Ontological Approach to Data Warehouse Source Integration

Francesco Di Tria, Ezio Lefons and Filippo Tangorra

Abstract In the early stages of data warehouse design, the integration of several source databases must be addressed. Data-oriented and hybrid methodologies need to consider a global schema coming from the integration of source databases, in order to start the conceptual design. Since each database relies on its own conceptual schema, in the integration process a reconciliation phase is necessary, in order to solve syntactical and/or semantic inconsistencies among concepts. In this paper, we present an ontology-based approach to perform the integration of different conceptual schemas automatically.

1 Introduction

The data warehouse design based on hybrid or data-driven methodologies [1, 2] always performs an analysis of the source databases, in order to understand the underlying semantic concepts inherent to the domain of interest [3]. Then, a global schema is produced that represents the source databases in an integrated way. To accomplish the integration process, a reconciliation phase is useful to solve syntactical and/or semantic inconsistencies among the concepts represented in the different databases.

While syntactical problems are traditionally solved using data dictionaries, the current trend to solve semantic problems is based on using an ontological approach [4] instead of the Entity/Relationship (ER) model. The reason is that ER schemas are used to represent locally-true concepts, or concepts that are true in the domain to be modeled. On the contrary, ontologies are used to represent necessarily-true concepts, or concepts that are true in any domain and, therefore, widely accepted and shared.

F. D. Tria (✉) · E. Lefons · F. Tangorra
Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro",
via Orabona 4, 70125 Bari, Italy
e-mail: francescoditria@di.uniba.it

E. Gelenbe and R. Lent (eds.), *Information Sciences and Systems 2013*,
Lecture Notes in Electrical Engineering 264, DOI: 10.1007/978-3-319-01604-7_25,

The process of constructing ontologies is called knowledge representation and it requires a lot of effort, because of the difficulties in formulating a comprehensive and rigorous conceptualization in the scope of a given domain. For these reasons, an ontology must be treated as a "reusable" artifact. When used in data warehousing, the designer must rely on a well-formed ontology, avoiding *ad hoc* modifications and extracting the parts of interest.

In this paper, we present an integration strategy based on an ontological approach to produce a global conceptual schema. This is then transformed into a relational schema to be given in input to a methodology for data warehouse design (in particular, the hybrid one we described in [5]).

The paper is organized as follows Sect. 2 presents the related work to the exploitation of ontologies in data warehouse design. Section 3 introduces our hybrid design methodology. Section 4 explains the integration strategy we propose. Section 5 shows a step-by-step example. Finally, Sect. 6 contains a few concluding remarks.

2 Related Work

The main issue in source integration deals with semantic inconsistencies among conceptual schemas. This can be addressed using techniques derived from artificial intelligence [6] and adopting an ontological approach, which is widely used in the semantic web [7].

An important work is described in [8]. The authors' approach is based on local ontologies for designing and implementing a single data source, inherent to a specific domain. Next, the data warehouse design process aims to create a global ontology coming from the integration of the local ontologies. Finally, the global ontology is used along with the logical schemas of the data sources to produce an integrated and reconciled schema, by mapping each local concept to a global ontological concept automatically.

The work of Romero and Abelló [9] is also based on an ontological approach but it skips the integration process and directly considers the generation of a multidimensional schema starting from a common ontology, namely *Cyc*. The final schema must be validated by the user in order to solve inconsistencies.

In [10], the authors propose a methodology to integrate data sources using a common ontology, enriched with a set of functional dependencies. These constraints support the designer in the choice of primary keys for dimension tables and allow the integration of similar concepts using common candidate keys.

3 Methodology Overview

The data warehouse design methodology we propose here is composed of the following phases, in that order:

- *Requirement analysis*. Decision makers' business goals are represented using the i^* framework for data warehousing [11]. The designer has to detect the information requirements and to translate them into a workload, containing the typical queries that allow the extraction of the required information. Then, the goals of the data warehouse must be transformed into a set of constraints, defining facts and dimensions to be included in the multidimensional schema. To this aim, both the workload and the constraints must be given in input to the conceptual design, in order to start the modeling phase in an automatic way.
- *Source analysis and integration*. The schemas of the different data sources must be analyzed and then reconciled, in order to obtain a global conceptual schema. The integration strategy is based on an ontological approach and, therefore, we need to work at the conceptual level. To this end, a reverse engineering from data sources to a conceptual schema is necessary, in order to deal with the concepts. The conceptual schema that results from the integration process must then be transformed into a relational schema, which constitutes the input to the data warehouse conceptual design. Since the transformation primitives from the conceptual to the logical levels are a well-known topic in literature, they are not addressed in this paper.
- *Data warehouse conceptual design*. This phase is based on the Graph-oriented Hybrid Multidimensional Model ($GrHyMM$, [5]) that identifies the facts in the source relational schema on the basis of constraints derived from the *Requirement analysis*. For each correctly-identified fact, it builds an attribute tree [12] to be remodeled using the constraints. Finally, the resulting attribute trees are checked in order to verify whether all the trees agree with the workload [13].
- *Data warehouse logical design*. The attribute trees are transformed into a relational schema—for instance, a snow-flake schema—considering each tree as a cube, having the root as the fact and the branches as the dimensions, possibly structured in hierarchies.
- *Data warehouse physical design*. The design process ends with the definition of the physical properties of the database on the basis of the specific features provided by the database system, such as indexing, partitioning, and so on.

We focus on *Source analysis and integration* phase of the methodology here.

4 Source Analysis and Integration

The preliminary step is the source analysis devoted to the study of the source databases. If necessary, the designer has to produce, for each data source, a conceptual schema along with a data dictionary, storing the description in the natural language

of the concepts modeled by the database. Then, the integration process proceeds incrementally using a binary operator that, given two conceptual schemas, produces a new conceptual schema.

Assumption Given the conceptual schemas $S_1, S_2, \ldots, S_n, n \geq 2$, we assume that
$G_1 = integration\,(S_1, S_2)$, and
$G_i = integration\,(G_{i-1}, S_{i+1})$, for $i = 2, \ldots, n-1$. □

In detail, the integration process of two databases S_i and S_j is composed of the following steps:

1. *Ontological representation*. In this step, we consider an ontology describing the main concepts of the domain of interest. If such an ontology does not exist, it must be built by domain experts. The aim is to build a shared and reusable *ontology*.
2. *Predicate generation*. For each concept in the ontology, we introduce a unary predicate. The output of this step is a set of *predicates*, which represents a vocabulary to build definitions of concepts using the first-order logic.
3. *Ontological definition generation*. For each concept in the ontology, we also introduce a definition on the basis of its semantic relationships. This definition is the description of the concept at the ontological level (that is, the common and shared definition). The output of this step is a set of *ontological definitions*.
4. *Entity definition generation*. For each entity present in the data sources and described in the data dictionary, we introduce a definition using the predicates. Therefore, an entity definition is a logic-based description of a concept in the database. The output of this step is a set of *entity definitions*.
5. *Similarity comparison*. Assuming that similar entities have a very close description, we can detect whether (a) entities that have different names refer to the same concept, and (b) entities that have the same name refer to different concepts. To do so, we utilize a set of inferring rules, the so-called *similarity comparison rules*, to analyze the logic-based descriptions and a metric to calculate the pairwise similarity of entity definitions.

In detail, given two schemas $S_i(A_i^1, A_i^2, \ldots, A_i^o)$ and $S_j(A_j^1, A_j^2, \ldots, A_j^m)$, where A_t^h is the hth entity of schema S_t, we compare the logic definition of A_t^h (for $h = 1, \ldots, o$) with that of A_j^q (for $q = 1, \ldots, m$). For each comparison, we calculate a similarity degree d and an output list K. The output list contains the possible ontological concepts shared by both the logic definitions.

Assuming we can compare the logical definitions of entities A_i^h and A_j^q and calculate both the similarity degree d and the output list K, we can observe one of the following cases:

(i) A_i^h is *equivalent* to A_j^q, if $d \geq x$;
(ii) A_i^h is a *generalization* of A_j^q, if the definition of A_j^q is part of the definition of A_i^h (or *vice versa*);
(iii) A_i^h and A_j^q are both *specializations* of a concept present in the ontology, if $0 < d \leq x$ and $K \neq \emptyset$;

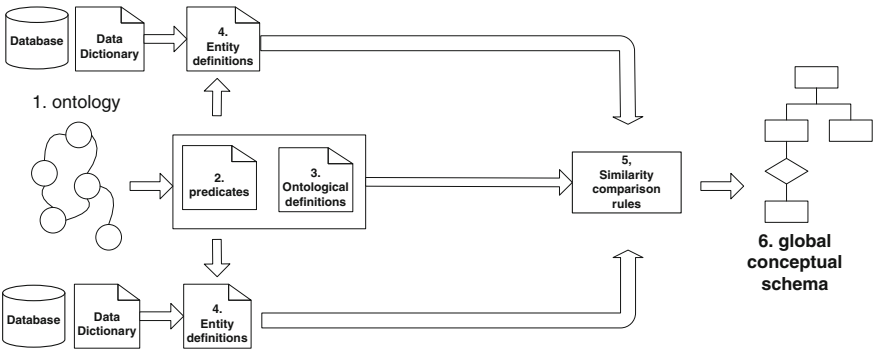

Fig. 1 Integration process diagram

(iv) A_i^h and A_j^q are *linked* via the relationship γ present in the ontology;

where x is a fixed threshold value. For convenience, we fixed x at 0.70.

6. *Global conceptual schema generation*. The final *global conceptual schema* G_u is built using the results obtained by the similarity comparison process and applying some generation rules.

   In detail, we have $G_u(A_u^1, A_u^2, \ldots, A_u^p)$, where for $s = 1, \ldots, p$,

   (i) $A_u^s = A_i^h \approx A_j^q$, if we observe case 5(i);
   (ii) $A_u^s = \{A_i^h, A_j^q\}$, if we observe case 5(ii);
   (iii) $A_u^s = \{K, A_i^h, A_j^q\}$, if we observe case 5(iii);
   (iv) $A_u^s = \{\gamma, A_i^h, A_j^q\}$, if we observe case 5(iv).

   Figure 1 shows the graphical representation of the integration process.

When a further schema S_w has to be integrated, the integration process starts from step 4, using the result of step 6 and the schema S_w.

5 Working Example

In this Section, we provide a complete example of source analysis and integration in order to highlight how the ontology supports the designer in the data warehouse conceptual design.

The case study aims to integrate two databases: (1) *Musical Instruments* and (2) *Fruit & Vegetables*. *Musical Instruments* is the database used by an on-line shop, in order to manage the sales of musical instruments and accessories. *Fruit & Vegetables* is the database used by a farm, in order to manage the wholesale of fruit and vegetables. Their essential conceptual schemas are provided in Fig. 2.

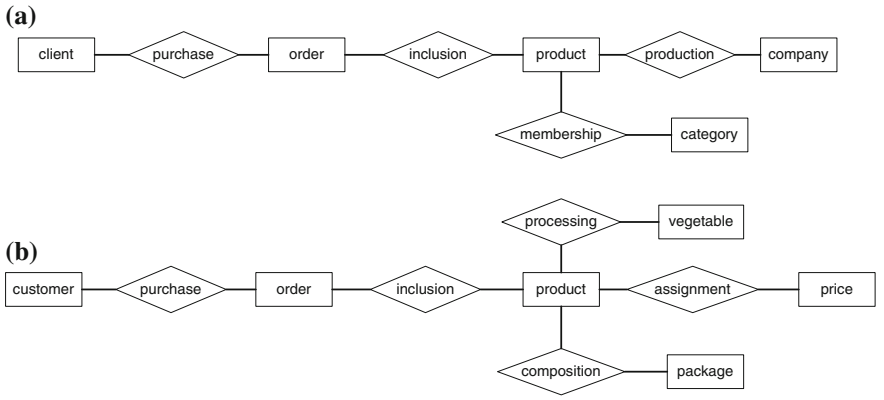

Fig. 2 Source databases: **a** *Musical Instruments*, and **b** *Fruit & Vegetables*

The first phase of the source integration is the ontological representation. To this end, we built our ontology starting from *OpenCyc*, the open source version of *Cyc* [14].

Therefore, we extracted from *OpenCyc* the concepts of interest [15] related to the business companies and sales activity, that is the most frequent domain in data warehousing. The relationships considered are $isA(X, Y)$ to indicate that X is a specialization of Y, and $has(X, Y)$ to indicate that X has an instance of Y.

Using the ontology previously introduced, we defined the predicates to be used as a vocabulary for the logical definitions of database entities. Each predicate corresponds to a concept present in the ontology. For each ontological concept, we also provide an extended definition, using the predicates previously introduced. So, we obtained a logical definition for each ontological concept.

The second phase is the generation of the entity definition.

For each database entity, we created a logical definition using the predicates we had previously generated. Indeed, such predicates represent the vocabulary for the construction of the concepts using the *first-order logic*.

Notice that these definitions often disagree with the ontological ones. In fact, entities are always defined without considering common and shared concepts, since entities represent local concepts. This means we assume that the database designer ignores the ontology. So, given $S_1(client, order, product, company, category)$ and $S_2(customer, order, product, vegetable, package, price)$, we have to create $G_1 = integration\,(S_1, S_2)$, by comparing each entity of S_1 with each entity of S_2.

The third and last phase is the comparison of the entity definitions in order to check whether two entities refer to the same concept or not. The comparison is done automatically using inferring rules defined in *first-order-logic*. These rules check the similarity degree between two lists L_1 and L_2 containing a logical definition of a database entity [16].

The similarity degree d is given by

$$d(n, l, m) = 0.5 \times \frac{l+1}{l+n+2} + (1 - 0.5) \times \frac{l+1}{l+m+2},$$

where

- n is the number of predicates p such that $p \in L_1$ and $p \in L_2$,
- l is the number of common predicates, and
- m is the number of predicates p such that $p \in L_1$ and $p \in L_2$.

The complete result of the case study is reported in Table 1. For each comparison between entities, both the similarity degree d (in the top cell) and the generalization list K (in the bottom cell) are reported. (The symbol "=" means that the entities are *equivalent.*)

5.1 Global Conceptual Schema Generation

Now we examine the results of the similarity comparison. We note that client and customer are always used as synonyms. However, the comparison results indicate that the client and customer have not been defined in the same manner and, therefore, they refer to different database entities. We observe that their similarity degree d is not zero and they present one common ontological concept (*viz*, *social Being*). This suggests introducing into the global schema G_1 the *Social Being* entity and two specializations corresponding to a client who is a social being with an account (that is, a registered user) and a client who is a social being with a legal title (that is, a company having a shop). This has been obtained by applying rule 6(iii) in Sect. 4.

Another generalization that has been detected is that between *product* in *Musical Instruments* and *product* in *Fruit & Vegetables*. Even if there is a syntactical concordance, the terms refer to very different items: the former refers to an instrument,

Table 1 Results of the comparison

		Fruit & Vegetables					
		customer	order	product	vegetable	package	price
Musical Instruments	client	0.367	0.128	0.166	0.183	0.183	0.25
		{*social Being*}	Ø	Ø	Ø	Ø	Ø
	order	0.155	0.888	0.252	0.284	0.311	0.222
		Ø	=	Ø	Ø	Ø	Ø
	product	0.162	0.118	0.583	0.325	0.287	0.229
		Ø	Ø	{*goods*}	Ø	Ø	Ø
	company	0.2	0.155	0.183	0.2	0.2	0.266
		Ø	Ø	Ø	Ø	Ø	Ø
	category	0.225	0.18	0.385	0.417	0.225	0.291
		Ø	Ø	Ø	Ø	Ø	Ø

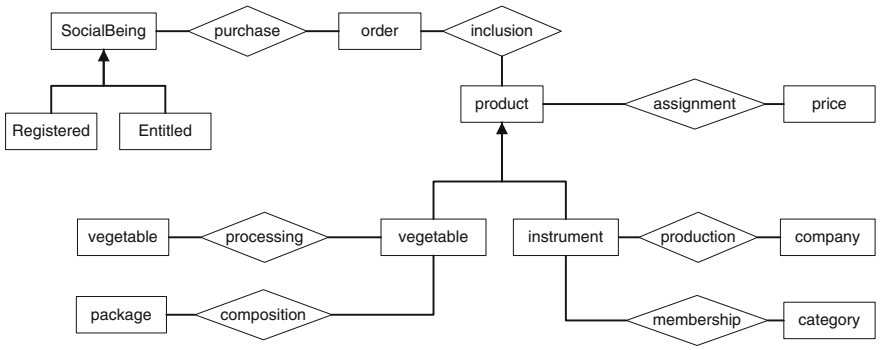

Fig. 3 Global conceptual schema

the latter to a fruit or a vegetable. However, these are both goods having a monetary value and are produced to be sold. Then, we created a generalization, namely *product*, which is an item having an assigned price. The specific products have been introduced as specializations, each with its own relationships. For example, an instrument is produced by a company. On the other hand, the producer of vegetables is missing information in the *Fruit & Vegetables* database. This has also been obtained by applying rule 6(iii) in Sect. 4.

Finally, it is worth noting that the *order* entities have been defined in the same way in both databases. So, they do not present a generalization because they refer to the same concept. This is the only overlapping concept. This has been obtained by applying rule 6(i) in Sect. 4.

The global conceptual schema G_1 is shown in Fig. 3. If we had to add the schema of another source database S_3, we should perform $G_2 = integration\,(G_1, S_3)$. After we have obtained a final global conceptual schema representing an integrated data source, we have to transform this schema into a relational one in order to use it in our hybrid data warehouse design methodology [5].

6 Conclusions

In this paper, we have presented an approach to construct a global conceptual schema coming from the integration of (two) relational databases. This approach is mainly based on an ontology containing common and shared concepts. The language we used is the predicate calculus, in order to define a set of inferring rules to automatically compare the similarity of two entities.

To this aim, we provide a logical definition for each database entity. For the sake of simplicity, we measure the similarity of two logical definitions and, using the comparison results, we are able to state whether the entities refer to the same concept or not. The final conceptual schema is built analyzing the comparison results. Thus,

the definition of an expert system able to reason on the comparison results is our next step to obtain an integrated schema automatically.

Since we claim that this approach can be applied also to attributes, future work will mainly focus on the problems arising when also the similarity between relationships has to be measured. Moreover, we intend to investigate the use of ontology in order to detect any type of ontological relationship existing between entities. In our opinion, this will allow the designer to discover inter-schema relationships.

References

1. Ballard C, Herreman D, Schau D, Bell R, Kim E, Valencic A (1998) Data modeling technique for data warehousing. IBM Corporation
2. Romero O, Abelló A (2009) A survey of multidimensional modeling methodologies. Int J Data Warehouse Min 5:1–23
3. Di Tria F, Lefons E, Tangorra F (2012) Hybrid methodology for data warehouse conceptual design by uml schemas. Inf Software Technol 54(4):360–379
4. Euzenat J, Shvaiko P (2007) Ontology matching. Springer
5. Di Tria F, Lefons E, Tangorra F (2011) GrHyMM: a graph-oriented hybrid multidimensional model. In: Proceedings of the 30th international conference on ER 2011, Brussels, Belgium, LNCS 6999. Springer, pp 86–97
6. Chen Z (2001) Intelligent data warehousing: from data preparation to data mining. CRC Press
7. Sure Y, Erdmann M, Angele J, Staab S, Studer R, Wenke D (2002) OntoEdit: collaborative ontology development for the semantic web. In: Proceedings of the 1st international semantic web conference, Sardinia, Italy, LNCS 2342. Springer Verlag, pp 221–235
8. Hakimpour F, Geppert A (2002) Global schema generation using formal ontologies. In: Proceedings of the 21st international conference on conceptual modeling, Tampere, Finland, LNCS 2503. Springer, pp 307–321
9. Romero O, Abelló A (2010) A framework for multidimensional design of data warehouses from ontologies. Data Knowl Eng 69:1138–1157
10. Bakhtouchi A, Bellatreche L, Ait-Ameur Y (2011) Ontologies and functional dependencies for data integration and reconciliation. In: Proceedings of the 30th international conference on ER 2011, Brussels, Belgium, LNCS 6999. Springer, pp 98–107
11. Mazón JN, Trujillo J, Serrano M, Piattini M et al (2005) Designing data warehouses: from business requirement analysis to multidimensional modeling. In: Cox K (ed) Requirements engineering for business need and it alignment. University of New South, Wales Press, pp 44–53
12. dell'Aquila C, Di Tria F, Lefons E, Tangorra F (2009) Dimensional fact model extension via predicate calculus. In: Proceedings of the 24th international symposium on computer and information sciences. IEEE Press, North Cyprus, pp 211–217
13. dell'Aquila C, Di Tria F, Lefons E, Tangorra F (2010) Logic programming for data warehouse conceptual schema validation. In: Proceedings of the 12th international conference on data warehousing and knowledge discovery, Bilbao, Spain, LNCS 6263. Springer, pp 1–12
14. Lenat DB (1995) Cyc: a large-scale investment in knowledge infrastructure. Commun ACM 38(11):32–38
15. Reed S, Lenat DB (2002) Mapping ontologies in Cyc. AAAI 2002 Conference workshop on ontologies for the semantic web. Edmonton, Canada
16. Ferilli S, Basile TMA, Biba M, Di Mauro N, Esposito F (2009) A general similarity framework for Horn clause logic. Fundam Inf 90(1–2):43–66