# Using OWL to integrate relational schemas

Alberto Abelló[1] and Fernando Carpani[2]

[1] Dept. de Llenguatges i Sistemes Informàtics, U. Politècnica de Catalunya
[2] Instituto de Computación, U. de la República Oriental del Uruguay

**Abstract.** Ontologies offer two contributions to the Semantic Web. On the first hand, they show a vocabulary consensus inside a community. On the other hand, they provide reasoning capabilities. In this paper we present a completely automatic translation from relational schemas to OWL, so that inference mechanisms can be used to integrate different schemas, by dealing with structure heterogeneities. The output of the translation algorithm, which explicits functional dependencies in the relational schema, belongs to OWL Full.

## 1 Introduction

In the last years, the appearance of the Web has changed the way of conceiving not only business, but also computers themselves. Integration and interoperability were already hot research topics, and the Web has just outlined their importance. However, the current state of the Web does not facilitate it, because only contains data instead of true information. Web pages are usually designed for humans, and their meaning cannot be captured by automatized engines. To ease the interpretation of data to the computers, the Web is being transformed into the "Semantic Web", by publishing the schema of data in the form of ontologies. As can be seen in [W3C04], the W3C has proposed a "Web Ontology Language" (OWL) for this ontologies.

Contribution of ontologies is two fold. On the one hand, an ontology should be generated by consensus of the actors in its usage domain. On the other hand, ontologies offer automatic reasoning mechanisms. From our point of view, such reasoning or inference mechanisms should be used to overcome schema conflicts. In most cases, due to the size of the schema manual integration can be really hard. But in some cases, like "Peer-to-Peer Database Management Systems" (P2PDBMS), this is not even feasible, and integration must be completely automated. As pointed out in [BHS05], interoperability and integration of ontologies is an important issue, and reasoning capabilities would help on testing for consistency and computing the integrated class hierarchy.

Therefore, in order to give access to our data to either humans or machines, we should offer the schema in such a way that it can be automatically interpreted. Unfortunately, in most cases, the conceptual schema of the data we want to publish is obsolete or lost. However, even in these cases, we can access the catalog of our DBMS to know the relations, attributes and constrains that are current.

In this work, we try to convert the relational information in the catalog (without considering the instance data) of the"Database Management System" (DBMS) into ontological knowledge, in such a way that inference mechanisms can be used to overcome schema discrepances.

Next section shows related work regarding some reverse engineering techniques; section 3 explains the notation used; section 4 introduces the algorithm; section 5 demonstrates how the obtained OWL schemas should be used; section 6 discusses the reasoning algorithms; and finally section 7 concludes the paper.

## 2 Related work

From the beginning, there has been a tight relationship between reasoning tools and databases. [BB93] shows a mechanism to translate from DL to a Relational Database. Every primitive (those not appearing on the left-hand side of a definition) concept or role is translated into a relation, while derived (those that do appear on the left-hand side of a definition) concepts or roles are smartly translated into SQL queries. The aim of this work is to be able to reason on the schema with a KBMD (Knowledge Based Management Systems), while keeping the instances in a DBMS. Our work is similar to [BB93] in the sense that our data is in a DBMS and want to publish the schema in such a way that reasoning algorithms can be used. The difference is that our starting point is the DBMS, while their is the KBMS.

More recently, with the appearance of the Web, as already pointed out in [BL98], there exists the need of publishing database contents. Thus, some works have been devoted to schema extraction and publication on the Web. [SSV02] shows a reverse engineering approach to extract an ontology (using Frame Logic) from a relational schema. They do not study the possibility of reasoning on the obtained ontology, and the process is not fully automatic, since the user chooses the translation rule to be applied in some cases. [Ast04] also shows a reverse engineering approach to transform a relational schema in 3NF into an ontology expressed in Frame Logic (which can be translated into RDF). They analyze keys, data and attribute correlations to be able to enrich the ontology with equivalence and inclusion relationships. However, they do not study the further use of inference mechanisms on their translation.

[dLC05] place their work at P2PDBMS area. They want to exchange data among a dynamic set of DBMS. To facilitate this, the authors propose the exchange of schema and instances together. Therefore, they choose OWL Full as ontology language, and the proposed translation from Relational model to OWL emphasizes that characteristic. However, their translation does not facilitate any kind of reasoning on the generated concepts.

## 3 Notation

Since it is a recommendation of W3C, we take OWL as the ontology language for this work. OWL provides three increasingly expressive sublanguages:

**Lite:** Supports subclass/superclass hierarchies, as well as properties besides some integrity constraints like cardinalities zero or one for those properties.

**DL:** DL comes from "Description Logics". Together with those in OWL Lite, this also supports complex definitions of classes like union, intersection, disjunction and complementariness; and any cardinality for the properties.

**Full:** Supports the same language constructs as DL. The difference lies on the usage restriction of those constructs. For example, while OWL DL requires the separation between classes and instances, OWL Full allows a class being an instance of another class. Another important characteristic of OWL Full is that it allows cardinality constraints placed on transitive properties.

```
<owl:Class rdf:ID="className">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class rdf:ID="attrName"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

**Fig. 1.** Example of OWL syntax for "$className \sqsubseteq \exists propName.attrName$"

As can be seen in figure 1, OWL syntax is too verbose. Therefore, through this paper, we will use DL syntax, which is much shorter. As can be seen in [HPS04] and [BHS05] OWL semantics can be translated into DL.

Thus, we consider that a knowledge base comprises two components, i.e. TBox (the terminology, we could recognize it as the schema) and ABox (the assertions about individuals, or instances). As explained in [BCM+03], the TBox contains concepts (i.e. classes), and to define a formal semantics of the logic we use an interpretation $\mathcal{I}$. An interpretation is a pair $[\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}]$, where $\Delta^{\mathcal{I}}$ is the domain (a non-empty set), and $\cdot^{\mathcal{I}}$ is an interpretation function that assigns to every atomic concept (i.e. class) $A$ a set ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$) and to every atomic role (i.e. property) $r$ a binary relation ($r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). A property is said to be functional iff $\{(a,b),(a,c)\} \subseteq r^{\mathcal{I}}$ implies $b = c$. A property is said to be transitive iff $\{(a,b),(b,c)\} \subseteq r^{\mathcal{I}}$ implies $(a,c) \in r^{\mathcal{I}}$. Inductively, this is extended to non-atomic classes by the following definitions (where $C$ and $D$ are classes ; $r$ and $s$ properties):

$$
\begin{array}{rr}
\text{Bottom } (\mathcal{S}) & \bot^{\mathcal{I}} = \emptyset \\
\text{Top } (\mathcal{S}) & \top^{\mathcal{I}} = \Delta^{\mathcal{I}} \\
\text{complementOf } (\mathcal{S}) & (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C \\
\text{intersectionOf } (\mathcal{S}) & (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
\text{unionOf } (\mathcal{S}) & (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
\text{someValuesFrom } (\mathcal{S}) & (\exists r.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
\text{allValuesFrom } (\mathcal{S}) & (\forall r.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.\ (a,b) \in r^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\} \\
\text{inverseOf } (\mathcal{I}) & (r^{-})^{\mathcal{I}} = \{(a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (b,a) \in r^{\mathcal{I}}\} \\
\text{FunctionalProperty } (\mathcal{F}) & (\leq 1\ r)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid\ \#\{b \mid (a,b) \in r^{\mathcal{I}}\} \leq 1\} \\
\text{min/maxCardinality } (\mathcal{N}) & (\leq n\ r)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid\ \#\{b \mid (a,b) \in r^{\mathcal{I}}\} \leq n\}
\end{array}
$$

Thus, this constructs can be used in the TBox to state a set of subsumption (i.e. subClassOf) and equivalence (i.e. equivalentClass) axioms of the form $C \sqsubseteq D$ and $C \equiv D$ (being $C$ and $D$ possibly complex classes).

$$X \sqsubseteq Y \Leftrightarrow \forall \mathcal{I} : X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$$

$$X \equiv Y \Leftrightarrow X \sqsubseteq Y \wedge Y \sqsubseteq X$$

Moreover, with $\mathcal{H}$ constructs in the logic, we can also state axioms respectively noted $r \sqsubseteq s$ to define a property hierarchy (i.e. subPropertyOf), on the understanding that cardinality restrictions are only allowed in "simple" properties (i.e. those non-transitive, and without transitive subproperties). Depending on the constructs allowed in the logic, we can talk about the complexity and expressive power of $\mathcal{SHIF}$, $\mathcal{SHOIN}$ (which corresponds to $\mathcal{SHIN}$, plus collections of individuals), etc.

| | |
|---|---|
| OWL-Lite | $\mathcal{SHIF}$ |
| OWL-DL | $\mathcal{SHOIN}$ |
| OWL-Full | Unconstrained $\mathcal{SHOIN}$ |

**Table 1.** Correspondence between OWL and DL

Table 1 shows the equivalences between OWL and DL. OWL-DL is equivalent to $\mathcal{SHOIN}$with nominals and concrete domains. Notice that OWL-Full removes the constraint on transitive roles and number restrictions, which is known to result in some undecidable combinations, as can be seen in [HST99].

## 4 Translation algorithm

Different translation possibilities (as shown in section 2) have already been defined in the literature from relatonal model to OWL. Ours differs from those because it facilitates reasoning by expliciting functional dependencies and smoothly deals with null values.

Similar to [AHV95], we assume the existence of a finite set $atts$ of attributes, and another finite set $rels$ (disjoint from $atts$) of relations. Being $A \in atts$, $Dom(A)$ is a set of constants called the domain of $A$. We also have a function $int : rels \rightarrow \mathcal{P}(atts)$, which represents the attributes of a relation (i.e. its intension). Moreover, we assume that attributes belong to, at most, one relation ($\forall R, S \in rels : R \neq S \Rightarrow int(R) \cap int(S) = \emptyset$).

The extension of a relation is a subset of the cartesian product of the domains of its attributes. Therefore, we define $ext(R) \subseteq Dom(A_1) \times ... \times Dom(A_n)$, being $int(R) = \{A_1, ..., A_n\}$. A tuple of a relation $R$ is an element of its extension. Thus, $\pi_{A_i}(t)$ is the value of $t$ belonging to $Dom(A_i)$.

A "Functional Dependency" (FD) is an expression of the form $X \rightarrow Y$ where $X, Y \subseteq atts$ fulfilling that $\forall s, t\ tuples : \pi_X(s) = \pi_X(t) \rightarrow \pi_Y(s) = \pi_Y(t)$. Then, a "Key Dependency" (KD) of a relation $R$ is a set of sets of attributes so that for each one of these sets it is subset of the intension of R and there exists an FD from it to the intension. Thus, $KD$ is a function $KD : rels \rightarrow \mathcal{P}(\mathcal{P}(atts))$, so that $\forall R : KD(R) \subseteq \mathcal{P}(int(R)) \wedge \forall X \in KD(R) : (X \rightarrow int(R) \in FD \wedge \nexists Y \subseteq X : Y \rightarrow int(R))$.

Finally, a "Foreign Key" (FK) between two sets of attributes $X$ and $Y$ belonging respectively to relations $R$ and $S$ (where $Y$ is a Key Dependency of S) is a bijection $pt$ (showing that an attribute points to another attribute) between the two sets of attributes so that for each tuple of $R$ exists a tuple in $S$ with the same values for attributes $X$ and $Y$, respectively. Thus, we define $FK$ as a function $FK : rels \rightarrow \mathcal{P}(atts) \times (atts \rightarrow atts)$, so that $\forall R : \forall < X, pt > \in FK(R) : (X \in \mathcal{P}(int(R)) \wedge \exists S \in rels : (\exists K \in KD(S) : \forall A \in X : pt(A) \in K \wedge \forall s \in ext(R) : \exists t \in ext(S) : \forall A_i \in X : \pi_{A_i}(s) = \pi_{pt(A_i)}(t)))$.

Thus, a relational schema $E_R$ is a tuple $< atts, rels, int, KD, FK, null >$, where $null$ is a function $null : atts \rightarrow \{$"T","F"$\}$ that shows whether an attribute admits null values or not. From here on, relational elements will have a subindex "R" to be distinguished from their corresponding OWL classes.

Figure 2 shows the translation algorithm from one of such schemas to OWL. First of all, we define four different superclasses, namely $Tuples$, $Values$, $Relations$, and $Keys$. By means of them, step 1 states that tuples are not attribute values; step 2 defines tuples as being exactly those of relations and keys; step 3 states that relations are disjoint from keys.

The main characteristic of this translation is that attributes are not classes but properties, which may look a bit strange. However, this is just a literal translation, because it is well known (see [EN89]) that attributes are functions from real world entities to data values. Therefore, they are translated into functional properties (step 4(b)i and 4(b)ii) and their domain into a derived class (step 4(b)iii and 4(b)iv). Step 4(b)v states that an attribute belongs to only one relation. We can extract more knowledge from the obligatory nature of the attribute. Thus, step 4(b)vi shows whether the attribute admits null values or not. We do not introduce any special value, the property instance just exists or not.

Conversely, relations, KDs and FKs are translated into classes (steps 4a, 4(c)i, and 4(d)i respectively). Nevertheless, they are related by means of properties. A

1. State $Tuples \sqsubseteq \neg Values$
2. State $Tuples \equiv Relations \sqcup Keys$
3. State $Relations \sqsubseteq \neg Keys$
4. For each $R_R \in rels$
   (a) Add class $R$
   (b) For each $A_R \in int(R_R)$
      i. Add property $r_A$
      ii. State $(= 1 r_A) \equiv \exists r_A . \top$
      iii. Add class $A \sqsubseteq Values$
      iv. State $A \equiv \exists r_A^- . R$
      v. State $R \sqsupseteq \exists r_A . A$
      vi. If $null(A_R) =$ "F"
         A. State $R \sqsubseteq \exists r_A . A$
   (c) For each $K_R \in KD(R_R)$
      i. Add class $K$
      ii. Add property $r_K$
      iii. State $(= 1 r_K) \equiv \exists r_K . \top$
      iv. State $(= 1 r_K^-) \equiv \exists r_K^- . \top$
      v. State $K \equiv \exists r_K^- . R$
      vi. State $R \equiv \exists r_K . K$
      vii. For each $A_R \in K_R$
         A. Add property $r_{K_A}$
         B. State $(= 1 r_{K_A}) \equiv \exists r_{K_A} . \top$
         C. State $K \equiv \exists r_{K_A} . A$
         D. State $\exists r_K . K \equiv (= 2 r_{T_{K_A}})$, being $r_{T_{K_A}}$ a transitive property so that $r_K \sqsubseteq r_{T_{K_A}}$; $r_A \sqsubseteq r_{T_{K_A}}$ and $r_{K_A} \sqsubseteq r_{T_{K_A}}$
   (d) For each $< F_R, pt > \in FK(R_R)$
      i. Add class $F$
      ii. Add property $r_F$
      iii. State $(= 1 r_F) \equiv \exists r_F . \top$
      iv. State $F \equiv \exists r_F^- . R$
      v. State $\exists r_F . F \equiv \bigsqcap_{A_R \in F_R} \exists r_A . A$
      vi. State $F \sqsubseteq K$ (being $K_R \in KD(S_R)$ so that $\forall A_R \in F_R : pt(A_R) \in K_R$)
      vii. For each $A_R \in F_R$
         A. Add property $r_{F_A}$
         B. State $(= 1 r_{F_A}) \equiv \exists r_{F_A} . \top$
         C. State $F \equiv \exists r_{F_A} . A$
         D. State $\exists r_F . F \equiv (= 2 r_{T_{F_A}})$, being $r_{T_{F_A}}$ a transitive property so that $r_F \sqsubseteq r_{T_{F_A}}$; $r_A \sqsubseteq r_{T_{F_A}}$ and $r_{F_A} \sqsubseteq r_{T_{F_A}}$
         E. State $r_{F_A} \sqsubseteq r_{K_{pt(A)}}$
         F. State $A \sqsubseteq pt(A)$
5. For each $R_R \in rels$
   (a) For each $S_R \in rels$
      i. If $R_R \neq S_R$: State $R \sqsubseteq \neg S$
      ii. For each $K_R \in KD(R_R)$
         A. For each $K'_R \in KD(S_R)$: if $K_R \neq K'_R$: State $K \sqsubseteq \neg K'$
6. State $Relations \equiv \bigsqcup_{R_R \in rels} R$
7. State $Keys \equiv \bigsqcup_{R_R \in rels} \bigsqcup_{K_R \in KD(R_R)} K$

**Fig. 2.** Translation algorithm to DL

relation is related to its KDs by means of a bijective complete property (steps 4(c)ii, 4(c)iii, 4(c)iv, 4(c)v and 4(c)vi), and to its FKs by means of a functional non-complete one (step 4(d)ii, 4(d)iii, 4(d)iv, and 4(d)v). Moreover, FKs also generate inclusion dependencies between classes, which are represented by means of subsumption axioms (step 4(d)vi). Then, as if they were relations, each KD and FK is related to its attributes (steps 4(c)vii and 4(d)vii respectively) by means of functional properties (steps 4(c)viiA, 4(c)viiB, 4(d)viiA and 4(d)viiB), stating them as mandatory (steps 4(c)viiC and 4(d)viiC). Steps 4(c)viiD and 4(d)viiD state that the value of an attribute of a relation must be the same if we go firstly through the KD or FK to find it. Similarly, step 4(d)viiE states that the value of an attribute of a FK coincides with the value of the pointed KD.

Finally, steps 5(a)i states that relations are disjoint between them; while 5(a)iiA states that KDs are disjoint between them. Steps 6 and 7 state that

all tuples of relations and keys are exactly those in the extensions of existing relations and keys, respectively.

Steps 4(c)viiD and 4(d)viiD should actually show property composition (i.e. $r_A \sqsupseteq r_K \circ r_{K_A}$ and $r_A \sqsupseteq r_F \circ r_{F_A}$), which guarantees the consistency between the different paths from a relation to its attributes. Nevertheless, this does not belong to $\mathcal{SHOIN}$and, consequently, cannot be represented in OWL. Thus, we have expressed it by means of a number restriction on a transitive property, which is equivalent in this case to the composition, but pushes us from OWL-DL to OWL-Full. However, not imposing it would not affect the reasoning performed in next section.

Moreover, this translation algorithm does not require any Normal Form for the relational schema to be translated. However, the most normalized it is, the more explicit KDs we find. Therefore, we will get a richer translation if the original relational schema was in Boice-Codd Normal Form (BCNF).

We have implemented this translation algorithm in a prototype and validated the results by introducing them in Protege 3.1.1 (which can be downloaded from http://protege.stanford.edu).

## 5 Usage

Once we have detected correspondences between attribute domains by means of name similarity measures like in [BSM04] and validating it maybe by mining their values, our OWL translation will allow to deal with structure heterogeneities, following the classification of semantic heterogeneities in [GSS96]. In the relational model, this can be seen like deciding whether two relations are equivalent or not, i.e the transitive closure of their FDs coincide. Notice that if one schema is not in BCNF, some FDs would not be captured by the translation algorithm, and we may make some mistakes in the comparison.

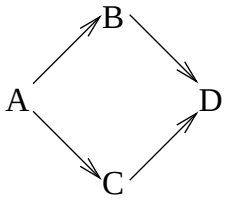
In order to be able to compare the schemas, we would need to define a property *det* showing that one class determines another. To do this, we need to use two DL constructs that do not belong to OWL, namely property union (i.e. role disjunction) and transitive closure of properties.

$$\begin{array}{rl} \text{Property union } (r \sqcup s)^{\mathcal{I}} & = \{(a,b) \mid (a,b) \in r^{\mathcal{I}} \vee (a,b) \in s^{\mathcal{I}}\} \\ \text{Transitive closure} \quad (r^{+})^{\mathcal{I}} & = \bigcup_{i \geq 1} (r^{\mathcal{I}})^{i} \end{array}$$

With this, *det* property is defined as follows:

$$det \equiv \Big( \bigsqcup_{R_R \in rels} \Big( \bigsqcup_{K_R \in KD(R_R)} (r_K \sqcup {r_K}^{-}) \sqcup \bigsqcup_{<F_R, pt> \in FK(R_R)} r_F \Big) \Big)^{+}$$

At this point it is important also to notice that if the relational schema is not a tree, we cannot guarantee that the transitive application of FKs results in a functional dependency. Let's pay attention to figure 3, where letters from A to D correspond to relations, and arrows show FKs between them. The query may show more than one value in D determined by each value of A. To guarantee




```
SELECT A.PK, D.PK
FROM A, B, D
WHERE A.FKB=B.PK AND B.FKD=D.PK
UNION
SELECT A.PK, D.PK
FROM A, C, D
WHERE A.FKB=C.PK AND C.FKD=D.PK;
```

**Fig. 3.** Example of non functionality in FKs

this is not happening, we should consider assertions and triggers, which have not been taken into account in this work. Under this conditions, the following theorems state how we can use *det* to automatically establish the equivalence between two schemas.

**Theorem 1.** *The tuples of $R_R$ that functionally determine values of an attribute $A_A$ are represented by:*

$$R \sqcap (\exists det.\exists r_A.A)$$

*Proof.* First of all, since we know that $r_K$, ${r_K}^-$ and $r_F$ are functional properties (steps 4(c)iii, 4(c)iv, and 4(d)iii) and the composition of functions is also functional, we can assure that each transitive chain in *det* determines exactly one value. Since we are assuming that it could not be that there is more than one chain of FKs from $R_R$ to $S_R$, and given that $r_A$ is functional, we can assure that $\exists det.\exists r_A.A \equiv (= 1\ det.\exists r_A.A)$.

Let's prove by induction on the length of the chain of FKs that the expression corresponds to the tuples that determine values of the attribute.

**Base case, there is no FK (i.e. $A_R \in int(R_R)$)** :
In step 4(b)i, we have defined $r_A$. By definition,$R \sqsupseteq \exists r_A.A$ (step 4(b)v). This implies

$$\exists r_A.A \equiv R \sqcap (\exists r_A.A)$$

Given $K_R$ the key of $R_R$, by definition of det, $(r_K \sqcup {r_K}^-)^+ \sqsubseteq det$. Therefore, $\forall a, a \in R^{\mathcal{I}} \Rightarrow (a,a) \in det^{\mathcal{I}}$. Again, given that $R \sqsupseteq \exists r_A.A$ (step 4(b)v),

$$R \sqcap (\exists r_A.A) \equiv R \sqcap (\exists det.\exists r_A.A)$$

Therefore, if $A_R \in int(R_R)$, we have shown that those tuples having values for attribute $A_R$ are exactly $R \sqcap (\exists det.\exists r_A.A)$.

**Inductive case, there is a chain of $n+1$ FKs (i.e. $A_R \notin int(R_R)$)** :
If so, there exists $S_R$ so that $A_R \in int(S_R)$ and there is a chain of $n+1$ FKs from $R_R$ to $S_R$. Thus, by induction hypothesis, $R \sqcap (\exists det.\exists r_B.B)$ shows those tuples of $R_R$ that would determine values of $B_R$ if $B_R \in int(T)$ and there is a chain of $n$ FKs; and $R \sqcap (\exists det.\exists r_F.\exists {r_K}^-.\exists r_A.A)$ corresponds to the tuples of $R_R$ that determine values of $A_R$ with a chain of $n+1$ FKs (being $K$ the key of $S_R$ and $F$ the FK of $T_R$ pointing to it).

By definition of $det$, $r_F$ and ${r_K}^-$ belong to it. Therefore, since it is a transitive closure, we know that $\exists det.\exists r_F.\exists {r_K}^-.S \sqsubseteq \exists det.S$. Thus,

$$R \sqcap (\exists det.\exists r_F.\exists {r_K}^-.\exists r_A.A) \sqsubseteq R \sqcap (\exists det.\exists r_A.A)$$

Moreover, since we assume that FKs have a tree structure, they must coincide. It cannot be that other tuples of $R_R$ than $R \sqcap \exists det.\exists r_F.K$ determine values of $K_R$.

$$R \sqcap (\exists det.\exists r_F.\exists {r_K}^-.\exists r_A.A) \equiv R \sqcap (\exists det.\exists r_A.A)$$

Thus, $R \sqcap (\exists det.\exists r_A.A)$ shows the instances of $R_R$ determining exactly one value for attribute $A_R$. □

**Lemma 1.** *Given two schemas $E_R^1$ and $E_R^2$ over the sets of attributes $atts^1 \supseteq atts^2$, and two relations $R_R^1 \in E_R^1$ and $R_R^2 \in E_R^2$, then the transitive closure of the KDs of $R_R^2$ is contained in that of $R_R^1$ ($R_R^1 \models R_R^2$) iff:*

$$\forall A_R \in {atts}^2 : R^2 \sqcap (\exists {det}^2.\exists r_A^2.A) \sqsubseteq \bot \vee R^1 \sqcap (\exists {det}^1.\exists r_A^1.A) \sqsubseteq R^2 \sqcap (\exists {det}^2.\exists r_A^2.A)$$

*Proof.* We must prove both directions:

$R_R^1 \models R_R^2$ :

If $R_R^1 \models R_R^2$, then the transitive closure of KDs of $R_R^2$ is a subset of that of $R_R^1$ (${R_R^2}^+ \subseteq {R_R^1}^+$) for the attributes of $E_R^2$. Therefore, $\forall A_R \in {atts}^2, \forall K_R \subseteq int(R_R^2) : K_R \rightarrow A_R \in {R_R^2}^+ \Rightarrow K_R \rightarrow A_R \in {R_R^1}^+$.
Since by theorem 1, $R \sqcap (\exists det.\exists r_A.A)$ shows the instances of $R_R$ determining exactly one value for attribute $A_R$, either does not exists any KD $K_R \rightarrow A_R$ in ${R_R^2}^+$ (i.e. $R^2 \sqcap (\exists {det}^2.\exists r_A^2.A) \sqsubseteq \bot$) or it exists and is fulfilled by both relations $R_R^1$ and $R_R^2$, which means that

$$R^1 \sqcap (\exists {det}^1.\exists r_A^1.A) \sqsubseteq R^2 \sqcap (\exists {det}^2.\exists r_A^2.A)$$

$\forall A_R \in {atts}^2 : R^2 \sqcap (\exists {det}^2.\exists r_A^2.A) \sqsubseteq \bot \vee R^1 \sqcap (\exists {det}^1.\exists r_A^1.A) \sqsubseteq R^2 \sqcap (\exists {det}^2.\exists r_A^2.A)$ :

Let's suppose not, i.e. there exists $K_R \rightarrow A_R \in {R_R^2}^+$ so that $K_R \rightarrow A_R \notin {R_R^1}^+$ and $R^1 \sqcap (\exists {det}^1.\exists r_A^1.A) \sqsubseteq R^2 \sqcap (\exists {det}^2.\exists r_A^2.A)$. By definition of KD, there should be an instance of $R_R^1$, that does not determine a value of $A_R$, while belonging to $R_R^2$ it does determine one. However, we also know that all instances of $R^1$ that determine one value of $A$ are subsumed by those instances of $R^2$ that determine one value of $A$. Moreover, since $K_R \rightarrow A_R \in {R_R^2}^+$, $R^2 \sqcap (\exists\ {det}^2.\exists r_A^2.A) \not\sqsubseteq \bot$. Therefore, we arrive to a contradiction and the hypothesis must be false.

□

**Corollary 1.** *Given two schemas $E_R^1$ and $E_R^2$ over the same attributes atts, and two relations $R_R^1 \in E_R^1$ and $R_R^2 \in E_R^2$, $R_R^1$ is equivalent to $R_R^2$ ($R_R^1 \equiv R_R^2$) iff:*

$$\forall A_R \in atts : R^1 \sqcap (\exists {det}^1.\exists r_A^1.A) \equiv R^2 \sqcap (\exists {det}^2.\exists r_A^2.A)$$

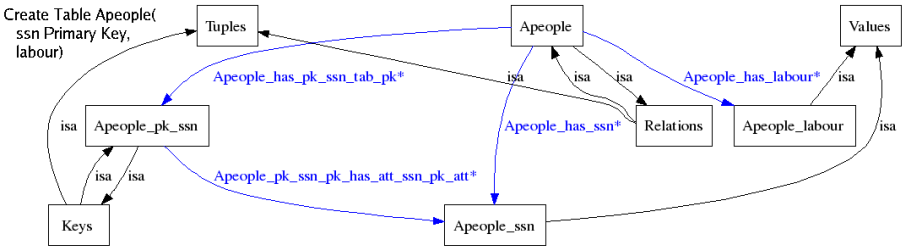



**Fig. 4.** Schema with an attribute admitting nulls

Let's consider two schemas that show people data. Schema A contains only one relation, which has an attribute "labour" that admits null values. On the other hand, schema B has specialized people, so that this attribute is in another relation "women", which does not allow null values. Figures 4 and 5 show the views generated by the "Ontoviz" tab of Protege 3.1.1 for those schemas, from the corresponding OWL representation (all generated classes are shown, but only $r_K$, $r_F$ and $r_A$ properties have been depicted).

Once we have detected, for example by means of name similarity, the equivalences between both sets of attributes (i.e. $Apeople\_ssn \equiv Bpeople\_ssn$, and $Apeople\_labour \equiv Bwomen\_labour$), we should state it in our knowledge base and check whether $Apeople \sqcap (\exists Adet.Apeople\_has\_ssn.Apeople\_ssn) \equiv Bpeople \sqcap (\exists Bdet.Bpeople\_has\_ssn.Bpeople\_ssn)$ and $Apeople \sqcap (\exists Adet.Apeople\_has\_labour.Apeople\_labour) \equiv Bpeople \sqcap (\exists Bdet.Bwomen\_has\_labour.Bwoman\_labour)$. If so, we can assure that $Apeople \equiv Bpeople$.

## 6 On reasoning algorithms

Our translation needs to be expressed in OWL-Full just because of the equivalences in steps 4(c)viiD and 4(d)viiD (the rest of classes and properties belong to $\mathcal{SHIF}$, which corresponds to OWL-Lite, as we can see in [HPS04]). These steps generate a transitive property whose cardinality needs to be constrained. This combination is not allowed in OWL-DL, because it results in undecidable reasoning problems in general, as we can see in [Hor05]. However, in our case, this will not result in infinite models.

On the other hand, the definition of "det" falls clearly out of OWL and nowadays there is no reasoner that supports transitive closure and union of properties. However, it seems to us an essential tool, not only in our case, but for ontologies in general. Moreover its worse case complexity is well known and coincides with that of $\mathcal{SHIQ}$(as can be seen in [BCM$^+$03]), which do have implemented reasoners with good response time in the common cases. DL is a relatively young area. Therefore, as expressed in [Hor05], we can expect some improvements in the expressive power of reasoners. Moreover, as shown in theorem 1, our problem is equivalent to obtaining the closure of a set of FD which is known to be decidable.

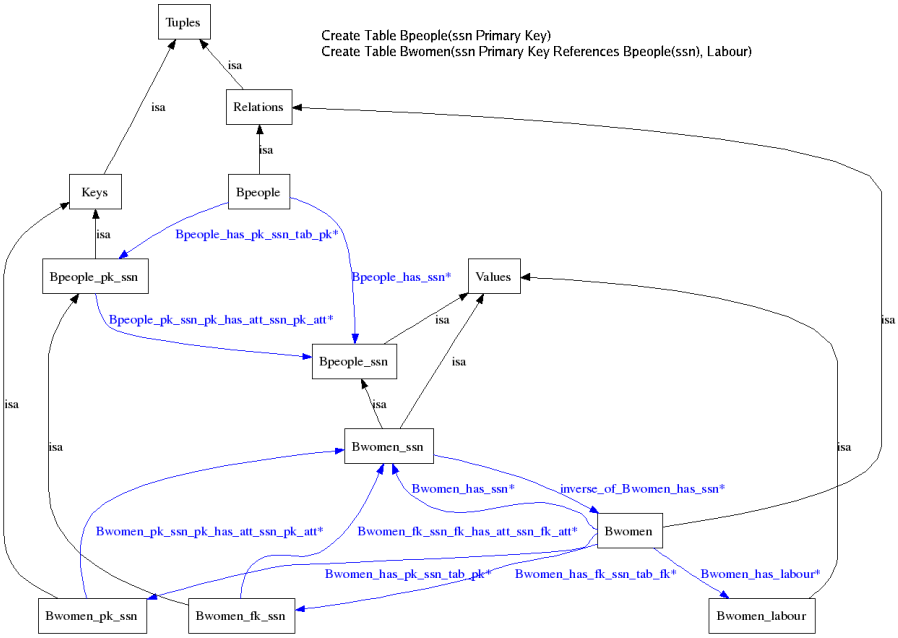



**Fig. 5.** Schema with an specialization forbidding nulls

In fact, it is an easy problem (a polynomial algorithm is in [AHV95]). Therefore, while no general algorithm is provided, we could use an efficient ad-hoc one.

## 7 Conclusions

In this work, we have shown an automatic translation from the information contained in the catalog of a relational DBMS to OWL. This translation allows to use a standard DL algorithm like subsumption to reason on the integration of different data sources. The reasoning is based on FDs whose transitive property difficulties the implementation of general algorithms. However, because of the existence of polynomial algorithms for the obtaining of the closure of FDs, we can assure that an ad-hoc tool will provide good performance. As future work, besides the implementation of such reasoning tool, we plan to extend the translation to deal with domains and checks.

## Acknowledgements

Our work has been partially supported by the Spanish Research Program PRONTIC and FEDER under project TIN2005-05406.

## References


[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Ast04] I. Astrova. Reverse Engineering of Relational Databases to Ontologies. In *Proc. of 1st European Semantic Web Symposium (ESWS 2004)*, volume 3053 of *LNCS*, pages 327–341. Springer, 2004.

[BB93] A. Borgida and R. J. Brachman. Loading Data into Description Reasoners. In *Proc. of 1993 ACM SIGMOD Int. Conf. on Management of Data*, pages 217–226. ACM Press, 1993.

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[BHS05] F. Baader, I. Horrocks, and U. Sattler. Description Logics as Ontology Languages for the Semantic Web. In *Mechanizing Mathematical Reasoning*, volume 2605 of *LNCS*, pages 228–248. Springer, 2005.

[BL98] T. Berners-Lee. Relational Databases on the Semantic Web, September 1998. http://www.w3.org/DesignIssues/RDB-RDF.html.

[BSM04] J. D. Bo, P. Spyns, and R. Meersman. Assisting Ontology Integration with Existing Thesauri. In *Proc. of OTM Confederated Int. Conf. (ODBASE'04)*, volume 3290 of *LNCS*, pages 801–818. Springer, 2004.

[dLC05] C. Pérez de Laborda and S. Conrad. Relational.OWL - A Data and Schema Representation Format Based on OWL. In *Proc. of 2nd Asia-Pacific Conf. on Conceptual Modelling (APCCM2005)*, pages 89–96, 2005.

[EN89] R. Elmasri and S. Navathe, editors. *Fundamentals of Database Systems*. Benjamin/Cummings, 1989.

[GSS96] M. García-Solaco and F. Saltor. *Object-Oriented Multidatabase Systems*, chapter Semantic Heterogeneity in Multidatabase Systems, pages 129–202. Prentice Hall, 1996.

[Hor05] I. Horrocks. Applications of Description Logics: State of the Art and Research Challenges. In *Proc. of 13th Int. Conf. on Conceptual Structures (ICCS 2005)*, volume 3596 of *LNCS*, pages 78–90. Springer, 2005.

[HPS04] I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Journal on Web Semantics*, 1(4):345–357, 2004.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In *Proc. of 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in LNAI, pages 161–180. Springer, 1999.

[SSV02] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the Semantic Web. In *Proc. of 2002 ACM Symposium on Applied Computing (SAC)*, pages 1100–1107. ACM, 2002.

[W3C04] W3C. Ontology Web Language (OWL)-Reference, February 2004. http://www.w3.org/TR/owl-ref.