

Reusing Relational Sources for Semantic Information Access

Lina Lubyte
Free University of Bozen-Bolzano
Faculty of Computer Science
Piazza Domenicani 3, I-39100
Bozen-Bolzano, Italy
lubyte@inf.unibz.it

ABSTRACT

The rapid growth of available data arises the need for more sophisticated techniques for semantic access to information. It has been proved that using conceptual model or ontology over relational data sources is necessary to overcome many problems related with accessing the structured data. However, the task of wrapping the data residing in a database by means of an ontology is mainly done manually.

The research we are carrying out studies the reuse of relational sources in the context of semantics-based access to information. This problem is tackled in two phases: (i) extracting semantics hidden in the relational sources by wrapping them by means of an ontology, (ii) understanding the methodology for semantic extension of such ontologies. In this paper we focus on the first sub-problem and present an automatic procedure for extracting from a relational database schema its conceptual view. The semantic mapping between the database and its conceptualization is captured by associating views over the data source to elements of the extracted ontology. To represent the extracted conceptual model we use an ontology language, rather than a graphical notation, in order to provide precise formal semantics. Our approach uses heuristics based on ideas of standard relational schema design and normalization. Under this we formally prove that our technique preserves the semantics of constraints in the database. Therefore, there is no data loss, and the extracted model constitutes a faithful wrapper of the relational database.

Categories and Subject Descriptors

I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods—*representation languages*; H.2 [Database Management]: Miscellaneous

General Terms

Algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PIKM'07, November 9, 2007, Lisboa, Portugal.

Copyright 2007 ACM 978-1-59593-832-9/07/0011 ...\$5.00.

1. INTRODUCTION

The use of a conceptual model (or an ontology) to describe relational data sources has been shown to be extremely useful to overcome many important data access problems. In particular, the notion of accessing information through the navigation of an ontology modeling the domain of an information system – which can be seen as a conceptual schema – only recently has gained interest in knowledge representation research. The ontology defines a vocabulary which is richer than the logical schema of the underlying data. Thus, the user would prefer to query the database using the rich vocabulary of the ontology. Moreover, this allows users (or applications) to formulate queries also in the case of schema mismatch, since the information system is accessed using the common and agreed-upon ontology vocabulary [17]. On the other hand, so-called *intentional navigation* can help a less skilled user during the initial step of query formulation, thus overcoming problems related with schema comprehension and so enabling him/her to easily formulate meaningful queries (e.g., Query Tool [14]).

To date, the task of wrapping relational data sources by means of an ontology is mainly done manually. However, large quantity of structured information sources available on the Web, as well as multiple databases in enterprises require automatic support for the conceptualization of the domain. Within this research area, we identify two key tasks that we discuss next.

Scenario 1. Suppose the goal is to design an ontology that is to be used as a unifying view in enterprise integration. Typically, there will be a number of database systems that are used in the enterprise to be modeled. Instead of constructing the ontology from scratch, ontology designer wants an automatic support for converting the database schema into an initial ontology (i.e., support for ontology bootstrapping). Once the core ontology is obtained from database schema, a user poses a query over the extracted ontology which is to be answered by using the data in the database.

Task 1. Automatic support for deriving an ontology from an existing database schema, yet capturing the semantic mapping between the database and the extracted ontology and guaranteeing that no information is lost.

As a first part of this doctoral work, we aim at defining a framework for automatically extracting from a relational database schema an ontology that is to be used as a unifying view over the data and is specifically tailored for information

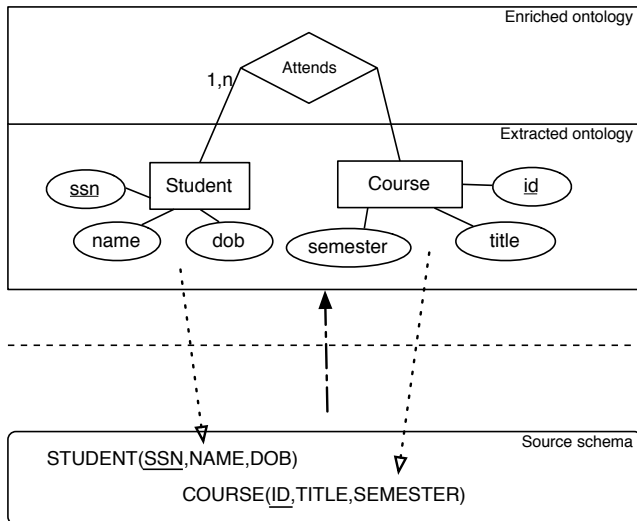


Figure 1: An example scenario

access. In such setting, the semantic mapping between the database and its conceptualization is nevertheless a crucial aspect, in order for queries over the ontology to be answered by using the data in the database. In addition, there must be no data loss while deriving ontology from the database schema, i.e. correctness must be preserved.

Scenario 2. Once a core ontology is obtained, ontology designer adds more details to a part of the ontology that has not yet been sufficiently described. An end user then uses the extended ontology for formulating his/her queries and an information system is still expected to return the answer (obviously using sophisticated query answering techniques, such as query rewriting [12]).

Task 2. Support for controlling the consequences of ontology extension.

When the extracted ontology is extended, it is important for an ontology engineer to be aware of the consequences of such modifications. In particular, when the extended ontology is used for querying the information residing in the database, it may happen that unintended answers are returned. Let us elaborate the matter with a specific example depicted in Figure 1. Please note that in this example the derived ontology is represented with Description Logic (DL) *DL-Lite* [11]. We use Entity-Relationship (ER) notation for a clear pictorial representation, and we do not discuss in this paper how common ER constructs can be expressed with *DL-Lite* (see [11] for details). At the lower level of this figure (below the dashed line) we have an input database schema, providing the actual data, whereas the bottom part of the upper level represents the ontology generated from the input database schema (Scenario 1). Assume we have a database schema with two relations *STUDENT* and *COURSE* (keys are underlined). Since no foreign keys are present in none of the two relations, with the ontology extraction procedure we obtain two corresponding entities, *Student* and *Course*, not related to each other via any relationship. The associated mappings are depicted with dashed arrows. When an

ontology engineer sees the derived ontology, he/she decides that, clearly, a relationship, say *Attends*, between the two entities exists. Now, suppose a user, looking at already extended ontology (the bottom part under the dashed line is hidden from the user), wants to know the pairs of students and courses that are being attended, i.e., he/she poses a query

$$q(x, y) \leftarrow \text{Attends}(x, y).$$

It is immediate to see that the evaluation of the above query gives an empty answer, since during query reformulation step [12] none of the inclusion dependencies will be applicable to an atom *Attends*. However, consider the situation when an ontology engineer adds mandatory participation constraint for the entity *Student*, stating that every student must attend at least one course. Then, the answer to the query

$$q(x) \leftarrow \text{Attends}(x, -)$$

is the set of all students.

To avoid unexpected results in such applications when using the extended ontology, the ontology engineer must have control of the consequences of his/her modifications. Thus, as a second part of this work, we aim at characterizing such a phenomenon. In other words, the goal is to understand the methodology for semantic extension of ontologies, i.e., in which cases such extensions are meaningful and give sensible results. Currently, there are no guidelines of how to avoid or solve the latter situations. Therefore, we are concerned with whether we can give guidelines for knowledge engineers such that this problem could be minimized.

Summing up, the focus of the current research is re-use of relational sources in the context of semantics-based access to information. Our goal is to understand which is the best way of building such an infrastructure by providing a uniform access to the data sources in a coherent and user-oriented way. We see the problem in two phases: eliciting the semantics of data hidden in the structure of relational sources by wrapping them by means of an ontology; and understanding the methodology for semantic extension of such ontology, i.e., when those extensions are meaningful and give sensible results.

We will discuss more on each of these two challenges in the following sections. Please note that the problem of extracting ontologies from relational data sources has been carried out from the beginning of the doctoral work, thus, in this paper we concentrate on the former. The second problem – that of semantic extension of extracted ontologies – is at the initial stage, and we only provide initial insights for approaching the matter. The rest of the paper is structured as follows. Section 2 surveys the work related to our research. In Section 3 we describe the approach proposed and results achieved so far. We conclude in Section 4.

2. RELATED WORK

The primary task of our work is extracting from a relational database schema its corresponding conceptual view. In this process we integrate aspects of *database reverse engineering* (DBRE) [18]. DBRE is defined as a process of recovering a conceptual model that represents the meaning of the logical schema by examining an existing database system to identify the database contents and their interrelationships. Approaches to recovering a conceptual schema from a relational database have appeared in the literature over the

years [26, 15, 5, 2, 6]. Four main sources have been explored for finding evidence to construct a conceptual schema from a logical database: the structures and integrity constraints of the database schema, the application programs that access the database, the data instances stored in the database, and the users and designers. In particular, [26] use schema structures and constraints by considering an input relational database being in BCNF. In Andersson’s work in [5] information about functional dependencies, keys and inclusion dependencies are deduced by looking into data manipulation statements that can be extracted from the application code. The approaches in [15, 6] analyze not only the database schema, but also data instances, while the work in [2] relies only on data instances in order to identify candidate keys, to locate foreign keys and to decide on the appropriate links between the given relations. Finally, the method of [19] decides the correct object types in the conceptual schemas by interacting with users. Unlike our work, DBRE approaches firstly produce just a pictorial representation of a conceptual model, without showing how it links to the database schema, and are thus used for “documenting” the database. Our approach, instead, is tailored for the direct use of the extracted ontology – accessing information mediated by the ontology. Secondly, most of DBRE methods are informal and do not specify the quality of the results. On the contrary, we provide formal results showing that our extraction technique is correct.

The recent call for a Semantic Web arose several approaches in bringing together relational databases and ontologies. Among them we mention [4], where the authors describe an automatic mapping between relations and ontologies, when given as input simple correspondences from attributes of relations to datatype properties of classes in an ontology. Unlike our approach, it requires a target existing ontology onto which the relations are mapped to. On the other hand, the approach of [21] extracts the schema information of the data source and converts it into an ontology. However, the latter technique extracts only the structural information about the ontology, so the constraints are not taken into account.

To obtain meaningful inter-operation between the data sources and extracted multiple ontologies, *semantic mappings* are the crucial aspect. Thus, schema and ontology mapping and matching is a relevant research topic. The problem of semi-automatically creating mappings has received significant attention recently in both the database and AI communities (see [29, 30]). Most of the approaches for discovering mappings focused on database schemas, e.g. Clio [27] and HePToX [10]. The primary principle for deriving mappings is using (referential) integrity constraints in a schema to assemble “logically connected elements”. These logical associations, together with the element correspondences, give rise to mappings between schemas. An interesting solution for discovering mappings semi-automatically is reported in [3]. It first derives semantic mappings from database schemas to ontologies, and then takes this advantage of the knowledge to produce a set of logical formulas representing mappings between different database schemas. In addition to tools for schema mapping and matching, a considerable body of work exists for ontology mapping and matching [20, 30].

Finally, let us note that the problem of accessing information through extended ontologies has not been tackled in the literature.

3. OUR SOLUTION

In this section we address several key technical issues and indicate solutions for the first phase of our research question, namely, for extracting semantics from relational sources.

3.1 Ontology Extraction

3.1.1 Formal Framework

We introduce the formal framework for representing the ontology and its relation with a relational data source. We adopt a standard relational model together with integrity constraints, detailed below. The ontology language adopted can be seen as an alternative to the use of standard modeling paradigms of Entity-Relationship or UML diagrams and enables the representation of their commonly used modeling constructs (see [9]). The advantage over these formalisms lies on the fact that the ontology language has a clear and unambiguous semantics which enables the use of automatic reasoning to support the designer.

Relational model We assume that the reader is familiar with standard relational database notions as presented, for example, in [1]. We assume that the database domain is a fixed denumerable set of elements Δ and that every such element is denoted uniquely by a constant symbol, called its *standard name* [24]. We make use of the standard notion of relational model by using named attributes, each with an associated datatype, instead of tuples.

A *relational schema* \mathcal{R} is a set of relations, each one with a fixed set of attributes (assumed to be pairwise distinct) with associated datatypes. We use $[s_1 : D_1, \dots, s_n : D_n]$ to denote that a relation has attributes s_1, \dots, s_n with associated data types D_1, \dots, D_n . We interpret relations over a fixed countable *domain* Δ of datatype elements, which we consider partitioned into the datatypes D_i . A *database instance* (or simply *database*) \mathcal{D} over a relational schema \mathcal{R} is an (interpretation) function that maps each relation R in \mathcal{R} into a set $R^{\mathcal{D}}$ of total functions from the set of attributes of R to Δ . Let $A = [s_1, \dots, s_m]$ be a sequence of m attribute names of a relation R of a schema \mathcal{R} . The *projection* of $R^{\mathcal{D}}$ over A is the relation $\pi_A R^{\mathcal{D}} \subseteq \Delta^m$, satisfying the condition that $\phi \in \pi_A R^{\mathcal{D}}$ iff $(\phi(s_1), \dots, \phi(s_m)) \in R^{\mathcal{D}}$.

The ontology extraction task takes as input a relational source. We abstract from any specific database implementation by considering an abstract *relational source* \mathcal{DB} , which is a pair (\mathcal{R}, Σ) , where \mathcal{R} is a relational schema and Σ is a set of *integrity constraints*. The semantics of relational schemata is provided in the usual way by means of the relational model. Given a relation r , let \mathbf{A} denote sequence of attributes of r . Below we briefly list the kind of database integrity constraints we consider in our framework (for more details the reader is referred to [25]).

- *Nulls-not-allowed constraints*: $\text{nonnull}(r, \mathbf{A})$, satisfied in a database when null values are not contained in any attribute of \mathbf{A} .
- *Unique constraints*: $\text{unique}(r, \mathbf{A})$, satisfied when the sequence of attributes \mathbf{A} are unique in a relation. Together with nulls-not-allowed constraints they correspond to *key constraints*, denoted $\text{key}(r, \mathbf{A})$.
- *Inclusion dependencies*: $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$, satisfied when projections over $\mathbf{A}_1, \mathbf{A}_2$ of relations r_1 and r_2 are included one in the other. If $\text{key}(r_2, \mathbf{A}_2)$, we call them *foreign key constraints*.

$R[c] \sqsubseteq R'[c'] \pi_c R^D \subseteq \pi_{c'} R'^D$	Subclass
$R[c] \text{ disj } R'[c'] \pi_c R^D \cap \pi_{c'} R'^D = \emptyset$	Disjointness
$\text{funct}(R[c])$ for all $\phi_1, \phi_2 \in R^D$ with $\phi_1 \neq \phi_2$, we have $\phi_1(s) \neq \phi_2(s)$ for some s in c	Functionality
$R_1[c_1], \dots, R_k[c_k] \text{ cover } R[c] \pi_c R^D \subseteq \bigcup_{i=1 \dots k} \pi_{c_i} R_i^D$	Covering

Figure 2: Syntax and semantics of *DLR-DB* axioms

- *Exclusion dependencies*: $(r_1[\mathbf{A}_1] \cap r_2[\mathbf{A}_2]) = \emptyset$, satisfied when the intersection of the projections over $\mathbf{A}_1, \mathbf{A}_2$ of relations r_1, r_2 is the empty set.
- *Covering constraints*:¹ $(r_1[\mathbf{A}_1] \cup \dots \cup r_m[\mathbf{A}_m]) \subseteq r_0[\mathbf{A}_0]$, satisfied when the projection of the relation r_0 over \mathbf{A}_0 is included in the union of the projections of the relations in the set.

Ontology language We call a *DLR-DB* system \mathcal{S} a triple $\langle \mathcal{R}, \mathcal{P}, \mathcal{K} \rangle$, where \mathcal{R} is a *relational schema*, \mathcal{P} is a *component structure* over \mathcal{R} , and \mathcal{K} is a set of assertions involving names in \mathcal{R} . The intuition behind a named component is the role name of a relationship in an ER schema (or UML class diagram). The *component structure* \mathcal{P} associates to each relation a mapping from *named components* to sequences of attributes. Let R be a relation in \mathcal{R} , to ease the notation we write \mathcal{P}_R instead of $\mathcal{P}(R)$.

Let R be a relation in \mathcal{R} , with attributes $[s_1 : D_1, \dots, s_n : D_n]$. \mathcal{P}_R is a non-empty (partial) function from a set of named components to the set of nonempty sequences of attributes of R . The domain of \mathcal{P}_R , denoted \mathcal{C}_R , is called the set of *components of R* . For a named component $c \in \mathcal{C}_R$, the sequence $\mathcal{P}_R(c) = [s_{i_1}, \dots, s_{i_m}]$, where each $i_j \in \{1, \dots, n\}$, is called the *c -component* of R . We require that the sequences of attributes for two different named components are not overlapping, and that each attribute appears at most once in each sequence. The *signature* of a component $\mathcal{P}_R(c)$, denoted $\tau(\mathcal{P}_R(c))$, is the sequence of types of the attributes of the component. Two components $\mathcal{P}_R(c_1)$ and $\mathcal{P}_R(c_2)$ are *compatible* if the two signatures $\tau(\mathcal{P}_R(c_1))$ and $\tau(\mathcal{P}_R(c_2))$ are equal.

The *DLR-DB* ontology language, used to express the assertions in \mathcal{K} , is based on the idea of modeling the domain by means of *axioms* involving the projection of the relation over the named components. An *atomic formula* is a projection of a relationship R over one of its components. The projection of R over the c -component is denoted by $R[c]$. When the relation has a single component, then this can be omitted and the atomic formula R corresponds to its projection over the single component. Given the atomic formulae $R[c], R'[c'], R_i[c_i]$, an *axiom* is an assertion of the form specified in Figure 2, where all the atomic formulae involved in the same axiom must be compatible. In the same figure, there is the semantics of a *DLR-DB* system which is provided in terms of relational models for \mathcal{R} , where \mathcal{K} plays the role of constraining the set of “admissible” models.

A database \mathcal{D} is said to be a *model* for \mathcal{K} if it satisfies all its axioms, and for each relationship R in \mathcal{R} with components

c_1, \dots, c_k , for any $\phi_1, \phi_2 \in R^D$ with $\phi_1 \neq \phi_2$, there is some s in c_i s.t. $\phi_1(s) \neq \phi_2(s)$. The above conditions are well defined because we assumed the compatibility of the atomic formulae involved in the constraints.

The *DLR-DB* ontology language enables the use of the most commonly used constructs in conceptual modeling (see [9] for details).

3.1.2 Principles of ontology extraction

The principles upon our technique are based on ideas of standard database design and normalization. Note that similar techniques are proposed in DBRE literature [18, 15] for recovering conceptual model from a relational database. We discussed how our work advances the DBRE approaches in Section 2. First, we assume that the relational source in third normal form (3NF). Second, in order to uncover the connections between relational schema and the conceptual model, the heuristics underlying the ontology extraction process rely on best practices of relational schema design from ER diagrams (or UML class diagrams) – a standard database modeling technique [16, 8]. One benefit of this approach is that it can be shown that our algorithm, though heuristic in general, is able to reconstruct the original ER diagram under some assumptions on the latter. Specifically, we consider ER models that support entities with attributes,² n -ary relationships which are subject to cardinality constraints, and inheritance hierarchies (IS-A) between entities (including multiple inheritance) which may be constrained to be disjoint or covering. Under these two assumptions we can formally prove that our extraction procedure preserves semantics of constraints in the relational database (detailed in Section 3.1.4).

As noted before, in the settings were data residing in relational database is accessed with the mediation of an ontology, semantic mappings are the crucial aspect. In our framework, this is achieved by associating (materialized) views over the data source to the elements of the extracted ontology. In this way, queries over the ontology can be answered by using data in the database.

Our proposed ontology extraction algorithm works in two phases. Firstly, a classification scheme for relations from the relational source is derived. Secondly, based on this classification, the ontology describing the data source is extracted. In addition, the process generates a set of view definitions, so connecting the database schema with the ontology. Thus, given relational source $\mathcal{DB} = (\Psi, \Sigma)$, our task is to extract from it the ontology in terms of axioms in \mathcal{K} of *DLR-DB* system, together with a set of views in *DLR-DB* \mathcal{R} , defined over the actual data (i.e., relational source \mathcal{DB}). In such

¹In ER terminology, this may also be indicated as *mandatory* for an IS-A relationship.

²We do not deal with multi-valued attributes in this paper.

setting, \mathcal{R} is seen as a new schema containing set of materialized views, obtained by transforming the source schema \mathcal{DB} .

Roughly speaking, we reverse the process of translating ER model to relational model. As a result, we identify that relations representing entities in ER schema have keys which are not part of their foreign keys, and every such foreign key represents functional binary relationship (i.e., one-to-one or one-to-many) with another entity. On the other hand, relations that correspond to n -ary relationships with cardinalities “many” for all participating entities have keys composed of their foreign keys. Since we do not allow inheritance between relationships, every such foreign key references key of a relation resulting from (sub-)entity. When a relation has key that is also foreign key, and no other foreign keys appear in that relation, then an inheritance relationship exists. If instead non-key foreign keys are present but the relation is the target of some foreign key, we are sure that this relation corresponds to sub-entity. Otherwise, such relation might also “look like” functional relationship (binary or n -ary), mapped directly to a relation. Relations of this type and all other relations not satisfying the conditions described above (e.g., in case of design errors) are classified as ambiguous relations (see below). This kind of reasoning gives rise to the classification of relations, which is the key concept of our extraction procedure.

3.1.3 Ontology Extraction Procedure

In this section we briefly sketch the actual ontology extraction procedure, more details and the algorithms can be found in [25].

First of all, relations are classified based on the appearance of their keys and foreign keys:

- we classify a relation r_i having key disjoint with every foreign key of r_i (if any) as *base relation*;
- a relation r_i having key which is also a foreign key and if r_i satisfies *one* of the following conditions
 - r_i has a single foreign key, or
 - r_i is referred to by some relation, i.e., r_i appears on the right-hand side of some foreign key constraint,

then r_i is classified as *specific relation*;

- a relation r_i having key entirely composed of foreign keys of r_i , and the number of foreign keys is greater than one, is classified as *relationship relation*;
- a relation r_i not satisfying any of the conditions above is classified as *ambiguous relation*.

Once relations in \mathcal{DB} are classified according to the conditions above, then the actual ontology extraction process returns a *DLR-DB* system as output. Given relation r_i , let A_i and FK_i denote the sequence of all attributes and sequence of all foreign key attributes of r_i . Table 1 provides summary of how components of *DLR-DB* system are generated for each class of relations. Specifically, for every base and specific relation r_i the algorithm generates a relation R_i by projecting on all non-foreign key attributes of r_i . R_i has a single c -component, where $\mathcal{P}_{R_i}(c)$ corresponds to the key attributes of r_i and thus functionality axiom $\text{func}(R_i[c])$ is added to \mathcal{K} .

A non-key foreign key in a base or specific relation r_i , with a referenced relation r_j , determines the association between relations R_i and R_j . Thus, for each such foreign key, relation R_k is generated by joining r_j with r_i and projecting on their keys. R_k has two components, c_i -component and c_j -component, where $\mathcal{P}_{R_k}(c_i)$ and $\mathcal{P}_{R_k}(c_j)$ correspond to key attributes of r_i and r_j , respectively. Since foreign key of r_i encodes a one-to-one or one-to-many relationship, c_i -component is functional, thus we have $\text{func}(R_k[c_i])$ in \mathcal{K} . For expressing an association between R_i and R_j the axioms of the form $R_k[c_i] \sqsubseteq R_i$ and $R_k[c_j] \sqsubseteq R_j$ are added to \mathcal{K} . Furthermore, whenever the latter foreign key of r_i participates in a nulls-not-allowed constraint, the axiom $R_i \sqsubseteq R_k[c_i]$ is generated stating mandatory participation for instances of R_i to R_k as values for the c_i -component; its participation to a unique constraint determines instead the functionality axiom $\text{func}(R_k[c_j])$ meaning that every value of the c_j -component appears in it at most once; finally, appearance of the foreign key of r_i in the right-hand side of an inclusion dependency determines mandatory participation for values of the only component of R_j to the relation R_k as values for the c_j -component, and thus the axiom $R_j \sqsubseteq R_k[c_j]$ is added to \mathcal{K} . For expressing an ISA between classes, for every specific relation r_i the subclass axiom $R_i \sqsubseteq R_j$ is added to \mathcal{K} , where R_j is the relation corresponding to (base or specific) relation r_j , that the foreign key which is also a key of r_i references. Additionally, each exclusion dependency on the set of specific relations induces the disjointness axioms $R_i \text{ disj } R_k$, for every pair of relations r_i, r_k appearing in the exclusion dependency. Similarly, every covering constraint on the set of specific relations induces the corresponding covering axiom in \mathcal{K} .

Similar reasoning accounts for relationship relations, as can be seen in Table 1. As for ambiguous relations, our heuristics “prefers” to recover a relationship, and thus the algorithm generates the structures corresponding to those defined for relationship relations. On the other hand, a user could decide which is the “best” structure for ambiguous relations. In this way, the ontology extraction task may be a completely automated procedure, or semi-automated process with a user intervention.

3.1.4 Correctness of the Technique

Our proposed ontology extraction technique can be seen as *schema transformation* as defined in [28]. An important consideration in such a process (i.e., when transforming one data model into another) is the potential for loss of information. We evaluate the correctness of our schema extraction procedure using the relative *information capacities* of the source and target schemata. In this section we briefly outline the main principles of this analysis; for full details and the actual proofs the reader is referred to [25].

In the following we denote by \mathcal{S} and \mathcal{T} source and target schemata corresponding to the input relational source \mathcal{DB} and relational schema \mathcal{R} of the extracted *DLR-DB* system. Let $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$ be consistent instances of schemata \mathcal{S} and \mathcal{T} , respectively. An *equivalence preserving mapping* between the instances of \mathcal{S} and \mathcal{T} is a bijection $\mu : \mathcal{D}_\mathcal{S} \rightarrow \mathcal{D}_\mathcal{T}$. Then \mathcal{S} and \mathcal{T} are said to be *equivalent* via μ , denoted $\mathcal{S} \equiv \mathcal{T}$. Given schemas \mathcal{S} and \mathcal{T} , a (*schema*) *transformation* is a total function $\mathcal{M} : \mathcal{S} \rightarrow \mathcal{T}$. \mathcal{M} is an *equivalence preserving transformation* if it induces an equivalence preserving mapping. To this end, we come up with the following result:

Relation type	\mathcal{R} & \mathcal{P}	Axiom in \mathcal{K}
base relation r_i $key(r_i, K_i)$	$R_i = \pi_{A_i - FK_i}(r_i)$ $\mathcal{P}_{R_i}(c_i) = K_i$	$func(R_i[c_i])$
specific relation r_i $key(r_i, K_i)$ $r_i[K_i] \subseteq r_j[K_j]$	$R_i = \pi_{A_i - FK_i}(r_i)$, s.t. $FK_i \cap K_i = \emptyset$ $\mathcal{P}_{R_i}(c_i) = K_i$	$func(R_i[c_i])$ $R_i \sqsubseteq R_j$
$r_{i_1}[K_{i_1}] \cap r_{i_2}[K_{i_2}] = \emptyset$ $r_j[K_j] \subseteq r_{i_1}[K_{i_1}] \cup \dots \cup r_{i_m}[K_{i_m}]$		$R_{i_1} \text{ disj } R_{i_2}$ $R_{i_1}[c_{i_1}], \dots, R_{i_m}[c_{i_m}] \text{ cover } R_j[c_j]$
base or specific relation r_i $r_i[FK_{i_j}] \subseteq r_j[K_j]$, for $j = 1, \dots, n$, n -number of f.k.s and $FK_{i_j} \neq K_i$	$R_k = \pi_{K_i, K_j}(r_j \bowtie r_i)$ $\mathcal{P}_{R_k}(c_i) = K_i; \mathcal{P}_{R_k}(c_j) = K_j$	$func(R_k[c_i])$ $R_k[c_i] \sqsubseteq R_i; R_k[c_j] \sqsubseteq R_j$
$nonnull(r_i, FK_{i_j})$ $unique(r_i, FK_{i_j})$ $r_j[K_j] \subseteq r_i[FK_{i_j}]$		$R_i \sqsubseteq R_k[c_i]$ $func(R_k[c_j])$ $R_j \sqsubseteq R_k[c_j]$
relationship relation r_i $r_i[FK_{i_j}] \subseteq r_j[K_j]$, for $j = 1, \dots, n$	$R_i = \pi_{A_i}(r_i)$ $\mathcal{P}_{R_i}(c_{i_j}) = FK_{i_j}$	$R_i[c_{i_j}] \sqsubseteq R_j$
$r_j[K_j] \subseteq r_i[FK_{i_j}]$		$R_j \sqsubseteq R_i[c_{i_j}]$
ambiguous relation r_i	repeat steps as for relationship relations	

Table 1: Summary of the extraction procedure

THEOREM 1. *The ontology extraction procedure is an equivalence preserving schema transformation.*

The actual proof of the above theorem can be found in [25]. Roughly speaking, we devise a bijective transformation for the respective models and we show that the constraints of the original schema and extracted ontology are satisfied by the models generated by this transformation.

The fact that our extraction procedure is equivalence preserving not only shows that there is no information loss, so the extracted schema can be used to access the data, but that we can evaluate queries expressed using the extracted ontology by simply expanding the generated views. This is no longer true in the case when the ontology is going to be modified. Then more sophisticated query answering techniques must be adopted in order to guarantee completeness (e.g. query rewriting [12]).

3.1.5 Experimentation

So far, we have developed the ontology extraction algorithm by investigating the constraints on the database schema. We have also shown that the extraction procedure is equivalence preserving for schemas designed following the principled methodology. Nonetheless, it is crucial to test the algorithm in real-world schemas in order to evaluate its adequacy. To do this, we have implemented our approach in a prototype system. Given a relational schema as input, our tool produces as output the ontology described in XML format, which is then graphically represented (using UML-like notation) with ICOM³ – Ontology Design tool – able to represent all constructs in our ontology language. The results of initial experiments we carried out are shown in this section.

The database schema used in this experiment implements the CERIF (Common European Research Information Format) 2000 Full Data Model. The ORACLE SQL script⁴ was taken. The schema is strongly structured as it stores 123 tables. The analysis of schema metadata was performed

³<http://www.inf.unibz.it/~franconi/icom>

⁴<http://cordis.europa.eu/cerif/src/toolkit.htm>

by our tool⁵, the classification on relations was derived, and the ontology extracted. Table 2 reports the summary of classified relations and extracted ontology components.

We observed that the tool produced correct ontology constructs for 77 tables (out of 123). We were particularly interested in the usefulness of our approach for ambiguous relations generated by our tool. We have identified that all 46 tables classified as ambiguous can be categorized into two types. That is, (i) those having foreign key properly included in its key, and (ii) those having key properly included in the set of its foreign keys, where the number of foreign keys is at least 3 and key spans at least 2 foreign keys. Relations of type (i) represent multivalued attributes in ER diagram. We have decided to treat them as having “hidden” foreign key, and to recover a many-to-many relationship. For example, the relation

PERSON_RESEARCH_INTEREST(LANG, TRANS_TYPE, PER_ID, KEYWORDS),

with primary key underlined, where PER_ID is a foreign key referencing PERSON relation. In this case, relation in *DLR-DB* having two components with sequences of attributes, PER_ID and LANGUAGE, TRANS_TYPE, respectively, should be generated. Relations of type (ii) correspond to n -ary relationships in ER diagram. However, we cannot deal with this case, since we consider the sequence of attributes of all the components of a relation as a key for this relation.

3.2 Ontology Extension

As noted in the introductory section of this paper, currently we are starting to look at the problem of accessing the data sources using the extended ontology. We do not have at present any results showing what kind of ontology extensions are meaningful and how meaningful. Nevertheless, in this section we briefly report on initial ideas regarding the matter.

We have decided to begin with the analysis of how added new terms reflect the result of query answering by rewriting

⁵As for now, we assume that inclusion and exclusion dependencies are explicitly available in database metadata.

Relation type	#Classified	<i>DLR-DB</i> construct	#Extracted
Base	52	Relation with single component	52
Relationship	25	Relation with 2 components	36
Specific	0	Subclass axiom	68
Ambiguous	46	Functionality axiom	63

Table 2: Summary of experimentation with CERIF database schema

for the families of Description Logics (DLs) of our interest. That is, we are concerned with whether (and if yes, then in which cases) the query over the new term can be rewritten.

We will concentrate on *Definitorially Complete* DLs, which is known as *Beth’s Definability property* for First Order Logic but then stated for DLs. Following [31], defining new concepts can be done in two ways: (1) in an explicit syntactical manner as in `NewConcept` \equiv C with C an expression in which `NewConcept` does not occur; (2) implicitly, by writing a set of general inclusion axioms T with the property that in any model of T, the interpretation of `NewConcept` is uniquely determined by the interpretation of the primitive concepts and relations. Then, a DL is Definitorially Complete, if every new concept defined in the implicit way can also be defined in the explicit manner. It was shown in [7] that *ALC* DL is definitorially complete; in [31] the same was shown for a number of extensions of *ALC*.

4. DISCUSSION AND CONCLUSIONS

The growth of available data call for ever more sophisticated techniques to manage information, and in particular to semantic, user-oriented access to information. Database community has recently started to support the claim for the need of much better understanding of *users* and their “pain points” when they work with structured data (see A. Halevy’s comments on SIGMOD 2007⁶).

We start with those observations and discuss a vision for building infrastructure for semantic access to information. Specifically, we concentrate on the process of extracting ontologies from relational data sources. Our extraction procedure uses heuristics based on principled database design methodologies and integrates database reverse engineering techniques. Nevertheless, it advances the DBRE area in that the resulting extracted ontology with a set of views represents the *meaning* of the data and the *semantic mappings* between the database schema and its conceptualization. Moreover, we provide formal results showing that our technique is correct, i.e. there is no data loss, while DBRE approaches are generally informal.

There are several issues worth noting. First, in principle, it is possible to use database schema directly (i.e., without extracting ontology and then using it to access information), e.g. using the rewriting based technique [22]. However, our approach makes explicit the “semantics” of relations, i.e. detecting and differentiating entities and relationships, multi-valued attributes, etc., and thus helps to understand this existing data.

Second, our adopted *DLR-DB* ontology language is close to the *DLR* family of ontology languages (see [13]). In particular, by virtue of the assumption that components do not share attributes, it is not difficult to show that the same reasoning mechanism used for *DLR* can be employed with

DLR-DB. The ability of employing correct and complete automated reasoning enables us to provide well founded tools to support the maintenance and evolution of the extracted ontology (see [23]), and to support the final user in formulating precise queries using the ontology (see [14]). Moreover, by taking away the covering axioms,⁷ this language corresponds exactly to *DLR-Lite* [12]. This means that we can use the same efficient query answering technique to evaluate conjunctive queries mediated by the ontology.

5. REFERENCES

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. 1995.
- [2] R. Alhajj. Extracting an extended entity-relationship model from a legacy relational database. *Information Systems*, 26(6):597–618, 2003.
- [3] Y. An. *Discovering and Using Semantics for Database Schemas*. PhD thesis, University of Toronto, 2007.
- [4] Y. An, A. Borgida, and J. Mylopoulos. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *ODBASE’05*, pages 1152–1169, 2005.
- [5] M. Andersson. Extracting an entity-relationship schema from a relational database through reverse engineering. In *Proc. of Int. Conf. on Entity-Relationship Approach (ER’94)*, pages 403–419, 1994.
- [6] I. Astrova. Reverse engineering of relational databases to ontologies. In *ESWS*, pages 327–341, 2004.
- [7] F. Baader and W. Nutt. *The Description Logic Handbook: Theory, Implementation and Applications*, chapter Basic Description Logics. Cambridge University Press, 2003.
- [8] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual Database Design. An Entity-Relationship Approach*. Benjamin/Cummings Publishing Company, Inc., 1992.
- [9] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on uml class diagrams. *Artificial Intelligence*, 168(1):70–118, 2005.
- [10] A. Bonifati, E. Q. Chang, T. Ho, V. S. Lakshmanan, and R. Pottinger. Heptox: Marring xml and heterogeneity in your p2p databases. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, pages 1267–1270, 2005.
- [11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Dl-lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.
- [12] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query

⁶<http://alanhalevy.blogspot.com/search/label/sigmod>

⁷Which are seldom used in logical schemata.

- answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
- [13] D. Calvanese, G. De Giacomo, and M. Lenzerini. Identification constraints and functional dependencies in description logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 155–160, 2001.
- [14] T. Catarci, P. Dongilli, T. D. Mascio, E. Franconi, G. Santucci, and S. Tessaris. An ontology based visual tool for query formulation support. In *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, 2004.
- [15] R. H. L. Chiang, T. M. Barron, and V. C. Storey. Reverse engineering of relational databases: extraction of an eer model from a relational database. *Data and Knowledge Engineering*, 12(2):107–142, 1994.
- [16] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison Wesley Publ. Co., fourth edition, 2004.
- [17] E. Franconi, S. Tessaris, B. C. Grau, B. Suntisrivaraporn, C. Lutz, R. Moller, and D. Lembo. Revised ontology task handbook. Deliverable D07, TONES EU-IST STREP FP6-7603, August 2006.
- [18] J. L. Hainaut. Database reverse engineering: models, techniques and strategies. In *Proc. of the 10th Conference on ER Approach*, 1998.
- [19] P. Johannesson. A method for transforming relational schemas into conceptual schemas. In *Proc. of the Int. Conf. on Data Engineering (ICDE'94)*, pages 190–201, 1994.
- [20] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. *Journal on Data Semantics*, 1:98–127, 2003.
- [21] C. P. Laborda and S. Conrad. Bringing relational data into the semantic web using sparql and relational.owl. In *Proc. of the 22nd Int. Conf. on Data Engineering Workshops (ICDEW'06)*, pages 55–62, 2006.
- [22] D. Lembo, M. Lenzerini, and R. Rosati. Methods and techniques for query rewriting. Deliverable D5.2, INFOMIX IST-2001-33570, April 2004.
- [23] D. Lembo, C. Lutz, and B. Suntisrivaraporn. Tasks for ontology design and maintenance. Deliverable D05, TONES EU-IST STREP FP6-7603, May 2006.
- [24] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.
- [25] L. Lubyte and S. Tessaris. Extracting ontologies from relational databases. Technical report, KRDB group – Free University of Bozen-Bolzano, 2007. Available at <http://www.inf.unibz.it/krdp/pub/TR/KRDB07-4.pdf>.
- [26] V. M. Markowitz and J. A. Makowsky. Identifying extended entity-relationship object structures in relational schemas. *IEEE Transactions on Software Engineering*, 16(8):777–790, 1990.
- [27] R. J. Miller, L. Haas, and M. A. Hernandez. Schema mapping as query discovery. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, pages 77–88, 2000.
- [28] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *Proc. of VLDB'93*, pages 120–133, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.
- [29] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–360, 2001.
- [30] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics IV*, pages 146–171, 2005.
- [31] B. ten Cate, W. Conradie, M. Marx, and Y. Venema. Definitorially complete description logics. In *Proc. of KR 2006*, 2006.