Reference Model and Perspective Schemata Inference for Enterprise Data Integration

Valéria Magalhães Pequeno and João Carlos Moura Pires

CENTRIA, Departamento de Informática,
Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa
2829-516, Caparica, Portugal
vmp@fct.unl.pt, jmp@di.fct.unl.pt

Abstract. We propose a declarative approach based on the creation of a reference model and perspective schemata to deal with the problem of integrating data from multiple, possibly heterogeneous and distributed, databases. The former provides a common semantic, while the latter connects schemata. This paper focuses on deduction of new perspective schemata using a proposed inference mechanism. A proof-of-concept prototype, based on Logic Programming, is slightly presented.

1 Introduction

One of the leading issues in database research is to develop flexible mechanisms for providing integrated access to multiple, distributed, heterogeneous databases and other information sources. A wide range of techniques has been developed to address this problem, including approaches based on creation of Data Warehouses (DWs), and Federated Database Systemss (FDBSs). DWs are highly specialised database systems which contain the unified history of an enterprise at a suitable level of detail for decision support. All data is integrated into, normally, a single repository, with a generalised and global schema. A FDBS enables a unified virtual view of one or more autonomous sources of information to hide data heterogeneity from applications and users. Closely coupled FDBSs, those that occur in DWs, provides a global schema expressed in a common, "canonical" data model. Unlike a DW, a FDBS leaves data at the source.

One of the main drawbacks of these approaches is the difficulty in developing a single (global or common) database schema that captures all the nuances of diverse data types, and expresses a unified view of the enterprise. The designer usually deal with incompatible data models, characterised by subtle differences in structure and semantic. He/she should define mappings between the global and source information schemata. These problems are hardest to deal with because of the rapid growth of the data volume and the data model complexity (both in sources and in global schema), which implies the rise of the difficulty of managing and understanding these models [1].

In order to deal with these problems, it is proposed to take a declarative approach, based on the creation of a reference model and, perspective schemata.

A Reference Model (conceptual model, business data model, or enterprise data model) is an abstract framework that represents the information used in an enterprise from a business viewpoint. It provides a common semantic that can be used to guide the development of other models and help with data consistency [1]. Its benefits include: a) reduction of the development risk by ensuring that all implemented systems correctly reflect the business environment [1]; b) helping with the project scope definition once the designer can use the reference model to identify the information that will be addressed by the systems; c) serving as basis for multiple products such as application systems, DWs, and FDBSs [2, 1], being a more stable basis for identifying information requirements to the DW systems than user query requirements, which are unpredictable and subject to frequent change [3].

A perspective schema describes a data model, part or whole (*the target*), in terms of other data models (*the base*). It represents the mapping between the base schemata (e.g., the information sources) and the target schema (e.g., the reference model). In the proposed approach, the relationship between the base and the target schemata is made explicitly and declaratively through correspondence assertions. By using the perspective schemata the designer has an explicit and formal representation, with well defined semantic, which allows to: evince diverse points of views of the (same or different) source information; b) deal with semantic heterogeneity in a declarative way; and c) reuse (a same perspective schema can used simultaneously in several (application, DW, FDBS) systems.

An advantage of the proposed approach is that by using the reference model the designer does not need to map each models. This effort is theoretically reduced since schemata (source or global) must only align with the reference model, rather than with each participating schema. Thus, the designer of the DW system (or FDBS) only needs to describe the mapping between the DW and the reference model, and he/she cannot be not involved with the mapping between the different sources. The designer can describe the global system without concerns about where the sources are or how they are stored. Furthermore, the mapping between the global schema and its sources are automatically generated by an inference mechanism. This paper focuses on the deduction of new perspective schemata using a proposed inference mechanism.

The remainder of this paper is laid out as follows. Section 2 concisely mentions representative works in the data integration area. Section 3 presents an overview of the reference model-based framework proposed in [4]. Section 4 briefly describes the language to define the perspective schemata. Section 5 details the process to infer new perspective schemata. The paper ends with Section 6, which points out new features of the approach presented here and for ongoing or planned future work on this topic.

2 Related Work

The database community has been engaged for many years with the problem of data integration. Research in this area has developed in several important

directions: schema matching and data quality, to cite a couple (see [5] for a survey), which covers different architectures (e.g., FDBSs and DWs), representation of data and involved data models (e.g., relational and non-structured). Recent research in FDBSs has included: behaviour integration [6], integration of non-traditional data (e.g., biomedical [7, 8], intelligence data [9], and web source [10]), interactive integration of data [11, 12], and federated data warehouse systems [13]. All these approaches use a global schema, but do not deal with a reference model. Similar to the research of the current paper, [10] uses correspondence assertions (in this case, for specifying the semantics of XML-based mediators). However, their CAs only deal with part of the semantic correspondence managed here. Furthermore, they assume that there is a universal key to determine when two distinct objects are the same entity in the real-world, which is often an unreal supposition.

Researches in DWs have focused on technical aspects such as multidimensional data models (e.g., [14–19]) as well as the materialised view definition and maintenance (e.g., [20]). In particular, the most conceptual multidimensional models are extensions to the Entity-Relationship model (e.g., [21–24]) or extensions to UML (e.g., [25–27]).

The work in [28] focuses on an ontology-based approach to determine the mapping between attributes from the source and the DW schemata, as well as to identify the transformations required for correctly moving data from source information to the DW. Their ontology, based on a common vocabulary as well as a set of data annotations (both provided by the designer), allows formal and explicit description of the semantic of the sources and the DW schemata. However, their strategy requires a deep knowledge of all schemata involved in the DW system, which is usually not the usual case. In our research, it is dispensable, since each schema (source or DW) needs to be related only to the reference model. Additionally, we deal with the matching of instances (i.e., the problem to identify the same instances of the real-world that are differently represented in the diverse schemata), and the work in [28] does not.

The closest approach to our research is described in [29]. Similar to our study, their proposal included a reference model (cited as "enterprise model") designed using an Enriched Entity-Relationship (EER) model. However, unlike our research, all their schemata, including the DW, are formed by relational structures, which are defined as views over the reference model. Their proposal provides the user with various levels of abstraction: conceptual, logical, and physical. In their conceptual level, they introduce the notion of intermodel assertions that precisely capture the structure of an EER schema or allow for the specifying of the relationship between diverse schemata. However, any transformation (e.g., restructuring of schema and values) or mapping of instances is deferred for the logical level, unlike the current work. In addition, they did not deal with complex data, integrity constraints, and path expressions, as our research does.

3 The Framework

The proposal presented in [4] offers a way to express the existing data models (source, reference model, and global/integrated schema) and the relationship between them. The approach is based on *Schema language* (L_S) and *Perspective schema language* (L_{PS}).

The language L_S is used to describe the actual data models (source, reference model, and global/integrated schema). The formal framework focuses on an object-relational paradigm, which includes definitions adopted by the main concepts of object and relational models as they are widely accepted([30, 31]).

The language L_{PS} is used to describe *perspective schemata*. A perspective schema is a special kind of model that describes a data model (part or whole) (*the target*) in terms of other data models (*the base*). In Fig. 1, for instance, $P_{s'1|RM}$, $P_{s'2|RM}$, $P_{s4|RM}$, ..., $\mathbf{P}_{sn|RM}$ are perspective schemata that map the reference model (**RM**) in terms of the sources ($\mathbf{S}'_1$, $\mathbf{S}'_2$, $\mathbf{S}_4$, ..., $\mathbf{S}_n$). L_{PS} mainly extends L_S with two components: Correspondence Assertions (CAs) and Matching Functions (MFs). CAs formally specify the relationship between schema components. MFs indicate when two data entities represent the same instance of the real world. L_{PS} includes data transformations, such as names conversion and data types conversion.

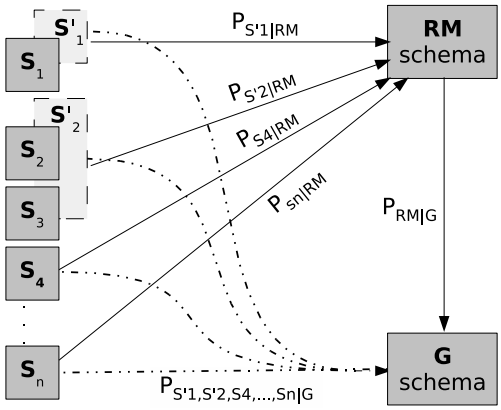

Fig. 1. Proposed architecture

Figure 1 illustrates the basic components of the proposed architecture and their relationships. The schemata **RM**, $\mathbf{S}_1$,...,$\mathbf{S}_n$ and **G** are defined using the language L_S and represent, respectively, the reference model, the sources $\mathbf{S}_1$,...,$\mathbf{S}_n$, and a global schema. The schemata $\mathbf{S'}_1$ and $\mathbf{S'}_2$ are defined using the language L_{PS}, and represent, respectively, a viewpoint of $\mathbf{S}_1$ and an integrated viewpoint of $\mathbf{S}_2$ and $\mathbf{S}_3$. $\mathbf{S'}_1$ and $\mathbf{S'}_2$ are special kinds of perspective schemata (called *view schema*), since the target schema is described in the scope of a perspective schema, instead of just referring to an existing model. The relationships

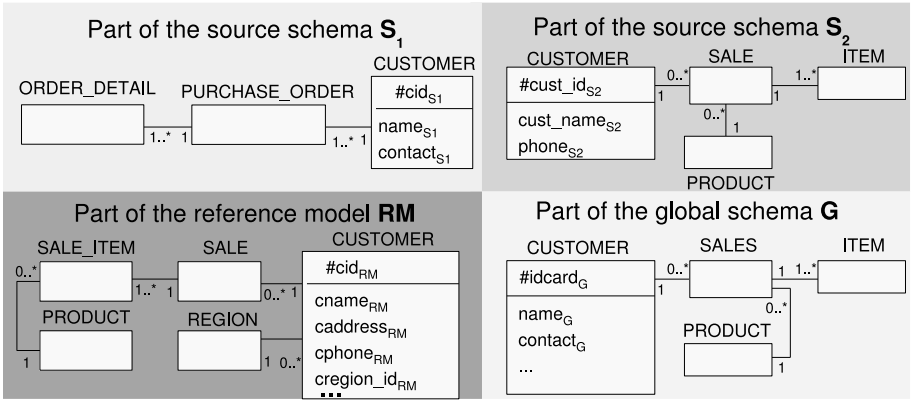

Fig. 2. Motivating example

between the target and the base schemata are shown through the perspective schemata $\mathbf{P}_{s'1|RM}$, $\mathbf{P}_{s'2|RM}$, $\mathbf{P}_{s4|RM}$, ..., $\mathbf{P}_{sn|RM}$, $\mathbf{P}_{RM|G}$, and $\mathbf{P}_{s'1,s'2,s4,...,sn|G}$ (denoted by arrows). In the current research, $\mathbf{P}_{s'1,s'2,s4,...,sn|G}$ can be automatically deduced by the proposed inference mechanism. The next Sect. illustrates, through the examples, the language L_{PS}, and the Sect. 5 presents the proposed inference mechanism. For a more detailed and formal description of L_S and L_{PS} languages, the reader is referred to [32, 33, 4].

4 Perspective Schema Language

The remainder of the paper, considers a simple sales scenario comprising two data sources $\mathbf{S}_1$ and $\mathbf{S}_2$, a reference model **RM**, and a global schema **G**. The schemata are shown in Fig. 2. All properties that are key to a relation (or class) are shown in Fig. 2 using "♯" before their names.

The language L_{PS}, as we mentioned before, is used to define perspective schemata. Usually, a perspective schema is formed by the following components:

1. *Name* is a schema name with the notation: $\mathbf{P}_{\mathbf{S}|\mathbf{T}}$, being **S** the name of one or more base schemata and **T** the name of the target schema. In Fig. 1, for instance, $\mathbf{P}_{s'1|RM}$ is a name of a perspective schema whose base is S'_1 and the target is **RM**;
2. *'Require' declarations* express the subset of the components of the target schema (classes, relations, keys, and foreign keys) that will be necessary in the perspective schema;
3. *Matching Function signatures* indicate which matching functions must be implemented to determine when two objects/tuples are distinct representations of the same object in the real-world;
4. *Correspondence Assertions* establish the semantic correspondence between schemata's components.

The target schema may have much more information than is required to represent in a perspective schema, namely when the target is the Reference Model. Hence, it is necessary to clearly indicate which elements of the target schema are in the scope of the perspective schema. This is done in L_{PS} using 'require' declarations. For instance, consider the perspective schema $\mathbf{P}_{\mathbf{S}_2|\mathbf{RM}}$ between the schemata **RM** and $\mathbf{S}_2$, both as presented in Fig. 2. For this perspective schema, four relations from **RM** are needed (PRODUCT, CUSTOMER, SALE, and SALE_ITEM). The 'require' declaration to relation CUSTOMER, for example, would be as follows:

$$\text{require}(\text{CUSTOMER}, \{\mathbf{cid}_{\text{RM}}, \mathbf{cname}_{\text{RM}}, \mathbf{cphone}_{\text{RM}}\})$$

Note that, for instance, the properties $\mathbf{cregion\_id}_{\text{RM}}$ and $\mathbf{caddress}_{\text{RM}}$ from **RM**.CUSTOMER are not declared as being required.

4.1 Matching Functions

From a conceptual viewpoint, it is essential to provide a way to identify instances of different models that represent the same entity in the real-world in order to combine them appropriately. This identification (we call it *the instance matching problem*) usually is expensive to compute, due to the complex structure and the character of the data. It is not part of the current research deal with the full instance matching problem. We assume, usually like the DW designers do, that data quality tools were used and that, for instance, duplicates were removed. Even then, in a data integration context, the instance matching problem still persists. The proposal presented in [4] is using MF signatures, which points to situations that should be considered in a data integration context. These signatures define a 1:1 correspondence between the objects/tuples in families of corresponding classes/relations. In particular, the work shown in [4] is based on the following matching function signature:

$$\mathbf{match} : ((\mathbf{S}_1\,[\mathrm{R}_1]\,, \tau_1) \times (\mathbf{S}_2\,[\mathrm{R}_2]\,, \{\tau_2\})) \rightarrow \text{Boolean}\ , \qquad (1)$$

being $\mathbf{S}_i$ schema names, R_i class/relation names, and τ_i the data type of the instances of R_i, for i $\in$ {1,2}. When both arguments are instanced, **match** verifies whether two instances are semantically equivalent or not. If only the first argument is instanced (i.e., $\mathbf{S}_1.\mathrm{R}_1$) then it obtains the semantically equivalent $\mathbf{S}_2.\mathrm{R}_2$ instance of the given $\mathbf{S}_1.\mathrm{R}_1$ instance, returning true when it is possible, and false when nothing is found or when there is more than one instance to match.

In some scenarios one-to-many correspondence between instances are common (e.g., when historical data is stored in the DW). In this case, a variant of **match** should be used, which has the following form:

$$\mathbf{match} : ((\mathbf{S}_1\,[\mathrm{R}_1]\,, \tau_1) \times (\mathbf{S}_2\,[\mathrm{R}_2\,(\mathbf{predicate})]\,, \{\tau_2\})) \rightarrow \text{Boolean}\ . \qquad (2)$$

In (2), **predicate** is a boolean condition that determines the context in which the instance matching must be applied in $\mathbf{S}_2.\mathrm{R}_2$. An example of a matching function signature involving schemata of Fig. 2 is as follows:

$$\textbf{match} : ((\textbf{RM}\,[\text{CUSTOMER}]\,, \tau_1) \times (\textbf{G}\,[\text{CUSTOMER}]\,, \{\tau_2\})) \rightarrow \text{Boolean} \qquad (3)$$

The implementation of the matching functions shall be externally provided, since their implementation is very close to the application domain and to the application itself.

4.2 Correspondence Assertions

The semantic correspondence between schemata's components is declared in the proposal presented in [4] through the CAs, which are used to formally assert the correspondence between schema components in a declarative way. CAs are classified in four groups: Property Correspondence Assertion (PCA), Extension Correspondence Assertion (ECA), Summation Correspondence Assertion (SCA), and Aggregation Correspondence Assertion (ACA). Examples of CAs are shown in Table 1 and explained in this Sect..

Table 1. Examples of correspondence assertions

Property Correspondence Assertions (PCAs)
ψ_1: $\mathbf{P_{RM\|G}}\,[\text{CUSTOMER}] \bullet \mathbf{idcard}_{\text{G}} \rightarrow \text{numberTOtext}\,(\mathbf{RM}\,[\text{CUSTOMER}] \bullet \mathbf{cid}_{\text{RM}})$
ψ_2: $\mathbf{P_{RM\|G}}\,[\text{CUSTOMER}] \bullet \mathbf{contact}_{\text{G}} \rightarrow \mathbf{RM}\,[\text{CUSTOMER}] \bullet \mathbf{cphone}_{\text{RM}}$
Extension Correspondence Assertions (ECAs)
ψ_3: $\mathbf{P_{RM\|G}}\,[\text{CUSTOMER}] \rightarrow \mathbf{RM}\,[\text{CUSTOMER}]$
ψ_4: $\mathbf{S}_v\,[\text{CUSTOMER}] \rightarrow \mathbf{S}_1\,[\text{CUSTOMER}] \sqsupset\!\bowtie\!\sqsubset \mathbf{S}_2\,[\text{CUSTOMER}]$
Summation Correspondence Assertion (SCA)
ψ_5: $\mathbf{P_{S_3\|RM}}\,[\text{PRODUCT}]\,(\mathbf{pid}_{\text{RM}}) \rightarrow \text{normalise}\,(\mathbf{S}_3\,[\text{PRODUCT\_SALES}]\,(\mathbf{product\_number}_{\text{S3}}))$

PCAs relate properties of a target schema to the properties of base schemata. They allow dealing with several kinds of semantic heterogeneity such as: *naming conflict* (for instance synonyms and homonyms properties), *data representation conflict* (that occur when similar contents are represented by different data types), and *encoding conflict* (that occur when similar contents are represented by different formats of data or unit of measures). For example, the PCAs ψ_1 and ψ_2 (see Table 1) deal with, respectively, *data representation conflict* and *naming conflict*. ψ_1 links the property $\mathbf{idcard}_G$ to the property $\mathbf{cid}_{RM}$ using the function **numberTOtext** to convert the data type from *number* to *text*. ψ_2 assigns $\mathbf{contact}_G$ to $\mathbf{cphone}_{RM}$.

ECAs are used to describe which objects/tuples of a base schema should have a corresponding semantically equivalent object/tuple in the target schema. For instance, the relation **G**.CUSTOMER is linked to relation **RM**.CUSTOMER through the ECA ψ_3 presented in Table 1. ψ_3 determines that **G**.CUSTOMER and **RM**.CUSTOMER are equivalent (i.e., for each tuple of CUSTOMER of the schema

RM there is one semantically equivalent tuple in CUSTOMER of the schema **G**, and vice-versa).

There are five different kinds of ECAs: equivalence, selection, difference, union, and intersection, being the ECA of union similar to the *natural outer-join* of the usual relational models. For instance, consider the view schema S_v (not presented in any figure) with the relation CUSTOMER, which is related to the relations CUSTOMER of the schemata S_1 and S_2 through the ECA ψ_4 shown in Table 1. ψ_4 determines that CUSTOMER in S_v is the union/join of CUSTOMER in S_1 and CUSTOMER in S_2 (i.e., for each tuple in CUSTOMER of S_1 there is one semantically equivalent tuple in CUSTOMER of S_v, or for each tuple in CUSTOMER of S_2 there is one semantically equivalent tuple in CUSTOMER of S_v, and vice-versa). In an ECA, any relation/class can appear with a selection condition, which determines the subset of instances of the class/relation being considered. This kind of ECA is especially important to the DW because through it the current instances of the DW can be selected and related to the instances of their sources (which usually do not have historical data).

SCAs are used to describe the summary of a class/relation whose instances are related to the instances of another class/relation by breaking them into logical groups that belong together. They are used to indicate that the relationship between classes/relations involve some type of aggregate functions (called SCA of groupby) or a normalisation process (called SCA of normalisation)[1]. For example, consider the source schema $\mathbf{S}_3$ (not presented in any figure), which contains a denormalised relation PRODUCT_SALES(**product_number**$_{S3}$, **product**$_{S3}$, **quantity**$_{S3}$, **price**$_{S3}$, **purchase_order**$_{S3}$) and the schema **RM** presented in Fig. 2. PRODUCT_SALES holds information about sold items in a purchase order as well as information logically related to products themselves, which could be in another relation, as occurring in schema **RM**. The SCA ψ_5, displayed in Table 1, determines the relationship between PRODUCT_SALES and **RM**.PRODUCT when a normalisation process is involved (i.e., it determines that **RM**.PRODUCT is a normalisation of $\mathbf{S}_3$.PRODUCT_SALES based on distinct values of property **product_number**$_{S3}$).

ACAs link properties of the target schema to the properties of the base schema when a SCA is used. ACAs associated to SCAs of groupby contains aggregation functions supported by most of the queries languages, like SQL-99 [34] (i.e., *summation, maximum, minimum, average* and *count*). The ACAs, similar to the PCAs, allow for the description of several kinds of situations; therefore, the aggregate expressions can be more detailed than simple property references. Calculations performed can include, for example, ordinary functions (such as sum or concatenate two or more properties' values before applying the aggregate function), and Boolean conditions (e.g., count all male students whose grades are greater or equal to 10).

[1] This research also deals with denormalisations, which is defined using *path expressions* (component of the language L_S).

5 Inference Mechanism

This proposal provides an inference mechanism to automatically infer a new perspective schema (see Fig. 3(c)), given:

1. a set of *origin* schemata and their associated perspective schemata, which take the *origin* schemata as *base* and the reference model as *target* (see Fig. 3(a));
2. a *destination* schema and its associated perspective schema, which take the reference model as *base* and the *destination* schema as *target* (see Fig. 3(b)).

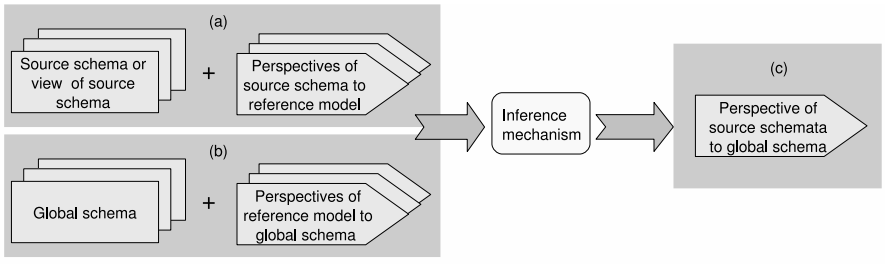

Fig. 3. Sketch of the inference mechanism

In context of the Fig. 1, the perspective schema $\mathbf{P}_{s'1,s'2,s4,...,sn|G}$ can be inferred taking as *origin* the schemata $\mathbf{S}'_1$, $\mathbf{S}'_2$, $\mathbf{S}_4$, ..., $\mathbf{S}_n$ as well as the perspective schemata $\mathbf{P}_{s'1|RM}$, $\mathbf{P}_{s'2|RM}$, $\mathbf{P}_{s4|RM}$, ..., $\mathbf{P}_{sn|RM}$, and as *destination* the schema $\mathbf{G}$ as well as the perspective schema $\mathbf{P}_{RM|G}$.

The inferred perspective schema will have as *base* a subset of *origin* schemata, and as *target* the *destination* schema. Its *'require' declarations* will be the same *'require' declarations* present in the perspective schema associated to the *destination* schema. The *MF signatures* and *CAs* of the inferred perspective schema will be automatically generated using a rule-based rewriting system.

5.1 The Rewriting System

The rule-based rewriting system is formed by a set of rules having the general form:

$$\textbf{Rule}: \frac{X \Rightarrow Y}{Z} \quad \text{(read } X \text{ is rewritten as } Y \text{ if } Z \text{ is valid)} \ , \tag{4}$$

In (4), **Rule** is the name of the rule. X and Y can be formed by any of the following expressions: a CA pattern expression, a MF pattern signature, or a component pattern expression. CA pattern expressions and MF pattern signatures are expressions conforming to the L_{PS} syntax to declare, respectively,

CAs and MF signatures, being that some of their elements are variables to be used in a unification process. Component pattern expressions are expressions conforming to the L_S or the L_{PS} syntax to represent components that can appear in CAs or in MF signatures (e.g., properties, path expressions, functions with n-ary arguments, values, or conditions of selection (predicates)), being that some of their elements are variables to be used in a unification process. Z is a condition formed by a set of CA pattern expressions, or expressions of the form $A \Rightarrow B$ such that A and B are component pattern expressions.

A condition Z is valid when all of its expressions are valid: a) the CA pattern expression is valid if there is a CA, which is declared in one of the perspective schemata associated to the *origin* or *destination* schemata, that unifies with it; b) the expression of the form $A \Rightarrow B$, such that A and B are component pattern expressions, is valid if there is a rule which unifies with it and which is recursively applied.

When X and Y are CA pattern expressions, the rules are rewritten-rules that rewrite CAs in other CAs (RR-CAs). When X and Y are MF pattern signatures, the rules are rewritten-rules that rewrite MFs in other MFs (RR-MFs). When X and Y are component pattern expressions, the rules are substitution-rules that rewrite components in other components (RR-Cs). The latter are used as an intermediary process by the RR-CAs and RR-MFs.

An example of a RR-CA is as follows:

$$\textbf{RR-CA1 :}\ \frac{\mathrm{P}_{\mathrm{RM|D}}\left[\underline{\mathrm{C}^{\mathrm{D}}}\right] \rightarrow \mathrm{RM}\left[\underline{\mathrm{C}^{\mathrm{RM}}}\right] \Rightarrow \mathrm{P}_{\underline{S}|\mathrm{D}}\left[\underline{\mathrm{C}^{\mathrm{D}}}\right] \rightarrow \underline{\mathbf{K}^{\mathrm{S}}}}{\mathrm{P}_{\underline{S}|\mathrm{RM}}\left[\underline{\mathrm{C}^{\mathrm{RM}}}\right] \rightarrow \underline{\mathbf{K}^{\mathrm{S}}}}\ . \tag{5}$$

In (5), all variables are indicated by an underline. **D** is the *destination* schema, **RM** is the reference model, and **S** is a variable that will be instantiated with some of the *origin* schemata. C^{D} is a variable that will be instantiated with a class/relation of the schema **D**; mutatis mutandis to C^{RM}. **K** is a variable that will be instantiated with the right side of a CA pattern expression of extension. The letter S in $\mathbf{K}^{\mathrm{S}}$ means that all elements in that expression belong to schema **S**. The value of **S** and **K** will depend on which CA, that is declared in the perspective schema associated to some *origin* schemata, will unify with the condition of the rule. The notation in (5) will be used through the paper to explain examples of rules.

The rule **RR-CA1** rewrites an ECA of equivalence connecting a class/relation C^{D} to a class/relation C^{RM}, into an ECA connecting C^{D} to a class/relation C^{S}; when is provided an ECA that connect C^{RM} to C^{S}.

An example of a RR-MF is as follows:

$$\textbf{RR-MF1 :}\ \frac{\begin{array}{l}\textbf{match} : \left(\left(\mathrm{RM}\left[\underline{\mathrm{C}^{\mathrm{RM}}}\right], \underline{\tau^{\mathrm{RM}}}\right) \times \left(\mathrm{D}\left[\underline{\mathrm{C}^{\mathrm{D}}}\right], \underline{\tau^{\mathrm{D}}}\right)\right) \rightarrow \mathrm{Boolean} \Rightarrow \\ \textbf{match} : \left(\left(\underline{\mathrm{S}}\left[\underline{\mathrm{C}^{\mathrm{S}}}\right], \underline{\tau^{\mathrm{S}}}\right) \times \left(\mathrm{D}\left[\underline{\mathrm{C}^{\mathrm{D}}}\right], \underline{\tau^{\mathrm{D}}}\right)\right) \rightarrow \mathrm{Boolean}\end{array}}{\mathrm{P}_{\underline{S}|\mathrm{RM}}\left[\underline{\mathrm{C}^{\mathrm{RM}}}\right] \rightarrow \underline{\mathrm{S}}\left[\underline{\mathrm{C}^{\mathrm{S}}}\right]}\ . \tag{6}$$

In (6), τ is a data type. The rule **RR-MF1** rewrites a MF signature that matches a class/relation C^{RM} to a class/relation C^{D}, into a MF signature that matches a class/relation C^{S} to C^{D}, when is provided an ECA of equivalence that connects C^{RM} to C^{S}.

An example of a RR-C is as follows:

$$\textbf{RR-C1}: \frac{\mathrm{RM}\left[\underline{\mathrm{C}^{RM}}\right] \bullet \underline{\boldsymbol{p}^{\mathrm{RM}}} \Rightarrow \underline{\mathbf{A}^{\mathrm{S}}}}{\mathrm{P}_{\underline{S}|\mathrm{RM}}\left[\underline{\mathrm{C}^{\mathrm{RM}}}\right] \bullet \underline{\boldsymbol{p}^{\mathrm{RM}}} \rightarrow \underline{\mathbf{A}^{\mathrm{S}}}} \quad . \tag{7}$$

In (7), $\boldsymbol{p}^{\mathrm{RM}}$ is a variable that will be instantiated with a property of a class/relation (the symbol "$\bullet$" means that $\boldsymbol{p}^{\mathrm{RM}}$ is defined in C^{RM}). $\mathbf{A}$ is a variable that will be instantiated with a component pattern expression. Similar to $\mathbf{K}^{\mathrm{S}}$ in (5), the letter S in $\mathbf{A}^{\mathrm{S}}$ means that all elements into that expression belong to schema $\mathbf{S}$. The value of $\mathbf{S}$ and $\mathbf{A}$ will depend on which CA declared in the perspective schema associated to some *origin* schemata will unify with the condition of the rule.

The rule **RR-C1** rewrites a property $\boldsymbol{p}^{\mathrm{RM}}$ into a property, a path expression, or a function of some *origin* schema; when is provided an PCA that connects $\boldsymbol{p}^{\mathrm{RM}}$ to that property, path expression, or function. The whole set of proposed rules can be found in [35].

5.2 Implementation Issues

A pseudo-code detailing as new CAs are deduced is shown in Fig. 4.

```
1: procedure INFER_CAs(A^G → A^RM, CAs)
2:     repeat
3:         find A^G → A^Si applying the inference rule R:
4:         R: (A^G → A^RM ⇒ A^G → A^Si) / conditions;
5:         add A^G → A^Si to CAs;
6:     until all rules for rewriting CAs have been tested
7: end procedure
```

Fig. 4. A pseudo-code of the inference mechanism to generate new CAs

In Fig. 4, $\mathbf{G}$ is a *destination* schema, $\mathbf{RM}$ the reference model, and $\mathbf{S}_i$, i ≥ 1, *origin* schemata. The algorithm tries to find, for each CA of the general form $A^{\mathbf{G}} \rightarrow A^{\mathbf{RM}}$ (assigning the global schema to the reference model), one or more CAs $A^{\mathbf{G}} \rightarrow A^{\mathbf{Si}}$ as a result of applying to $A^{\mathbf{G}} \rightarrow A^{\mathbf{RM}}$ some rule for rewriting CAs. Notice that, in the condition of the rule can appear expressions of the form $A \Rightarrow B$. In this case, the recursivity will be present. In order to reduce the search space, all rules of the inference mechanism are oriented. Let us see an example. A new ECA:

$$\mathbf{P_{S1|G}}\,[\text{CUSTOMER}] \rightarrow \mathbf{S1}\,[\text{CUSTOMER}]$$

can be created based on ψ_3 (see Table 1) by using the rule **RR-CA1**, since the CA ψ_6 is defined in perspective schema $\mathbf{P}_{S1|RM}$ (see Table 2).

Table 2. More examples of correspondence assertions

Extension Correspondence Assertion (ECA)
ψ_6: $\mathbf{P_{S1\|RM}}\,[\text{CUSTOMER}] \rightarrow \mathbf{S1}\,[\text{CUSTOMER}]$
ψ_7: $\mathbf{P_{S2\|RM}}\,[\text{CUSTOMER}] \rightarrow \mathbf{S2}\,[\text{CUSTOMER}]$

A pseudo-code detailing as new MF signatures are deduced is shown in Fig. 5. In Fig. 5, **K** and **L** are pairs (classes/relations, data type) of the reference model or of the *destination* schema, while $\mathbf{K}'$ and $\mathbf{L}'$ are pairs (classes/relations, data type) of some *origin* or *destination* schemata. For each MF **M** that is declared in the perspective schema associated to the *destination* schema, the algorithm tries to find one or more MFs as a result of applying to **M** some rule for rewriting MFs. For instance, two new MF signatures:

$$\textbf{match}:((\mathbf{S1}[\text{CUSTOMER}],\tau_1)\times(\mathbf{G}[\text{CUSTOMER}],\{\tau_2\}))\rightarrow\text{Boolean}$$
$$\textbf{match}:((\mathbf{S2}[\text{CUSTOMER}],\tau_1)\times(\mathbf{G}[\text{CUSTOMER}],\{\tau_2\}))\rightarrow\text{Boolean}$$

can be created based on MF signature presented in (3) by using the rule **RR-MF1** twice, since as the CAs ψ_6 and ψ_7 are defined, respectively, in perspective schemata $\mathbf{P}_{S1|RM}$ and $\mathsf{P}_{S2|RM}$ (see Table 2).

```
1: procedure INFER_MFS(match:(K×L)→Boolean ,MFs)
2:    repeat
3:       find match:(K′×L′)→Boolean applying the inference rule R:
4:       R: match:(K×L)→Boolean ⇒ match:(K′×L′)→Boolean / conditions ;
5:       add match:(K′×L′) → Boolean to MFs;
6:    until all rules for rewriting MFs have been tested
7: end procedure
```

Fig. 5. A pseudo-code of the inference mechanism to generate new MFs

A pseudo-code with the iteration of the process to generate a new perspective schema is shown in Fig. 6. In Fig. 6, P_T is a perspective schema from the reference model to the global model; P_j, $1 \leq \text{j} \leq \text{n}$, are perspective schemata from the sources to the reference model; and P_I is the inferred perspective schema from the sources to the global model. All elements of the perspective schemata are grouped in lists: $classList$, $relationList$, $keyList$, $caList$, and $mfList$. The three first lists hold 'require' declarations of, respectively, classes, relations, and keys

```
1: procedure GENERATENEWPERSPECTIVE(P_T, P_1, ..., P_n, P_I)
2:     for each CA A^G → A^RM in P_T.caList do
3:         infer_CAs(A^G → A^RM,{A^G → A^Si});
4:         add CAs A^G → A^Si to P_I.caList;
5:     end for
6:     for each MF m in P_T.mfList do
7:         infer_MFs(m,{m'_i});
8:         add MFs m'_i to P_I.mfList
9:     end for
10:    for each E in classList/relationList/keyList do
11:        create a require declaration to P_I;
12:        add it, appropriately, to P_I.classList/
13:        P_I.relationList/P_I.keyList
14:    end for
15: end procedure
```

Fig. 6. A pseudo-code to the creation of inferred perspective schemata

(including foreign keys). *caList* contains CA declarations, and $mfList$ has MF signatures.

This mechanism has been developed as part of a proof-of-concept prototype using a Prolog language. Beside the inference mechanism module, the prototype consists of another five modules, such as the *schema manager*, and the *ISCO translator*. The *schema manager* module is employed by the designer to manage the schemata (in language L_S) as well as the perspective schemata (in language L_{PS}). The *ISCO translator* performs the mapping between schemata written in L_S or L_{PS} languages to schemata defined in a language programming called Information Systems Construction (ISCO) language [36]. ISCO is based on a contextual constraint logic programming that allows the construction of information systems. It can define (object) relational schemata, represent data, and transparently access data from various heterogeneous sources in a uniform way, like a mediator system [37]. Thus, it is possible to access data from information sources using the perspective schema in ISCO. Furthermore, once the perspective schema from source schemata to the global schema has been inferred, as well as the new match functions have been implemented, it can be translated to ISCO language and so the data of the global schema can be queried. Details about the prototype can be found in [38].

6 Conclusions and Future Works

In this paper, the authors have presented a proposal to automatically connect a global schema to its sources by using an inference mechanism taking into account a reference model. The proposal approach makes clear the mappings that there are in a DW system, and uncouples them in order to make their maintenance easier. Besides, the relationship between the global schema and the

source schemata is made explicitly and declaratively through correspondence assertions.

The current approach is particularly useful in data integration systems that define a common or canonical schema, such as in DW systems and in FDBSs. An advantage of the proposed approach is that by using the reference model the designer does not need to have an in depth knowledge of all schemata involved in the DW system or in the FDBSs, since each schema (source or global) needs to be related only to the reference model. Thus, the effort to describe the mappings between schemata is reduced, since mappings between the DW and each sources (and between sources themselves) is automatically done by the inference mechanism. Besides, the DW designer can describe the global system without concerns about where the sources are or how they are stored. The inference mechanism also allows that changes changes in the actual source schemata, in the global schema, or in the mapping between schemata, which are common in the life cycle of any system, are completely transparent to the DW systems (or FDBSs).

Another advantage of our approach is that the process of data integration can be incrementally done in two ways:

1. View schemata can be created as a middle process to relate sections of data that have been integrated (those view schemata, in turn, are related to the reference model). Thus the data integration process can be divided in small parts, rather than being seen as a whole, making the integration task easier.
2. New source schemata can be added gradually, due to the inference mechanism.

A prototype Prolog-based has been developed to allow the description of schemata and perspective schemata in the proposed language as well as to infer new perspective schemata based on other ones. The matching functions can be implemented using Prolog itself or external functions. In addition, the prototype includes translators from the proposed language to the ISCO one. ISCO [36] allows access to heterogeneous data sources and to perform arbitrary computations. Thus, user-queries can be done, in a transparent way, to access the information sources, like occurs in mediator systems [37].

For future work, investigations will be made into how the perspective schemata can be used to automate the materialisation of the data in the DWs or in another repository of a data integration environment. Another important direction for future work is the development of a graphical user-friendly interface to declare the schemata in the proposed language, and thus, to hide some syntax details.

References

1. Imhoff, C., Galemmo, N., Geiger, J.G.: Mastering Data Warehouse Design - Relational and Dimensional Techniques. Wiley Publishing (2003)
2. Geiger, J.G.: Why build a data model? Information Management Magazine (June 2009)

3. Moody, D.L.: From enterprise models to dimensional models: A methodology for data warehouse and data mart design. In: Proc. of the Intl. Workshop on Design and Management of Data Warehouses. (2000)
4. Pequeno, V.M., Pires, J.C.G.M.: Using perspective schemata to model the ETL process. In: ICMIS 2009 :Intl. Conf. on Management Information Systems, France (June 2009)
5. Halevy, A.Y., Rajaraman, A., Ordille, J.J.: Data integration: The teenage years. In: VLDB. (2006) 9–16
6. Stumptner, M., Schrefl, M., Grossmann, G.: On the road to behavior-based integration. In: APCCM: First Asia-Pacific Conf. on Conceptual Modelling. (2004) 15–22
7. Louie, B., Mork, P., Martin-Sanchez, F., Halevy, A., Tarczy-Hornoch, P.: Data integration and genomic medicine. Journal of Biomedical Informatics **40** (2007) 5–13
8. Naidu, P.G., Palakal, M.J., Hartanto, S.: On-the-fly data integration models for biological databases. In: SAC'07: Proc. of the 2007 ACM symp. on Applied computing, USA, ACM (2007) 118–122
9. Yoakum-Stover, S., Malyuta, T.: Unified architecture for integrating intelligence data. In: DAMA: Europe Conf., UK (2008)
10. Vidal, V.M.P., Lóscio, B.F., Salgado, A.C.: Using correspondence assertions for specifying the semantics of XML-based mediators. In: Workshop on Information Integration on the Web. (2001) 3–11
11. Ives, Z.G., Knoblock, C.A., Minton, S., Jacob, M., Talukdar, P.P., Tuchinda, R., Ambite, J.L., Muslea, M., Gazen, C.: Interactive data integration through smart copy & paste. In: CIDR:4th Biennial Conference on Innovative Data Systems Research, www.crdrdb.org (2009)
12. Mccann, R., Doan, A., Varadarajan, V., Kramnik, E.: Building data integration systems via mass collaboration. In: WebDB: Intl. Workshop on the Web and Databases, USA (2003)
13. Berger, S., Schrefl, M.: From federated databases to a federated data warehouse system. In: HICSS'08: 41st Annual Hawaii Intl. Conf. on System Sciences, USA, IEEE Computer Society (2008) 394
14. Dori, D., Feldman, R., Sturm, A.: From conceptual models to schemata: An object-process-based data warehouse construction method. Inf. Syst. **33**(6) (2008) 567–593
15. Malinowski, E., Zimányi, E.: A conceptual model for temporal data warehouses and its transformation to the ER and the object-relational models. Data knowl. eng. **64**(1) (2008) 101–133
16. Pérez, J.M., Berlanga, R., Aramburu, M.J., Pedersen, T.B.: A relevance-extended multi-dimensional model for a data warehouse contextualized with documents. In: DOLAP'05: Proc. of the 8th ACM Intl. Workshop on Data Warehousing and OLAP, USA, ACM (2005) 19–28
17. Golfarelli, M., Maniezzo, V., Rizzi, S.: Materialization of fragmented views in multidimensional databases. Data Knowl. Eng. **49**(3) (2004) 325–351
18. Husemann, B., Lechtenborger, J., Vossen, G.: Conceptual data warehouse modeling. In: DMDW: Design and Management of Data Warehouses. (2000) 6
19. Rizzi, S.: Conceptual modeling solutions for the data warehouse. In Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications (2008) 208–227 copyright 2008 by Information Science Reference, formerly known as Idea Group Reference (an imprint of IGI Global).

20. Wrembel, R.: On a formal model of an object-oriented database with views supporting data materialisation. In: Proc. of the Conf. on Advances in Databases and Information Systems. (1999) 109–116
21. Franconi, E., Kamble, A.: A data warehouse conceptual data model. In: SSDBM'04: Proc. of the 16th Intl. Conf. on Scientific and Statistical Database Management, USA, IEEE Computer Society (2004) 435–436
22. Kamble, A.S.: A conceptual model for multidimensional data. In: APCCM'08: Proc. of the 15th on Asia-Pacific Conf. on Conceptual Modelling, Australia, Australian Computer Society, Inc. (2008) 29–38
23. Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R model for the multidimensional paradigm. In: Proc. of the Workshops on Data Warehousing and Data Mining. (1999) 105–116
24. Tryfona, N., Busborg, F., Christiansen, J.G.B.: starER: a conceptual model for data warehouse design. In: DOLAP'99: Proc. of the 2nd ACM Intl. Workshop on Data warehousing and OLAP, USA, ACM (1999) 3–8
25. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modelling in data warehouses. Data Knowl. Eng. **59**(3) (2005) 725–769
26. Nguyen, T.B., Tjoa, A.M., Wagner, R.: An object oriented multidimensional data model for OLAP. In: Web-Age Inf. Management. (2000) 69–82
27. Trujillo, J., Palomar, M., Gómez, J.: Applying object-oriented conceptual modeling techniques to the design of multidimensional databases and OLAP applications. In: WAIM'00: Proc. of the 1st Intl. Conf. on Web-Age Information Management, UK, Springer-Verlag (2000) 83–94
28. Skoutas, D., Simitsis, A.: Designing ETL processes using semantic web technologies. In: DOLAP'06: Proceedings of the 9th ACM international workshop on Data warehousing and OLAP, USA, ACM (2006) 67–74
29. Calvanese, D., Dragone, L., Nardi, D., Rosati, R., Trisolini, S.M.: Enterprise modeling and data warehousing in TELECOM ITALIA. Inf. Syst. **31**(1) (2006) 1–32
30. Codd, E.F.: A relational model of data for large shared data banks. In: Communications of the ACM. (1970) 377–387
31. Cattell, R.G., Barry, D., eds.: The Object Database Standard ODMG 3.0. Morgan Kaufmann Publishers (2000)
32. Pequeno, V.M., Pires, J.C.G.M.: A formal object-relational data warehouse model. Technical report, Universidade Nova de Lisboa (November 2007)
33. Pequeno, V.M., Pires, J.C.G.M.: Using perspective schemata to model the ETL process. Technical report, Universidade Nova de Lisboa (2009)
34. Elmasri, R., Navathe, S.B.: Fundamentals of database systems. 5th edn. Addison Wesley (2006)
35. Pequeno, V.M., Pires, J.C.G.M.: Reference model and perspective schemata inference for enterprise data integration. Technical report, Universidade Nova de Lisboa (2009)
36. Abreu, S., Nogueira, V.: Using a logic programming language with persistence and contexts. In: INAP'05: 16th Intl. Conf. on applications of declarative programming and knowledge management. Volume 4369 of Lecture Notes in Computer Science., Springer (2006) 38–47 (Revised Selected Papers).
37. Wiederhold, G.: Mediators in the architecture of future information systems. Computer **25**(3) (1992) 38–49
38. Pequeno, V.M., Abreu, S., Pires, J.C.G.M.: Using contextual logic programming language to acess data in warehousing systems. In: 14th Portuguese Conference on Artificial Intelligence, Portugal (October 2009) (to appear).