KRDB Research Centre Technical Report:

Extracting Ontologies from Relational Databases

L. Lubyte, S. Tessaris

Affiliation	KRDB, Faculty of Computer Science, FUB Piazza Domenicani 3, 39100 Bolzano, Italy
Corresponding author	L. Lubyte lubyte@inf.unibz.it
Keywords	ontology extraction, conceptual modelling, normalization, semantic mapping
Number	KRDB07-04
Date	06-04-07
URL	http://www.inf.unibz.it/krdb/

Abstract

The use of a conceptual model (or an ontology) to describe relational data sources has been proved to be extremely useful to overcome many important data access problems. However, the task of wrapping relational data sources by means of an ontology is mainly done manually. In this paper we introduce an automatic procedure for extracting a conceptual view from a relational database. The semantic mapping between the database schema and its conceptualisation is captured by associating views over the data source to elements of the extracted conceptual model. To represent the conceptual model we use an ontology language, rather that a graphical notation, in order to provide a precise formal semantics. In particular we adopt a variant of the *DLR-Lite* description logic because of its nice computational properties, and ability to express the mostly used modelling constraints.

In order to uncover the connections between relational schema and the conceptual model, the heuristics underlying the ontology extraction process are based on ideas of standard relational schema design and normalisation. In fact, we assume that the relational source is in third normal form, and designed by means of a principled methodology; e.g. using Entity-Relationship or UML diagrams.

Contents

Chapter 1

Introduction

The use of a conceptual model or an ontology over data sources has been shown to be necessary to overcome many important database problems. These include federated databases [18], data warehousing [7], information integration through mediated schemas [14], and the Semantic Web [11] (for a survey see [19]). Since ontologies provide a conceptual view of the application domain, the recent trend to employ such ontologies for navigational (and reasoning) purposes when accessing the data gives additional motivation for the problem of extracting the ontology from database schema [13]. When such an ontology exists, modelling the relation between the data sources and an ontology is a crucial aspect in order to capture the semantics of the data.

In this paper we define the framework for extracting from a relational database an ontology that is to be used as a conceptual view over the data, where the semantic mapping between the database schema and the ontology is captured by associating a view over the source data to each element of the ontology. Thus, the vocabulary over the ontology can be seen as a set of (materialised) views over the vocabulary of the data source [10] (i.e., similar to the technique known as GAV approach in the information integration literature [14]). The advantages of such a scenario are clear since it enables to access and query the underlying data using an ontology vocabulary.

To describe the extracted conceptual model, we provide an expressive ontology language which can capture features from Entity-Relationship and UML class diagrams, as well as variants of Description Logics. The heuristics underlying the ontology extraction process are based on ideas of standard relational schema design from ER diagrams in order to uncover the connections between relational constructs and those of ontologies. Besides the latter assumption the procedure presented in this paper takes into consideration relations being in third normal form (3NF) – in such a way best reflecting concepts of object-oriented data models.

The rest of the paper is structured as follows. In the next section we present the formal framework, where first the constraints expressed over the relational schema are defined, and the ontology language is presented. In the section on ontology extraction we describe an intuitive progression of ideas underlying our approach, and discuss the extraction procedure. Then, we provide an example illustrating all the process. In the last section we report on related work, and we draw some conclusions.

Chapter 2

Preliminaries

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 in the next section. The ontology language adopted enables the representation of "standard" modelling constructs which are commonly used in Entity-Relationships or UML diagrams (see [5]).

The use of an ontology language can be seen as an alternative to the use of standard modelling paradigms of Entity Relationships or UML diagrams. 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.

2.1 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* [15]. We make use of the standard notion of relational model by using named attributes, each with an associated datatype, instead of tuples.

Definition 1 *A* relational schema $\mathcal{R}$ *is a set of relationships, each one with a fixed set of attributes (assumed to be pairwise distinct) with associated datatypes. We use* $[s_1 : D_1, \ldots, s_n : D_n]$ *to denote that a relationship has attributes* $s_1, \ldots, s_n$ *with associated data types* $D_1, \ldots, D_n$*. We interpret relationships over a fixed countable* domain Δ *of datatype elements, which we consider partitioned into the datatypes* D_i*. The domain contains a special constant (*null*) called the* NULL value.

A database instance *(or simply* database*)* $\mathcal{D}$ *over a relational schema* $\mathcal{R}$ *is an (interpretation) function that maps each relationship* R *in* $\mathcal{R}$ *into a set* $R^{\mathcal{D}}$ *of total functions from the set of attributes of* R *to* Δ*. The instance must satisfy the attribute datatypes specified in the relational schema. I.e., if* R *has attributes* $[s_1 : D_1, \ldots, s_n : D_n]$*, and* $\phi \in R^{\mathcal{D}}$*, then* $\phi(s_i) \in D_i$ *or* $\phi(s_i) = \mathsf{null}$*.*

To ease the presentation of the semantics we make use of relational algebra. To this purpose, we introduce the definition of a projection of a relationship over a sequence of attributes.

Let $A = [s_1, \ldots, s_m]$ be a sequence of m attribute names of a relationship R of a schema $\mathcal{R}$, and $\mathcal{D}$ a database over $\mathcal{R}$. The *projection* of $R^{\mathcal{D}}$ over A is the relation $\pi_A R^{\mathcal{D}} \subseteq \Delta^m_{\mathcal{D}}$, satisfying the condition that $\phi \in R^{\mathcal{D}}$ iff $(\phi(s_1), \ldots, \phi(s_n)) \in \pi_A R^{\mathcal{D}}$. Note that the order of the attributes in A fixes the correspondence between positional arguments of $\pi_A R^{\mathcal{D}}$ and the attribute names of R.

As for the semantics of null values, we assume that they represent *unknown* values. That is, a null value is considered different from any other constant and from a null value in any other tuple. Formally, this has an impact on the equality among tuples; that is, two tuples $(t_1, \ldots, t_n)$ and $(t'_1, \ldots, t'_n)$ are equal iff $t_i = t'_i$ and they do not contain null values.[1]

The ontology extraction task takes as input a relational source; e.g. a DBMS. We abstract from any specific database implementation by considering an abstract *relational source* $\mathcal{DB}$, which is a pair $(\mathcal{R}, \Sigma)$, 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. Below we briefly list the kind of database integrity constraints we consider in our framework.

- *nulls-not-allowed constraints*: given a relation r in the schema, a nulls-not-allowed constraint over r is an assertion of the form $\mathit{nonnull}(r, \mathbf{A})$, where $\mathbf{A}$ is a sequence of attributes of r. Such a constraint is satisfied in a database $\mathcal{D}$ if for each $\phi \in r^{\mathcal{D}}$ we have $\phi(a) \neq \mathsf{null}$ for each $a \in \mathbf{A}$.

- *unique constraints*: given a relation r in the schema, a unique constraint over r is an assertion of the form $\mathit{unique}(r, \mathbf{A})$, where $\mathbf{A}$ is a sequence of attributes of r. Such a constraint is satisfied in a database $\mathcal{D}$ if for each $\phi_1, \phi_2 \in r^{\mathcal{D}}$, with $\phi_1 \neq \phi_2$, we have $\phi_1(\mathbf{A}) \neq \phi_2(\mathbf{A})$.[2] When we have $\mathit{unique}(r, \mathbf{A})$ and $\mathit{nonnull}(r, \mathbf{A})$ for r, then a *key constraint* $\mathit{key}(r, \mathbf{A})$ is associated to r.

- *inclusion dependencies*: an inclusion dependency is an assertion of the form $r_1\,[\mathbf{A_1}] \subseteq r_2\,[\mathbf{A_2}]$, where r_1, r_2 are relations, $\mathbf{A_1}$, $\mathbf{A_2}$ are sequences of distinct attributes of r_1 and r_2, respectively. Such a constraint is satisfied in a database $\mathcal{D}$ if $\pi_{\mathbf{A_1}}(r_1^{\mathcal{D}}) \subseteq \pi_{\mathbf{A_2}}(r_2^{\mathcal{D}})$. We call an inclusion dependency $r_1\,[\mathbf{A_1}] \subseteq r_2\,[\mathbf{A_2}]$ where $\mathbf{A_2}$ is in $\mathit{key}(r_2, \mathbf{A_2})$ a *foreign key constraint.*

- *exclusion dependencies*: an exclusion dependency is an assertion of the form $(r_1\,[\mathbf{A_1}] \cap \ldots \cap r_m\,[\mathbf{A_m}]) = \emptyset$, where $m \geq 2$ $r_1, \ldots, r_m$ are relations, $\mathbf{A_1}, \ldots, \mathbf{A_m}$ are sequences of attributes of $r_1, \ldots, r_m$, respectively. Such a constraint is satisfied in a database $\mathcal{D}$ if $\pi_{\mathbf{A_1}}(r_1^{\mathcal{D}}) \cap \ldots \cap \pi_{\mathbf{A_m}}(r_m^{\mathcal{D}}) = \emptyset$.

- *covering constraints*:[3] a covering constraint is an assertion of the form $(r_1\,[\mathbf{A_1}] \cup \ldots \cup r_m\,[\mathbf{A_m}]) \subseteq r_0\,[\mathbf{A_0}]$, where $m \geq 2$, $r_1, \ldots, r_m, r_0$ are relations, $\mathbf{A_1}, \ldots, \mathbf{A_m}, \mathbf{A_0}$ are sequences of attributes of $r_1, \ldots, r_m, r_0$, respectively. Such a constraint is satisfied in a database $\mathcal{D}$ if $\pi_{\mathbf{A_0}}(r_0^{\mathcal{D}}) \subseteq \bigcup_{i=1,\ldots,m} \pi_{\mathbf{A_i}}(r_i^{\mathcal{D}})$.

[1]Assuming this semantics for null values is not crucial, different ones can be accommodated.

[2]Given a function ϕ and a sequence of attributes $A = [s_1, \ldots, s_m]$, we use the notation $\phi(A)$ to indicate the tuple composed by the values of the attributes; i.e. $(\phi(s_1), \ldots, \phi(s_n))$.

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

2.2 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 $\mathcal{R}$. In this section we describe these concepts.

In addition to the standard definition of a relational schema, we introduce the concept of named components. 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 relationship a mapping from *named components* to sequences of attributes. Let R be a relationship in $\mathcal{R}$, to ease the notation we write $\mathcal{P}_R$ instead of $\mathcal{P}(R)$.

Definition 2 *Let R be a relationship in $\mathcal{R}$, with attributes $[s_1 : D_1, \ldots, s_n : D_n]$. $\mathcal{P}_R$ 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 $\mathcal{P}_R(c) = [s_{i_1}, \ldots, s_{i_m}]$, where each $i_j \in \{1, \ldots, n\}$, is called the c-*component *of R. We denote with $\delta(R)$ the set of all attributes not belonging to any c-component.*

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. I.e., given $\mathcal{P}_R(c_i) = [s_{i_1}, \ldots, s_{i_k}]$ and $\mathcal{P}_R(c_j) = [s_{j_1}, \ldots, s_{j_m}]$, if $s_{i_\ell} = s_{j_r}$ then $c_i = c_j$ and $\ell = r$.

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. Specifically, if the attributes of R are $[s_1 : D_1, \ldots, s_n : D_n]$, the signature of the component $\mathcal{P}_R(c) = [s_{i_1}, \ldots, s_{i_m}]$ is the sequence $[D_{i_1}, \ldots, D_{i_m}]$.*

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 constraints in $\mathcal{K}$, is based on the idea of modelling the domain by means of *axioms* involving the projection of the relationship 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 relationship has a single component, then this can be omitted and the atomic formula R corresponds to its projection over the single component.

Two atomic formulae $R[c]$ and $S[c']$ are *compatible* iff the two corresponding components $\mathcal{P}_R(c)$ and $\mathcal{P}_S(c')$ are compatible. Given the atomic formulae $R[c], R'[c'], R_i[c_i]$, an *axiom* is an assertion of the form specified in Figure 2.2, 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 $\langle \mathcal{R}, \mathcal{P}, \mathcal{K} \rangle$, 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 $c_1, \ldots, c_k$, for any $\phi_1, \phi_2 \in R^{\mathcal{D}}$ 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. Note that, in the definition above, we require the satisfiability of all the axioms, and in addition we consider the sequence of attributes of all the components of a relationship as a key for the relationship itself. This reflects the fact that in conceptual models the additional attributes not belonging to any component are not considered relevant to identify an element of an entity or a relationship.

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

Figure 2.1: Syntax and semantics of *DLR-DB* axioms.

The *DLR-DB* ontology language enables the use of the most commonly used constructs in conceptual modelling. In particular, among these we mention:

ISA, using assertions of the form $E_1 \sqsubseteq E_2$, stating that the class E_1 is a subclass of the class E_2;[4]

Disjointness, using assertions of the form E_1 disj E_2, stating disjointness between the two classes E_1 and E_2;

Role typing, using assertions of the form $R[c] \sqsubseteq E$, stating that the role corresponding to the c component of the relationship R is of type E;

Participation constraints, using assertions of the form $E \sqsubseteq R[c]$, stating that instances of class E participate to the relationship R as value for the c component;

Non-participation constraints, using assertions of the form E disj $R[c]$, stating that instances of class E do not participate to the relationship R as value for the c component;

Functionality, using assertions of the form $\text{funct}(R[c])$, stating that an object can appear in the c component of the relationship R at most once;

Covering, using the corresponding assertion to state that each member of a class must be contained in (at least) one of the covering classes.

Note that by taking away the covering axioms and considering only components containing single attributes this ontology language corresponds exactly to *DLR-Lite* (see [6]). By virtue of the assumption that components do not share attributes, it is not difficult to show that the same reasoning mechanism of *DLR-Lite* can be used in our case. The discussion on the actual reasoning tasks which can be employed in the context of *DLR-DB* systems is out of the scope of this paper. Herewith we are mainly interested of the use of the language to express data models extracted from the relational data sources.

[4]As mentioned before, when a relation has a single component, the component name can be omitted.

Chapter 3

Ontology extraction

3.1 Principles of Ontology Extraction

The principles upon our technique are based on best practices on relational schema design from ER diagrams – a standard database modelling technique [9, 4]. 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,[1] 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.

According to the methodology of translating ER model to relational model (we follow mostly [4, 9]), entities and some relationships generate relations. These relationships always include many-to-many and n-ary relationships. The corresponding relation includes the identifier of each of the participating entities and the additional attributes, if any. The remaining two types of relationships, one-to-one and many-to-one, usually result in adding attribute(s) that constitute the key of one relation to the other, thus acting as a foreign key. However, if, for instance, null values are not tolerated, the latter relationship may result in a separate relation. Sub-entities in ISA relationship are accounted for by creating a relation for each sub-entity, having key that corresponds to the identifier of the super-entity. The key of each sub-entity relation acts as a foreign key and references the key of the super-entity relation. Table 3.1 specifies a translation $\lambda(C)$ returning a relational table scheme being in 3NF for every ER component C, where C is either an entity or a n-ary relationship ($n \geq 2$).

3.2 The Extraction Process

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, expressing the mapping between the database schema and the ontology. Thus, given a relational source $\mathcal{DB} = (\Psi, \Sigma)$, our task is to extract from it the ontology in

[1]We do not deal with multi-valued attributes in this document

ER component C	**Relational table** $\lambda(C)$
Entity E $X = \mathsf{attrs}(E)$ $K = \mathsf{id}(E)$	**Table** $\lambda(E)$ columns: X $\mathit{key}(\lambda(E), K)$
Sub-entity $S(E)$ $X = \mathsf{attrs}(S)$ $K = \mathsf{id}(E)$	**Table** $\lambda(S)$ columns: KX $\mathit{key}(\lambda(S), K)$ $\lambda(S)\,[K] \subseteq \lambda(E)\,[K]$

if $S_1(E), \ldots, S_m(E)$ are disjoint
if $S_1(E), \ldots, S_m(E)$ are covering
	$(\lambda(S_1)\,[K] \cap \ldots \cap \lambda(S_m)\,[K]) = \emptyset$
	$(\lambda(S_1)\,[K] \cup \ldots \cup \lambda(S_m)\,[K]) = \lambda(E)\,[K]$

n-ary relationship $R(E_1, \ldots, E_n)$, $n \geq 1$ $X = \mathsf{attrs}(R)$ $K_i = \mathsf{id}(E_i)$, for $i = 1, \ldots, n$ if R is functional on E_i side if R is non-functional	**Table** $\lambda(R)$ columns: $K_1 \ldots K_n X$ $\mathit{key}(\lambda(R), K_i)$, for each $i = 1 \ldots n$ $\mathit{key}(\lambda(R), K_1 \ldots K_n)$ $\lambda(R)\,[K_i] \subseteq \lambda(E_i)\,[K_i]$, for $i = 1, \ldots, n$
if R is total on E_i side	$\lambda(E_i)\,[K_i] \subseteq \lambda(R)\,[K_i]$
N:M relationship $R(E_1, E_2)$ $X = \mathsf{attrs}(R)$ $K_1 = \mathsf{id}(E_1)$, $K_2 = \mathsf{id}(E_2)$	**Table** $\lambda(R)$ columns: $K_1 K_2 X$ $\mathit{key}(\lambda(R), K_1 K_2)$ $\lambda(R)\,[K_i] \subseteq \lambda(E_i)\,[K_i]$, for $i = 1, 2$
if R is total on E_i side, $i = 1, 2$	$\lambda(E_i)\,[K_i] \subseteq \lambda(R)\,[K_i]$
1:N relationship $R(E_1, E_2)$ $X = \mathsf{attrs}(R)$ $K_1 = \mathsf{id}(E_1)$, $K_2 = \mathsf{id}(E_2)$ Let R be functional on E_1 side	**Option (a): Table** $\lambda(R)$ columns: $K_1 K_2 X$ $\mathit{key}(\lambda(R), K_1)$ $\lambda(R)\,[K_i] \subseteq \lambda(E_i)\,[K_i]$, for $i = 1, 2$
if R is total on E_i side, $i = 1, 2$	$\lambda(E_i)\,[K_i] \subseteq \lambda(R)\,[K_i]$
1:N relationship $R(E_1, E_2)$ $K_1 = \mathsf{id}(E_1)$, $K_2 = \mathsf{id}(E_2)$ Let R be functional on E_1 side	**Option (b): Add foreign key to** $\lambda(E_1)$ $\lambda(E_1)\,[K_2] \subseteq \lambda(E_2)\,[K_2]$

if R is total from E_1 to E_2
if R is total from E_2 to E_1
	$\mathit{nonnull}(\lambda(E_1), K_2)$
	$\lambda(E_2)\,[K_2] \subseteq \lambda(E_1)\,[K_2]$

1:1 relationship $R(E_1, E_2)$ $X = \mathsf{attrs}(R)$ $K_1 = \mathsf{id}(E_1)$, $K_2 = \mathsf{id}(E_2)$	**Option (a): Table** $\lambda(R)$ columns: $K_1 K_2 X$ $\mathit{key}(\lambda(R), K_i)$, $i = 1, 2$ $\lambda(R)\,[K_i] \subseteq \lambda(E_i)\,[K_i]$, for $i = 1, 2$
 if R is total on E_i side, $i = 1, 2$	$\mathit{unique}(\lambda(R), K_2)$ $\lambda(E_i)\,[K_i] \subseteq \lambda(R)\,[K_i]$
1:1 relationship $R(E_1, E_2)$ $K_1 = \mathsf{id}(E_1)$, $K_2 = \mathsf{id}(E_2)$	**Option (b): Add foreign key to** $\lambda(E_1)$ $\lambda(E_1)\,[K_2] \subseteq \lambda(E_2)\,[K_2]$

if R is total on E_1 side
if R is total on E_2 side
	$\mathit{unique}(\lambda(E_1), K_2)$
	$\mathit{nonnull}(\lambda(E_1), K_2)$
	$\lambda(E_2)\,[K_2] \subseteq \lambda(E_1)\,[K_2]$

Table 3.1: Mapping ER model to relational model.

terms of *DLR-DB* ontology language (i.e., set of axioms in $\mathcal{K}$), together with a set of view definitions over *DLR-DB* relational source.

Roughly speaking, we reverse the process of translating ER model to relational model (i.e., function λ). 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 (which, in turn, reference keys of base or specific relations). Another type of relations are those corresponding to sub-entities in an inheritance relationship. When a relation has a key that is also a foreign key, and no other non-key foreign keys appear in that relation, then, clearly, an inheritance relationship exists. On the other hand, if 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[2]. Otherwise, such relation might also "look like" functional relationship (binary or n-ary), mapped directly to a relation, and therefore relations of this type are classified as ambiguous relations (see below). The classification function is provided in Appendix A.

Summarising, 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 having key which is one of the foreign keys of r_i is classified as an *ambiguous relation.*

Once the relations in $\mathcal{DB}$ are classified according to the conditions defined above, then the actual ontology extraction process returns a *DLR-DB* system as output. The pseudo code for extraction algorithm can be found in Appendix A. For every base and specific relation r_i the algorithm generates a relationship R_i with the attributes in a one-to-one correspondence with non-key foreign key attributes of r_i, and a single c-component, where $\mathcal{P}_{R_i}(c)$ corresponds to the key attributes of r_i and thus functionality axiom funct$(R_i[c])$ is added to $\mathcal{K}$;[3] a view is defined by projecting on all non-key foreign key attributes of r_i.

Once relationships for base and specific relations are defined, associations between those relationships must be identified. Specifically, a non-key foreign key in a relation r_i referencing

[2]Observe that only relations corresponding to (sub-)entities are referred to by other relations.

[3]This is done because a foreign key in a base relation references either other base relation or a specific relation, and vice versa. I.e., an association may exist between a (strong) entity and a sub-entity.

relation, r_j, determines the association between relationships R_i and R_j. Thus, for each such foreign key, a relationship R_k is generated, having 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, where c_i-component is functional, i.e., we have $\text{funct}(R_k[c_i])$ in $\mathcal{K}$; a corresponding view is defined by joining r_j with r_i and projecting on their keys. For expressing an association, determined by R_k, 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 $\text{funct}(R_k[c_j])$ 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 relationship 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 relationship corresponding to (base or specific) relation, r_j, that the key foreign key of r_i references. Additionally, each exclusion dependency on the set of specific relations induces the disjointness axioms R_i 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 $\mathcal{K}$.

Each relationship relation r_i is accounted for by generating a relationship R_i, with attributes in a one-to-one correspondence with those of r_i, and n components, where n is the number of foreign keys of r_i. Each $\mathcal{P}_{R_i}(c_{i_l})$ ($l \in \{1, \ldots, n\}$) has sequence of attributes corresponding to the l-th foreign key attributes of r_i; the corresponding view is defined by projecting on all attributes of r_i. Then, for each foreign key of r_i referencing relation r_j (that is already represented with a relationship R_j having a single component), the algorithm generates an axiom $R_i[c_{i_l}] \sqsubseteq R_j$ stating that the role corresponding to the c_{i_l}-component of R_i is of type R_j. Furthermore, if this foreign key appears on the right-hand side of an inclusion dependency, the axiom $R_j \sqsubseteq R_i[c_{i_l}]$ is added to $\mathcal{K}$ that states mandatory participation for instances of R_j to the relationship R_i as values for the c_{i_l} component.

Finally, the appropriate structures for ambiguous relations must be identified. As already discussed before, an ambiguous relation may correspond in ER schema to either sub-entity, which also participates with cardinality "one" in a binary relationship, or a functional relationship that was directly mapped to a relation. Following the idea that all functional binary relationships should be represented in a relational model with an embedded foreign key, e.g., in order to obtain the relational schema with a minimum number of relations, and that n-ary relationships ($n \geq 3$) are relatively unusual, our heuristics "prefers" to recover an inheritance relationship, and thus the algorithm generates the structures corresponding to those defined for specific 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.

3.3 Example

As an example of the ontology extraction process, consider the relational schema (primary keys are underlined) with constraints of Figure 3.1.

$\text{Scholar}_r(\underline{\text{ssn}}, \text{name}, \text{deptNo})$	$\text{Publication}_r(\underline{\text{id}}, \text{title}, \text{year})$
$\text{IsAuthorOf}_r(\underline{\text{schSsn}, \text{publId}})$	$\text{Department}_r(\underline{\text{no}}, \text{name})$
$\text{PostDoc}_r(\underline{\text{ssn}}, \text{scholarship})$	$\text{Professor}_r(\underline{\text{ssn}}, \text{salary})$
(a) Scholar_r [deptNo] $\subseteq$ Department_r [no]	(g) Publication_r [id] $\subseteq$ IsAuthorOf_r [publId]
(b) IsAuthorOf_r [schSsn] $\subseteq$ Scholar_r [ssn]	(h) Department_r [no] $\subseteq$ Scholar_r [deptNo]
(c) IsAuthorOf_r [publId] $\subseteq$ Publication_r [id]	(i) $\mathit{unique}(\text{Scholar}_r, \text{deptNo})$
(d) PostDoc_r [ssn] $\subseteq$ Scholar_r [ssn]	(j) $\mathit{nonnull}(\text{Scholar}_r, \text{deptNo})$
(e) Professor_r [ssn] $\subseteq$ Scholar_r [ssn]	(k) PostDoc_r [ssn] $\cap$ Professor_r [ssn] $= \emptyset$
(f) Scholar_r [ssn] $\subseteq$ IsAuthorOf_r [schSsn]	(l) Scholar_r [ssn] $\subseteq$ PostDoc_r [ssn] $\cup$ Professor_r [ssn]

Figure 3.1: Relational schema with constraints.

At the initial step of extraction process, relations Scholar, Publication and Department are classified as base relations, i.e. their keys and foreign keys do not share any attributes; IsAuthorOf relation is classified as relationship relation – its key is entirely composed from foreign keys; while relations PostDoc and Professor satisfy the conditions required for specific relations, i.e. the key ssn is their single foreign key.

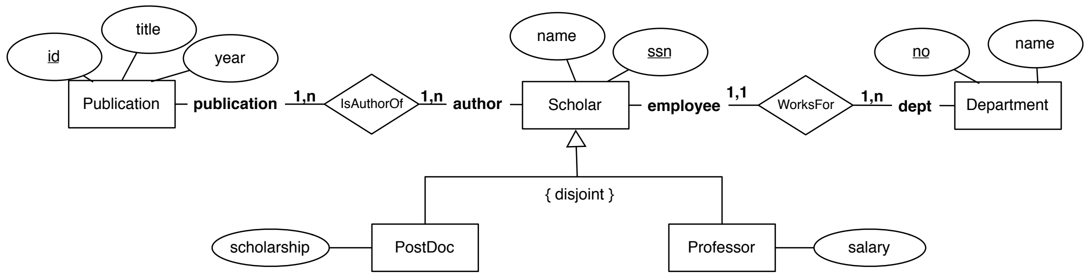
Without going again into details of the algorithm, we list below the extracted relationships of *DLR-DB* $\mathcal{R}$ together with the devised component structure $\mathcal{P}$, by considering the relation names and their corresponding attributes in the input relational source. Starting with base and specific relations, we have the corresponding relationships with single components. Since the component names for the latter relationships are not relevant (they can be omitted), we choose a common name id for all the five of them.[4] Figure 3.2 shows the extracted ontology together with the corresponding ER diagram.

Relationship	**Component** c	$\mathcal{P}_R(c)$	**Additianal attr.**	**View definition**
Scholar	id	ssn	name	$\pi_{\text{ssn,name}}(\text{Scholar}_r)$
Publication	id	id	title, year	$\pi_{\text{id,title,year}}(\text{Publication}_r)$
Department	id	no	name	$\pi_{\text{no,name}}(\text{Department}_r)$
PostDoc	id	ssn	scholarship	$\pi_{\text{ssn,scholarship}}(\text{PostDoc}_r)$
Professor	id	ssn	salary	$\pi_{\text{ssn,salary}}(\text{Professor}_r)$
WorksFor	employee	ssn		$\pi_{\text{ssn,no}}(\text{Department}_r \bowtie \text{Scholar}_r)$
	dept	no		
IsAuthorOf	author	schSsn		$\pi_{\text{schSsn,publId}}(\text{IsAuthorOf}_r)$
	publication	publId		

The constraints on the input relational source reflect those added to $\mathcal{K}$ of *DLR-DB* system. We list all the extracted axioms by the algorithm below. In particular, foreign key constraints (a)-(c) for base and relationship relations induce axioms (1) to (4) stating role typing of the corresponding relationships. The inclusion dependencies in (f)-(h) and nulls-not-allowed constraint in (j) above induce the axioms (5) to (8) stating mandatory participation for Scholar

[4]For the sake of clarity, the naming of the components for relationships WorksFor and IsAuthorOf, as well as the name of the WorksFor relationship itself, are determined by domain knowledge.

and Publication to the relationship IsAuthorOf, and for Scholar and Department to WorksFor. Functionality axiom in (9) is used to impose the upper limit for the multiplicity on employee component, which is induced by the uniqueness constraint in (i). Subclass axioms in (10) and (11) express inheritance relationship, determined by foreign key constraints associated to specific relations in (d), (e). Finally, based on (k) and (l) above, axioms (12) and (13), respectively, are extracted.



(1)	IsAuthorOf[author] $\sqsubseteq$ Scholar	(7)	Scholar $\sqsubseteq$ WorksFor[employee]
(2)	IsAuthorOf[publication] $\sqsubseteq$ Publication	(8)	Department $\sqsubseteq$ WorksFor[dept]
(3)	WorksFor[employee] $\sqsubseteq$ Scholar	(9)	funct(WorksFor[employee])
(4)	WorksFor[dept] $\sqsubseteq$ Department	(10)	PostDoc $\sqsubseteq$ Scholar
(5)	Scholar $\sqsubseteq$ IsAuthorOf[author]	(11)	Professor $\sqsubseteq$ Scholar
(6)	Publication $\sqsubseteq$ IsAuthorOf[publication]	(12)	PostDoc disj Professor

Figure 3.2: Extracted ontology and corresponding ER diagram.

3.4 Information Capacity Preserving

Our proposed ontology extraction technique can be seen as a *schema transformation* as defined in [17]. An important consideration in such a process (i.e., transforming one data model into another) is the potential for loss of information. We may evaluate the correctness of our schema extraction procedure based on the relative *information capacities* of the source and target schemas. In this section we show that our ontology extraction procedure is *equivalence preserving*.

In the following we denote by $\mathcal{S}$ and $\mathcal{T}$ source and target schemas corresponding to the input relational source $\mathcal{DB}$ and relational schema $\mathcal{R}$ of the extracted *DLR-DB* system.

Definition 3 *Let $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$ be consistent instances of schemas $\mathcal{S}$ 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:

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

Proof. Let $\mathcal{S}$ and $\mathcal{T}$ be source and target schemas, and let $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$ be databases over $\mathcal{S}$ and $\mathcal{T}$ respectively. Let μ denote the mapping function as defined in Definition 3. We have to show that

1. $\mu(\mathcal{D}_\mathcal{S}) = \mu(\mathcal{D}'_\mathcal{S}) \Rightarrow \mathcal{D}_\mathcal{S} = \mathcal{D}'_\mathcal{S}$, i.e., μ is injective;
2. If $\mathcal{D}_\mathcal{S}$ is consistent w.r.t. $\mathcal{S} \Rightarrow \mu(\mathcal{D}_\mathcal{S})$ is consistent w.r.t. $\mathcal{T}$;
3. If $\mathcal{D}_\mathcal{T}$ is consistent w.r.t. $\mathcal{T} \Rightarrow \exists \mathcal{D}_\mathcal{S}$ s.t. $\mathcal{D}_\mathcal{S}$ is consistent w.r.t. $\mathcal{S} \wedge \mu(\mathcal{D}_\mathcal{S}) = \mathcal{D}_\mathcal{T}$.

Let X, K and FK_i denote respectively the sequence of non-key, key and foreign key attribute names of a source relation r, where every FK_i, $1 \leq i \leq n$, references relation r_i. Let A denote the sequence of all attribute names of r. First of all observe that the mapping function μ applied for each class of relations results in the following:

- for a *base relation* r,
$$\mu(r) = \begin{cases} \pi_{K,X}(r), \\ \pi_{K,K_1}(r_1 \bowtie r), \\ \dots \\ \pi_{K,K_n}(r_n \bowtie r); \end{cases}$$
- for a *relationship relation* r,
$$\mu(r) = \pi_A(r);$$
- for a *specific relation* r, $\mu(r)$ is identical to the one of base relation;
- for an *ambiguous relation* r, $\mu(r)$ is identical to the one of base relation, when extraction procedure is completely automatic; otherwise it coincides with the one defined for relationship relation.

Note that attribute names are in a one-to-one correspondence in r and $\mu(r)$, e.g. the sequence of key attributes K of r coincide with K in $\mu(r)$. Moreover, as mapping definitions for specific and ambiguous relations coincide with those of base and relationship relations, in the following, without loss of generality, we will only distinguish between base and relationship relations.

Proof of statement 1. Let by contradiction $\mathcal{D}_\mathcal{S} \neq \mathcal{D}'_\mathcal{S}$. Then $\exists t \in \mathcal{D}_\mathcal{S} \wedge t \notin \mathcal{D}'_\mathcal{S}$. Let $t \in r$. We distinguish two cases, namely when r is a base and relationship relation.

Case 1: r is a base relation. Since $\mu(\mathcal{D}_\mathcal{S}) = \mu(\mathcal{D}'_\mathcal{S})$, then $\pi_{K,X}(r^{\mathcal{D}_\mathcal{S}}) = \pi_{K,X}(r^{\mathcal{D}'_\mathcal{S}})$. Hence $\exists t' \in r^{\mathcal{D}'_\mathcal{S}}$ such that $t' \neq t$ and $t.\{K,X\} = t'\{K,X\}$. We get a contradiction, since key requirement being unique is violated.

Case 2: r is a relationship relation. Proof by contradiction as in previous case.

Proof of statement 2. Let γ be axiom over $\mathcal{T}$ and suppose by contradiction $\mu(\mathcal{D}_\mathcal{S})$ is inconsistent w.r.t. $\mathcal{T}$, i.e. $\exists \gamma$ s.t. γ is not satisfied in $\mu(\mathcal{D}_\mathcal{S})$. Consider $r \in \mathcal{S}$. Following the notation used throughout the document, we denote by R a relation in $\mathcal{T}$ obtained by extraction procedure.

- γ is a functionality axiom funct$(R[c])$. This may correspond in $\mathcal{S}$ to the following constraints:

- *key*(r, K), where r is a base relation and R is a relation obtained by $\pi_{K,X}(r)$. Then the c-component of R is in a one-to-one correspondence with the key attributes K of r. If the functionality axiom is not satisfied in $\mathcal{D}_{\mathcal{T}}$ (i.e., c does not contain unique values), then the key K condition being unique is violated. We get a contradiction.
- *unique*(r, FK_i), where r is a base relation and R is a relation obtained by $\pi_{K,K_i}(r_i \bowtie r)$, where r_i is a relation referenced by FK_i. Then the c-component of R is in a one-to-one correspondence with the key attributes K_i of r_i. If the functionality axiom is not satisfied in $\mathcal{D}_{\mathcal{T}}$, then the key K_i condition being unique is violated, which is a contradiction.

• γ is a subclass axiom $R[c] \sqsubseteq R'[c']$. This correspond in $\mathcal{S}$ to the following constraints:

- $r\,[FK_i] \subseteq r_i\,[K_i]$, where r is a base or specific relation and r_i is a base relation. Then R' is a relation generated by $\pi_{K,X}(r)$ and R - a relation obtained by $\pi_{K,K_i}(r_i \bowtie r)$ respectively. The c-component and c'-component of R and R' are in a one-to-one correspondence with the key attributes K of r. If the subclass axiom is not satisfied in $\mathcal{D}_{\mathcal{T}}$ (i.e., there are values of c-component that are not contained in c'-component), then we get a contradiction because of violating the key K.
- $r\,[FK_i] \subseteq r_i\,[K_i]$, where r is a relationship relation and r_i is a base or specific relation. Then R is a relation generated by $\pi_A(r)$ and R' - a relation obtained by $\pi_{K_i,X_i}(r_i)$. The contradiction is obtained following the reasoning of the previous case.
- *notnull*(r, FK_i), where r is a base relation. R is a relation generated by $\pi_{K,X}(r)$ and R' - relation obtained by $\pi_{K,K_i}(r_i \bowtie r)$, where r_i is a relation referenced by FK_i. Then c-component of R and c'-component of R' are in a one-to-one correspondence with the key attributes K of r. If the subclass axiom is not satisfied in $\mathcal{D}_{\mathcal{T}}$ (i.e., there are c-component values that are not included in c'-component values), we get a contradiction because violating the key K.

• γ is a disjointness axiom R_1 disj R_2. This correspond in $\mathcal{S}$ to exclusion dependency $\pi_{K_1}(r_1) \cap \pi_{K_2}(r_2) = \emptyset$, where r_1 and r_2 are specific relations. The single components of R_1 and R_2 are in a one-to-one correspondence with the key attributes of r_1 and r_2. If the disjointness axiom is not satisfied in $\mathcal{D}_{\mathcal{T}}$ then the corresponding exclusion constraint is not satisfied which is a contradiction.

• γ is a covering axiom $R_1[c_1], \ldots, R_m[c_m]\,\mathrm{cover}\,R$. This correspond in $\mathcal{S}$ to a covering constraint $\pi_K(r) \subseteq \pi_{K_1}(r_1) \cup \ldots \cup \pi_{K_m}(r_m)$, where r is a base relation and $r_1, \ldots, r_m$ are specific relations. The contradiction follows based on the reasoning in the previous case.

Proof of statement 3. We start by showing that μ is reversible (or lossless), i.e., we can define a function μ^{-1}, the inverse of μ, such that every relation $r \in \mathcal{S}$ can be built back by applying μ^{-1}. That is, $\forall r \in \mathcal{S}$, $\mu^{-1}(\mu(r)) = r$. Specifically, for building back a *base relation* r, μ^{-1} is defined as follows:

$$\mu^{-1}(R) = R \bowtie R_{rel_1} \bowtie \ldots \bowtie R_{rel_n},$$

where R corresponds to relation $\pi_{K,X}(r)$ and each R_{rel_i} $(1 \leq i \leq n)$ is a relation obtained by $\pi_{K,K_i}(r_i \bowtie r)$. For a *relationship relation* r, μ^{-1} has the form

$$\mu^{-1}(R) = \pi_B(R),$$

where R is a relation obtained by $\pi_A(r)$ and B is a sequence of attribute names of R[5].

Since inverse functions exist for each $r \in \mathcal{S}$, there exists a database instance, $\mathcal{D}_\mathcal{S}$ over $\mathcal{S}$. Let ϵ be constraint over $\mathcal{S}$ and suppose by contradiction $\mathcal{D}_\mathcal{S}$ is not consistent with $\mathcal{S}$, i.e., $\exists\epsilon$ s.t. ϵ is not satisfied in $\mathcal{D}_\mathcal{S}$. Consider $r \in \mathcal{S}$. Again, we distinguish two cases:

Case 1: r is a base relation built by $R \bowtie R_{rel_1} \bowtie \ldots \bowtie R_{rel_n}$.

- ϵ is a key constraint $\mathit{key}(r, K)$. Then we have in $\mathcal{T}$ a functionality axiom $\text{funct}(R[c])$ satisfied in $\mathcal{D}_\mathcal{T}$. The single c-component of R is in a one-to-one correspondence with the attributes in K, and if $\mathit{key}(r, K)$ is not satisfied in $\mathcal{D}_\mathcal{S}$ (i.e., key is not functional or contains null values) then the corresponding functionality axiom is not satisfied in $\mathcal{D}_\mathcal{T}$, which is a contradiction.
- ϵ is a nulls-not-allowed constraint $\mathit{notnull}(r, FK_i)$. Then we have in $\mathcal{T}$ a subclass axiom $R \sqsubseteq R_{rel_i}[c_1]$ satisfied in $\mathcal{D}_\mathcal{T}$. The single component of R and c_1-component of R_{rel_i} are in a one-to-one correspondence with the key attributes K of r. If $\mathit{notnull}(r, FK_i)$ is not satisfied in $\mathcal{D}_\mathcal{S}$ (i.e., if FK_i contains null values), the c-component and c_1-component contain null values, which is a contradiction.
- ϵ is a unique constraint $\mathit{unique}(r, FK_i)$. Then we have in $\mathcal{T}$ a functionality axiom $\text{funct}(R_{rel_i}[c_2])$ satisfied in $\mathcal{D}_\mathcal{T}$. c_2-component of R_{rel_i} is in a one-to-one correspondence with the attributes of K_i of r_i. If $\mathit{unique}(r, FK_i)$ is not satisfied in $\mathcal{D}_\mathcal{S}$ (i.e., FK_i is not functional), then the c_2-component is not functional, which is a contradiction.
- ϵ is a foreign key constraint $r\,[FK_i] \subseteq r_i\,[K_i]$. Then there are in $\mathcal{T}$ two subclass axioms $R_{rel_i}[c_1] \sqsubseteq R$ and $R_{rel_i}[c_2] \sqsubseteq R_i$ satisfied in $\mathcal{D}_\mathcal{T}$. c_1-component of R_{rel_i} and a single component of R (resp. c_2-component of R_{rel_i} and a single component of R_i) are in a one-to-one correspondence with K attributes of r (resp. K_i attributes of r_i). If the foreign key constraint is not satisfied (i.e., FK_i contains value(s) that is not contained in K_i) we get a contradiction by violating the key K.
- ϵ is an inclusion dependency $r_i\,[K_i] \subseteq r\,[FK_i]$. Then there is in $\mathcal{T}$ a subclass axiom $R_i \sqsubseteq R[c_2]$ satisfied in $\mathcal{D}_\mathcal{T}$. Following reasoning of the previous case, if the inclusion dependency is not satisfied in $\mathcal{D}_\mathcal{S}$, the corresponding subclass axiom is not satisfied in $\mathcal{D}_\mathcal{T}$ which is a contradiction.
- ϵ is an exclusion dependency $\pi_{K_1}(r_1) \cap \pi_{K_2}(r_2) = \emptyset$[6]. Then there is in $\mathcal{T}$ the disjointness axiom R_1 disj R_2 satisfied in $\mathcal{D}_\mathcal{T}$, where R_1 and R_2 are defined as R above and single components of R_1 and R_2 correspond to the attributes of K_1 of r_1 and K_2 of r_2, respectively. Thus, if the exclusion dependency is not satisfied in $\mathcal{D}_\mathcal{S}$, then the components of R_1 and R_2 share values which is a contradiction.

[5]Note that the sequence of attributes B coincide with that of A

[6]Exclusion dependency (as well as covering constraint below) is applicable only to specific relations. Since mapping function for base and specific relations coincide, we include this constraint here.

- ϵ is a covering constraint $\pi_K(r) \subseteq \pi_{K_1}(r_1) \cup \ldots \cup \pi_{K_m}(r_m)$. Then there is in $\mathcal{T}$ the covering axiom $R_1[c_1], \ldots, R_m[c_m]$ cover R satisfied in $\mathcal{D}_{\mathcal{T}}$. We get a contradiction based on the reasoning of the previous case.

Case 2: r is a relationship relation built by $\pi_A(R)$. The constraints applicable to r are the following:

- ϵ is a key constraint $key(r, K)$. All attributes of $R \in \mathcal{T}$ are in a one-to-one correspondence with those of r and thus the key of R is composed of K and is satisfied in $\mathcal{D}_{\mathcal{T}}$. If $key(r, K)$ is not satisfied in $\mathcal{D}_{\mathcal{S}}$, then the key of R is not satisfied in $\mathcal{D}_{\mathcal{T}}$, which is a contradiction.
- ϵ is a foreign key constraint $r\,[FK_i] \subseteq r_i\,[K_i]$. Then there is in $\mathcal{T}$ a subclass axiom $R[c_i] \sqsubseteq R_i$ satisfied in $\mathcal{D}_{\mathcal{T}}$. c_i-component of R and a single component of R_i are in a one-to-one correspondence with FK_i and K_i attributes of r and r_i, respectively. If $r\,[FK_i] \subseteq r_i\,[K_i]$ is not satisfied in $\mathcal{D}_{\mathcal{S}}$ (i.e., FK_i contains values that are not part of K_i), then c_i-component of R contains values that are not contained in a single component of R_i, which is a contradiction.
- ϵ is an inclusion dependency $r_i\,[K_i] \subseteq r\,[FK_i]$. Then we have in $\mathcal{T}$ a subclass axiom $R_i \sqsubseteq R[c_i]$ satisfied in $\mathcal{D}_{\mathcal{T}}$. Following reasoning of the previous case, if the inclusion dependency is not satisfied in $\mathcal{D}_{\mathcal{S}}$, the corresponding subclass axiom is not satisfied in $\mathcal{D}_{\mathcal{T}}$ which is a contradiction.

Finally, we have to show that $\mu(\mathcal{D}_{\mathcal{S}}) = \mathcal{D}_{\mathcal{T}}$. Suppose by contradiction $\mu(\mathcal{D}_{\mathcal{S}}) \neq \mathcal{D}_{\mathcal{T}}$ and consider $r \in \mathcal{S}$. We distinguish two cases:

Case 1: r is a base relation. Then there is $R \in \mathcal{T}$ s.t. R either has the sequence of attributes in a one-to-one correspondence with non-foreign key attributes of r (denoted by $R(K, X)$) or R has attributes in a one-to-one correspondence with key attributes of r and its referenced relation r_i (denoted by $R(K, K_i)$). Consider $\mu(r)$. By assuming that $\mu(r) \neq R$ we get immediately contradiction.

Case 2: r is a relationship relation. Proof follows based on the reasoning above.

Chapter 4

Related Work

The primary motivation of our work is a scenario for extracting from a relational database schema its corresponding conceptual view. In this process we integrate aspects of database reverse engineering, and since our ontology language is closely related to ER or UML formalisms, the works on extracting ER (or EER) and object models are of particular interest. Much work has been addressed on the issue of explicitly defining semantics in database schemas [2, 5] and extracting semantics out of database schemas [8, 16]. The work described in [2] provides algorithms that investigate data instances of an existing legacy database in order to identify candidate keys of relations, to locate foreign keys, and to decide on the appropriate links between the given relations. However, the results on querying data instances cannot "guarantee" satisfaction of certain constraints; e.g., the fact that there are no null values in a particular attribute does not mean that it is NOT NULL. As a result, user involvement is always required. The approach in [4] depends on the user to resolve name conflicts, and to specify the required semantic information that leads to identify the keys. In our work we instead assume the knowledge on key and foreign key constraints, as well as non null and unique values on attributes, inclusion and disjointness between relations, etc. exist in the schema. The work in [16] propose transformations that are applied to produce the re-engineered schema and handles the establishment of inheritance hierarchies. However, it considers relations in BCNF and thus every relation is in a one-to-one correspondence with an object in the extracted schema. The main idea of the methodology described in [8] comes close to ours in the sense that it derives classification for relations and attributes based on heuristics of what kind of ER components would give rise to particular relations.

The recent call for a Semantic Web arose several approaches in bringing together relational databases and ontologies. Among them we mention [3], 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 ontology onto which the relations are mapped to. On the other hand, the approach of [12] 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.

Chapter 5

Conclusions

We have described an heuristic procedure for extracting from relational database its conceptual view, where the wrapping of relational data sources by means of an extracted ontology is done by associating view over the original data to each element of the ontology. To represent the extracted ontology, instead of a graphical notation, we employ an ontology language thus providing a precise semantics to extracted schema. Our extraction procedure relies on information from the database schema, (i.e., key and foreign key structure, restrictions on attributes and dependencies between relations), and automatically extracts all the relevant semantics if an input relational schema was designed using a standard methodology. The resulting ontology can be used to access the data source by means of query rewriting using the defined views. However, this can be done as long as the ontology itself is not going to be modified. In such a case, query rewriting techniques should be used to retrieve the data (see [6]).

We are currently following several directions to continue the work reported in this paper. First, conceptual modelling constructs as multi-valued attributes and weak entities, alternative techniques for inheritance representation in relational tables (e.g., when a super-class is embedded in its subclass relations). On the other hand, additional sources of knowledge, e.g. linguistic relationships between keys, can be used to disambiguate relations. The results of our technique should be validated by an empirical analysis. To this purpose we are starting to experiment with real database schemas to evaluate the quality of the extracted ontologies.

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor 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, A. Borgida, and J. Mylopoulos. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *ODBASE'05*, pages 1152–1169, 2005.

[4] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual Database Design. An Entity-Relationship Approach.* Benjamin/Cummings Publishing Company, Inc., 1992.

[5] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on uml class diagrams. *Artificial Intelligence*, 168(1):70–118, 2005.

[6] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

[7] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. 10(3):237–271, 2001.

[8] 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.

[9] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Addison Wesley Publ. Co., fourth edition, 2004.

[10] 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.

[11] J. Heflin and J. Hendler. A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[12] 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.

[13] D. Lembo, C. Lutz, and B. Suntisrivaraporn. Tasks for ontology design and maintenance. Deliverable D05, TONES EU-IST STREP FP6-7603, May 2006.

[14] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS02)*, pages 233–346, 2002.

[15] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases.* The MIT Press, 2001.

[16] 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.

[17] Ren&#233;e J. Miller, Yannis E. Ioannidis, and Raghu Ramakrishnan. The use of information capacity in schema integration and translation. In *VLDB '93: Proceedings of the 19th International Conference on Very Large Data Bases*, pages 120–133, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[18] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[19] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner. Ontology-based integration of information - a survey of existing approaches. In *Proc. of IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.

Appendix A

Extraction Algorithms

Function classify classifies the relations in the input relational source into the base (denoted by Ψ_B), relationship (denoted by Ψ_R), specific (denoted by Ψ_S) and ambiguous (denoted by Ψ_A) relations. Let $card(X)$ denote cardinality of set X and $refrels$ denote the set of all relations referenced by foreign keys.

```
**Function** classify($\mathcal{DB}$)
**Input:** Relational source $\mathcal{DB} = (\Psi, \Sigma)$
**Output:** Sets $\Psi_B, \Psi_R, \Psi_S, \Psi_A$ of classified relations
$\Psi_B := \{\}$;
$\Psi_R := \{\}$;
$\Psi_S := \{\}$;
$\Psi_A := \{\}$;
for each $r_i \in \Psi$ do
  if $PK_i \cap \mathcal{FK}_i = \emptyset$
    then $\Psi_B := \Psi_B \cup r_i$;
    else if $PK_i = \mathcal{FK}_i$ and $card(\mathcal{FK}_i) > 1$
      then $\Psi_R := \Psi_R \cup r_i$;
      else if $PK_i = \mathcal{FK}_i$ and $card(\mathcal{FK}_i) = 1$ or $r_i \in$ refrels
        then $\Psi_S := \Psi_S \cup r_i$;
        else $\Psi_A := \Psi_A \cup r_i$
end
```

Algorithm extract returns the *DLR-DB* system with the set of relationships in $\mathcal{R}$ and set of axioms in $\mathcal{K}$. We denote by $R(C_1, \ldots, C_n)$ the relationship $R \in \mathcal{R}$ with n components, where every c_i-component of R is in a one-to-one correspondence with the sequence C_i of attributes of a source relation r. Let $r_i \in \Psi$ correspond to a relationship $R_i \in \mathcal{R}$.

```
Algorithm extract($\mathcal{DB}$)
Input: Relational source $\mathcal{DB} = (\Psi, \Sigma)$
Output: DLR-DB system $\mathcal{S}$ and set of view definitions $\mathcal{V}$
$\mathcal{R} := \{\}$;
$\mathcal{K} := \{\}$;
```

```
$\mathcal{V} := \{\}$;
classify($\mathcal{DB}$);
(a) for each $r_i \in \Psi_B$ do
   $\mathcal{R} := \mathcal{R} \cup R_i(PK_i)$;
   $\mathcal{K} := \mathcal{K} \cup \text{funct}(R_i[PK_i])$;
   $\mathcal{V} := \mathcal{V} \cup V_i : \pi_{PK_i, X_i}(r_i)$;
end;
for each $r_i \in \Psi_S$ do
perform sequence inside (a);
end;
for each $r_i \in \Psi_A$ do
   perform sequence inside (a);
end;
(b) for each $r_i \in \Psi_B$ s.t. $\mathcal{FK}_i \neq \emptyset$ do
   for each $FK_{i_j} \in \mathcal{FK}_i$ do
      $\mathcal{R} := \mathcal{R} \cup R_k(PK_i, PK_j)$;
      $\mathcal{K} := \mathcal{K} \cup R_k[PK_i] \subseteq R_i$;
      $\mathcal{K} := \mathcal{K} \cup R_k[PK_j] \subseteq R_j$;
      $\mathcal{V} := \mathcal{V} \cup V_{i,j} : \pi_{PK_i, PK_j}(r_j \bowtie r_i)$;
      if $notnull(r_i, FK_{i_j})$ is satisfied
         then $\mathcal{K} := \mathcal{K} \cup R_i \subseteq R_k[PK_i]$;
      if $unique(r_i, FK_{i_j})$ is satisfied
         then $\mathcal{K} := \mathcal{K} \cup \text{funct}(R_k[PK_j])$;
      if $FK_{i_j}$ appears on the right-hand side of an inclusion dep.
         then $\mathcal{K} := \mathcal{K} \cup R_j \subseteq R_k[PK_j]$;
end;
for each $r_i \in \Psi_S$ s.t. $\mathcal{FK}_i > 1$ do
   for each $FK_{i_j} \in \mathcal{FK}_i$ s.t. $FK_{i_j} \neq PK_i$ do
      perform sequence inside (b);
end;
for each $r_i \in \Psi_A$ s.t. $\mathcal{FK}_i > 1$ do
   for each $FK_{i_j} \in \mathcal{FK}_i$ s.t. $FK_{i_j} \neq PK_i$ do
      perform sequence inside (b);
end;
for each $r_i \in \Psi_R$ do
   $\mathcal{R} := \mathcal{R} \cup R_i(PK_{i_1}, \ldots, PK_{i_n})$;
   $\mathcal{V} := \mathcal{V} \cup V_i : \pi_{A_i}(r_i)$;
   for each $FK_{i_j} \in \mathcal{FK}_i$ do
      $\mathcal{K} := \mathcal{K} \cup R_i\left[FK_{i_j}\right] \subseteq R_j$;
      if $FK_{i_j}$ appears on the right-hand side of an inclusion dep.
         then $\mathcal{K} := \mathcal{K} \cup R_j \subseteq R_i\left[FK_{i_j}\right]$;
end;
for each exclusion dependency $(r_1 \cap r_2 = \emptyset) \in \Sigma$ do
   $\mathcal{K} := \mathcal{K} \cup R_1 \text{ disj } R_2$;
end;
for each covering constraint $r \subseteq r_1 \cup \ldots \cup r_m \in \Sigma$ do
```

```
    $\mathcal{K}$ := $\mathcal{K} \cup R_1, \ldots, R_m$ cover $R$;
return $\mathcal{S}, \mathcal{V}$
```