

# Schema Exchange: a Template-based Approach to Data and Metadata Translation

Paolo Papotti and Riccardo Torlone

Università Roma Tre  
{papotti,torlone}@dia.uniroma3.it

**Abstract.** We study the schema exchange problem, a natural extension of the data exchange problem to an intensional level. To this end, we first introduce the notion of schema template, a tool for the representation of a class of schemas sharing the same structure. We then define the schema exchange notion as the problem of (i) taking a schema that matches a source template, and (ii) generating a new schema for a target template, on the basis of a set of dependencies defined over the two templates. This framework allows the definition, once for all, of generic transformations that work for several schemas. A method for the generation of a “correct” solution of the schema exchange problem is proposed and a number of general results are given. We also show how it is possible to automatically generate a data exchange setting from a schema exchange solution. This allows the definition of queries to migrate data from a source database into the one obtained as a result of a schema exchange.

## 1 Introduction

In the last years, we have witnessed an increasing complexity of database applications, especially when several data sources need to be accessed, transformed and merged. There is a consequent growing need for advanced tools and flexible techniques supporting the management, the exchange, and the integration of different and heterogeneous sources of information.

In this trend, data exchange is a problem that has received recently great attention, both from a theoretical [12, 11] and a practical viewpoint [18]. In a data exchange scenario, given a set of correspondences between a source and a target schema, the goal is the automatic generation of queries able to transform data over the source into a format conforming to the target.

In this paper, we address the novel problem of *schema exchange*, which naturally extends the data exchange scenario to *sets* of similar schemas. To this aim, we first introduce the notion of *schema template*, which is used to represent a class of different database schemas sharing the same structure. Then, given a set of correspondences between the components of a source and a target template, the goal is the translation of any data source whose schema conforms to the source template into a format conforming to the target template. This framework allows the definition, once for all, of “generic” transformations that

works for different but similar schemas, such as the denormalization of a pair of relation tables based on a foreign key between them.

To tackle this problem, we introduce a formal notion of *solution* for a schema exchange setting and propose a technique for the automatic generation of solutions. This is done by representing constraints over templates and correspondences between them with a special class of first order formulas, and then using them to chase [2] the source schema. Also, we show how it is possible to automatically generate a data exchange setting from a schema exchange solution. This allows the definition of a set of queries to migrate data from a source database into the one obtained as a result of the schema exchange.

From a practical point of view, in our scenario the user can: (i) describe a collection of databases with common characteristics, by means of a source template  $T_1$ , (ii) impose the structure of a translation of the source through a target template  $T_2$ , (iii) define how to exchange information from source to target by means of simple correspondences, graphically represented by lines between  $T_1$  to  $T_2$ , and (iv) translate any data source over a schema matching with  $T_1$  into a format conforming to a schema matching with  $T_2$ .

We advocate that the relational model is adequate for implementing such approach. In particular we show how existing repositories for relational database management systems are already usable for such purpose. In fact, metadata can easily be modeled through relations and attributes and therefore can be queried using a standard relational query language, independently of whether or not it is associated to some data.

The approach presented in this paper is new and, to our knowledge, there is no work in the literature that is strictly related. In general, we can say that our contribution can be set in the framework of *metadata management*. Metadata can generally be thought of as information that describes, or supplements, the central data. Examples of metadata include schema definitions [14, 19], integrity constraints, annotations and comments on data [9, 6, 13], data provenance [8], and a large list of more specific information, like quality issues [16]. While the list is not exhaustive, it provides a representative sample of the kinds of metadata that have been addressed. Most of the above proposals focus on specific kinds of metadata and are not directly applicable to other kinds without major modifications. Bernstein set the various problems within a very general framework called *model management* [3, 4]. In [5] the authors show the value of this framework to approach several metadata related problems, with a significant reduction of programming effort. Our contribution goes in this direction: as in model management, schemas and mappings are treated as first class citizens. In particular, the schema exchange problem has some points in common with the *ModelGen* operator, but with crucial differences. The *ModelGen* operator realizes a *schema translation* from a source data model  $M_s$  to a target data model  $M_t$  of schemas conforming to  $M_s$ , for instance, a translation of a schema from the ER to an XML-based data model. Several approaches to this problem have been proposed in the last years [1, 7, 15, 17]. In this paper, we provide a novel contribution to this problem by studying a framework for schema translation with a clear and

precise semantics, that can be at the basis of an innovative tool supporting an important activity of model management.

The structure of the paper is as follows. In Section 2 we briefly set the basic definitions and recall some results of the data exchange problem. In Section 3, we introduce the notions of template and schema exchange and we show how they can be implemented with the relational database technology. In Section 4 we describes how templates and schemas are related and, in Section 5 we show how a data exchange problem can be obtained from a schema exchange setting. Finally, in Section 6, we draw some conclusions and sketch some future directions of research.

## 2 Preliminaries

### 2.1 Basics

A (*relational*) *schema*  $\mathbf{S}$  is composed by a set of *relations*  $R(A_1, \dots, A_n)$ , where  $R$  is the *name* of the relation and  $A_1, \dots, A_k$  are its *attributes*. Each attribute is associated with a set of values called the *domain* of the attribute. An instance of a relation  $R(A_1, \dots, A_n)$  is a set of tuples, each of which associates with each  $A_i$  a value taken from its domain. An instance  $I$  of a schema  $\mathbf{S}$  contains an instance of each relation in  $\mathbf{S}$ .

A *dependency* over a schema  $\mathbf{S}$  is a first order formula of the form:  $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \chi(\mathbf{x}))$  where  $\phi(\mathbf{x})$  and  $\chi(\mathbf{x})$  are formulas over  $\mathbf{S}$ , and  $\mathbf{x}$  are the free variables of the formula, ranging over the domains of the attributes occurring in  $\mathbf{S}$ .

As usual, we will focus on two special kind of dependencies: the tuple generating dependencies (tgd) and the equality generating dependencies (egd), as it is widely accepted that they include all of the naturally-occurring constraints on relational databases. A tgd has the form:  $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}(\psi(\mathbf{x}, \mathbf{y})))$  where  $\phi(\mathbf{x})$  and  $\psi(\mathbf{x}, \mathbf{y})$  are conjunction of atomic formulas, whereas an egd has the form:  $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow (x_1 = x_2))$  where  $\phi(\mathbf{x})$  is a conjunction of atomic formulas and  $x_1, x_2$  are variables in  $\mathbf{x}$ .

### 2.2 Data Exchange

In the relational-to-relational data exchange framework [11], a data exchange setting is described by  $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ , where: (i)  $\mathbf{S}$  is a source schema, (ii)  $\mathbf{T}$  is a target schema, (iii)  $\Sigma_{st}$  is a finite set of *s-t* (source-to-target) tgds  $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}(\chi(\mathbf{x}, \mathbf{y})))$  where  $\phi(\mathbf{x})$  is a conjunction of atomic formulas over  $\mathbf{S}$  and  $\chi(\mathbf{x}, \mathbf{y})$  is a conjunction of atomic formulas over  $\mathbf{T}$ , and (iv)  $\Sigma_t$  is a finite set of tgds or egds over  $\mathbf{T}$ . Given an instance  $I$  of  $\mathbf{S}$ , a solution for  $I$  under  $M$  is an instance  $J$  of  $\mathbf{T}$  such that  $(I, J)$  satisfies  $\Sigma_{st} \cup \Sigma_t$ . A solution may have distinct labeled nulls denoting unknown values.

In general, there are many possible solutions for  $I$  under  $M$ . A solution  $J$  is *universal* if there is a homomorphism from  $J$  to every other solution for  $I$  under  $M$ . A homomorphism from an instance  $I$  to an instance  $J$  is a function  $h$  from

constant values and nulls occurring in  $I$  to constant values and nulls occurring in  $J$  such that: (i) it is the identity on constants, and (ii) (with some abuse of notation)  $h(I) \subseteq J$ .

In [12] it was shown that a universal solution of  $I$  under  $M$  can be obtained by applying the chase procedure to  $I$  using  $\Sigma_{st} \cup \Sigma_t$ . This procedure takes as input an instance  $I$  and generates another instance by applying *chase steps* based on dependencies in  $\Sigma_{st} \cup \Sigma_t$ . There are two kinds of chase steps: (1) a tgd  $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}(\psi(\mathbf{x}, \mathbf{y})))$  can be applied to  $I$  if there is a homomorphism  $h$  from  $\phi(\mathbf{x})$  to  $I$ ; in this case, the result of its application is  $I \cup h'(\psi(\mathbf{x}, \mathbf{y}))$ , where  $h'$  is the extension of  $h$  to  $\mathbf{y}$  obtained by assigning fresh labeled nulls to the variables in  $\mathbf{y}$ ; (2) an egd  $\phi(x) \rightarrow (x_1 = x_2)$  can be applied to  $I$  if there is a homomorphism  $h$  from  $\phi(\mathbf{x})$  to  $I$  such that  $h(x_1) \neq h(x_2)$ ; in this case, the result of its application is the following: if one of  $h(x_1)$  and  $h(x_2)$  is a constant and the other is a variable then the variable is changed to the constant, otherwise the values are equated unless they are both constants, since in this case the process *fails*. The chase of  $I$  is obtained by applying all applicable chase steps exhaustively to  $I$ .

### 3 Schema Exchange Semantics

In this section we define the schema exchange problem as the application of the data exchange problem to *templates* of schemas.

#### 3.1 Schema templates

We fix a finite set  $\mathcal{C}$  of *construct names*. A *construct*  $C(p_0, p_1, \dots, p_k)$  has a name  $C$  in  $\mathcal{C}$  and a finite set  $p_1, \dots, p_k$  of distinct *properties*, each of which is associated with a set of values called the *domain* of the property. In principle, the set  $\mathcal{C}$  can contain construct names from different data models so that we can define transformations between schemes of different models. In this paper however, for sake of simplicity, we focus on schema exchange between schema templates of relational schemas; the approach can be extended to other types of templates, but challenging issues already arise in the relational case.

Therefore, we fix the following relational construct names and properties:

Construct Names	Properties (domain)
Relation (or R)	name (strings)
Attribute (or A)	name (strings), nullable (booleans), in (strings)
AttributeKey (or AK)	name (strings), in (strings)
AttributeFKey (or AFK)	name (strings), in (strings), refer (strings)

Note that the **Relation** construct is associated only to the **name** property, whose domain is a set of strings. The same domain is also associated with the property **in** of the constructs **Attribute**, **AttributeKey** and **AttributeFKey**, and the property **refer** of the construct **AttributeFKey**: these properties are used to specify references between constructs. Clearly, other properties can be considered.

Basically, a template is a set of constructs with a set of dependencies associated with them, which are used to specify constraints over single constructs and semantic associations between different constructs.

**Definition 1 (Template).** A (schema) template is a pair  $(\mathbf{C}, \Sigma_{\mathbf{C}})$ , where  $\mathbf{C}$  is a finite collection of constructs and  $\Sigma_{\mathbf{C}}$  is a set of dependencies over  $\mathbf{C}$ .

*Example 1.* An example of a template  $\mathcal{T} = (\mathbf{C}, \Sigma_{\mathbf{C}})$  contains the following set of constructs:

$$\mathbf{C} = \{ \text{Relation}(\text{name}), \text{AttributeKey}(\text{name}, \text{in}), \text{Attribute}(\text{name}, \text{nullable}, \text{in}), \\ \text{AttributeFKKey}(\text{name}, \text{in}, \text{refer}) \}$$

and the dependencies:

$$\Sigma_{\mathbf{C}} = \{ d_1 = \text{AttributeKey}(n_K, n_R) \rightarrow \text{Relation}(n_R), \\ d_2 = \text{Attribute}(n_A, u, n_R) \rightarrow \text{Relation}(n_R), \\ d_3 = \text{AttributeFKKey}(n_F, n_R, n'_R) \rightarrow \text{Relation}(n_R), \text{Relation}(n'_R), \\ d_4 = \text{Attribute}(n_A, u, n_R) \rightarrow (u = \text{true}) \}$$

The tgds  $d_1$  and  $d_2$  state the membership of keys and attributes to relations, respectively. The dependency  $d_3$  states the membership of a foreign key to a relation and its reference to another relation. Finally, the egd  $d_4$  states that we are considering only relations with attributes that allow null values.

For simplicity, in the following we will omit the membership dependencies between constructs (like  $d_1$ ,  $d_2$  and  $d_3$  in Example 1), assuming that they belong to  $\Sigma_{\mathbf{C}}$ .

Let us now introduce the notion of e-schemas. Basically, an e-schema corresponds to the *encoding* of a (relational) schema and is obtained by instantiating a template.

**Definition 2 (E-schemas).** An e-schema component  $S$  over a construct  $C$  is a function that associates with each property  $p_1, \dots, p_k$  of  $C$  a value  $a_i$  taken from its domain. A e-schema  $S$  over a template  $(\mathbf{C}, \Sigma_{\mathbf{C}})$  is a finite set of e-schema components over constructs in  $\mathbf{C}$  that satisfy  $\Sigma_{\mathbf{C}}$ .

*Example 2.* A valid e-schema for the template of Example 1 is the following:

Relation		AttributeKey	
name		name	in
EMP		EmpName	EMP
DEPT		DeptNo	DEPT
Attribute		AttributeFKKey	
name	nullable	name	in
Salary	true	name	refer
Building	false	Dept	EMP DEPT

It is easy to see that this e-schema represents a relational table EMP with EmpName as key, Salary as attribute and Dept as foreign key, and a relational table DEPT with DeptNo as key and Building as attribute.

Note that e-schemas in Example 2 remind the common way commercial databases use to store metadata in catalogs. We can therefore easily verify whether a relational schema stored in a DBMS matches a given template definition: this can be done by querying the catalog of the system and checking the satisfaction of the dependencies.

In the following, an e-schema component over a construct  $C(p_1, \dots, p_k)$  will be called a *relation* component if  $C = \text{Relation}$ , an *attribute* component if  $C = \text{Attribute}$ , a *key* component if  $C = \text{AttributeKey}$ , a *foreign key* component if  $C = \text{AttributeFKey}$ . Moreover, we will denote an e-schema component over a construct  $C(p_1, \dots, p_k)$  by  $C(p_1 : a_i, \dots, p_k : a_k)$ . Alternatively, we will use, for each construct, a tabular notation with a column for each property.

### 3.2 Schema exchange

Given a source template  $\mathcal{T}_1 = (\mathbf{C}_1, \Sigma_{\mathbf{C}_1})$ , a target template  $\mathcal{T}_2 = (\mathbf{C}_2, \Sigma_{\mathbf{C}_2})$ , and a set  $\Sigma_{\mathbf{C}_1\mathbf{C}_2}$  of *source-to-target dependencies*, that is, a set of tgds on  $\mathbf{C}_1 \cup \mathbf{C}_2$ , we denote a *schema exchange setting* by the triple  $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$ .

**Definition 3 (Schema exchange).** *Given a schema exchange setting  $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$  and a source e-schema  $\mathcal{S}_1$  over  $(\mathbf{C}_1, \Sigma_{\mathbf{C}_1})$ , the schema exchange problem consists in finding a finite target e-schema  $\mathcal{S}_2$  over  $(\mathbf{C}_2, \Sigma_{\mathbf{C}_2})$  such that  $\mathcal{S}_1 \cup \mathcal{S}_2$  satisfies  $\Sigma_{\mathbf{C}_1\mathbf{C}_2}$ . In this case  $\mathcal{S}_2$  is called a solution for  $\mathcal{S}_1$  or, simply a solution.*

*Example 3.* Consider a schema exchange problem in which the source template  $\mathcal{T}_1 = (\mathbf{C}_1, \Sigma_{\mathbf{C}_1})$  and the target template  $\mathcal{T}_2 = (\mathbf{C}_2, \Sigma_{\mathbf{C}_2})$  are the following:

$$\mathbf{C}_1 = \{ \text{Relation}(\text{name}), \text{AttributeKey}(\text{name}, \text{in}), \text{Attribute}(\text{name}, \text{in}) \}$$

$$\mathbf{C}_2 = \{ \text{Relation}(\text{name}), \text{AttributeKey}(\text{name}, \text{in}), \text{Attribute}(\text{name}, \text{in}), \\ \text{AttributeFKey}(\text{name}, \text{in}, \text{refer}) \}$$

with the corresponding membership constraints in  $\Sigma_{\mathbf{C}_1}$  and in  $\Sigma_{\mathbf{C}_2}$ .

Assume now that we would like to split relations over  $\mathcal{T}_1$  into a pair of relations over  $\mathcal{T}_2$  related by a foreign key. This scenario is graphically shown (informally) in Figure 1 and is precisely captured by the following set of tgds  $\Sigma_{\mathbf{C}_1, \mathbf{C}_2}$ :

$$\Sigma_{\mathbf{C}_1, \mathbf{C}_2} = \{ \text{Relation}(n_R), \text{AttributeKey}(n_K, n_R), \text{Attribute}(n_A, n_R) \rightarrow \\ \text{Relation}(n_R), \text{AttributeKey}(n_K, n_R), \text{AttributeFKey}(n_F, n_R, n'_R), \\ \text{Relation}(n'_R), \text{AttributeKey}(n_F, n'_R), \text{Attribute}(n_A, n'_R) \}$$

Consider now the following e-schema valid for  $\mathcal{T}_1$ :

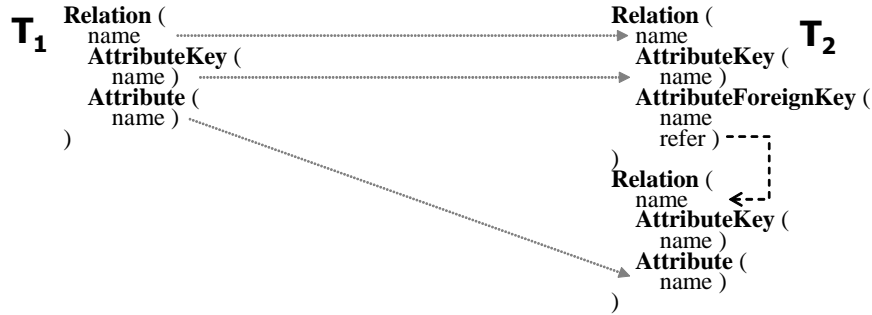


Fig. 1. Schema exchange scenario for Example 3

Relation
name
EMP

AttributeKey
name in
EmpName EMP

Attribute
name in
DeptName EMP
Floor EMP

This e-schema has one relation called EMP with EmpName as key and two attributes: DeptName and Floor. A possible solution  $S'_1$  for this setting is:

Relation
name
EMP
$N_0$
$N_2$

AttributeKey
name in
EmpName EMP
$N_1$ $N_0$
$N_3$ $N_2$

Attribute
name in
DeptName $N_0$
Floor $N_2$

AttributeFKKey
name in refer
$N_1$ EMP $N_0$
$N_3$ EMP $N_2$

where  $N_0, \dots, N_3$  are labelled nulls. This solution contains three relations: EMP,  $N_0$  and  $N_2$ . Relation EMP has EmpName as key and  $N_1, N_3$  as foreign keys for  $N_0$  and  $N_2$  respectively. Relation  $N_0$  has  $N_1$  as key and DeptName as attribute. Finally, relation  $N_2$  has  $N_3$  as key and Floor as attribute. There are several null values because the dependencies in  $\Sigma_{C_1, C_2}$  do not allow the complete definition of the target e-schema.

Consider now another solution  $S'_2$ :

Relation
name
EMP
$N_0$

AttributeKey
name in
EmpName EMP
$N_1$ $N_0$

Attribute
name in
DeptName $N_0$
Floor $N_0$

AttributeFKKey
name in refer
$N_1$ EMP $N_0$

with  $N_0$  and  $N_1$  as labelled nulls. This solution contains two relations named EMP and  $N_0$ . Relation EMP has EmpName as key and  $N_1$  as foreign key, relation  $N_0$  has  $N_1$  as key and DeptName and Floor as attributes.

From Example 3 two issues arise: which solution to choose and how to generate it. Solution  $S'_2$  in the example seems to be less general than  $S'_1$ . This is

captured precisely by the notion of homomorphisms. In fact, it is easy to see that, while there is a homomorphism from  $S'_1$  to  $S'_2$ , there is no homomorphism from  $S'_2$  to  $S'_1$ . It follows that  $S'_2$  contains “extra” information whereas  $S'_1$  is a more general solution. As in data exchange [11, 12], we argue that the “correct” solution is the most general one, in the sense above. This solution is called universal.

**Definition 4 (Universal solution).** *A solution  $S$  of the schema exchange problem is universal if there exists a homomorphism from  $S$  to all other solutions.*

The following result follows from analogous results of the data exchange problem.

**Theorem 1.** *Let  $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$  be a data exchange setting and  $S_1$  be an e-schema over  $\mathcal{T}_1$ . The chase procedure over  $S_1$  using  $\Sigma_{\mathbf{C}_1\mathbf{C}_2} \cup \Sigma_{\mathbf{C}_2}$  terminates and generates a universal solution.*

## 4 Decoding and encoding of relational schemas

In this section we describe how the notion of e-schema introduced in Section 3 can be converted into a “standard” relational schema, and vice versa.

### 4.1 Relational decoding

Basically, the transformation of an e-schema in a relational schema requires the definition of formulas that describe the semantics of the various components of an e-schema, according to the intended meaning of corresponding constructs.

Let  $S$  be an e-schema over a template  $\mathcal{T} = (\mathbf{C}, \Sigma_{\mathbf{C}})$ . The *relational decoding* of  $S$ , denoted by  $\mathbf{R}\text{-DEC}(S)$ , is a pair  $(\mathbf{S}, \Sigma_{\mathbf{S}})$  where:

- $\mathbf{S}$  contains a set of objects  $R(A_1, \dots, A_n)$  for each Relation component  $m \in S$  such that:  $m(\text{name}) = R$  and
  - for each Attribute component  $m'_i \in m$ :  $A_i = m'_i(\text{name})$ ,
  - for each AttributeKey component  $m''_k \in m$ :  $A_k = m''_k(\text{name})$ ,
  - for each AttributeFKey component  $m'''_f \in m$ :  $A_f = m'''_f(\text{name})$ ;
- $\Sigma_{\mathbf{S}}$  contains an egd over  $R(A_1, \dots, A_n) \in \mathbf{R}$  of the form:

$$R(x_1, x_2, \dots, x_n), R(x_1, x'_2, \dots, x'_n) \rightarrow (x_2 = x'_2, \dots, x_n = x'_n)$$

for each AttributeKey component  $m' \in m$  for the Relation component  $m \in S$  such that  $R = m(\text{name})$  and  $x_1 = m'(\text{name})$ ;

- $\Sigma_{\mathbf{S}}$  contains a set of tgds over a pair  $R(A_1, \dots, A_k, \dots, A_m), R'(A_k, A'_1, \dots, A'_n)$  in  $\mathbf{S}$  of the form:

$$R(x_1, \dots, x_k, \dots, x_m) \rightarrow R'(x_k, x'_1, \dots, x'_n)$$

for each AttributeFKey component  $m \in m'$  for the Relation  $m' \in \mathbf{S}$  such that  $R = m'(\text{name})$  and: (i)  $x_k = m(\text{name})$  in  $R$ , (ii) Relation component  $m''$  such that  $R' = m''(\text{name})$ , (iii) for the AttributeKey component  $m'''$  in  $m''$ :  $m(\text{refer}) = m'''(\text{name})$ , and (iv) in  $R'$ :  $x_k = m'''(\text{name})$ .

*Example 4.* Let us consider the e-schema  $\mathbf{S}$  of Example 2 reported below:

Relation	AttributeKey
name	name in
EMP	EmpName EMP
DEPT	DeptNo DEPT

Attribute	AttributeFKey
name nullable in	name in refer
Salary true EMP	Dept EMP DEPT
Building false DEPT	

The relational representation of  $\mathbf{S}$  is:  $\mathbf{R}\text{-DEC}(\mathbf{S}) = (\mathbf{S}, \Sigma_{\mathbf{S}})$  where:

$$\mathbf{R} = \{\text{EMP}(\text{EmpName}, \text{Salary}, \text{Dept}), \text{DEPT}(\text{DeptNo}, \text{Building})\}$$

$$\Sigma_{\mathbf{R}} = \{\text{EMP}(x_1, x_2, x_3), \text{EMP}(x_1, x'_2, x'_3) \rightarrow (x_2 = x'_2, x_3 = x'_3), \\ \text{DEPT}(x_1, x_2), \text{DEPT}(x_1, x'_2) \rightarrow (x_2 = x'_2), \\ \text{EMP}(x_1, x_2, x_3) \rightarrow \text{DEPT}(x_3, x'_2)\}$$

In the same line, a procedure for the *encoding* of a relational schema, that is for the transformation of a relational schema  $(\mathbf{S}, \Sigma_{\mathbf{S}})$  into an e-schema  $\mathbf{S}$ , can also be defined.

## 4.2 Relational encoding

Let  $\mathbf{S}$  be a relational schema with a set of schema dependencies  $\Sigma_{\mathbf{S}}$ . The *encoding* of  $\mathbf{S}$ , denoted by  $\mathbf{R}\text{-ENC}(\mathbf{S}, \Sigma_{\mathbf{S}})$ , is an e-schema  $\mathbf{S}$  where:

- for each relation  $R(A_1, \dots, A_n)$  in  $\mathbf{S}$ ,  $\mathbf{S}$  has a Relation component  $m$  such that  $m(\text{name}) = R$  and
  - for each attribute  $A_i \in R$  there is an Attribute component  $m_i \in m$  with properties:
    - \*  $m_i(\text{name}) = A_i$ ,
    - \*  $m_i(\text{nullable}) = \text{true}$  if  $A_i$  is nullable,
    - \*  $m_i(\text{in}) = m(\text{name})$ ;
- for each egd in  $\Sigma_{\mathbf{S}}$  of the form:

$$R(x_1, x_2, \dots, x_n), R(x_1, x'_2, \dots, x'_n) \rightarrow (x_2 = x'_2, \dots, x_n = x'_n)$$

such that there is an Relation  $m$  in  $\mathbf{S}$  with  $m(\text{name}) = R$ :

- for each attribute  $x_i \notin x_2, \dots, x_n$  (attributes in the right hand side of the egd), the **Attribute** component  $m' \in m$  with  $m'(\mathbf{name}) = x_i$  changes its metaprimitive name from **Attribute** to **AttributeKey** and the nullable property is deleted;
- for each **tgd** in  $\Sigma_{\mathbf{S}}$  of the form:

$$R(x_1, \dots, x_k, \dots, x_m) \rightarrow R'(x_k, x'_1, \dots, x'_n)$$

such that there is an **Relation**  $m$  in  $\mathbf{S}$  with  $m(\mathbf{name}) = R$  and an **Relation**  $m'$  with  $m'(\mathbf{name}) = R'$ :

- for each attribute  $x_k$  (both in the left and right hand side of the **tgd**) the **Attribute** component  $m_i \in R$  with  $m_i(\mathbf{name}) = x_k$  changes its name from **Attribute** to **AttributeFKey** with  $m_i(\mathbf{refer}) = m'_i(\mathbf{name})$  where  $m'_i \in R'$  and  $m'_i(\mathbf{name}) = x_k$ .

## 5 From Schema to Data Exchange

In this section we propose a transformation process that generates a data exchange from a given schema exchange setting.

### 5.1 Metaroutes and Value Correspondences

Before discussing the transformation process, two preliminary notions are needed. First of all, in order to convert the schema exchange setting into a *data* exchange setting, we need to keep track of the correspondences between the source schema and the solution of the schema exchange problem. This can be seen as an application of the data provenance problem to schema exchange. To this end, by extending to our context a notion introduced in [10], we make use of *metaroutes* to describe the relationships between source and target metadata.

**Definition 5.** *Given an e-schema  $\mathbf{S}$  and a set of dependencies  $\Sigma$ , a metaroute for  $\mathbf{S}$  is an expression of the form*

$$I_0 \rightarrow_{\sigma_1, h_1} I_1 \dots I_{n-1} \rightarrow_{\sigma_n, h_n} I_n$$

where  $I_0 \subseteq \mathbf{S}$  and, for each  $I_{i-1} \rightarrow_{\sigma_i, h_i} I_i$  ( $1 \leq i \leq n$ ), it is the case that  $I_i$  is the result of the application of a chase step on  $I_{i-1}$  based on the dependency  $\sigma_i \in \Sigma$  and the homomorphism  $h_i$ .

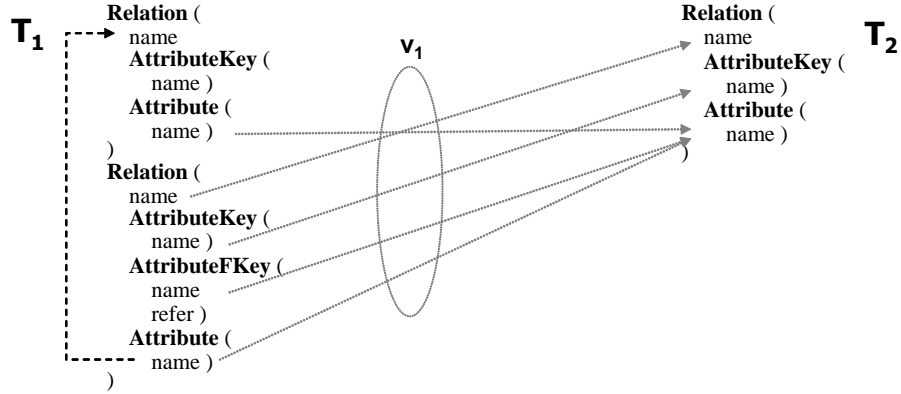
Note that since a reduced number of elements are involved in schema exchange, we can store all the metaroutes and we do not need to compute them partially and incrementally as in [10]. Metaroutes and homomorphisms are then used to derive *value correspondences* between source and target schemas.

**Definition 6.** *A value correspondence over two schemes  $\mathbf{S}$  and  $\mathbf{S}'$  is a triple  $v = (t \in R, t' \in R', t.A_i = t'.A_j)$  where  $R \in \mathbf{S}$ ,  $R' \in \mathbf{S}'$ , and  $A_i = A_j$  is a set of equalities over the attributes of  $R$  and  $R'$  respectively.*

## 5.2 The S-D transformation process

Given a relational database over a schema  $\mathbf{S}_1$  and schema exchange setting  $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1, \mathbf{C}_2})$  such that the encoding  $\mathbf{S}_1$  of  $\mathbf{S}_1$  is an instance of  $\mathcal{T}_1$ , we aim at generating a target database over a schema  $\mathbf{S}_2$  such that the encoding  $\mathbf{S}_2$  of  $\mathbf{S}_2$  is a universal solution for  $\mathbf{S}_1$ . We call such generation process *S-D transformation* and it can be summarized as follows.

1.  $\mathbf{S}_1$  is encoded into an e-schema  $\mathbf{S}_1$ ;
2. the chase procedure is applied to  $\mathbf{S}_1$  using  $\Sigma_{\mathbf{C}_1, \mathbf{C}_2}$  and metaroutes are generated during the execution of the procedure: each chase step based on the dependency  $\sigma_i \in \Sigma$  and the homomorphism  $h_i$  adds an element  $I_{i-1} \rightarrow_{\sigma_i, h_i} I_i$  to the metaroute;
3. the result  $\mathbf{S}_2$  of the chase procedure is decoded into a schema  $\mathbf{S}_2$ ;
4. for each attribute  $A$  occurring in  $\mathbf{S}_2$ : (i) we select the metaroute  $I_0 \rightarrow_{\sigma_1, h_1} I_1 \dots I_{n-1} \rightarrow_{\sigma_n, h_n} I_n$  such that  $A$  occur in  $I_n$ , and (ii)  $A$  is annotated in  $\mathbf{S}_1$  and  $\mathbf{S}_2$  with  $h^{-1}(A)$ , where  $h = h_1 \circ \dots \circ h_n$ ;
5. the annotations of the attributes in  $\mathbf{S}_1$  and  $\mathbf{S}_2$  are used to derive value correspondences between them;
6. a data exchange setting is generated from  $\mathbf{S}_1$  and  $\mathbf{S}_2$  using the generated value correspondences, on the basis of the method presented in [18].



**Fig. 2.** Schema exchange scenario for Example 5.

*Example 5.* Let us consider the schema exchange setting described graphically in Figure 2 and represented by the following set of tgds  $\Sigma_{\mathbf{C}_1, \mathbf{C}_2}$ :

$$\{ v_1 = \text{Relation}(n_r), \text{AttributeKey}(n_k, n_r), \text{Attribute}(n_a, n_r), \text{Relation}(n'_r), \text{AttributeKey}(n'_k, n'_r), \text{Attribute}(n'_a, n'_r), \text{AttributeFKey}(n_f, n'_r, n_r) \rightarrow \text{Relation}(n'_r), \text{AttributeKey}(n'_k, n'_r), \text{Attribute}(n'_a, n'_r), \text{Attribute}(n_f, n'_r), \text{Attribute}(n_a, n'_r) \}$$

Intuitively, the only constraint occurring in  $\Sigma_{C_1, C_2}$  specifies that the target is obtained by joining two source relations according to a foreign key defined between them. Now consider the following source schema:

$$\mathbf{S} = \{\text{DEPT}(\text{id}, \text{dname}), \text{EMP}(\text{id}, \text{ename}, \text{dep})\}$$

$$\Sigma_{\mathbf{S}} = \{\text{DEPT}(x_1, x_2), \text{DEPT}(x_1, x'_2) \rightarrow (x_2 = x'_2), \\ \text{EMP}(x_1, x_2, x_3), \text{EMP}(x_1, x'_2, x'_3) \rightarrow (x_2 = x'_2, x_3 = x'_3), \\ \text{EMP}(x_1, x_2, x_3) \rightarrow \text{DEPT}(x_3, x'_1)\}$$

The encoding of  $\mathbf{S}$  is the e-schema  $\mathbf{S}$  that follows:

Relation	AttributeKey	Attribute	AttributeFKey																
$s_1$ <table border="1"><tr><td>name</td></tr><tr><td>DEPT</td></tr></table>	name	DEPT	$s_3$ <table border="1"><tr><td>name</td><td>in</td></tr><tr><td>id</td><td>DEPT</td></tr></table>	name	in	id	DEPT	$s_5$ <table border="1"><tr><td>name</td><td>in</td></tr><tr><td>dname</td><td>DEPT</td></tr></table>	name	in	dname	DEPT	$s_7$ <table border="1"><tr><td>name</td><td>in</td><td>refer</td></tr><tr><td>dep</td><td>EMP</td><td>DEPT</td></tr></table>	name	in	refer	dep	EMP	DEPT
name																			
DEPT																			
name	in																		
id	DEPT																		
name	in																		
dname	DEPT																		
name	in	refer																	
dep	EMP	DEPT																	
$s_2$ <table border="1"><tr><td>EMP</td></tr></table>	EMP	$s_4$ <table border="1"><tr><td>id</td><td>EMP</td></tr></table>	id	EMP	$s_6$ <table border="1"><tr><td>ename</td><td>EMP</td></tr></table>	ename	EMP												
EMP																			
id	EMP																		
ename	EMP																		

Let  $\{s_1, \dots, s_7\}$  be the e-components of  $\mathbf{S}$ . The application of the chase based on the given tgd produces the set of e-schema components  $\{t_1, \dots, t_5\}$ :

Relation	AttributeKey	Attribute										
$t_1$ <table border="1"><tr><td>name</td></tr><tr><td>EMP</td></tr></table>	name	EMP	$t_2$ <table border="1"><tr><td>name</td><td>in</td></tr><tr><td>id</td><td>EMP</td></tr></table>	name	in	id	EMP	$t_3$ <table border="1"><tr><td>name</td><td>in</td></tr><tr><td>ename</td><td>EMP</td></tr></table>	name	in	ename	EMP
name												
EMP												
name	in											
id	EMP											
name	in											
ename	EMP											
		$t_4$ <table border="1"><tr><td>dep</td><td>EMP</td></tr></table>	dep	EMP								
dep	EMP											
		$t_5$ <table border="1"><tr><td>dname</td><td>EMP</td></tr></table>	dname	EMP								
dname	EMP											

The metaroute generated by this chase step is:  $\{s_1, \dots, s_7\} \rightarrow_{v_1, h_1} \{t_1, \dots, t_5\}$ , where  $h_1$  is the homomorphism:

$$\{n_r \mapsto \text{DEPT}, n_k \mapsto \text{id}, n_a \mapsto \text{dname}, n'_r \mapsto \text{EMP}, n'_k \mapsto \text{id}, n'_a \mapsto \text{ename}, n_f \mapsto \text{dep}\}$$

The chase ends successfully and produce an e-schema  $\mathbf{S}'$  whose decoding is the schema  $(\mathbf{S}', \Sigma_{\mathbf{S}'})$  where:

$$\mathbf{S}' = \{\text{EMP}(\text{id}, \text{ename}, \text{dep}, \text{dname})\}$$

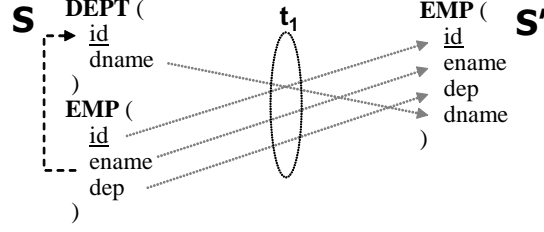
$$\Sigma_{\mathbf{S}'} = \{\text{DEPT}(x_1, x_2, x_3, x_4), \text{DEPT}(x_1, x'_2, x'_3, x'_4) \rightarrow (x_2 = x'_2, x_3 = x'_3, x_4 = x'_4)\}$$

Now, on the basis of the above metaroute, source and target schema can be annotated as follows:

$$\mathbf{S} = \{\text{DEPT}(\text{id}[n_k], \text{dname}[n_a]), \text{EMP}(\text{id}[n'_k], \text{ename}[n'_a], \text{dep}[n_f])\} \\ \mathbf{S}' = \{\text{EMP}(\text{id}[n'_k], \text{ename}[n'_a], \text{dep}[n_f], \text{dname}[n_a])\}$$

The value correspondences between  $\mathbf{S}$  and  $\mathbf{S}'$  easily follow:

$$vc_1 = (d \in \mathbf{S}.\text{DEPT}, e \in \mathbf{S}'.\text{EMP}, d.\text{dname} = e.\text{dname}) \\ vc_2 = (e \in \mathbf{S}.\text{EMP}, e' \in \mathbf{S}'.\text{EMP}, e.\text{id} = e'.\text{id}, e.\text{ename} = e'.\text{ename}, e.\text{dep} = e'.\text{dep})$$



**Fig. 3.** Data exchange scenario for Example 5.

We then obtain the data mapping scenario reported graphically in Figure 3. In the spirit of [18] we are now able to automatically generate a schema exchange setting. Given the source schema  $\mathbf{S}$ , the target schema  $\mathbf{S}'$  with its constraints, and the value correspondences we obtain the following tgdt:

$$t_1 = \mathbf{S}.\text{EMP}(ss, en, d), \mathbf{S}.\text{DEPT}(d, dn) \rightarrow \mathbf{S}'.\text{EMP}(ss, en, d, dn)$$

A number of general results can be shown. First, the fact that the output of the S-D process is a “correct” result, that is, the solution of the data exchange problem reflects the semantics of the schema exchange problem given as input. In order to introduce the concept of correctness in this context, a preliminary notion is needed.

Given a tgdt  $t$ , an *encoding* of  $t$  is the tgdt obtained by applying the encoding procedure defined in Section 4 considering the atoms of the formula as they were relational schemas and using the dependencies in  $\Sigma_{\mathbf{S}}$  for the left side of  $t$  and the dependencies  $\Sigma_{\mathbf{S}'}$  for the right side of  $t$ . Note that the tgdt we obtain is defined on templates.

For instance, given the tgdt  $t_1$  of the example above, the encoding of  $t_1$  is the following tgdt on templates:

$$v_2 = \text{Relation}(\text{EMP}), \text{AttributeKey}(ss, \text{EMP}), \text{Attribute}(en, \text{EMP}), \\ \text{AttributeFKKey}(d, \text{EMP}, \text{DEPT}), \text{Relation}(\text{DEPT}), \text{AttributeKey}(d, \text{DEPT}), \\ \text{Attribute}(dn, \text{DEPT}) \rightarrow \text{Relation}(\text{EMP}), \text{AttributeKey}(ss, \text{EMP}), \\ \text{Attribute}(en, \text{EMP}), \text{Attribute}(d, \text{EMP}), \text{Attribute}(dn, \text{EMP})$$

This tgdt  $v_2$  is different from the original tgdt  $v_1$  for the schema exchange scenario described in Example 5. However, it can be verified that they generate the same output  $\mathbf{S}'$  on the given input  $\mathbf{S}$ . This exactly captures the fact that the data exchange problem obtained as output fulfils the semantics of the schema exchange problem given as input.

This intuition is captured by the following correctness result.

**Theorem 2.** *Let  $(\mathbf{S}, \mathbf{S}', \Sigma_{\mathbf{S}\mathbf{S}'})$  be the output of the S-D transformation process when  $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$  and  $\mathbf{S}$  are given as input and let  $\overline{\Sigma}$  be the set of s-t tgds obtained by encoding the tgds in  $\Sigma_{\mathbf{S}\mathbf{S}'}$ . The e-schema  $\mathbf{S}'$  is a universal solution of the schema exchange setting  $(\mathcal{T}_1, \mathcal{T}_2, \overline{\Sigma})$ .*

The following completeness result can also be shown.

**Theorem 3.** *Any data exchange setting can be obtained from the S-D transformation process over some schema exchange setting.*

## 6 Conclusion and Future work

We have introduced the schema exchange problem, a generalization of data exchange. This problem consists of taking a schema that matches a source template, and generating a new schema for a target template, on the basis of a set of dependencies defined over the two templates. To tackle this problem, we have presented a method for the generation of a “correct” solution of the problem and a process aimed at automatically generating a data exchange setting from a schema exchange solution.

We believe that several interesting directions of research can be pursued within this framework. We just sketch some of them.

- *Metaquerying.* A template is actually a schema and it can therefore be queried. A query over a template is indeed a *meta* query since it operates over meta-data. There are a number of meta-queries that are meaningful. For instance, we can retrieve with a query over a template the pairs of relations that can be joined, being related by a foreign key. Also, we can verify whether there is a join path between two relations.
- *Special class of solutions.* Given a schema exchange problem, can we verify whether all the solutions of the problem satisfy some relevant property? For instance, we would like to obtain only relations that are acyclic or satisfy some normal form. We are also investigating under which conditions a schema exchange problem generates a data exchange setting with certain properties, e.g., the fact that the dependencies belong to some relevant class.
- *Combining data and metadata.* The framework we have presented can be extended to support mappings and constraints involving data and metadata at the same time. This scenario also allows the user to specify the transformation of metadata into data and vice versa. For instance, we could move the name of a relational attribute into a tuple of a relation.

## References

1. P. Atzeni, P. Cappellari, P. A. Bernstein. Model-independent schema and data translation. *EDBT*, pages 368–385, 2006.
2. C. Beeri, M. Y. Vardi. A Proof Procedure for Data Dependencies. *J. ACM*, 31(4):718–741, 1984.
3. P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *First Conf. on Innovative Database Research (CIDR), Asilomar*, pages 209–220, 2003.
4. P. A. Bernstein, A. Y. Levy, and R. A. Pottinger. A Vision for Management of Complex Models. *SIGMOD Record*, 29(4):55–63, December 2000.

5. P. A. Bernstein and E. Rahm. Data Warehouse Scenarios for Model Management. In *19th International Conference on Conceptual Modeling (ER), USA*, pages 1–15, 2000.
6. D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya. An Annotation Management System for Relational Databases. *VLDB*, 900911, 2004.
7. S. Bowers, L. M. L. Delcambre. The uni-level description: A uniform framework for representing information in multiple data models. *ER*, pages 45–58, 2003.
8. P. Buneman, S. Khanna, and W. C. Tan. Why and Where: A Characterization of Data Provenance. *ICDT*, 2001.
9. P. Buneman, S. Khanna, and W. C. Tan. On Propagation of Deletion and Annotations Through Views. *PODS*, 2002.
10. L. Chiticariu, W. C. Tan. Debugging Schema Mappings with Routes. *VLDB*, 79–90, 2006.
11. R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.
12. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and Query Answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
13. F. Geerts, A. Kementsietsidis, and D. Milano. MONDRIAN: Annotating and querying databases through colors and blocks. *ICDE*, 2006.
14. L. V. S. Lakshmanan, F. Sadri, and S. N. Subramanian. SchemaSQL: An extension to SQL for multidatabase interoperability. *ACM Trans. Database Syst.*, 26(4):476–519, 2001.
15. P. McBrien and A. Poulouvasilis. Data Integration by Bi-Directional Schema Transformation Rules. *ICDE*, pages 227–238, 2003.
16. G. Mihaila, L. Raschid, and M.-E. Vidal. Querying “quality of data” metadata. *IEEE META-DATA*, 1999.
17. P. Papotti and R. Torlone. Heterogeneous Data Translation through XML Conversion. *J. Web Eng.*, 4(3):189–204, 2005.
18. L. Popa, Y. Velegarakis, R. J. Miller, M. A. Hernández, R. Fagin. Translating Web Data. *VLDB*, 598–609, 2002.
19. C. M. Wyss and E. Robertson. Relational Interoperability. *TODS*, 30:2, 2005.