ROMA TRE

Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 84 – 00146 Roma, Italy.

Schema Exchange: Generic Mappings for Transforming Data and Metadata

Paolo Papotti, Riccardo Torlone

RT-DIA-125-2008 **June 2008**

Dipartimento di Informatica e Automazione
Università di Roma Tre
00146 Roma – Italy

http://www.dia.uniroma3.it

ABSTRACT

In this paper we present and study the problem of schema exchange, a natural extension of the data exchange problem. 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 mapping between the two templates defined by means of FO dependencies. This framework allows the definition, once for all, of generic transformations that can be applied to 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 generate automatically, from a schema exchange solution, a data exchange setting that reflects the semantics of the mappings between the original templates. This allows the definition of queries to migrate data from a source database into the one obtained as a result of a schema exchange.

Contents

1 Introduction **4**

2 Motivating Examples **5**

3 Background **7**
3.1 Schemas and dependencies 7
3.2 Data Exchange 7

4 Schema Templates and Encoding of Schemas **8**
4.1 Schema templates 9
4.2 Relational decoding 10
4.3 Relational encoding 11

5 Schema Exchange **12**
5.1 Source-to-target template dependency 12
5.2 The problem of schema exchange 13
5.3 Universal solutions and core 14

6 From Schema to Data Exchange **16**
6.1 Metaroutes and S-D transformation process 16
6.2 Properties of the S-D transformation process 19

7 Related work **21**

8 Conclusion and Future work **22**

List of Figures

1 An example of data exchange. 5
2 Another example of data exchange. 6
3 An example of schema exchange that generalizes the data exchanges in Figures 1 and 2. 6
4 Schema exchange scenario for Example 5 14
5 Schema exchange scenario for Example 7. 17
6 Data exchange scenario for Example 7. 19

1 Introduction

In the last years, we have witnessed an increasing complexity of database applications, especially when several and possibly autonomous data sources need to be accessed, transformed and combined. 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, the data exchange problem has received recently great attention, both from a theoretical [12, 14] and a practical point of view [22]. In a data exchange scenario, given a mapping between a schema, called the source schema, and another schema, called the target schema, the goal is to take data structured under the source schema and transforming it into a format conforming to the target schema.

In this paper, we address the novel problem of *schema* exchange, which naturally extends the data exchange process to *sets* of similar schemas: while the data exchange process operates over specific source and target schemas, the goal of schema exchange is rather the definition of generic transformations of data under structurally similar schemas. To this aim, we introduce the notion of *schema template*, which is used to represent a class of different database schemas sharing the same structure. Then, given a mapping between the components of a source and a target template, the goal is the translation of any database whose schema conforms to the source template into a format conforming to the target template.

This framework can be used to support several activities involved in the management of heterogeneous data sources. First, it allows the definition, once for all, of "generic" transformations that work for different but similar schemas, such as the denormalization of a pair of relation tables based on a foreign key between them. Then, it can support the *reuse* of a data exchange setting, since a mapping between templates can be derived from a mapping between schemas for later use in similar scenarios. Finally, it can be used to implement *model translations*, that is, translations of schemas and data from one data model to another (e.g., from relational to XML), a problem largely studied in recent years [1, 4, 20].

As it has been done for data exchange [14], we introduce a formal notion of *solution* for the schema exchange problem and present a technique for the automatic generation of solutions. This is done by representing constraints over templates and mappings between them with a special class of first order formulas. These dependencies are used to generate the solution by applying the chase procedure [2] to an "encoding" of the source schema. Then, we propose an algorithm for deriving automatically a data exchange setting from a schema exchange solution and we show that this setting reflects the semantics of the mappings defined over templates at an higher level of abstraction. In this way, it is possible to migrate data from a source database into the database obtained as a result of the schema exchange.

The final goal of our research is the development of a design a tool in which the user can: (i) describe data collections presenting structural similarities, by means of a source template T_1, (ii) define the structure of a possible transformation of the source by means of a target template T_2 and a set correspondences over T_1 and T_2, graphically represented by lines between T_1 to T_2, (iii) translate any data source over a schema matching with T_1 into a format described by a schema matching with T_2, (iv) make use of a set of predefined schema exchange settings as *design patterns* for the development of new data exchange settings, and (v) convert a data exchange setting into a schema exchange one for its reuse.

The structure of the paper is as follows. In Section 2 we illustrate and motivate the schema exchange problem by means of a number of practical examples. In Section 3 we set the basic definitions and recall some useful results on the data exchange problem. In Section 4 we introduce formally the notion of template and show how templates and schemas are related. In Section 5, we

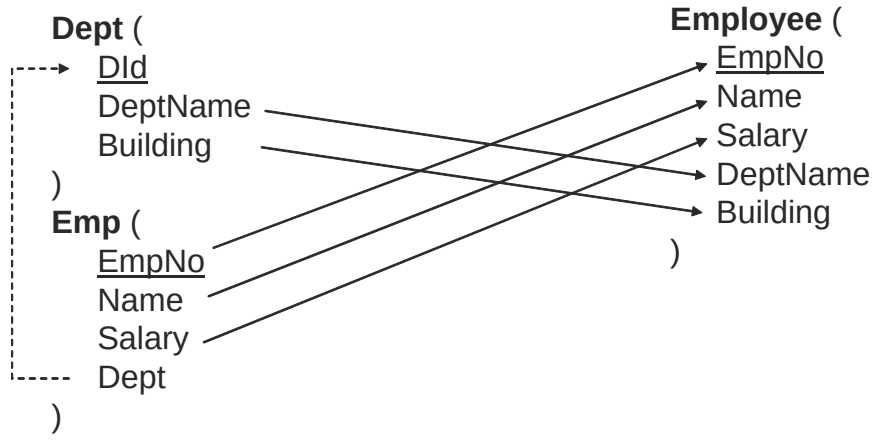

Figure 1: An example of data exchange.

define the problem of schema exchange and show how "correct" solutions can be generated and, in Section 6, we present a method for converting a schema exchange into a data exchange setting. Finally, in Section 7 we compare our work to existing literature and in Section 8 we draw some conclusions and sketch future directions of research.

2 Motivating Examples

A graphical example of a data exchange scenario is reported in Figure 1. On the left hand side there is a *source* schema involving the relations `Emp` and `Dept` linked by a foreign key and, on the right hand side, a *target* schema that involves the single relation `Employee`. The correspondences between the source and the target, expressed by means of arrows, suggest that the target can be obtained from the source by joining the relations `Emp` and `Dept` according to the foreign key, projecting the result on the attributes involved by the arrows, and storing the result into the relation `Employee`. This transformation can be specified, in a precise way, by means of the following first order rule, called *source-to-target dependency*:

$$\texttt{Emp}(e, n, s, d), \texttt{Dept}(d, dn, b) \rightarrow \texttt{Employee}(e, n, s, dn, b)$$

The goal of the given scenario is indeed to derive in an automatic way the queries needed to translate data structured under the source schema into a format conforming to the target schema, according to the given source-to-target dependencies.

Let us now consider the data exchange setting reported graphically in Figure 2. The example is different (we now have students and courses) but the underlying transformation to obtain the target is indeed similar: again, we need to "denormalize" the source by joining two relations using a referential constraint between them.

The basic idea of our approach is that this transformation can be conveniently represented, in general terms, by means of a framework, which we have called *schema exchange*, reported graphically in Figure 3. In this scenario we have *templates* of schema, that is, generic structures representing a set of database schemas with a similar configuration, and a mapping between them, again informally represented by means of arrows. The aim is to represent a generic transformation that can be applied to different, but structurally similar, data sources.

Intuitively, the semantics of this scenario, in the example at the hand, is the following. Given a relational schema involving at least two relations related by a foreign key, *for each* pair of relations

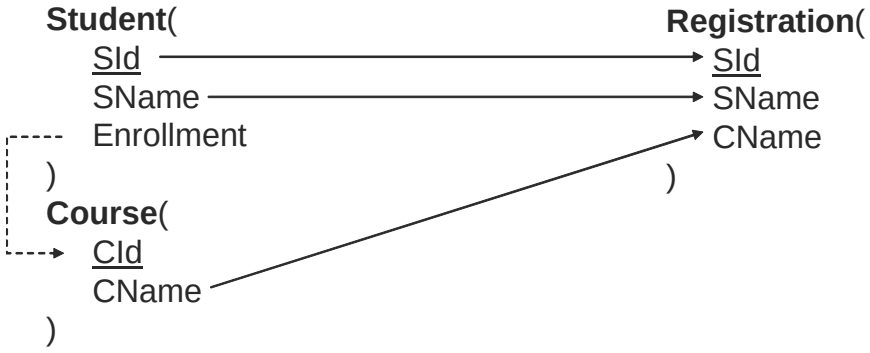

Figure 2: Another example of data exchange.

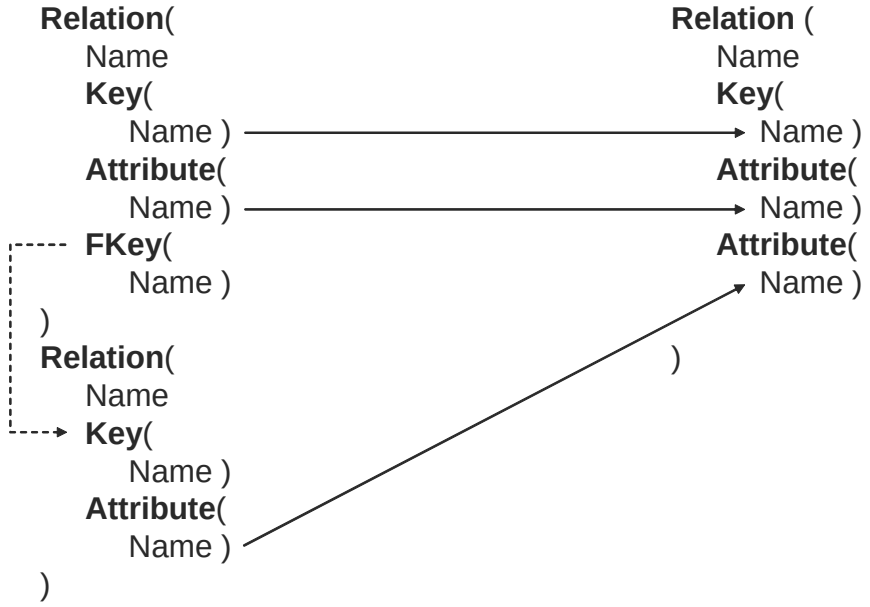

Figure 3: An example of schema exchange that generalizes the data exchanges in Figures 1 and 2.

R and R' such that: (i) R has a key, a non-empty set of attributes and at least a foreign key attribute that refers to (the key of) R', and (ii) R' has a key and a non-empty set of attributes, *generate* a target relation with the same attributes and the same key attributes of R, and with the attributes of R' not involved in the key.

Also in this case, the semantics can be precisely specified by a source-to-target dependency defined over templates, as follows:

$$\begin{aligned}&\mathsf{Relation}(n_R), \mathsf{Key}(n_K, n_R), \mathsf{Attribute}(n_A, n_R), \mathsf{FKey}(n_F, n_R, n'_R),\\ &\mathsf{Relation}(n'_R), \mathsf{Key}(n'_K, n'_R), \mathsf{Attribute}(n'_A, n'_R) \rightarrow\\ &\mathsf{Relation}(n''_R), \mathsf{Key}(n_K, n''_R), \mathsf{Attribute}(n_A, n''_R), \mathsf{Attribute}(n'_A, n''_R)\end{aligned}$$

It is easy to verify that given as input the two sources schemas in the examples in Figures 1 and 2 we obtain the corresponding target schemas as result of the generic transformation represented in Figure 3. The source-to-target dependencies between the source and target schema that reflects the semantics of the mappings defined on the original templates can also be derived.

The notion of schema exchange can be used to support several practical problems related to the exchange of information between heterogeneous repositories of data:

- First, it can be used to support the design of data exchange setting, since a schema exchange models naturally a *design pattern* for data exchange, that is, a general repeatable solution to a commonly occurring data transformation from a format into another;
- Moreover, it can support the *reuse* of a data exchange setting, since a schema exchange that generalizes a given data exchange mapping can be easily derived and later used to implement a transformation similar to the original one;
- Finally, a schema exchange framework can be used to represent and tackle the problem of *model translation* [1, 4, 20], in which the goal is the translation of schemas and data between heterogenous data models. In fact, even if we mainly refers to the relational model, our notion of schema template can represent indeed schemas of a large variety of data models.

The rest of this paper is devoted to the precise definition of this notion of schema exchange, and the investigation of its general properties. As we have said in the introduction, the final goal of our research is the design of a practical system, based on schema exchange, supporting the user in the design, reuse and maintenance of data exchange settings.

3 Background

In this section we illustrate the notions that are at the basis of our work: relational schemas, data dependencies and the problem of data exchange.

3.1 Schemas and dependencies

A *schema* $\mathbf{S}$ is composed by a set of *relations* $R(A_1, \ldots, A_n)$, where R is the *name* of the relation and $A_1, \ldots, A_k$ are its *attributes*. Each attribute is associated with a set of values called the *domain* of the attribute. An *instance* of a relation $R(A_1, \ldots, A_n)$ is a finite 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 $\mathbf{S}$.

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. If $\mathbf{y}$ is empty (no existentially quantified variables), then we speak about a *full* tgd. 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 $\mathbf{x}$.

For simplicity, hereinafter we will omit all the quantifiers in formulas assuming that variables occurring in the left hand size of a formula are universally quantified and those occurring only in the right hand size are existentially quantified.

3.2 Data Exchange

In the relational-to-relational data exchange framework [12], a *data exchange setting* is described by a four-tuple $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where:

- $\mathbf{S}$ is a source schema,
- $\mathbf{T}$ is a target schema,
- Σ_{st} is a finite set of *s-t* (source-to-target) tgds $\phi(\mathbf{x}) \rightarrow \chi(\mathbf{x}, \mathbf{y})$ 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
- Σ_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 Σ_{st} and J satisfies Σ_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 [14] it was shown that a universal solution of I under M certainly exists when the dependencies in Σ_t are either egds or *weakly-acyclic* tgds, a class of tgds which admits limited forms of cycles among different relation arguments. In this case, if a solution exists, a universal solution can be computed in polynomial time 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 the given dependencies. There are two kinds of chase steps:

- a tgd $\phi(\mathbf{x}) \rightarrow \psi(\mathbf{x}, \mathbf{y})$ 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 $\mathbf{y}$;
- an egd $\phi(\mathbf{x}) \rightarrow (x_1 = x_2)$ 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.

Universal solutions are particularly important also for query answering, since Fagin et al. [14] have shown that the (certain) answers to a query q over the target schema can be obtained by evaluating q over any universal solution.

The problem is that, in general, there may exist multiple, different universal solutions for a data exchange problem. However, in [12] it has been shown that there is one particular universal solution, called the *core*, which is the most compact one. More specifically, the core of a universal solution I is (up to isomorphism) the smallest subset J of I such that J is homomorphically equivalent to I. Gottlob and Nash [16] have shown that the core of a universal solution of a data exchange problem whose source-to-target constraints are tgds and whose target constraints consist of egds and weakly acyclic tgds can be computed in polynomial time.

4 Schema Templates and Encoding of Schemas

In this section we introduce the notion of *schema template*, which is used to encode database schemas sharing the same structure. We also show how instances of templates, called e-schemas, can be converted to relational schemas and vice versa.

4.1 Schema templates

We fix a finite set $\mathcal{C}$ of *construct names*. A *construct* $C(p_0, p_1, \ldots p_k)$ has a name C in $\mathcal{C}$ and a finite set $p_1, \ldots, p_k$ 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 constructs from several data models so that we can include in our framework schemas of different models. In this paper however, for sake of simplicity, we focus on relational schemas; the approach can be extended to richer data models, but challenging issues already arise in the relational model.

Therefore, given a set $\mathcal{R}$ *relation names*, we fix the following relational constructs and properties:

Construct Names	Properties (domain)
Relation	name ($\mathcal{R}$)
Attribute	name (`string`), nullable (`boolean`), in ($\mathcal{R}$)
Key	name (`string`), in ($\mathcal{R}$)
FKey	name (`string`), in ($\mathcal{R}$), refer ($\mathcal{R}$)

The Relation construct has only the name property, whose domain is $\mathcal{R}$. The constructs Key and FKey denote attributes that belong to a primary key and to a foreign key, respectively. The property in of the constructs Attribute, Key and FKey has $\mathcal{R}$ as domain and is used to specify the relation that includes the construct. Finally, the property refer of the construct FKey has also $\mathcal{R}$ as domain and is used to specify the relation it refers to. Clearly, other properties can be considered for every construct and have not been included to keep the notation simple. For instance, we could associate the properties basic_type and default_value with the construct Attribute.

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:

$$\mathbf{C} = \{ \quad \mathsf{Relation}(\mathsf{name}), \mathsf{Key}(\mathsf{name}, \mathsf{in}), \mathsf{Attribute}(\mathsf{name}, \mathsf{nullable}, \mathsf{in}), \\ \mathsf{FKey}(\mathsf{name}, \mathsf{in}, \mathsf{refer}) \ \}$$

and the dependencies:

$$\begin{aligned} \Sigma_{\mathbf{C}} = \{ \quad & d_1 = \mathsf{Key}(n_K, n_R) \rightarrow \mathsf{Relation}(n_R), \\ & d_2 = \mathsf{Attribute}(n_A, u, n_R) \rightarrow \mathsf{Relation}(n_R), \\ & d_3 = \mathsf{FKey}(n_F, n_R, n'_R) \rightarrow \mathsf{Relation}(n_R), \mathsf{Relation}(n'_R), \\ & d_4 = \mathsf{Attribute}(n_A, u, n_R) \rightarrow (u = \mathtt{true}) \ \} \end{aligned}$$

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. ■

Note that the dependencies d_1, d_2 and d_3 in Example 1 should always hold in a relational schema. Actually, they can be considered as axioms of the relational model. For this reason, in the following we will call them *relational* dependencies and assume that they always belong to the dependencies associated with a relational template.

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 of* C *a value 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

name
EMP
DEPT

Key

name	in
EmpName	EMP
DeptNo	DEPT

Attribute

name	nullable	in
Salary	true	EMP
Building	true	DEPT

FKey

name	in	refer
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 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 $C(p_1, \ldots, p_k)$ will be called a *relation* component if $C = \mathsf{Relation}$, an *attribute* component if $C = \mathsf{Attribute}$, a *key* component if $C = \mathsf{Key}$, a *foreign key* component if $C = \mathsf{FKey}$. Moreover, we will denote an e-schema component over a construct $C(p_1, \ldots, p_k)$ by $C(p_1 : a_i, \ldots, p_k : a_k)$. Alternatively, we will use, for each construct, a tabular notation with a column for each property.

In the next sections, we describe how the notion of e-schema can be converted into a "standard" relational schema, and vice versa.

4.2 Relational decoding

Basically, in order to convert an e-schema into a relational schema we need to define the formulas describing the semantics of the various e-schema components, 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 R-Dec(S), is a pair $(\mathbf{S}, \Sigma_{\mathbf{S}})$ where:

- $\mathbf{S}$ contains a relation $R(K_1, \ldots, K_m, A_1, \ldots, A_n, F_1, \ldots, F_p)$ for each relation component $r \in \mathsf{S}$ such that:
  - $R = r(\mathsf{name})$;
  - $K_i = k_i(\mathsf{name})$ $(1 \leq i \leq m)$ for each key component $k_i \in \mathsf{S}$ such that $k_i(\mathsf{in}) = R$,
  - $A_j = a_j(\mathsf{name})$ $(0 \leq j \leq n)$ for each attribute component $a_j \in \mathsf{S}$ such that $a_j(\mathsf{in}) = R$,
  - $F_k = f_k(\mathsf{name})$ $(0 \leq k \leq p)$ for each foreign key component $f_k \in \mathsf{S}$ such that $f_k(\mathsf{in}) = R$.
- $\Sigma_{\mathbf{S}}$ contains an egd over the relation $R(K_1, \ldots, K_m, A_1, \ldots, A_n) \in \mathbf{S}$ of the form:

$$R(x_1, \ldots, x_m, y_1, \ldots, y_n), R(x_1, \ldots, x_m, y'_1, \ldots, y'_n) \rightarrow (y_1 = y'_1, \ldots, y_n = y'_n)$$

  for each relation component $r \in \mathsf{S}$ such that:

– $R = r(\mathsf{name})$;

– $K_i = k_i(\mathsf{name})$ $(1 \leq i \leq m)$ for each key component $k_i \in \mathsf{S}$ such that $k_i(\mathsf{in}) = R$

- $\Sigma_{\mathbf{R}}$ contains a tgd over a pair of relation schemas $R(A_1, \ldots, A_m, F_1, \ldots, F_n)$ and $R'(K_1, \ldots, K_n, A'_1, \ldots, A'_p)$ in $\mathbf{S}$ of the form:

$$R(x_1, \ldots, x_m, y_1, \ldots, y_n) \rightarrow R'(y_1, \ldots, y_n, z_1, \ldots, z_p)$$

for each pair of relation components r and r' in S such that:

– $R = r(\mathsf{name})$ and $R' = r'(\mathsf{name})$;

– $F_i = f_i(\mathsf{name})$ $(1 \leq i \leq n)$ for each foreign key component $f_i \in \mathsf{S}$ such that $f_i(\mathsf{in}) = R$ and $f_i(\mathsf{refer}) = R'$.

Example 3 Let us consider the e-schema S of Example 2, which is reported below for convenience:

Relation

name
EMP
DEPT

Key

name	in
EmpName	EMP
DeptNo	DEPT

Attribute

name	nullable	in
Salary	true	EMP
Building	false	DEPT

FKey

name	in	refer
Dept	EMP	DEPT

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

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

$$\begin{aligned}\Sigma_{\mathbf{S}} = \{\quad & \mathtt{EMP}(x_1, x_2, x_3), \mathtt{EMP}(x_1, x'_2, x'_3) \rightarrow (x_2 = x'_2, x_3 = x'_3),\\ & \mathtt{DEPT}(x_1, x_2), \mathtt{DEPT}(x_1, x'_2) \rightarrow (x_2 = x'_2),\\ & \mathtt{EMP}(x_1, x_2, x_3) \rightarrow \mathtt{DEPT}(x_3, x'_2)\ \}\end{aligned}$$

■

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 S, can also be defined. This procedure will be illustrated in the following section.

4.3 Relational encoding

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

- S contains a relation component r for each relation $R(A_1, \ldots, A_n) \in \mathbf{S}$ such that $r(\mathsf{name}) = R$;
- S contains an attribute component a for each attribute A of a relation $R \in \mathbf{S}$ not involved in an egd or in a tgd $\Sigma_{\mathbf{S}}$ over R such that: $a(\mathsf{name}) = A_i$ and $a(\mathsf{in}) = R$;

- S contains a key component k_i $(1 \leq i \leq m)$ for each egd in $\Sigma_{\mathbf{S}}$ over a relation schema $R(K_1, \ldots, K_m, A_1, \ldots, A_n) \in \mathbf{S}$ of the form:

$$R(x_1, \ldots, x_m, y_1, \ldots, y_n), R(x_1, \ldots, x_m, y'_1, \ldots, y'_n) \rightarrow (y_1 = y'_1, \ldots, y_n = y'_n)$$

  such that: $k_i(\mathsf{name}) = K_i$ and $k_i(\mathsf{in}) = R$;

- S contains a foreign key component f_i $(1 \leq i \leq n)$ for each tgd over a pair of relation schemas $R(A_1, \ldots, A_m, F_1, \ldots, F_n)$ and $R'(K_1, \ldots, K_n, A'_1, \ldots, A'_p)$ in $\mathbf{S}$ of the form:

$$R(x_1, \ldots, x_m, y_1, \ldots, y_n) \rightarrow R'(y_1, \ldots, y_n, z_1, \ldots, z_p)$$

  such that: $f_i(\mathsf{name}) = F_i$, $f(\mathsf{in}) = R$, and $f(\mathsf{refer}) = R'$.

Example 4 Let us consider the relational schema $(\mathbf{S}, \Sigma_{\mathbf{S}})$ where:

$$\mathbf{S} = \{\ \texttt{ORDER}(\texttt{OrderNo}, \texttt{Item}, \texttt{Quantity}), \texttt{PRODUCT}(\texttt{ProdNo}, \texttt{Price})\ \}$$

$$\Sigma_{\mathbf{S}} = \{\ \begin{array}{c} \texttt{ORDER}(x_1, x_2, x_3), \texttt{ORDER}(x_1, x'_2, x'_3) \rightarrow (x_2 = x'_2, x_3 = x'_3), \\ \texttt{PRODUCT}(x_1, x_2), \texttt{PRODUCT}(x_1, x'_2) \rightarrow (x_2 = x'_2), \\ \texttt{ORDER}(x_1, x_2, x_3) \rightarrow \texttt{PRODUCT}(x_2, x'_2)\ \} \end{array}$$

The encoding of $(\mathbf{S}, \Sigma_{\mathbf{S}})$ is the following e-schema:

Relation

name
ORDER
PRODUCT

Key

name	in
OrderNo	ORDER
ProdNo	PRODUCT

Attribute

name	in
Quantity	ORDER
Price	PRODUCT

FKey

name	in	refer
Item	ORDER	PRODUCT

■

5 Schema Exchange

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

5.1 Source-to-target template dependency

In order to investigate the problem of data and metadata translation between templates we introduce the following notion.

Definition 3 (s-t template dependency) *Given a* source *template* $\mathcal{T}_1$ *and a* target *template* $\mathcal{T}_2$*, a* source-to-target (s-t) template dependency *is a tuple generating dependency of the form:* $\phi(\boldsymbol{x}) \rightarrow \psi(\boldsymbol{x}, \boldsymbol{y})$*, where* $\phi(\boldsymbol{x})$ *is a conjunction of atomic formulas over the components of* $\mathbf{C}_1$ *and* $\psi(\boldsymbol{x}, \boldsymbol{y})$ *is a conjunction of atomic formulas over the components of* $\mathbf{C}_2$*.*

There are two different but equivalent semantics that can be associated with s-t template dependencies. In a declarative semantics, they represent constraints required to hold on pairs of instances over the source and the target template. Under this semantics, multiple pairs of e-schemas may satisfy the relationship. Alternatively, under the transformational semantics, the dependencies are viewed as transformations from the source to the target: given a source e-schema S over the source template, there is a procedural way of computing a target e-schema S' over the target template. This transformational semantics follows the data exchange line of work and it is the one we adopt in the schema exchange definition we introduce next.

5.2 The problem of 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 s-t template dependencies, we denote a *schema exchange setting* by the triple $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$.

Definition 4 (Schema exchange) *Let* $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$ *be a schema exchange setting and* S_1 *a source e-schema over* $(\mathbf{C}_1, \Sigma_{\mathbf{C}_1})$*. The* schema exchange problem *consists in finding a finite target e-schema* S_2 *over* $(\mathbf{C}_2, \Sigma_{\mathbf{C}_2})$ *such that* $\mathsf{S}_1 \cup \mathsf{S}_2$ *satisfies* $\Sigma_{\mathbf{C}_1\mathbf{C}_2}$*. In this case* S_2 *is called* a solution for S_1 *or, simply a* solution.

Example 5 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:

$$\mathbf{C}_1 = \{ \quad \mathsf{Relation}(\mathsf{name}), \mathsf{Key}(\mathsf{name}, \mathsf{in}), \mathsf{Attribute}(\mathsf{name}, \mathsf{in}) \ \}$$

$$\mathbf{C}_2 = \{ \quad \mathsf{Relation}(\mathsf{name}), \mathsf{Key}(\mathsf{name}, \mathsf{in}), \mathsf{Attribute}(\mathsf{name}, \mathsf{in}), \mathsf{FKey}(\mathsf{name}, \mathsf{in}, \mathsf{refer}) \ \}$$

with the relational dependencies in $\Sigma_{\mathbf{C}_1}$ and in $\Sigma_{\mathbf{C}_2}$ shown in Example 1.

Assume now that we would like to split each relation 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 4 and is precisely captured by the following set of tgds $\Sigma_{\mathbf{C}_1,\mathbf{C}_2}$:

$$\begin{aligned}\Sigma_{\mathbf{C}_1,\mathbf{C}_2} = \{ \quad & \mathsf{Relation}(n_R), \mathsf{Key}(n_K, n_R), \mathsf{Attribute}(n_A, n_R) \rightarrow \\ & \mathsf{Relation}(n_R), \mathsf{Key}(n_K, n_R), \mathsf{FKey}(n_F, n_R, n'_R), \\ & \mathsf{Relation}(n'_R), \mathsf{Key}(n_F, n'_R), \mathsf{Attribute}(n_A, n'_R) \ \}\end{aligned}$$

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

Relation

name
EMP

Key

name	in
EmpName	EMP

Attribute

name	in
DeptName	EMP
Floor	EMP

According to the decoding procedure described in Section 4.2, this e-schema encode the relational schema: $\mathbf{S} = \{ \ \mathtt{EMP}(\mathtt{EmpName}, \mathtt{DeptName}, \mathtt{Floor})\}$ in which $\mathsf{EmpName}$ is the key. A possible solution S'_1 for this setting is:

Relation

name
EMP
$\mathtt{R}_1$
$\mathtt{R}_2$

Key

name	in
EmpName	EMP
$\mathtt{K}_1$	$\mathtt{R}_1$
$\mathtt{K}_2$	$\mathtt{R}_2$

Attribute

name	in
DeptName	$\mathtt{R}_1$
Floor	$\mathtt{R}_2$

FKey

name	in	refer
$\mathtt{K}_1$	EMP	$\mathtt{R}_1$
$\mathtt{K}_2$	EMP	$\mathtt{R}_2$

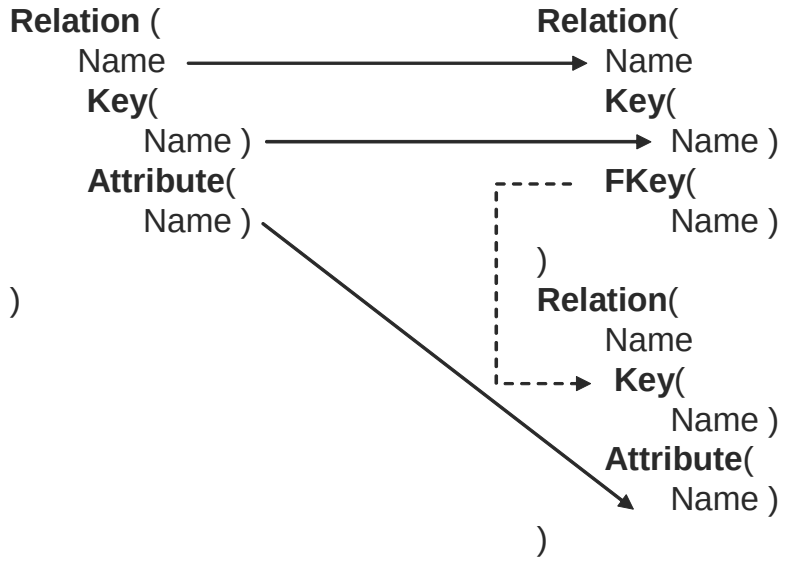

Figure 4: Schema exchange scenario for Example 5

where $\mathtt{R}_1, \mathtt{R}_2, \mathtt{K}_1, \mathtt{K}_2$ are labelled nulls. The decoding of this solution contains three relations: $\mathtt{EMP}(\underline{\mathtt{EmpName}}, \mathtt{K}_1, \mathtt{K}_2)$, $\mathtt{R}_1(\underline{\mathtt{K}_1}, \mathtt{DeptName})$, and $\mathtt{R}_2(\underline{\mathtt{K}_2}, \mathtt{Floor})$, in which the attributes $\mathtt{K}_1, \mathtt{K}_2$ of relation EMP are foreign keys for $\mathtt{R}_1$ and $\mathtt{R}_2$, respectively. There are several null values because the dependencies in $\Sigma_{\mathbf{C}_1,\mathbf{C}_2}$ do not allow the complete definition of the target e-schema.

Actually many other solutions for the same schema exchange problem exist. For instance the following e-schema S'_2:

Relation

name
EMP
$\mathtt{R}_1$

Key

name	in
EmpName	EMP
$\mathtt{K}_1$	$\mathtt{R}_1$

Attribute

name	in
DeptName	$\mathtt{R}_1$
Floor	$\mathtt{R}_1$

FKey

name	in	refer
$\mathtt{K}_1$	EMP	$\mathtt{R}_1$

where $\mathtt{R}_1$ and $\mathtt{K}_1$ are labelled nulls. By decoding this solution we obtain two relations: $\mathtt{EMP}(\underline{\mathtt{EmpName}}, \mathtt{K}_1)$ and $\mathtt{R}_1(\underline{\mathtt{K}_1}, \mathtt{DeptName}, \mathtt{Floor})$, where $\mathtt{K}_1$ of relation EMP is a foreign key for $\mathtt{R}_1$. ■

Two issues arise from Example 5: 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 homomorphisms from S'_1 to S'_2 ($R_2 \mapsto R_1, K_2 \mapsto K_1$), 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 [12, 14], we argue that the "correct" solution is the most general one, in the sense above. This solution is called universal, as illustrated in the next section.

5.3 Universal solutions and core

The discussion on the possible solutions for a schema exchange problem leads to the following definition.

Definition 5 (Universal solution) *A solution S of the schema exchange problem is* universal *if there exists a homomorphism from S to all the other solutions.*

The next result provides a method for the generation of a universal solutions of a given schema exchange problem. It follows from an analogous result for 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* $\mathcal{T}_1$ *and assume that* $\Sigma_{\mathbf{C}_2}$ *includes weakly acyclic tgds. 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 if a solution exists and fails otherwise.*

Proof: A schema exchange setting can be viewed as a data exchange setting over a source schema S_1 and a target schema S_2 that involve a relation for each construct of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Under this view, S_1 is an instance of S_1. It follows that the results on data exchange shown in [14] can be applied to our scenario. In particular, the fact that, if a solution exists, the application of the chase procedure to an instance of S_1 using the given dependencies terminates and generates a universal solution. In this case, by construction, the output is an instance of $\mathcal{T}_2$ and, by definition, is also a universal solution of the given schema exchange problem. □

Theorem 1 provides a procedural way to generate a universal solution of a schema exchange problem. However, what we obtain is only one of the possible universal solutions and it turns out that, in general, it is not necessarily the best in terms of size. This is clarified in the following example.

Example 6 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:

$$\mathbf{C}_1 = \{\mathsf{Relation}(\mathsf{name}), \mathsf{Key}(\mathsf{name}, \mathsf{in}), \mathsf{Attribute}(\mathsf{name}, \mathsf{in})\ \}$$

$$\mathbf{C}_2 = \{\mathsf{Relation}(\mathsf{name}), \mathsf{Key}(\mathsf{name}, \mathsf{in})\}$$

with the usual relational constraints in $\Sigma_{\mathbf{C}_1}$ and in $\Sigma_{\mathbf{C}_2}$ shown in Example 1.

Assume now that we would like just to copy each relation over $\mathcal{T}_1$ into a relation over $\mathcal{T}_2$ having a new key. This transformation can be implemented by the following tgd:

$$\Sigma_{\mathbf{C}_1,\mathbf{C}_2} = \{\quad \mathsf{Relation}(n_R), \mathsf{Key}(n_K, n_R), \mathsf{Attribute}(n_A, n_R) \rightarrow \\ \mathsf{Relation}(n_R), \mathsf{Key}(n'_K, n_R)\ \}$$

Let us consider the following e-schema S over $\mathcal{T}_1$:

Relation

name
ORDER

Key

name	in
OrderNo	ORDER

Attribute

name	in
Item	ORDER
Quantity	ORDER

If we apply the chase over S using over $\Sigma_{\mathbf{C}_1,\mathbf{C}_2}$ we obtain the following universal solution S':

Relation

name
ORDER

Key

name	in
$\mathtt{K}_1$	ORDER
$\mathtt{K}_2$	ORDER

where $\mathtt{K}_1$ and $\mathtt{K}_2$ are labelled nulls and form a composite key for the relation ORDER.

However, it is easy to see that the following e-schema S'' is also a universal solution:

Relation

name
ORDER

Key

name	in
$\mathtt{K}_1$	ORDER

Note that S'' is homomorphically equivalent to S' (that is, there is an homomorphism from S' to S'' and vice versa) but it is more compact than S'. ■

Solutions S' and S'' of Example 6 are both universal but S'' is clearly preferable to S' since it is smaller in size. This solution is actually the *core* of all the universal solutions for S. It has been shown that the core is unique, as all universal solutions for a schema exchange problem have the same core up to isomorphism [12]. The core for a schema exchange problem with only egd as target constraints can be computed in naive way by successively removing tuples from a universal solution, as long as the solution resulting in each step satisfy the dependencies. Recently, Gottlob and Nash [16] have proposed a polynomial time algorithm that computes the core of a universal solution of a data exchange problem whose source-to-target constraints are tgds and whose target constraints consist of egds and weakly acyclic tgds.

6 From Schema to Data Exchange

In this section we propose a transformation process that generates a *data* exchange setting as an instance of a schema exchange setting for a given data source.

6.1 Metaroutes and S-D transformation process

Before discussing the transformation process, a preliminary notion is needed. 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 [11], we introduce the notion of *metaroutes* to describe the relationships between source and target metadata.

Definition 6 *Let S be an e-schema and Σ be a set of s-t template dependencies. A* metaroute *for S is an expression of the form $I \rightarrow_{\sigma,h} I'$ where $I \subseteq \mathsf{S}$ and I' is the result of the application of a chase step on I based on the dependency $\sigma \in \Sigma$ and the homomorphism h.*

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 [11].

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 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 S_2 of $\mathbf{S}_2$ is a universal solution for S_1. We call such process S-D *transformation* and it can be summarized as follows.

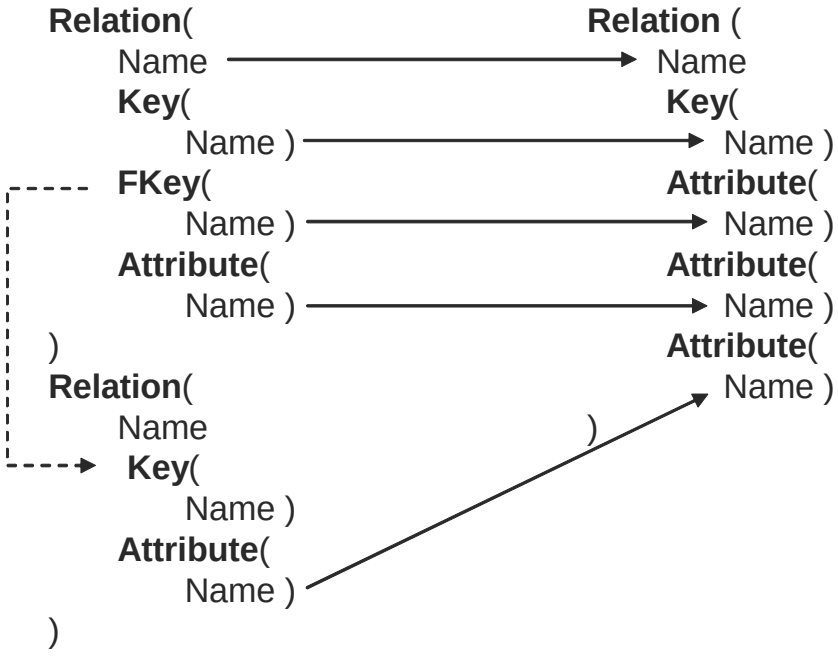

Figure 5: Schema exchange scenario for Example 7.

Input: A schema $\mathbf{S}_1$ and a schema exchange setting $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$;
Output: A data exchange setting $(\mathbf{S}_1, \mathbf{S}_2, \Sigma_{\mathbf{S}_1\mathbf{S}_2})$;

---

(1) Encode $\mathbf{S}_1$ into an e-schema S_1;
(2) Apply the chase procedure to S_1 using $\Sigma_{\mathbf{C}_1\mathbf{C}_2}$ saving metaroutes in a set $\mathcal{M}$ during the execution: each chase step based on the dependency $\sigma \in \Sigma$ and the homomorphism h adds to $\mathcal{M}$ a metaroute $I \rightarrow_{\sigma,h} I'$;
(3) Decode the result S_2 of the chase procedure into a schema $\mathbf{S}_2$;
(4) For each metaroute $I \rightarrow_{\sigma,h} I'$ in $\mathcal{M}$:

(a) let L be the set of relations in $\mathbf{S}_1$ mentioned in I: annotate all the attributes A in L with $h^{-1}(A)$;

(b) let R be the set of relations in $\mathbf{S}_2$ mentioned in I': annotate all the attributes A' in R with $h^{-1}(A')$;

(c) replace all the attributes in L and R with variables by associating the same variable to the attributes with the same annotation;

(c) add the tgd $L \rightarrow R$ to a set $\Sigma_{\mathbf{S}_1\mathbf{S}_2}$;
(5) Return the data exchange setting $(\mathbf{S}_1, \mathbf{S}_2, \Sigma_{\mathbf{S}_1\mathbf{S}_2})$.

Example 7 Let us consider the schema exchange setting described graphically in Figure 5 and represented by the following set of tgds $\Sigma_{\mathbf{C}_1,\mathbf{C}_2}$:

$$\begin{aligned}\{\ v_1 = \ & \mathsf{Relation}(n_R), \mathsf{Key}(n_K, n_R), \mathsf{FKey}(n_F, n_R, n'_R), \mathsf{Attribute}(n_A, n_R), \\ & \mathsf{Relation}(n'_R), \mathsf{Key}(n'_K, n'_R), \mathsf{Attribute}(n'_A, n'_R) \rightarrow \\ & \mathsf{Relation}(n_R), \mathsf{Key}(n_K, n_R), \mathsf{Attribute}(n_F, n_R), \mathsf{Attribute}(n_A, n_R), \\ & \mathsf{Attribute}(n'_A, n_R)\ \}\end{aligned}$$

Intuitively, the single dependency occurring in $\Sigma_{\mathbf{C}_1,\mathbf{C}_2}$ 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:

$$\mathbf{S} = \{\ \texttt{EMP}(\texttt{eid}, \texttt{name}, \texttt{did}), \texttt{DEPT}(\texttt{eid}, \texttt{dname}),\ \}$$

$$\begin{aligned}\Sigma_{\mathbf{S}} = \{\quad & \texttt{EMP}(x_1, x_2, x_3), \texttt{EMP}(x_1, x'_2, x'_3) \rightarrow (x_2 = x'_2, x_3 = x'_3),\\ & \texttt{DEPT}(x_1, x_2), \texttt{DEPT}(x_1, x'_2) \rightarrow (x_2 = x'_2),\\ & \texttt{EMP}(x_1, x_2, x_3) \rightarrow \texttt{DEPT}(x_3, x'_1)\ \}\end{aligned}$$

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

Relation

	name
s_1	EMP
s_2	DEPT

Key

	name	in
s_3	eid	EMP
s_4	did	DEPT

Attribute

	name	in
s_5	name	EMP
s_6	dname	DEPT

FKey

	name	in	refer
s_7	did	EMP	DEPT

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

Relation

	name
t_1	EMP

Key

	name	in
t_2	eid	EMP

Attribute

	name	in
t_3	name	EMP
t_4	did	EMP
t_5	dname	EMP

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

$$\begin{gathered}\{\ n_R \mapsto \texttt{EMP}, n_K \mapsto \texttt{eid}, n_F \mapsto \texttt{did}, n_A \mapsto \texttt{name}, n'_R \mapsto \texttt{DEPT},\\ n'_K \mapsto \texttt{did}, n'_A \mapsto \texttt{dname}\ \}\end{gathered}$$

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

$$\mathbf{S}' = \{\texttt{EMP}(\texttt{eid}, \texttt{name}, \texttt{did}, \texttt{dname})\}$$

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

Now, on the basis of the above metaroute, we derive the following sets of annotated relations:

$$\begin{gathered}L = \{\texttt{EMP}(\texttt{eid}[n_K], \texttt{name}[n_A], \texttt{did}[n_F]), \texttt{DEPT}(\texttt{did}[n_F], \texttt{dname}[n'_A])\}\\ R = \{\texttt{EMP}(\texttt{eid}[n_K], \texttt{name}[n_A], \texttt{did}[n_F], \texttt{dname}[n'_A])\}\end{gathered}$$

By replacing all the attributes in L and R with variables in such a way that the attributes with the same annotation are replaced with the same variable we obtain the following s-t tgd:

$$t_1 = \quad \mathsf{S}.\texttt{EMP}(x_1, x_2, x_3), \mathsf{S}.\texttt{DEPT}(x_3, x_4) \rightarrow \mathsf{S}'.\texttt{EMP}(x_1, x_2, x_3, x_4)$$

The final data mapping scenario is reported graphically in Figure 6. ■

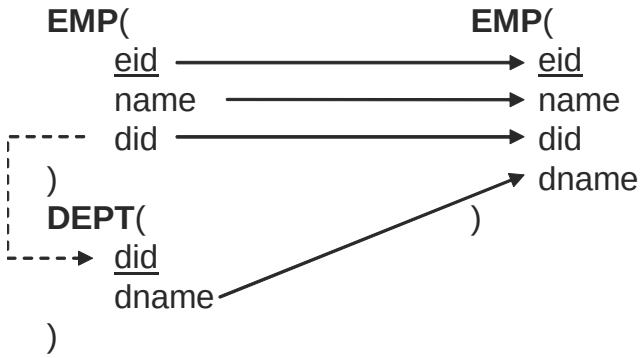

Figure 6: Data exchange scenario for Example 7.

6.2 Properties of the S-D transformation process

A number of general results on the S-D transformation process can be shown. First, the fact that the output of the 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 s-t tgd t over relational schemas, the *encoding* of t is a tgd over schema templates obtained by applying a procedure similar to the one defined in Section 4.3 for schemas.

More precisely, let t a s-t tgd over a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, and let $\Sigma_{\mathbf{S}}$ and $\Sigma_{\mathbf{T}}$ be two set of dependencies over $\mathbf{S}$ and $\mathbf{T}$, respectively.

The *encoding* of t is a s-t template tgd $\hat{t}$ such that:

- the left-hand-side (right-hand-side) of the tgd $\hat{t}$ contains an atom of the form $\mathsf{Relation}(R)$ for each relation R occurring in the lhs (rhs) of t;
- the lhs (rhs) of $\hat{t}$ contains a set of atoms of the form $\mathsf{Key}(K_i, R)$ $(1 \leq i \leq m)$ for each relation $R(K_1, \ldots, K_m, A_1, \ldots, A_n)$ occurring in t such that there is an egd:

$$R(x_1, \ldots, x_m, y_1, \ldots, y_n), R(x_1, \ldots, x_m, y'_1, \ldots, y'_n) \rightarrow (y_1 = y'_1, \ldots, y_n = y'_n)$$

  in $\Sigma_{\mathbf{S}}$ $(\Sigma_{\mathbf{T}})$;
- the lhs (rhs) of $\hat{t}$ contains a set of atoms of the form $\mathsf{FKey}(F_i, R, R')$ $(1 \leq i \leq n)$ for each relation $R(A_1, \ldots, A_m, F_1, \ldots, F_n)$ occurring in t such that there is a tgd:

$$R(x_1, \ldots, x_m, y_1, \ldots, y_n) \rightarrow R'(y_1, \ldots, y_n, z_1, \ldots, z_p)$$

  in $\Sigma_{\mathbf{S}}$ $(\Sigma_{\mathbf{T}})$;
- the lhs (rhs) of $\hat{t}$ contains a set of atoms of the form $\mathsf{Attribute}(A_i, R)$ $(1 \leq i \leq p \leq n)$ for each relation $R(K_1, \ldots, K_m, A_1, \ldots, A_n)$ occurring in the lhs (rhs) of t such that A_i in not involved in the rhs of an egd in $\Sigma_{\mathbf{S}}$ $(\Sigma_{\mathbf{T}})$ or in both the lhs and rhs of a tgd in $\Sigma_{\mathbf{S}}$ $(\Sigma_{\mathbf{T}})$.

Example 8 Let us consider the tgd t_1 of the Example 7, which is reported here for convenience:

$$t_1 = \quad \mathtt{S.EMP}(x_1, x_2, x_3), \mathtt{S.DEPT}(x_3, x_4) \rightarrow \mathtt{S'.EMP}(x_1, x_2, x_3, x_4)$$

The encoding of t_1 is the following s-t template tgd.

$$\begin{aligned} v_2 = \quad & \mathsf{Relation}(\mathtt{EMP}), \mathsf{Key}(\mathtt{eid}, \mathtt{EMP}), \mathsf{Attribute}(\mathtt{name}, \mathtt{EMP}), \\ & \mathsf{FKey}(\mathtt{did}, \mathtt{EMP}, \mathtt{DEPT}), \mathsf{Relation}(\mathtt{DEPT}), \mathsf{Key}(\mathtt{did}, \mathtt{DEPT}), \\ & \mathsf{Attribute}(\mathtt{dname}, \mathtt{DEPT}) \rightarrow \mathsf{Relation}(\mathtt{EMP}), \mathsf{Key}(\mathtt{eid}, \mathtt{EMP}), \\ & \mathsf{Attribute}(\mathtt{name}, \mathtt{EMP}), \mathsf{Attribute}(\mathtt{did}, \mathtt{EMP}), \mathsf{Attribute}(\mathtt{dname}, \mathtt{EMP}) \end{aligned}$$

■

The tgd v_2 in the example above is less general than the original tgd v_1 for the schema exchange scenario described in Example 7. However, it generates the same output S' on the given input S. This exactly captures the fact that the data exchange problem obtained as output captures the semantics of the schema exchange problem given as input.

This intuition is captured by the following correctness result.

Theorem 2 *Let $(\mathbf{S}, \mathbf{S}', \Sigma_{\mathbf{SS}'})$ 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 $\widehat{\Sigma}$ be the set of s-t tgds obtained by encoding the s-t tgds in $\Sigma_{\mathbf{SS}'}$. The encoding S' of $\mathbf{S}'$ is a universal solution for the encoding S of $\mathbf{S}$ under the schema exchange setting $(\mathcal{T}_1, \mathcal{T}_2, \widehat{\Sigma})$.*

Proof: The e-schema S' is obtained in step 2 of the S-D transformation process by chasing the encoding S of $\mathbf{S}$ using the dependencies in $\Sigma_{\mathbf{C}_1\mathbf{C}_2}$. The proof proceeds by induction on the number n of chase steps needed to generate S'. Specifically, we show that, for every $0 \leq i \leq n$, indicating with S^i the e-schema produced after the i-th step in chasing S using dependencies in $\Sigma_{\mathbf{C}_1\mathbf{C}_2}$, there is an e-schema $\widehat{\mathsf{S}^i}$ produced as an intermediate result in chasing S using dependencies in $\widehat{\Sigma}$ such that S^i is homomorphic to $\widehat{\mathsf{S}^i}$. Since, by Theorem 1, if a solution exists, $\widehat{\mathsf{S}^n}$ is a universal solution for S under $(\mathcal{T}_1, \mathcal{T}_2, \widehat{\Sigma})$, if there is an homomorphism from $\mathsf{S}^n = \mathsf{S}'$ to $\widehat{\mathsf{S}^n}$ then the former is also a universal solution for S under $(\mathcal{T}_1, \mathcal{T}_2, \widehat{\Sigma})$ and the claim follows. The basis is immediate since, for $n = 0$, the e-schema S' as well as the sets of dependencies $\Sigma_{\mathbf{SS}'}$ and $\widehat{\Sigma}$ are empty by construction, and so S' is also a trivial universal solution for $(\mathcal{T}_1, \mathcal{T}_2, \widehat{\Sigma})$. With respect to the induction, assume that there is an homomorphism h^i from S^{i-1} to $\widehat{\mathsf{S}^{i-1}}$ and let $\mathsf{S}^{i-1} \rightarrow_{\sigma,h} \mathsf{S}^i$ be the metaroute corresponding to the i-th chase step over S based on the tgd $\sigma \in \Sigma_{\mathbf{C}_1\mathbf{C}_2}$ and the homomorphism h. The result of the application of this chase step is, by definition of chase step, $\mathsf{S}^i = \mathsf{S}^{i-1} \cup h(R)$, where R is the right hand side of σ. By construction (step 4 of the S-D transformation process), there is a tgd $\sigma' \in \Sigma_{\mathbf{SS}'}$ which is built from the above metaroute by associating the same variable to the attributes of $\mathbf{S}$ and $\mathbf{S}'$ that are mapped by σ. Since the encoding of a tgd preserves the bindings of the variables, it follows that there is an homomorphism h^* from σ to $\widehat{\sigma}$. Moreover, by definition of encoding of a tgd it easily follows that there is an homomorphism $\widehat{h}$ from the left hand side $\widehat{L}$ of $\widehat{\sigma}$ to the encoding S of $\mathbf{S}$. Since S is not modified by the chase procedure, we have that $\widehat{h}$ is also an homomorphism from the left hand side of $\widehat{\sigma}$ and $\widehat{\mathsf{S}^{i-1}}$. Hence, we can apply a chase step based on $\widehat{\sigma}$ and $\widehat{h}$ to $\widehat{\mathsf{S}^{i-1}}$. The result of the application of this chase step is, by definition, $\widehat{\mathsf{S}^i} = \widehat{\mathsf{S}^{i-1}} \cup \widehat{h}(\widehat{R})$, where $\widehat{R}$ is the right hand side of $\widehat{\sigma}$, and since $\widehat{R} = h^*(R)$, we have that $\widehat{\mathsf{S}^i} = \widehat{\mathsf{S}^{i-1}} \cup \widehat{h}(h^*(R))$. Now let L be the left hand side of σ: since, by the inductive hypothesis, there is an homomorphism h^i from S^{i-1} and $\widehat{\mathsf{S}^{i-1}}$, we have that $h^i \circ h$ is an homomorphism from L to $\widehat{\mathsf{S}^{i-1}}$. At the same time, the composition of the homomorphisms h^*, from L to $\widehat{L}$, and $\widehat{\sigma}$, from $\widehat{L}$ to $\widehat{\mathsf{S}^{i-1}}$, is also an homomorphism from L to $\widehat{\mathsf{S}^{i-1}}$. It follows that, on the variables occurring in L and R, $h^i \circ h = \widehat{h} \circ h^*$. Hence, we have that $h^i(\mathsf{S}^i) = h^i(\mathsf{S}^{i-1}) \cup h^i(h(R)) = \widehat{\mathsf{S}^{i-1}} \cup \widehat{h}(h^*(R)) = \widehat{\mathsf{S}^i}$, and so h^i is an homomorphism from S^i to $\widehat{\mathsf{S}^i}$. □

The following completeness result can also be shown. We say that a data exchange setting is *constant-free* if no constants are used in formulas.

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

Proof: Given a data exchange setting $M = (\mathbf{S}, \mathbf{S}', \Sigma_{\mathbf{SS}'})$ we can derive a schema exchange setting $(\mathcal{T}_1, \mathcal{T}_2, \Sigma_{\mathbf{C}_1\mathbf{C}_2})$ from which M can be obtained with the S-D transformation process by: (i) defining two templates $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\mathbf{S}$ is an instance of $\mathcal{T}_1$ and $\mathbf{S}'$ is an instance of $\mathcal{T}_2$ (this can be easily done on the basis of the structure of the encodings of $\mathbf{S}$ and $\mathbf{S}'$), and (ii) encoding the tgds in $\Sigma_{\mathbf{SS}'}$. The schema exchange setting we obtain is already suitable for our purposes, but it can be made even more general by replacing each constant with a unique variable. It is also easy to see that this cannot be done if constants appear in $\Sigma_{\mathbf{SS}'}$ since the encoding procedure does not consider them. □

7 Related work

To our knowledge, the notion of schema exchange studied in this paper is new. In general, we can say that our contribution can be set in the framework of *metadata management*. Metadata can generally be thought as information that describes, or supplements, actual data. Several studies have addressed metadata related problems, such as, interoperability [17, 23], annotations and comments on data [7, 10, 15], data provenance [9], and a large list of more specific problems, like data quality [19]. While the list is not exhaustive, it witnesses the large interest in this important area and the different facets of the problem.

Most of the proposed approaches focus on a specific kind of metadata and are not directly applicable to other cases without major modifications. Bernstein set the various problems within a very general framework called *model management* [3, 4, 5]. In [6] 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 offers some novel research opportunity in the context of the *ModelGen* operator. The ModelGen operator realizes a *schema translation* from a source data model M_s to a target data model M_t and returns an executable mapping (or a transformational script) for the translation of the instances of the source schema. For instance, the ModelGen operator could be used to translate an relational database into a schema for an XML document (e.g., a DTD). ModelGen has been implemented in commercial products as a non-generic way to translate schemas between specific pairs of data models. In particular, the most supported scenario in available tools is the translation of ER diagrams into relational schemas.

Several proposals for the general problem have also been proposed in the last years [1, 21, 24]. All the frameworks share the same approach: (i) the system rewrites source schema S into a representation S' for a universal metamodel; (ii) a sequence of rule-based transformations modifies S' to translate or eliminate the constructs that are not allowed in the target data model; (iii) after n transformations, S' can be rewritten into the representation for the target data model. It is evident that in this translation process a crucial role is done by the rule-based transformations. Surprisingly all the frameworks lack a tool for design of such transformations at the data model level. 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 the design and the automatic generation of transformations over schemas.

It also important to locate schema exchange in the context of data exchange. Our work is largely inspired by the theoretical foundations explored by Fagin *et al* in the last years [12, 14]. We remark that schema exchange is the first proposal to extend their results to metadata and to introduce the novel notion of encoding of schemas and tgds. One of the most important practical contributions in data exchange are the algorithms for the generation of the tgds representing the schema mappings [13, 22]. Those algorithms take as input the schemas (with their constraints) and the arrows between the elements of the schemas (named *correspondences* in literature). We point out that in many practical settings the target schema does not exist, and it must be designed from scratch with a manual process before designing the mapping. This means that the user must deal with at least two manual steps: (i) the definition of the target schema and (ii) the definition of the correspondences between the elements. Many attempts have been done to automate the generation of the target schema [1, 8, 18, 20, 21, 24] and the identification of the correspondences between different schemas (the *schema matching* problem, see [25] for a survey). Proposed solutions gave important but partial contributions: in general settings, with existing tools it is not possible to avoid the manual time-consuming work we highlighted. Our approach effectively tackles this problem. As we have shown, in many cases it is possible to define generic transformations over templates of schemas and the use of these transformations can support the activity described above.

8 Conclusion and Future work

We have introduced the schema exchange problem, a generalization of data exchange to sets of schemas with similar structure. 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 are currently developing a prototype to generate semi-automatically dependencies over templates, given two template definitions and a set of correspondences between their components. The prototype presents some features that are direct applications of the results shown in this paper. Given a set of arrows between the template elements displayed in a Graphical User Interface, the user can easily define design patterns over classes of schemas. At design time, the system generates the generic mappings formulas that express the transformation to be applied on the database schemas conforming to the class defined by the source template. Finally, to be used on the databases at hand, such mappings are automatically rewritten as executable SQL scripts over the DBMS catalog. We remark that such prototype can be used to assist users in the design of transformations at the data model level, as required for the ModelGen implementations in literature [4].

We believe that other interesting directions of research can be pursued within the schema exchange settings. We sketch some of them.

- *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 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, or we could generate a target schema with a set of attributes identified by some data in the source. It is evident that, adding support to data and metadata in the same dependency, is a required step to extend the class of data exchange settings that can be generated in the $\mathsf{S}-\mathsf{D}$ transformation process.

- *Reuse of existing data exchange.* We have mentioned that the framework can be used to study the reuse of a data exchange setting. We are developing algorithms to generalize into a generic mapping data exchange mappings given by the user. The idea is to extract the design pattern for each data exchange scenario, and possibly find common patterns when the given scenarios are two or more. Once a transformation is expressed in a schema exchange setting, it can be later used to derive a data exchange transformation similar to the original one for a different pair of schemas.

- *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 classes.

References

[1] P. Atzeni, P. Cappellari, P. A. Bernstein. Model-independent schema and data translation. In *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 *CIDR*, pages 209–220, 2003.

[4] P. A. Bernstein and S. Melnik. Model Management 2.0: Manipulating Richer Mappings. In *SIGMOD*, pages 1–12, 2007.

[5] 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.

[6] P. A. Bernstein and E. Rahm. Data Warehouse Scenarios for Model Management. In *ER*, pages 1–15, 2000.

[7] D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya. An Annotation Management System for Relational Databases. In *VLDB*, pages 900-911, 2004.

[8] S. Bowers, L. M. L. Delcambre. The uni-level description: A uniform framework for representing information in multiple data models. In *ER*, pages 45–58, 2003.

[9] P. Buneman, S. Khanna, and W. C. Tan. Why and Where: A Characterization of Data Provenance. In *ICDT*, pages 316–330, 2001.

[10] P. Buneman, S. Khanna, and W. C. Tan. On Propagation of Deletion and Annotations Through Views. In *PODS*, pages 150–158, 2002.

[11] L. Chiticariu, W. C. Tan Debugging Schema Mappings with Routes. In *VLDB*, pages 79–90, 2006.

[12] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.

[13] A. Fuxman, M. A. Hernández, H. Ho, R. J. Miller, P. Papotti, and L. Popa. Nested Mappings: Schema Mapping Reloaded. In *VLDB*, pages 67–78, 2006.

[14] 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.

[15] F. Geerts, A. Kementsietsidis, and D. Milano. MONDRIAN: Annotating and querying databases through colors and blocks. In *ICDE*, pages 82–93, 2006.

[16] G.Gottlob and A. Nash. Data Exchange: Computing Cores in Polynomial Time. In *PODS*, pages 40–49, 2006.

[17] 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.

[18] P. McBrien and A. Poulovassilis. Data Integration by Bi-Directional Schema Transformation Rules. In *ICDE*, pages 227–238, 2003.

[19] G. Mihaila, L. Raschid, and M.-E. Vidal. Querying "quality of data" metadata. In *IEEE META-DATA*, 1999.

[20] P. Papotti and R. Torlone. Heterogeneous Data Translation through XML Conversion. *J. Web Eng.*, 4(3):189–204, 2005.

[21] P. Papotti and R. Torlone. Automatic Generation of Model Translations. In *CAiSE*, pages 36–50, 2007.

[22] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.

[23] C. M. Wyss and E. Robertson. Relational languages for metadata integration. *TODS*, 30(2):624-660, 2005.

[24] P. Mork, P. A. Bernstein, S. Melnik. Teaching a Schema Translator to Produce O/R Views. In *ER*, pages 102–119, 2007.

[25] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.