Ontology Engineering Relationally

Jaroslav Pokorný, Jana Pribolová, Peter Vojtáš

Tech.Report No 2009/2 **April 2009**

Charles University Prague
Department of Software Engineering
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

Ontology Engineering Relationally

Jaroslav POKORNÝ [a], Jana PRIBOLOVÁ [b] and Peter VOJTÁŠ [a]
[a] *Department of Software Engineering, Charles University, Prague, Czech Republic*
[b] *Institute of Computer Science, P. J. Šafárik University, Košice, Slovakia*

Abstract. The retrieval problem is one of the main reasoning problems for ontology based systems. The retrieval problem for concept C consists in finding all individuals a which satisfy $C(a)$. We present ontology transformation which can help to improve evaluating queries over (sublanguage of) OWL ontologies. Our solution is based on translating retrieval concepts into relational algebra expressions and consequently to SQL queries over a database constructed from the original ontology. Ontology transformation into database is ontology-dependent but fully automatic and it is provided by system TORQue.

Keywords. Knowledge base, Ontology, Relational database, Query, Query evaluation

Introduction

Ontology is a knowledge base that can describe concepts, their instances and relations between them. Definitions of concepts and relations are formulated in an ontology language, for more detail see Section 1.2. One of the main kinds of inferencing for knowledge based systems is so-called *retrieval problem* [2]. The retrieval problem in ontologies is equivalent to querying in relational databases.

In present there are lot of methods and tools for creating, designing and reasoning the ontologies. Some of them have been influenced by techniques of relational databases, e.g. [8]. However the problem is an effective maintenance and reusing them through the running queries. There exist some open source systems that can handle ontology with large amount of the data. We have an experience with Sesame [23] within the projects described in [7,10]. But we have found out that query evaluation is very expensive. However the problem is not just problem of Sesame but also of other systems. To speed up the evaluation of some specific queries (those known in advance) the transformation to optimized indexes is very effective [11].

In this work we would like to present a model of ontology storage based on relational databases without loosing any ontological advantage. Combination of ontologies and relational databases has the advantage that all query optimization techniques provided by relational database systems can be used. This makes possible to work with huge datasets. We also present test results and compare them with other systems to consider contributions and disadvantages of our approach.

There are some research groups interested in storing ontologies into databases. One of the prominent projects is realized by the system HAWK [19] and its predecessor project DLDB [12]. These systems work similarly to ours, i.e. they recompute IS-A hi-

erarchy of concepts and store it into a database. Ontology queries can be then translated into standard SQL queries. After that the queries are evaluated by the relational database system. Unfortunately, the problem of these systems is that the query answers are not complete [9].

Another remarkable approach working with OWL-QL language and a relational database is described in [13]. The translation process of OWL-QL queries to SQL queries for the class-based relations can be found in [14].

For RDF [22] data, the simplest variant of relational approach to ontology engineering is to load all RDF triples into one table with three columns as it is in the case of Sesame version 1 in some kinds of its repositories or in the system called DataPile [3]. Sesame upgrade – Sesame version 2 has some other improvements. Instances names are stored out of the mentioned triple-table to speed up join-queries. This system also stores triples into more than one table to spread out large amount of triples.

Other approaches have a small number of tables, each to store concepts names, role names, individuals, and assertions about concepts, respectively. This systems are similar to the previous ones because all instances are stored into single table, for more details see [9,18].

There exist other experiments to improve ontology querying based on reasoning with logic databases [5,4]. They combine description logic expressions with logic programs (LP) and create the description logic programs (DLP). They adapt LP reasoners to run DLP. But this approaches need to find efficient means for implementing the approach as a whole.

The rest of the report is organized as follows. In Section 1 we present a short introduction to relational databases and the description logic we deal with, namely $\mathcal{EL}$-description logic [1]. In Section 2 we present the framework for mapping an ontology expressed in $\mathcal{EL}$-description logic into a relational database. As a result we obtain a relational database scheme and extensions of relations. In Section 3 we provide a mapping transforming such abstract database into SQL language. The resulted SQL database will be composed from tables and views. Observations about ontology vs. conceptual modelling are presented in Section 4. Section 5 is devoted to experiments with our implementation called TORQue (Translation of Ontology into Relational Queries) and comparisons with system Sesame. Finally, Section 6 concludes the paper.

1. Ontologies and Relational Databases

Since our appraoch extends relational databases with ontologies we first remind terminology of relational databases and ontologies.

1.1. Relational Databases

First, we remind database notation. A *relational scheme* $R(\Omega)$ describes a relation named R with a set of attributes $\Omega = \{A_1, A_2, \ldots, A_n\}, n \geq 1$. Alternatively, $R(\Omega)$ can be expressed directly as $R(A_1, A_2, \ldots, A_n)$. Every attribute A_i is associated with a set of values called *domain* D_i. A *relation* associated with $R(\Omega)$ is a subset of $D_1 \times D_2 \times \ldots \times D_n$. We denote it R^*. If it is evident whether R^* is a relation or its scheme, we can omit the symbol *. A relation consists of *tuples*. A relation scheme can be associated

with several *candidate keys* from which one *primary key* is chosen. Each key is a subset of Ω. In our approach we will suppose only the primary key of relation. The primary key attributes will be underlined.

A *relation database scheme* $\mathcal{D}$ is a pair $(\mathbf{R}, \mathbf{I})$, where $\mathbf{R}$ is a set of relational schemes and $\mathbf{I}$ is a set of integrity constraints over $\mathbf{R}$. A *relational database* $\mathcal{D}^*$ associated with $\mathcal{D}$ consists of the relations associated with the relational schemes of $\mathbf{R}$. The database is consistent iff it satisfies the constraints of $\mathbf{I}$.

Now we will present a few operations enabling to build new relations. Such a set of operations is called *relational algebra*. Its most important set operations include *union*, *intersection*, and *difference*. For database querying we need mainly *(natural, θ, equi) join*, *projection*, and *selection*. These operations (except difference) are specified in rows 3–7 of Table 1. Row 8 – 10 explain some usual predicates over relations.

Table 1. Relational algebra syntax and semantics.

#	Name	RA syntax	RA Semantics
1	*Extensional relation*	$R(A_1, \ldots, A_n)$ $S(B_1, \ldots, B_m)$	$R^* \subseteq \Pi_{i=1}^{n} D_{A_i}$ $S^* \subseteq \Pi_{i=1}^{m} D_{B_i}$
2	*Actual domain*	$Adom_i$	$Adom_i \subseteq D_i$
3	*Intersection*	$R \cap S$	$R^* \cap S^*$, where $n = m$ and $\forall i \in [1, n] : D_{A_i} = D_{B_i}$
4	*Union*	$R \cup S$	$R^* \cup S^*$, where $n = m$ and $\forall i \in [1, n] : D_{A_i} = D_{B_i}$
5	*Natural Join*	$R \bowtie S$	$(x, y, z) \in (R \bowtie S)^*$ if $(x, y) \in R^*$ and $(y, z) \in S^*$
5a	*θ-Join*	$R[\theta]S$	$(x, y, z, w) \in (R[\theta]S)^*$ if $(x, y) \in R^*$ and $(z, w) \in S^*$ and $y\theta z$
6	*Projection*	$R[A]$	$x \in (R[A])^*$ if $\exists y, (x, y) \in R$
7	*Selection*	$R(\sigma)$	$x \in (R(\sigma))^*$ if $\sigma(x) = TRUE$ and $x \in R^*$
8	*Inclusion*	$R \subseteq S$	$R^* \subseteq S^*$
9	*Equality*	$R = S$	$R^* = S^*$
10	*tuple in relation*	$R(a, b)$	$(a, b) \in R^*$

In practice, relational databases are stored and maintained by a relational database management systems (RDBMS).

1.2. Description Logic vs. OWL

We describe an ontology through the Description Logic (DL), for more details see [2]. In this work we deal only with $\mathcal{EL}$-description logic without role constructors and role assertion constructions (see [1]).

In Table 2 we list some constructors and possible assertions, their syntax and semantics in the columns DL Syntax and DL Semantics, respectively. In practice we depict an ontology as OWL[20] (Web Ontology Language) document, therefore in the Table 2 there is a column to describe constructors and assertions in OWL language with its DL equivalent in the same line. Note also, we do not deal here with concrete domains

Table 2. A part of OWL 2 $\mathcal{EL}$ and $\mathcal{EL}$-DL Syntax and Semantics.

#	Name	OWL Syntax	DL Syntax	DL Semantics
1	*Atomic concept*	`owl:Class`	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
2	*Atomic role*	`Property`	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
3	*Functional role*	`owl:Functional Property`	R	$R^{\mathcal{I}} : \Delta^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}}$
4	*Top concept*	`owl:Thing`	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
5	*Conjunction*	`owl:intersectionOf`	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
6	*Full existential quantification*	`owl:someValuesFrom`	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C\}$
7	*at most number restriction*	`owl:maxCardinality`	$(\leq nR)$	$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \lvert b.(a,b) \in R^{\mathcal{I}} \rvert \leq n\}$
8	*at least number restriction*	`owl:minCardinality`	$(\geq nR)$	$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \lvert b.(a,b) \in R^{\mathcal{I}} \rvert \geq n\}$
9	*Definition*	`owl:equivalentClass`	$A := C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
10	*Inclusion*	`owl:subClassOf`	$A \sqsubseteq B$	$A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
11	*Atomic Concept assertation*	`rdf:type`	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
12	*Role assertation*	`Property`	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

and do not distinguish among object-data variants of syntax (e.g. ObjectMaxCardinality, ObjectMinCardinality, ObjectSomeValuesFrom) in OWL2 ([21]) ontology.

Basic elements of DL are *concepts* and *roles*. Elementary descriptions are *atomic concepts* (denoted A, B) and *atomic roles* (denoted R). Concept constructions can be built from them inductively according to the syntax rules denoted in Table 2 (rows 4 – 8). Concept constructions are denoted C, D, which we consider to be nonterminal symbols, these are not names of concepts.

We divide names of concepts, roles and instances into categories as follows:

Label	Set of all	Condition
$\mathbf{NC}$	atomic concepts	
$\mathbf{NFR}$	atomic functional roles	
$\mathbf{NNR}$	atomic non-functional roles	$\mathbf{NNR} \cap \mathbf{NFR} = \emptyset$
$\mathbf{NR}$	atomic roles	$\mathbf{NR} = \mathbf{NNR} \cup \mathbf{NFR}$
$\mathbf{NI}$	instances	

In this work we use the word concept instead of concept names for short.

In order to define formal semantics of concepts and roles, we consider an interpretation $\mathcal{I}$ that consist of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every atomic role $R \in \mathbf{NNR}$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to every functional role $R \in \mathbf{NFR}$ a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}}$ (Table 2, rows 1 – 3). The interpretation function is extended to concept descriptions by the inductive definitions shown in the Table 2, in rows 4 – 8.

A *DL knowledge base* usually consists of a set of *terminological axioms* (called *TBox*) and a set of *assertional axioms* or *assertions* (often called *ABox*). TBox axioms are shown in Table 2, rows 9 and 10, and axioms of ABox are described in rows 11 and 12. Interpretation $\mathcal{I}$ satisfies axiom α if it satisfies the condition in column DL Semantics.

Then $\mathcal{I}$ is called a *model* of α. We write $\mathcal{T} \models \alpha$ (or $\mathcal{A} \models \alpha$) where α is terminological (assertional) axiom.We denote the set of terminological axioms by $\mathcal{T}$, the set of assertions we write as $\mathcal{A}$. Then a knowledge base $\mathcal{O}$ is specified as $\mathcal{O} = (\mathcal{T}, \mathcal{A})$.

In DL equivalence (equality) of concepts (concept constructions) is usually denoted by $\equiv$. In our DL we consider only special type of equivalence. The equality whose left-hand side is atomic concept A is called a definition of A. In this report we denote definition equality as the symbol $:=$. We assume, there are no cycles in definitions.

We emphasize that, all important concepts have a name, i.e. all of them are atomic. Non-atomic concepts – concept constructions – can only help to define some of the atomic. For example atomic concept $Student$ can be defined as $Person$ who takes $Course$. The situation is also the same in OWL. All atomic concepts have an URI. Non-atomic ones are in OWL called blank nodes and do not have any URI. They can have only internal unique name. Non-atomic concepts would be denoted by non-terminals in a formal grammar.

Let us note that we understand both ABox $\mathcal{A}$ and TBox $\mathcal{T}$, as sets of assertions about concepts or individuals, respectively. The set of assertions can be divided into two categories. One, denoted by subscript $\mathcal{E}$, includes *extensional assertions*, the second one comprehends as set of additionally *deduced assertions* and it is denoted by subscript $\mathcal{D}$. The set $\mathcal{T}_\mathcal{D}$ is derived with respect to (symmetric, transitive) properties of the assertions $:=$ and $\sqsubseteq$. For all $\mathcal{T}$ sets the following holds:

- $\mathcal{T} = \mathcal{T}_\mathcal{E} \cup \mathcal{T}_\mathcal{D}$,
- $\mathcal{T}_\mathcal{E} \cap \mathcal{T}_\mathcal{D} = \emptyset$.

Also mention that $\mathcal{A} = \mathcal{A}_\mathcal{E} \cup \mathcal{A}^\mathcal{T}_\mathcal{D}$. The set $\mathcal{A}^\mathcal{T}_\mathcal{D}$ depends on the TBox $\mathcal{T}$, because we derive assertions on basis of $\mathcal{A}_\mathcal{E}$ and TBox assertions. If it is evident which $\mathcal{T}$ inducted $\mathcal{A}^\mathcal{T}_\mathcal{D}$, we omit superscript $\mathcal{T}$. We note the set of TBox subsumptions as $T_\sqsubseteq$ and the set of equalities as $T_:=$.

Example 1.1 Let us have a knowledge base $\mathcal{O}$ with the following sets
$\mathbf{NC} = \{Org, Univ, Faculty, Person, Employee, FStuff, Student, Dean\}$
$\mathbf{NFR} = \{hasName, hasEmail\}$, and $\mathbf{NNR} = \{headOf\}$.

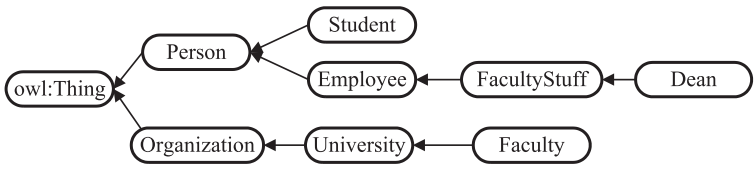

Figure 1. IS-A hierarchy with respect to $\mathcal{T}_\mathcal{D}$.

Org, $Univ$ and $FStuff$ are abbreviations of Organization, University, and FacultyStuff, respectively.

Assume that we have following assertions in $\mathcal{T}_\mathcal{E}$ (see Figure 1):

$$\begin{aligned} Univ &\sqsubseteq Org, & Student &\sqsubseteq Person, \\ Employee &\sqsubseteq Person, & FStuff &\sqsubseteq Employee, \\ Dean &\sqsubseteq FStuff, & Univ &\sqsubseteq Faculty, \\ Dean &:= Person \sqcap \exists headOf.Faculty \end{aligned}$$

Assume that we have following assertions in $\mathcal{A}_\mathcal{E}$:

$hasName(P_1, Name_1)$,	$headOf(P_2, F_2)$,	$Employee(P_2)$,
$hasName(P_2, Name_2)$,	$headOf(D_1, F_1)$,	$Person(P_1)$,
$hasName(P_3, Name_3)$,	$headOf(P_3, F_3)$,	$Person(P_3)$,
$hasName(S_1, Name_4)$,	$Student(S_1)$,	$Faculty(F_1)$,
$hasName(U_1, Name_5)$,	$Dean(D_1)$,	$Faculty(F_2)$,
$hasEmail(S_1, Email_1)$,	$Univ(U_1)$,	$Faculty(F_3)$,
$hasEmail(D_1, Email_2)$,		

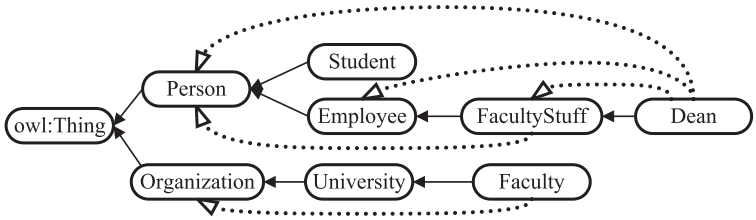

Figure 2. IS-A hierarchy with respect to the whole $\mathcal{T}$.

The TBox assertions that belongs to $\mathcal{T}_\mathcal{D}$ are shown in Figure 2 as dotted arrows. All concepts are also subsumed by top concept also. We can derive the following assertions of the $\mathcal{A}_\mathcal{D}^{\mathcal{T}_\sqsubseteq}$ from the subsumptions:

$Org(U_1)$,	$Org(F_3)$,	$Univ(F_1)$,	$Person(P_2)$,
$Org(F_1)$,	$Employee(D_1)$,	$Univ(F_2)$,	$Person(S_1)$,
$Org(F_2)$,	$FStuff(D_1)$,	$Univ(F_3)$,	$Person(D_1)$.

If we take definition of the concept equalities (in our example it is $Dean$ definition) into account, we receive the following:

$$\mathcal{A}_\mathcal{D}^{\mathcal{T}_=} = \{Dean(P_3)\}$$

Finally, if we take all assertions of $\mathcal{T}$ into account, then $\mathcal{A}_\mathcal{D}^{\mathcal{T}}$ contains:

$Org(U_1)$,	$Univ(F_1)$,	$Employee(P_3)$,	$FStuff(D_1)$,
$Org(F_1)$,	$Univ(F_2)$,	$Person(S_1)$,	$FStuff(P_2)$,
$Org(F_2)$,	$Univ(F_3)$,	$Person(D_1)$,	$FStuff(P_3)$,
$Org(F_3)$,	$Employee(D_1)$,	$Person(P_2)$,	$Dean(P_3)$,
$Dean(P_2)$.			

Also assertions of the type $\top(I)$ are omitted.

2. Mapping From $\mathcal{O}$ To $\mathcal{D}$

The ontology translation creates database of relations that satisfy integrity constraints. Nevertheless for optimization task we need to define so called potential domains and potential ranges of roles.

Definition 2.1 Let $M \subseteq \mathcal{A}$. A concept $A \in \mathbf{NC}$ is said to be a *potential domain* for role $R \in \mathbf{NR}$ with respect to the set M if there is $R(a, b) \in M$ such that $A(a) \in M$. The set of potential domains for role R with respect to M is denoted $\mathbf{PD}_R^M$. If it is evident which M induces potential domains, we omit superscript M and denote them as $\mathbf{PD}_R$.

Note, that definition of potential domains depends on $\mathcal{A}$.

Algorithms 1, 2 use potential domains with respect to the $\mathcal{A}_\mathcal{E}$ therefore the following example takes into account only $\mathcal{A}_\mathcal{E}$ assertions.

Example 2.2 According to Example 1.1 and the Definition 2.1 the set of potential domains for roles $hasName$ (grey soft-boxes in Figure 3), $hasEmail$ (depicted as rectangles around the soft-boxes representing concepts in Figure 3) and $headOf$ are:

$$\begin{aligned}
\mathbf{PD}^{\mathcal{A}_\mathcal{E}}_{hasName} &= \{Univ, Person, Employee, Student\},\\
\mathbf{PD}^{\mathcal{A}_\mathcal{E}}_{hasEmail} &= \{Student, Dean\},\\
\mathbf{PD}^{\mathcal{A}_\mathcal{E}}_{headOf} &= \{Person, Dean\}.
\end{aligned}$$

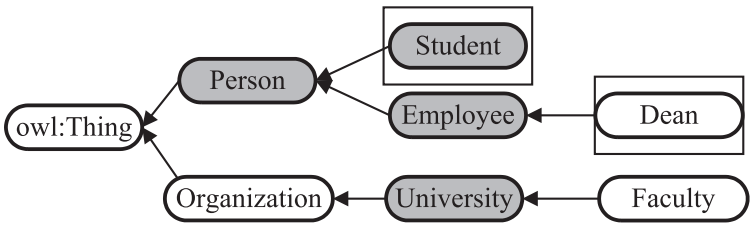

Figure 3. Potential and valid domains.

Example 2.3 According to Example 1.1 and the Definition 2.1 the set of potential domains for roles $hasName$, $hasEmail$ and $headOf$ are:

$$\begin{aligned}
\mathbf{PD}^{\mathcal{A}^{\mathcal{T}_\sqsubseteq}_\mathcal{D}}_{hasName} &= \{Org, Univ, Person, Employee, Student\},\\
\mathbf{PD}^{\mathcal{A}^{\mathcal{T}_:=}_\mathcal{D}}_{hasName} &= \{Univ, Person, Employee, Student\},\\
\mathbf{PD}^{\mathcal{A}^{\mathcal{T}}_\mathcal{D}}_{hasName} &= \{Org, Univ, Person, Employee, FStuff, Dean, Student\},\\
\mathbf{PD}^{\mathcal{A}^{\mathcal{T}_\sqsubseteq}_\mathcal{D}}_{hasEmail} &= \{Person, Student, Employee, FStuff, Dean\},\\
\mathbf{PD}^{\mathcal{A}^{\mathcal{T}_:=}_\mathcal{D}}_{hasEmail} &= \{Student, Dean\},\\
\mathbf{PD}^{\mathcal{A}^{\mathcal{T}}_\mathcal{D}}_{hasEmail} &= \{Person, Student, Employee, FStuff, Dean\}.
\end{aligned}$$

We divided concepts and roles into various categories. The reason is that we will translate them from different categories into relations (attributes) in different ways. Searching for the domains of functional properties helps us to translate this properties in the attributes of relations representing domains instead of translating them into new relations. But we decided to do experiments also with reducing a number of the relations with the attribute representing functional role. We used two observations. First, if there exist assertions $C \sqsubseteq D$ and $C(a)$, then $D(a)$ can be deduced as well. Second, if knowledge base includes the assertions $D(b)$, $R(a, c)$ and $R(b, d)$, then it is not necessary encode the functional role R into an attribute of both, relational scheme representing C and relational scheme representing D. We choose so called valid domain to encode R. In our case it is only relational scheme representing the concept D. But if the knowledge base does not consider any assertion of the type $R(b, d)$, then the functional role R is encoded as attribute of the relational scheme representing the concept C.

Definition 2.4 Let $M \subseteq \mathcal{A}$. A *valid domain* for role $R \in \mathbf{NFR}$ with respect to the set M is a potential domain $A \in \mathbf{PD}_R^M$ with property that does not exists $B,\ B \in \mathbf{PD}_R^M$ so that $A \sqsubseteq B \in \mathcal{T}$ and $A \equiv B \notin \mathcal{T}$, as well as $B \sqsubseteq A \notin \mathcal{T}$. The set of valid domains for role R is denoted as $\mathbf{VD}_R^{M,\mathcal{T}}$. If we are interesting in valid domains with respect to whole $\mathcal{T}$, we can omit the superscript $\mathcal{T}$.

Example 2.5 According to the facts from Examples 1.1, 2.2 and the Definition 2.4 we obtain the valid domains of the functional roles:

$$\mathbf{VD}_{hasName}^{\mathcal{A}_\mathcal{E}} = \{Univ, Person\}, \quad \mathbf{VD}_{hasEmail}^{\mathcal{A}_\mathcal{E}} = \{Student, Dean\}.$$

According to Example 1.1 and the Definition 2.4 the sets of valid domains with respect to $\mathcal{A}_\mathcal{D}^{\mathcal{T}_:=}$, $\mathcal{A}_\mathcal{D}^{\mathcal{T}_\sqsubseteq}$ and $\mathcal{A}_\mathcal{D}^{\mathcal{T}}$ are:

$$\begin{array}{llll}
\mathbf{VD}_{hasName}^{\mathcal{A}_\mathcal{D}^{\mathcal{T}_\sqsubseteq}} = \{Org, Person\}, & \mathbf{VD}_{hasEmail}^{\mathcal{A}_\mathcal{D}^{\mathcal{T}_\sqsubseteq}} = \{Person\}, \\
\mathbf{VD}_{hasName}^{\mathcal{A}_\mathcal{D}^{\mathcal{T}_:=}} = \{Univ, Person\}, & \mathbf{VD}_{hasEmail}^{\mathcal{A}_\mathcal{D}^{\mathcal{T}_:=}} = \{Student, Dean\}, \\
\mathbf{VD}_{hasName}^{\mathcal{A}_\mathcal{D}^{\mathcal{T}}} = \{Org, Person\}, & \mathbf{VD}_{hasEmail}^{\mathcal{A}_\mathcal{D}^{\mathcal{T}}} = \{Person\}.
\end{array}$$

The mechanism of choosing valid domains of the role R from the potential ones is simple. The potential domains of the role R are divided into maximal groups linearly ordered by subsumption. For each pair of the potential domains A, B of such a group, either $A \sqsubseteq B \in \mathcal{T}$ or $B \sqsubseteq A \in \mathcal{T}$. It is possible that the intersection of these groups is not empty. The reason is that a concept can be subsumed by two other concepts.

Then for each such a group we choose the most general concept with respect to subsumption hierarchy. This ideas are validated by the following claim.

Claim 2.6 For each $R \in \mathbf{NFR}$ and for each maximal group $\{A_1, A_2, \ldots, A_n\} \subseteq \mathbf{PD}_R^{\mathcal{A}_\mathcal{E}}$ so that $A_1 \sqsubseteq A_2 \sqsubseteq \ldots \sqsubseteq A_n$, only for A_n

$$A_n \in \mathbf{VD}_R^{\mathcal{A}_\mathcal{E}}$$

holds.

Moreover, for every $i < n$ so that $A_i \sqsubseteq A_{i+1} \sqsubseteq \ldots \sqsubseteq A_n$

$$A_n \in \mathbf{VD}_R^{\mathcal{A}_\mathcal{E}}$$

holds.

Definition 2.7 The role $R \in \mathbf{NFR}$ is said to be a *role defined on the concept A with respect to the set of assertions M* if $A \in \mathbf{VD}_R^{M,\mathcal{T}}$. We denote the set of all roles defined on the concept A as $isIn\mathbf{VD}_A^{M,\mathcal{T}}$. Similarly as in Definition 2.4 if we use all assertions of $\mathcal{T}$, we can omit $\mathcal{T}$ and denote the set of all roles defined under concept A as $isIn\mathbf{VD}_A^M$.

Example 2.8 In the running example of this paper an interesting point is to compute e.g. $isIn\mathbf{VD}_{Person}^{\mathcal{A}_\mathcal{E}}$:

$$isIn\mathbf{VD}_{Person}^{\mathcal{A}_\mathcal{E}} = \{hasName\}$$

Definition 2.9 A *potential range* for role $R \in \mathbf{NR}$ (with respect to $\mathcal{A}$) is a concept A for which there exists $R(a,b) \in \mathcal{A}$ and $A(b) \in \mathcal{A}$. We denote the set of potential ranges of role R as $\mathbf{PR}_R$.

Let us illustrate how to find out potential range for roles supposing situation shown in Example 1.1.

Example 2.10 For role $hasName$ the set

$$\mathbf{PR}_{hasName} = \{String\},$$

similarly for $hasEmail$ the set

$$\mathbf{PR}_{hasEmail} = \{String\}.$$

For role $headOf$ it is the set

$$\mathbf{PR}_{headOf} = \{Faculty\}.$$

Now we will show how to create a relational database scheme from the ontology. By $resource$ we denote the attribute whose domain contains some identifiers, e.g. OIDs, URI, etc.

Algorithm 1 Let $\mathcal{O}$ be a knowledge base with TBox $\mathcal{T}$ and ABox $\mathcal{A}$. $\mathcal{T}$, $\mathcal{A}$, concept's names, and role's name are translated into relational database $\mathcal{D} = (\mathbf{R}, \mathbf{I})$. Here $\mathbf{R}$ denotes a relational database scheme consisting of basic relational schemes and view definitions using relational algebra expressions (RA expressions). $\mathbf{I}$ denotes a set of integrity constraints. The translation is done by induction as described below.

First part of translation depends only on the language, the second part depends also on ABox and the last depends on the TBox, too.

Note that names of attributes are motivated by RDF *(subject, predicate, object)* and *resource* terminology.

The construction is based on the following steps:

First translation steps are based solely on the description logic language.

1. For all $A \in \mathbf{NC}$ we add to $\mathbf{R}$ new relation T_A with scheme $T_A(\underline{resource})$.
2. For all $R \in \mathbf{NNR}$ we add to $\mathbf{R}$ a relation scheme $T_R(\underline{subject, object})$.

Following translation steps depend on the ABox (and deduced valid domains)

3. For all $R \in \mathbf{NFR}$ for which $\mathbf{VD}_R^{\mathcal{A}_\mathcal{E}} = \emptyset$, we add to $\mathbf{R}$ a new relational scheme $T_R(\underline{subject}, object)$.
4. For all $A \in \mathbf{NC}$ for which $isIn\mathbf{VD}_A^{\mathcal{A}_\mathcal{E}} = \{R_1, R_2, \ldots, R_n\}, n \geq 1$ we modify $T_A(resource) \in \mathbf{R}$ to relation $T_A^{mod} \in \mathbf{R}$ with scheme
$$T_A^{mod}(\underline{resource}, R_1.object, \ldots, R_n.object)$$

The following translations depend on the TBox. First we deal with definitions:

5. For all $A \in \mathbf{NC}$ such that there is a concept construction C with $A := C \in \mathcal{T}$ we add to $\mathbf{R}$ a new relation T_A^{view} with scheme $T_A^{view}(\underline{resource})$ and view definitions so that (S_D and S_E are defined in step 6):
   - If $C := D \sqcap E$ then
     $$T_A^{view} = T_A \cup (S_D \cap S_E)$$
   - If $C := \exists R.D$ and $R \in \mathbf{NNR}$ or $\mathbf{VD}_R^{\mathcal{A}\varepsilon} = \emptyset$ then
     $T_A^{view} = T_A \cup (T_R(subject, object)$
     $[T_R.object = S_D.resource]$
     $S_D(resource))[T_R.subject]$.
   - If $C = \exists R.D$ and $R \in \mathbf{NFR}$ and $\mathbf{VD}_R^{\mathcal{A}\varepsilon} \neq \emptyset, n > 0$ then
     $T_A^{view} = T_A \cup (S_R^{rec}(subject, object)$
     $[S_R^{rec}.object = S_D.resource]$
     $S_D(resource))[S_R^{rec}.subject]$
     where S_R^{rec} is the RA expression for reconstruction of the role R from appropriate columns of T_B^{mod} tables
     $$S_R^{rec}(subject, object) = \bigcup_{B \in \mathbf{VD}_R^{\mathcal{A}\varepsilon}} \left(T_B^{mod}[resource, R.object]\right)$$
     were D and E are concept constructions. Each atomic concept is also a concept construction by definition. Here we assume that this is a lossless encoding of all ABox information about R.
6. A concept construction can be an atomic concept A. In this case $S_A = T_A$. For a non-atomic concept construction C such that there is in $\mathcal{T}$ no definition with right hand side C and C is a subconstruction of a concept definition in $\mathcal{T}$, then we create a new RA expression S_C with the only attribute *resource* so that:
   - If $C = D \sqcap E$ then
     $$S_C = (S_D \cap S_E)$$
   - If $C = \exists R.D$ and $R \in \mathbf{NNR}$ or $\mathbf{VD}_R^{\mathcal{A}\varepsilon} = \emptyset$ then
     $S_C = (T_R(subject, object)[T_R.object = S_D.resource]S_D(resource))$
     $[T_R.subject]$
   - If $C = \exists R.D$ and $R \in \mathbf{NFR}$ and $\mathbf{VD}_R^{\mathcal{A}\varepsilon} \neq \emptyset$ then
     $S_C = (S_R^{rec}(subject, object)[S_R^{rec}.object = S_D.resource]S_D(resource))$
     $[S_R^{rec}.subject]$
7. To transform axioms in $\mathcal{T}$, we add the following integrity constraint to $\mathbf{I}$:
   - if $C \equiv D \in \mathcal{T}$ and C, D are non-atomic concepts constructions, then $S_C = S_D \in \mathbf{I}$,
   - if $C \sqsubseteq D \in \mathcal{T}$ then $S_C \subseteq S_D \in \mathbf{I}$.

Note that for a functional role the attribute *R.object* can be added to more than one relation of the type T_C^{mod}. Then data $R(a, b),\ R(c, d)$ of such a role R can be split up into several relations with the attribute *R.object*, e.g. it can happen, that $T_{C_1}^{mod}(a, \ldots, b, \ldots)$ and $T_{C_2}^{mod}(c, \ldots, d, \ldots)$. If $\mathbf{VD}_R$ (and equivalently $\mathbf{PD}_R$) is empty (typically when there are no concepts with ABox axioms witnessing non-empty intersection with the subject attribute of the role R), data in R is handled in same way as for non-functional roles.

Example 2.11 According to the previous examples and Algorithm 1 we acquire database with this relations (we omit in the table the T in the $T_A, T_A^{mod/view}, T_R$ notation) created by the Step 1:

$Org(\underline{resource})$, $Person(\underline{resource})$, $Dean(\underline{resource})$,
$Univ(\underline{resource})$, $Employee(\underline{resource})$, $Student(\underline{resource})$,
$Faculty(\underline{resource})$, $FStuff(\underline{resource})$.

The Step 2 creates following scheme:

$$headOf(\underline{subject}, \underline{object})$$

There are no results of the Step 3. However the results of the Step 4 are:

$Person^{mod}$ $(\underline{resource}, hasName.object)$,
$Univ^{mod}$ $(\underline{resource}, hasName.object)$,
$Student^{mod}$ $(\underline{resource}, hasEmail.object)$,
$Dean^{mod}$ $(\underline{resource}, hasEmail.object)$.

Step 5 creates relation scheme $Dean^{view}(\underline{resource})$ for which the following holds:

$$Dean^{view} = Dean \cup \big(Person \cup (headOf(subject, object) \\ [headOf.object = Faculty.resource] \\ (Faculty)[headOf.subject])\big)$$

The right-hand side of the equality is a construction created in Step 6. Finally, Step 7 defines the following:

$$\mathbf{I}_{\mathcal{T}} = \{Faculty \subseteq Univ, Univ \subseteq Org, Dean \subseteq FStuff, FStuff \subseteq Employee, \\ Emplyee \subseteq Person, Student \subseteq Person\}$$

In our work the computation of valid domain is done with respect to $\mathcal{A}_{\mathcal{E}}$. We use neither $\mathcal{A}_{\mathcal{D}}^{\mathcal{T}_{\sqsubseteq}}$, $\mathcal{A}_{\mathcal{D}}^{\mathcal{T}_{:=}}$ nor $\mathcal{A}_{\mathcal{E}}^{\mathcal{T}}$ for it. Supposing computation of valid domains were done with respect to $\mathcal{A}_{\mathcal{E}}^{\mathcal{T}}$, top concept would be only one valid domain per each role. It means that the table representing top concept would have as many columns as the number of functional role is. Or without considering top concept as a valid domain for every role, each functional role would be transformed into database as attribute of all direct successors whose at least one instance (with respect to whole $\mathcal{T}$) participates on the role.

Suppose computation of valid domains were done with respect to $\mathcal{A}_{\mathcal{D}}^{\mathcal{T}_{:=}}$, there would be one or more tables to transfer only one role assertion. This situation seems to cause redundancy. Imagine if a was the instance of A known as extensional fact and B known as deducted fact and instance a took part in role R as assertion $R(a, b)$. Both A and B would be valid domains of the role R what would mean that either $\langle a, b\rangle \in T_A^{mod}[T_A.resource, R.object]$ or $\langle a, b\rangle \in T_B^{mod}[T_B.resource, R.object]$ or both.

Lemma 2.12 *For all $T_A^{mod}(\Omega_A), T_B^{mod}(\Omega_B) \in \mathbf{R}$ so that $A \sqsubseteq B \in \mathcal{T}_{\mathcal{E}}$ and $A \equiv B \notin \mathcal{T}_{\mathcal{E}}$, as well as $B \sqsubseteq A \notin \mathcal{T}_{\mathcal{E}}$, the following equality holds:*

$$(\Omega_A \setminus \{resource\}) \cap (\Omega_B \setminus \{resource\}) = \emptyset$$

We created a relation scheme. Finally we can handle the data e. g. assertions from ABox $\mathcal{A}$. In fact, this data is translated into tuples of the associated database relations.

Algorithm 2 Suppose that $\mathcal{T}$, $\mathbf{NC}$ and $\mathbf{NR}$ are translated into database scheme $\mathcal{D}$. ABox $\mathcal{A}$ is transferred into $\mathcal{D}$ by induction as follows:

1. If $B(a) \in \mathcal{A}_\mathcal{E} \cup \mathcal{A}_\mathcal{D}^{\mathcal{T}_\sqsubseteq}$ and also $B \in \mathbf{NC}$, then $\langle a \rangle \in T_B$.
2. If $R(a,b) \in \mathcal{A}$ and $R \in \mathbf{NNR}$, then $\langle a, b \rangle \in T_R$.
3. If $R(a,b) \in \mathcal{A}$ and $R \in \mathbf{NFR}$, then one of the following items:
   (a) if $\mathbf{VD}_R^{\mathcal{A}_\mathcal{E}} = \emptyset$ then
   $$\langle a, b \rangle \in T_R,$$
   (b) if there exists $A \in \mathbf{VD}_R^{\mathcal{A}_\mathcal{E}}$ so that $A(a) \in \mathcal{A}_\mathcal{E}$, then
   $$\langle a, b \rangle \in T_A^{mod}[T_A.resource, R.object],$$
   (c) if there exists $A \in \mathbf{PD}_R \setminus \mathbf{VD}_R^{\mathcal{A}_\mathcal{E}}$ so that $A(a) \in \mathcal{A}_\mathcal{E}$, then there exists a maximal sequence $A = B_1, B_2, \ldots, B_n \in \mathbf{NC}$ so that $B_n \in \mathbf{VD}_R^{\mathcal{A}_\mathcal{E}}$ and $B_i \sqsubseteq B_{i+1} \in \mathcal{T}_\mathcal{D}$ for $i = 1, \ldots, n-1$ Then
   $$\langle a, b \rangle \in T_{B_n}^{mod}[T_B.resource, R.object].$$

In fact, according to the rule 2 in Algorithm 1 we obtain in step 3b in Algorithm 2 a tuple $\langle a, x_1, x_2, \ldots, x_n \rangle$, where some x_i are possibly empty. This is expressed in databases by introducing a special *null* value. Similarly, we can consider result of 3c.

Note also, that in order to have loss less transport of data from ontology into a relational database we have to deal with the case when $\mathbf{VD}_R^{\mathcal{A}_\mathcal{E}} = \emptyset$. This can cause problems in querying, when one tuple of relation R is in T_R another are in T_C^{mod}.

One possibility is to assume that for all $R \in \mathbf{NFR}$

$$\{a : (\exists b) R(a,b) \in \mathcal{A}_\mathcal{E}\} \subseteq \bigcup \{a : B(a) \in \mathcal{A}_\mathcal{E} \;\&\; B \in \mathbf{PD}_R^{\mathcal{A}_\mathcal{E}}\} \tag{1}$$

Assumption 1 guaranties that steps 3b and 3c of algorithm 2 suffice to cover all data from a functional role R. We do not discuss this further in this paper.

Example 2.13 According to all facts mentioned in previous examples and steps 1 and 2 of the Algorithm 2 we obtain the following relational database storing our ontology:

$$\begin{aligned}
Org &= \{\langle F_1 \rangle, \langle F_2 \rangle, \langle F_3 \rangle, \langle U_1 \rangle\}, & Employee &= \{P_2, D_1\},\\
Univ &= \{\langle F_1 \rangle, \langle F_2 \rangle, \langle F_3 \rangle, \langle U_1 \rangle\}, & FStuff &= \{\langle D_1 \rangle\},\\
Faculty &= \{\langle F_1 \rangle, \langle F_2 \rangle, \langle F_3 \rangle\}, & Dean &= \{\langle D_1 \rangle\},\\
Person &= \{\langle P_1 \rangle, \langle P_2 \rangle, \langle P_3 \rangle, \langle D_1 \rangle, \langle S_1 \rangle\}, & Student &= \{\langle S_1 \rangle\},\\
headOf &= \{\langle D_1, F_1 \rangle, \langle P_2, F_2 \rangle, \langle P_3, F_3 \rangle\}
\end{aligned}$$

Using step 3b we modify just the following relations:

$$\begin{aligned}
Univ^{mod}(resource, hasName.object) &= \{\langle U_1, Name_5 \rangle\},\\
Person^{mod}(resource, hasName.object) &= \{\langle P_1, Name_1 \rangle, \langle P_3, Name_3 \rangle\},\\
Student^{mod}(resource, hasEmail.object) &= \{\langle S_1, Email_1 \rangle\}\\
Dean^{mod}(resource, hasEmail.object) &= \{\langle D_1, Email_2 \rangle\}
\end{aligned}$$

Finally we apply step 3c with the following changes:

$$\begin{aligned}
Person(resource, hasName.object) = \{&\langle P_1, Name_1 \rangle, \langle P_2, Name_2 \rangle,\\
&\langle P_3, Name_3 \rangle, \langle S_1, Name_4 \rangle\}.
\end{aligned}$$

Let us mention that the relation $Dean^{view}$ looks as follows:

$$Dean^{view} = \{\langle D_1\rangle, \langle P_2\rangle, \langle P_3\rangle\}.$$

ABox assertions correspond to relational algebra constructions given by the Table 3.

Table 3. Translation of DL Syntax to relational algebra.

#	Name	DL Syntax	RA construction
1	*Concept assertation*	$C(a)$	$\langle a\rangle \in T_C$
2	*Role assertation*	$R(a,b)$	$\langle a,b\rangle \in T_R$

Theorem 2.14 *Let $\mathcal{O}$ be a finite knowledge base, A is a concept and a be an individual. We have that*

$$\langle a\rangle \in T_A^*[resource] \Longrightarrow \mathcal{O} \models A(a).$$

Proof.
Assume $\langle a\rangle \in T_A^*[resource]$ but $\mathcal{O} \not\models A(a)$. It means there exists interpretation $\mathcal{I}$ for which $a^{\mathcal{I}} \notin A^{\mathcal{I}}$. It is necessary to find out why individual a should be in relation T_A^*. Inserting the tuples into relations representing concepts is done in Algorithm 2 either in steps 1, 3b or 3c. Hence there is only one potential insert where it could arise an unsoundness, namely insert in the step 3c. Other cases contradict because in 1 and 3b there is assumption $A(a) \in \mathcal{A}$ and this is true if and only if for any interpretation $\mathcal{I}$ $a^{\mathcal{I}} \in A^{\mathcal{I}}$ holds.

Let us assume that statement of Theorem fails due to a data record obtained in the step 3c of the algorithm 2. Let A be a concept such that $A = B_n$ and $\exists B_1, B_2, \ldots, B_{n-1} \in \mathbf{PD}_R^{\mathcal{A}_\varepsilon} : B_1 \sqsubseteq B_2 \sqsubseteq \ldots \sqsubseteq B_n$ and $\langle a\rangle \in T_A^*[resource]$. Then in all interpretation $\mathcal{I}$ which are model of $\mathcal{O}$ we have $a^{\mathcal{I}} \in A^{\mathcal{I}}$.

□

3. Relational to SQL mapping

A relational database expressed in SQL consists usually of definitions of basic relations and views. Each relational scheme and its associated integrity constraints are represented by a standalone `CREATE TABLE` statement. One possibility how to approach the constructions obtained by Algorithm 1 in SQL is based on the strategy to assign one `CREATE TABLE` for each scheme $T_A(resource)$ and $T_R(subject, object)$, respectively. The same will be hold for $T_A^{mod}(resource, R_1^A.object, ..., R_n^A.object)$. `NULL` values in its rows for those $R_i^A.object$ attributes for which $A(a)$ holds and $a \notin R_i^A[subject]$. The relations coming from defined concepts will be modeled by views in the SQL database. We remind that views in SQL are virtual relations defined by a `SELECT` statement in `CREATE VIEW` definition. Sometimes views can be materialized, i.e. they look like basic tables in the database. Obviously the `SELECT` in a `CREATE VIEW` definition can contain other views, etc.

We will use here a little different approach to express the root `SELECT` in `CREATE VIEW` directly with nested `SELECT`s and `INTERSECT` operations.

A decision which method to use in practice depends on real data used in the ontology and performance considerations.

For purposes of this report we will call the set of operations projection, equi-join, intersection and union as a *relational DL-algebra*. Our next steps are directed to finding rules for translation of expressions of this algebra to SQL language. Particularly for each expression E which has been generated by Algorithm 1 we show that there is one `SELECT-FROM-WHERE` statement implementing E. To do this process as straightforward as possible we start with two lemmas that let us to modify the original expression into expressions with more convenient properties. First we will propagate all inner $\cap s$ to higher $\cap s$ in E.

Lemma 3.1 *Every expression of relational DL-algebra E is equivalent to a set expression composed of $\cap$ and operators and operands containing no occurrence of $\cap$ and $\cup$.*

Proof.
We prove the claim by induction on the number of intersections in E. The basis, zero $\cap$s or $\cup$, is trivial. The only one place where $\cap$ and $\cup$ are nested in a relational subexpression of E is $T_D(resource)$ in the right-side of an equi-join.

Let $T_D = T_{D_1} \diamondsuit T_{D_2}$, where $\diamondsuit$ is $\cap$ or $\cup$. Since T_{D_i} have fewer $\diamondsuit$s than T_D, we can suppose that they fulfill the statement. Then, due to the additivity of operators $\diamondsuit$,

$$T_R(subject, object)[T_R.object = T_D.resource]T_D(resource)$$

can be replaced by equivalent expression

$$T_R(subject, object)[T_R.object = T_{D_1}.resource]T_{D_1}(resource)\diamondsuit$$
$$T_R(subject, object)[T_R.object = T_{D_2}.resource]T_{D_2}(resource).$$

We have omitted projection in our consideration since it does not influence these set transformations.

□

The second case concerns nested expressions given by $T_D(resource)$ looking like

$$(T_{R_1}(subject, object)[T_{R_1}.object = T_{C_1}.resource]T_{C_1}(resource))[T_{R_1}.subject]$$

for $C \in \mathbf{NC}$ and $R_1 \in \mathbf{NR}$. We show how to replace the whole expression E by a sequence of equi-joins of all participated relations followed by a projection. Such expressions can be easily transformed to the SQL language.

Lemma 3.2 *Let $D = (R_1(A, B)[R_1.B = T_1.C]T_1(C))[R_1.A]$. Then*

$$(R(A, B)[R.B = D.C]D(C))[A] \equiv$$
$$((R(A, B)[R.B = R_1.A]R_1(A, B))[R_1.B = T_1.C]T_1(C))[R.A]$$

Proof.
After a substitution for D we obtain
$(R(A,B)[R.B = D.C](R_1(A,B)[R_1.B = T_1.C]T_1(C))[R_1.A])[R.A] \equiv$
$(R(A,B)[R.B = R_1.A](R_1(A,B)[R_1.B = T_1.C]T_1(C))[R_1.A])[R.A] \equiv$
$((R(A,B)[R.B = R_1.A]R_1(A,B))[R_1.B = T_1.C]T_1(C))[R.A]$

Remark: the distributive laws for union and intersection

i. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
ii. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

We can adjust the our relational expressions into a standard form, i.e. either intersection of unions or union of intersections of atomic relational expressions E_i, each of them being composed from equi-joins and projections.

□

Theorem 3.3 *(relational DL-algebra → SQL transformation). Let E be an expression of relational DL-algebra. Then there is an SQL-expression S defining under the same interpretation $\mathcal{I}$ the same relation.*

Proof.
By Lemma 3.1 and 3.2 we can assume that E is an intersection of unions. Then E is translated to SQL as an SQL-expression composed from `INTERSECT` operators, simple `SELECT`s, union of `SELECT`s and `TABLE` R, where R is a relation associated with an atomic concept.

Each simple `SELECT` arises from relational expressions considered in Lemma 3.2. We will construct them as follows. Suppose the expression $E_1 = (R(A,B)[R.B = D.C]D(C))[A]$ where $D = (R_1(A,B)[R_1.B = T_1.C]T_1(C))[R_1.A]$. The resulted SQL expression is obtained easily from the expression equivalent to E_1 according to Lemma 3.2 as

```
SELECT R.A
FROM R, R1,T1
WHERE R.B=R1.A AND R1.B=T1.C
```

Obviously, the relational expression with equi-join can be further nested. By induction it is possible to prove that it means to add other equalities to the `WHERE` clause.

□

Example 3.4 Let tables `PERSON(RESOURCE)`, `HEAD_OF(SUBJECT,OBJECT)`, and `CLLG(RESOURCE)` be defined by associated `CREATE TABLE` statements. Abbreviating the respective `resource`, `object`, and `subject`, we will to transform the construction (here we assume that $T_{Dean} = \emptyset$)

$$T^{view}_{Dean} = T_{Person} \cap (T_{headOf}(headOf.subject, headOf.object)\\ [headOf.object = Cllg.resource]\\ T_{Cllg}(Cllg.resource))\\ [headOf.subject],$$

for the concept *Dean* as follows:

```
CREATE VIEW DEAN (RESOURCE) AS
(TABLE PERSON
 INTERSECT
 SELECT HEAD_OF.SUBJ
   FROM HEAD_OF, CLLG
  WHERE HEAD_OF.OBJ = CLLG.RES)
```

To do the database scheme complete we need to transform integrity constraints into SQL `CONSTRAINT` statements included in `CREATE TABLE` expression. Subset containment and equality of two sets C and D can be expressed by referential integrities between two tables, if C and D are in **NC**.

4. Ontologies vs. Conceptual Modelling

Now we mention some observations on different approaches to database modelling. The approach to build a database dependent only on existence of a knowledge base is completely different from that one in the world of usual transactional databases. Such databases arise mostly from a conceptual scheme that says, e.g., that entities $Student$ and $Course$ participate in the relationship $takesCourse$. In any conceptual model $takesCourse(S_1, C_1)$ implies $Student(S_1)$ and $Course(C_1)$. In the world of ontologies the assertion $Student(S_1)$ holds if S_1 is a person, i.e. $Person(S_1)$ should hold. According to definition of the $Student$ concept, for $Student(S_1)$ we can deduce that $Person(S_1)$. But such a student need not be enrolled in any course. The same is usual in transactional databases. On the other hand, we can have $takesCourse(S_2, C_1)$ in the same ontology without assertion that S_2 is a student or even a person. Also C_1 need not be in the list of courses.

Consequently, any similarity of ontology construction to conceptual modeling is only partial. Often we define new concepts representing an entity type via other concepts and roles, i.e., via other entity types and relationship types (see, e.g., $Student$ via $takesCourse$). It reminds so called reverse engineering in which from relationship types participating entity types are recognized and defined. In fact, this knowledge is partial, since the notion $Course$ only belongs to $\mathbf{PR}^{\mathcal{A}}_{takeCourse}$. Maybe yet something else can be characterized by the role $takesCourse$. These phenomena can be explained with the help of notions of close world assumption (CWA) and open world assumption (OWA). The *close world assumption* implies that everything we do not know is false, while the *open world assumption* states that everything we do not know is undefined. The presence of OWA in our considerations results also in situation that usual principles known from ISA-hierarchies do not work here. For example, attribute inheritance is generally not guaranteed. The reason is that the existence of an entity attribute can be not deduced only from extensional assertions using roles.

5. Experiments

5.1. System Setup

To verify the real query evaluation cost we have designed and implemented the system TORQue – Translation of Ontology into Relational Queries. The TORQue is an implementation of theoretical ideas described in previous chapters.

In experiments we wanted to evaluate scalability of our system and compare it with non-commercial system. As it was said we have had an experience with the system Sesame, therefore we have chosen it. For this propose we used the TORQue in version 1.0 and the Sesame in version 2.2.1. We used MySQL version 5.0 as the underlying RDBMS.

We have done the tests on desktop computer with 1.83 GHz Intel Core2 CPU, 2 GB of RAM, 80 GB hard disk, Windows XP Professional OS, and Java 6.

5.1.1. OWL Datasets

Test data is extensional data created over Univ-Bench ontology [6]. The data, that is generated by UBA generator, has synthetic origin. Each dataset can consist of one or more OWL files. The datasets are called *LUBM*. To identify the datasets LUBM we use notation LUBM$_n$, where n is a number of universities contained in the dataset. In our experiments we created 3 sets of the test data: LUBM$_1$, LUBM$_5$ and LUBM$_{10}$, which include OWL files for 1, 5 and 10 universities, respectively. The largest one has over 100 MB size of the OWL files with instances saved on hard disk.

5.2. Loading Time and Repositories

In this section we want discuss loading time of the different storages. In Table 4 we listed data loading time for both systems, Torque as well as Sesame. For each dataset we show number of files that belong to the dataset and their total size on disk. Also we present number of triples of every dataset, loading time of the repository and repository size on disk.

Table 4. Data loading parameters

	Dataset data	Number of Files	Total Size (MB)	Number of Triples	Load Time (hh:mm:ss)	Repository Size (MB)
Sesame	LUBM$_1$	15	8.1	103 397	00:02:20	19.1
TORQue					00:33:18	18.7
Sesame	LUBM$_5$	93	50.5	646 128	00:49:37	117
TORQue					10:07:35	82.1
Sesame	LUBM$_{10}$	189	103	1 316 993	03:07:02	240
TORQue					40:26:11	185.5

The TORQue system seems to be economical considering database size on the disk that it takes. On the other hand, the loading time is much more worse than the loading the Sesame repositories. We should notice that loading time of the database created by TORQue consists of the time spent on computing things like potential domains and valid

domains, time spent on communication with the database and time spent on communication with the Sesame. Communication with Sesame cost us a lot of time. For now we ask the system Sesame everything about the ontology (names of the concepts, roles, subsumption and equality assertions, concepts assertion, and role assertions).

Let us mention that Sesame offers repository based on an RDBMS but we decided not to use this kind of repository. The reason is, that this repository does not provide any reasoning tasks. We remind also that Sesame does not find answers to almost all test queries. Even the memory repository is not usable in practice. So, our experiments use only native Sesame repository.

5.3. Test Queries

Our set of test queries consists of the 12 queries, 10 becomes from LUBM benchmark set, namely queries 1 – 8, 12 and 14. We preserve the numbering of the selected benchmark queries. The rest of queries are constructed by us (numbers 15 and 16). We have decided to add these queries to our sample because this queries use functional properties. The LUBM queries are presented in SPARQL language [25], what is a standard language recommended by W3C Consortium. However, we want to provide the tests of Sesame also with language SeRQL [24], what was, for Sesame, the first supported query language.

5.4. Query Response Time

The test system receives the test set of the queries (in SPARQL language) and *id* of the repository that should be tested. It runs query by query from the test sample and repeats them m times. The number m is in this case 10. For each query, each system, and each dataset we compute average response time. Figure 4 shows the graphs with test results for all datasets.

Each figure consists of two graphs. The graph on the left-hand side informs about all our test queries. The right-hand side one concerns the queries with complete answer. Order of the queries presented in all graphs is defined by the number of tuples in the query answers.

To verify completeness of query answers we used results presented in [6]. The really important feature of the system TORQue is that it computes the complete answers in all cases. However, the system Sesame returns (independently of the query language) only 91% answer in case of the query 7, 83% for queries 6 and 8. Even for the query 12 it does not find any answer though it should return some answer.

6. Conclusion

We have proposed a model of translation of the ontology to relational database and developed the system TORQue.

DBMSs with their wide scale of query optimization techniques offer us a lot of abilities to improve our system. One of them is a technique which speeds up query evaluation through indexing of some columns. It helps with relational joins. To further improve the indexes there is an idea inspired with Sesame 2. All names of concepts, roles

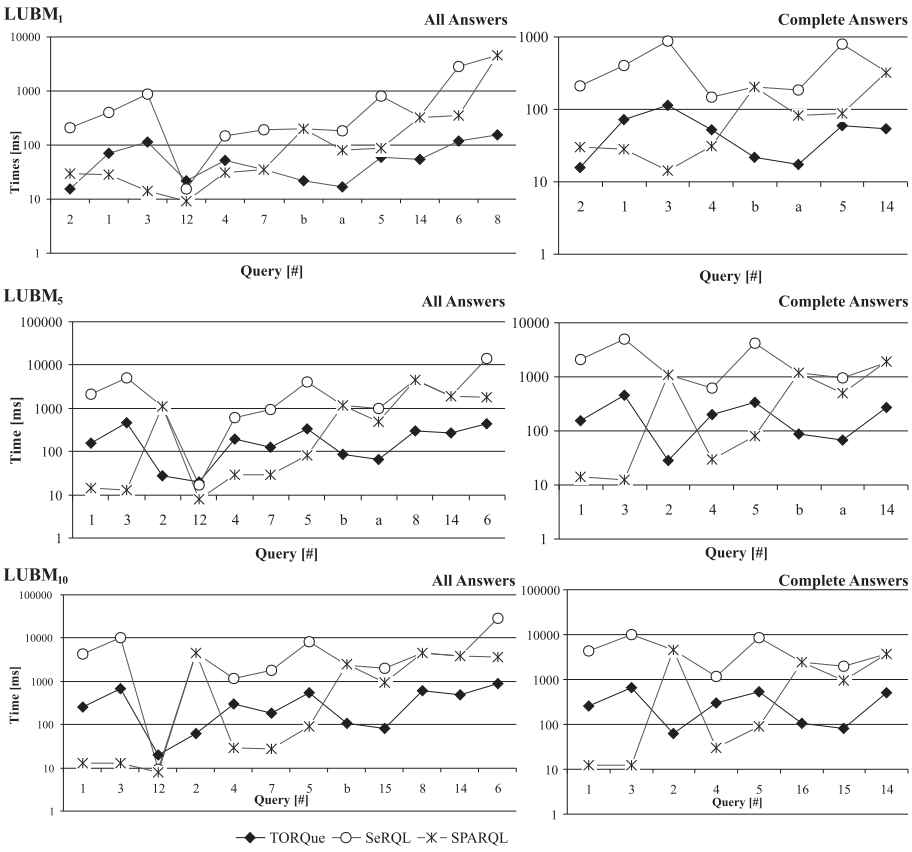

Figure 4. Test results for LUBM$_1$, LUBM$_5$, LUBM$_{10}$.

and instances refer to numeric indexes instead of to their origin names of string type. In databases it is more effective to join tables through numerical columns.

In the introduction we mentioned projects (described in [7,10]) which we are participated in. As we mentioned, required information in the systems was static. There was finite number of query types over the ontology. Therefore we could store ontology information into fixed index structure and the rest of application could work just with the indexes, what is really fast. But if we want to modify object of our interest we have to modify a configuration file (add or modify query that creates index). In future work we would like to supersede this ontology information retrieval with the system described in this work, i.e. the system TORQue.

To increase usability of our system, we want to implement SPARQL-compiler as well. Hence the query language SPARQL is a standard ontology query language based on RDF triples. Finally, we remind that relational databases offer additional operators to operators mentioned in Table 1. Therefore we also plan to go behind the $\mathcal{EL}$ description logic and to develop algorithms for translations of these extended constructions. Consequently, providing experiments with a wider class of description logics will be possible.

Acknowledgements

The work presented in the paper is supported by Slovak project VEGA 1/0131/09, Czech project GAČR 201/09/0990 and IS 1ET100300517.

References

[1] F. Baader, S. Brandt, and C. Lutz. *Pushing the EL Envelope.* In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005), 2005.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider: *The Description Logic Handbook. Theory, implementation, and application*. Cambridge University Press, 2003 United Kingdom.

[3] J. Dokulil, J. Tykal, J. Yaghob and F. Zavoral: *Semantic Web Repository And Interfaces*. In Mobile Ubiquitous Computing, Systems, Services and Technologies, 2007, 223–228.

[4] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits: *Combining Answer Set Programming with Description Logics for the Semantic Web.* Artificial Intelligence 172 (2008) 1495–1539

[5] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker: *Description Logic Programs: Combining Logic Programs with Description Logic*. WWW2003, Hungary, 2003.

[6] Y. Guo, Z. Pan, and J. Heflin: *LUBM: A benchmark for OWL knowledge base systems*. Journal of Web Semantics, 3:158–182, 2005.

[7] P. Gurský, T. Horváth, J. Jirásek, S. Krajči, R. Novotný, J. Pribolová, V. Vaneková, and P. Vojtáš: *User Preference Web Search - Experiments with a System Connecting Web and User*. To appear in COMPUTING AND INFORMATICS, 2009

[8] dldb2B. Hüsemann and G. Vossen: *Ontology Engineering from a Database Perspective* (Extended Abstract), ASIAN 2005, LNCS 3818, Springer, pages 49-63, 2005.

[9] N. Kottmann and T. Studer: *Improving semantic query answering*. DEXA 2007, LNSC 4653, Springer, 2007, pages 671 – 679.

[10] P. Návrat, P. Bartoš, M. Bieliková, L. Hluchý, and P. Vojtáš: *Acquiring, organizing and presenting information and knowledge in an environment of heterogeneous information sources.* In Tools for Aquisition, Organisation and Presenting of Information and Knowledge, 2007, pages 1 – 19.

[11] P. Gurský: *Towards better semantics in the multifeature querying*. Proceedings of Dateso 2006, CEUR Workshop Proceedings Vol. 176, pages 63-73, 2006.

[12] Z. Pan and J. Heflin: *DLDB: Extending relational databases to support semantic web queries*. In Workshop on Practical and Scaleable Semantic Web Systems, ISWC 2003, pages 109–113, 2003.

[13] M.J. Park, J.H. Lee, C.H. Lee, J. Lin, O. Serres, and C.W. Chung: *ONTOMS: An Efficient and Scalable Ontology Management System.* In: KAIST CS/TR-2005-246, 2005.

[14] M.J. Park, J.H. Lee, C.H. Lee, J. Lin, O. Serres, and C.W. Chung: *An Efficient and Scalable Management of Ontology.* Journal on Data Semantics X, LNCS 4900, pages 133-173, 2008.

[15] J. Pokorný, J. Pribolová, and P. Vojtáš: *Ontology Engineering Relationally*. In EJC 2009, T. Tokuda etal eds. Publ. University of Maribor 2009, 270-277

[16] J. Pokorný, J. Pribolová, and P. Vojtáš: *Translation of Ontology Retrieval Problem into Relational Queries*. In Dateso 2009, Eds. K. Richta, J. Pokorný, V. Snášel, CEUR Workshop Proceedings Vol. 471, ISSN 1613-0073, online CEUR-WS.org/Vol-471/paper5.pdf

[17] R. Ramakrishnan and J. Gehrke: *Database Management Systems*. Third Edition, McGraw Hill, 2003.

[18] R. Veselý and M. Bieliková: *Representing ontologies by relational model and query realization* (In Slovak). Proceedings of ZNALOSTI 2008, pages 280 – 290.

[19] HAWK: http://swat.cse.lehigh.edu/projects/index.html#hawk

[20] OWL: http://www.w3.org/2004/OWL/

[21] OWL2: http://www.w3.org/TR/2008/WD-owl2-profiles-20081008/

[22] RDF: http://www.w3.org/RDF/

[23] Sesame: http://www.openrdf.org/

[24] SeRQL: http://www.openrdf.org/doc/sesame2/users/ch09.html

[25] SPARQL: http://www.w3.org/TR/rdf-sparql-query/

[26] Univ-Bench Artificial data generator: http://swat.cse.lehigh.edu/projects/lubm/index.htm