Evaluating DBOWL: A Non-materializing OWL Reasoner based on Relational Database Technology

Maria del Mar Roldan-Garcia, Jose F. Aldana-Montes

University of Malaga, Departamento de Lenguajes y Ciencias de la Computacion
Malaga 29071, Spain,
(mmar,jfam)@lcc.uma.es,
WWW home page: http://khaos.uma.es

Abstract. DBOWL is a scalable reasoner for OWL ontologies with very large Aboxes (billions of instances). DBOWL supports most of the fragment of OWL covering OWL-DL. DBOWL stores ontologies and classifies instances using relational database technology and combines relational algebra expressions and fixed-point iterations for computing the closure of the ontology, called knowledge base creation. In this paper we describe and evaluate DBOWL. For the evaluation both the standard datasets provided in the context of the ORE 2012 workshop and the UOBM (University Ontology Benchmark) are used. A demo of DBOWL is available at http://khaos.uma.es/dbowl.

1 Introduction

With the explosion of Linked Data[1], some communities are making an effort to develop formal ontologies for annotating their databases and are publishing these databases as RDF triples. Examples of this are biopax[2] in the field of Life Science and LinkedGeoData[3] in the field of Geographic Information Systems. This means that formal ontologies with a large number (billions of) instances are now available. In order to manage these ontologies, current platforms need a scalable, high-performance repository offering both light and heavy-weight reasoning capabilities. The majority of current ontologies are expressed in the well-known Web Ontology Language (OWL) that is based on a family of logical formalisms called Description Logic (DL). Managing large amounts of OWL data, including query answering and reasoning, is a challenging technical prospect, but one which is increasingly needed in numerous real-world application domains from Health Care and Life Sciences to Finance and Government.

In order to solve these problems, we have developed DBOWL, a scalable reasoner for very large OWL ontologies. DBOWL supports most of the fragment of OWL covering OWL 1 DL. DBOWL stores ontologies and classifies instances

[1] http://linkeddata.org/
[2] http://www.biopax.org/
[3] http://linkedgeodata.org/About

using relational database technology. The state-of-the-art algorithm for achieving soundness and completeness in reasoning with expressive DL ontologies is the so-called Tableau procedure. Current Tableau-based implementations such as Pellet, Racer and HermiT show very good behavior in practice, but are completely memory-based and thus cannot cope with ontologies that have a large ABox. Several alternative approaches using disk-oriented implementations have been presented. These proposals can be classified into three categories. (1) Those which combine a DL main-memory based reasoner with a database, (2) Those which translate the ontology to Datalog and use a deductive database to evaluate it, and (3) Those which extend the database with reasoning capabilities. Our proposal follows a different approach: The state-of-the-art OWL reasoner Pellet[4] is currently used to classify the ontology Tbox. Information returned by Pellet is stored in a relational database. Class and property instances are also stored in the relational database as relation tuples. An algorithm which combines relational algebra expressions with fixed-point iterations is used to compute the closure of the of the ontology, called *knowledge base creation*. The use of an OWL reasoner, like Pellet, to classify the Tbox is crucial in our approach. This allows the capture of some Tbox inferences that cannot be obtained by other similar proposals such as those based on disjunctive datalog [1].

This paper presents a description and an evaluation of DBOWL. In order to evaluate DBOWL we use the standard datasets provided in the context of the ORE 2012 workshop[5] and the UOBM (University Ontology Benchmark) [2], the extremely well known benchmark for comparing ontology repositories in the Semantic Web. The rest of the paper is organized as follows. Section 2 introduces the theoretical concepts on which DBOWL is based. Section 3 describes the theoretical foundation of DBOWL, presenting the process for computing the ontology closure. Section 4 discusses the advantages and limitations of DBOWL. The evaluation of DBOWL is presented in Section 5. Finally Section 6 concludes the paper.

2 Preliminaries

2.1 The Relational Model

The relational model was first introduced by Ted Codd of IBM Research in 1970 in a classic paper [3]. The relational model is characterized by its simplicity and mathematical foundation. The relational model represents the database as a collection of *relations*.

A **domain** D is a set of atomic values. Atomic means that each value in the domain is indivisible as far as the relational model is concerned. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain, to help in interpreting its values. A **relation schema** R, denoted

[4] http://clarkparsia.com/pellet
[5] http://www.cs.ox.ac.uk/isg/conferences/ORE2012/

by $R(A_1, A_2, \ldots, A_n)$ is made up of a relation name R and a list of attributes $A_1, A_2, \ldots, A_n$. Each **attribute** A_i is the name of a role payed by some domain D in the relation schema R. D is called the **domain** of A_i and is denoted by $dom(A_i)$. The **degree** (or arity) of a relation is the number of attributes n of its relation schema. A **relation** (or **relation state**) r of the relation schema $R(A_1, A_2, \ldots, A_n)$, also denoted by $r(R)$, is a **mathematical relation** of degree n on the domains $A_1, A_2, \ldots, A_n)$, which is a **subset** of the **cartesian product** of the domains that define R:

$$r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$$

$r(R)$ is defined more informally as a set of n-tuples $r = \{t_1, t_2, \ldots, t_m\}$. Each n-tuple t is an ordered list of n values $t =< v_1, v_2, \ldots, v_n >$, where each value v_i, $1 \leq i \leq n$, is an element of $dom(A_i)$, or is a special NULL value. NULL is used to represent the values of attributes that may be unknown or may not apply to a tuple. This notation is used in the rest of the paper.

The terms **relation intension** for the schema R and **relation extension** for a relation state $r(R)$ are also commonly used. A relation is defined as a set of tuples. Mathematically elements of a set have *no order* among them.

2.2 The Relational Algebra

The basic set of operations for the relational model has an algebraic topology, and is known as the **Relational Algebra**. Operands in the Relational Algebra are Relations. Relational Algebra is closed with respect to the relational model: Each operation takes one or more relations and returns a relation. Given closure property, operations can be composed.

Relational Algebra operations enable a user to specify basic retrieval requests. The result of a retrieval is a new relation, which may have been formed from one or more relations. A sequence of relational algebra operations forms a **relational algebra expression**, the result of which will also be a relation that represents the result of a database query (or retrieval request). Therefore, it is possible to assign a new relation name to a relational algebra expression, in order to simplify its use by other relational algebra expressions. Such relations are called *idb* (Intensional) relations, unlike the relations in **R**, which are called *edb* (Extensional) relations.

Operations in relational algebra can be divided into two groups: Set operations from mathematical set theory (UNION ($\cup$), INTERSECTION ($\cap$), SET DIFFERENCE ($\setminus$) and CARTESIAN PRODUCT ($\times$)), and operations developed specifically for relational databases (SELECT (σ), which selects a *subset* of the tuples from a relation that satisfied a selection condition, PROJECT (π), which selects certain attributes from the relation and discard the other attributes, JOIN ($\bowtie$), which combines related tuples from two relations into single tuples) among others.

2.3 DBOWL Ontologies

In order to simplify the implementation of the reasoner, some restrictions are imposed on the OWL ontologies supported by DBOWL. Even so, the ontologies supported by DBOWL are expressive enough for real application in the Semantic Web. DBOWL covers all of OWL 1 DL including inverse, transitive and symmetric properties, cardinality restrictions, simple XML schema defined datatypes and instance assertions. Enumerate classes (a.k.a, nominals) are only partially supported.

Let P and Q be properties, x be an individual and n be a positive number, class descriptions in DBOWL ontologies are formed according to the following syntax rule:

$$C, D \rightarrow A\ (NamedClass) \mid \neg A\ (complementOf\ NamedClass) \mid$$
$$C \sqcap D\ (intersectionOf\ ClassDescriptions) \mid$$
$$C \sqcup D\ (unionOf\ ClassDescriptions) \mid \forall P.C\ (allValuesFrom) \mid$$
$$\exists P.C\ (someValuesFrom) \mid \exists P.\{x\}\ (hasValue) \mid$$
$$\{x_1, \ldots, x_n\}\ (oneOf) \mid >= nP\ (minCardinality) \mid <= nP\ (maxCardinality)$$

Tbox Axiom	**DL syntax**
SubClassOf	$A \sqsubseteq B$
equivalentClasses	$A \equiv B$
SubPropertyOf	$P \sqsubseteq Q$
equivalentProperty	$P \equiv Q$
disjointWith	$A \sqsubseteq \neg B$
inverseOf	$P \equiv Q^-$
transitiveProperty	$P^+ \sqsubseteq P$
symmetricProperty	$P \equiv P^-$
functionalProperty	$\top \sqsubseteq \leq 1P$
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$
domain	$\geq 1P \sqsubseteq A$
range	$\top \sqsubseteq \forall P.A$
Abox Axiom	**DL syntax**
class instance	$A(x)$
property instance	$P(x, y)$
sameAs	$x_1 \equiv x_2$

Table 1. DBOWL ontologies axioms

Table 1 shows the Tbox and Abox axioms for DBOWL ontologies. A and B are used for specifying Named Classes and C and D for specifying Class Descriptions. DBOWL assumes that all individuals are different unless the ontology includes an *owl:sameAs* assertion or you inferred it. This is important in real applications with a large number of individuals where usually it is easier to specify if two individuals represent the same resource than which individuals are different to others. The following restrictions are imposed on the DBOWL

ontologies. These restrictions are related more to the ontology syntax than to the ontology expressivity:

1. In the Tbox, all OWL constructors are supported. However, class descriptions always appear in the ontology as an equivalence or as a superclass of a Named Class. In an RDF/XML OWL ontology, class descriptions are always involved in the definition of a Named Class.
2. In the Abox, only assertions of Named Classes and Property Names are supported.
3. Only negation of Named Classes is allowed. Nevertheless, a negation of a class description could be included in the ontology defining a Named Class as equivalent to a Class Description and negating this Named Class.
4. Properties' domain and range must be Named Classes. In the same way, for asserting a complex property's domain or range we must define a Named Class as equivalent to a Class Description and use this Named Class as Property domain or range.
5. Only disjointness of Named Classes is allowed. As in the previous cases, a disjointness of a class description could be included in the ontology defining a Named Class as equivalent to a Class Description and disjoining this Named Class.

3 DBOWL Theoretical Foundations

In this section we present the theoretical foundations of our approach to scalable OWL reasoning. Although DBOWL is basically a Description Logic reasoner, it has been designed as an OWL reasoner. This implies that not all the DL inferences are supported. The main objective of DBOWL is to classify instances in Named Classes and Properties. In order to do this, for each Named Class and Property in the ontology, a *edb* relation $R_{A_1}(id), \ldots, R_{A_n}(id)$, $R_{P_1}(subject, object), \ldots, R_{P_m}(subject, object)$ is defined, being n and m the number of Named Classes and Properties in the ontology respectively. These relations contain one tuple for each individual or pair of individual asserted as member of such Named Class or Property.

3.1 Classification Function

In order to classify instances in Names Classes and Properties, we define a *classification function* $\mathcal{F}$ (see table 2). This function takes as input a DBOWL property axiom, a DBOWL domain or range axiom (see table 1), or an axiom $(A \equiv C)$, where C is a DBOWL class description and A is a Named Class in the ontology or an auxiliary name. The function define a new *idb* relation by means of a relational algebra expression, depending on the input type, or invoke the function with a new input.

For each Named Class in the ontology, a set of *idb* relations $S_{A_{i_0}}(id), \ldots, S_{A_{i_k}}(id), i : 1 \ldots n$ are defined. In the same way, for each Property in the ontology

a set of *idb* relations $S_{P_{j_0}}(subject, object), \ldots, S_{P_{j_l}}(subject, object)$, $j : 1 \ldots m$ are defined. The values of k and l depend on the number of axioms in the ontology evolving C_i and P_j respectively.

Each $S_{A_{i_x}}$, $x : 0 \ldots k$ has the following features (similarly for each $S_{P_{j_x}}$):

- $S_{A_{i_x}} = Q_{A_{i_x}}$, where $Q_{A_{i_x}}$ is a relational algebra expression,
- $S_{A_{i_0}} = R_{A_i}$,
- $S_{A_{i_{(x-1)}}}$ always occurs in Q_{i_x}, for $x : 1 \ldots k$, and
- if $S_{A_{j_r}}$ or $S_{P_{j_s}}$ occur in Q_{i_x}, they represent the last *idb* relation defined for A_j and P_j respectively.

3.2 Knowledge base Creation

In order to create the DBOWL knowledge base, function $\mathcal{F}$ is evaluate iteratively, defining the corresponding *idb* relations, until no new tuples are generated, i.e. until a fixed-point is reached. ($S_{A_{i_x}} = S_{A_{i_{(x-1)}}}$, $i : 1, \ldots, n$, and $S_{P_{j_x}} = S_{P_{j_{(x-1)}}}$, $j : 1, \ldots, m$).

In order to improve the efficiency of the evaluation, $\mathcal{F}$ is expressed as a composition of four functions, i.e.

$\mathcal{F} = \mathcal{F}_1 \circ \mathcal{F}_2 \circ \mathcal{F}_3 \circ \mathcal{F}_4$, where,

- $\mathcal{F}_1$ takes as input only axioms such as $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^-$, $P^+ \sqsubseteq P$, $P \equiv P^-$
- $\mathcal{F}_2$ takes as input only axioms such as $\geq 1P \sqsubseteq A$, $\top \sqsubseteq \forall P.A$
- $\mathcal{F}_3$ takes as input only axioms such as $A \sqsubseteq B$, $A \equiv B$, $A \equiv C \sqcap D$, $A \equiv C \sqcup D$, $A \sqsubseteq \forall P.C$, $A \equiv \exists P.C$, $A \equiv \neg B$, $A \equiv \{v_1, \ldots, v_n\}$, $A \equiv \exists P.\{v\}$
- $\mathcal{F}_4$ takes as input only axioms such as $A \equiv \exists P.\{v\}$

The algorithm proceeds as follows:

1. $\mathcal{F}_1$ is evaluated iteratively, defining the corresponding *idb* relations, until no new tuples are generated, i.e. until a fixed-point is reached. ($S_{P_{j_x}} = S_{P_{j_{(x-1)}}}$, $j : 1, \ldots, m$).
2. $\mathcal{F}_2$ is evaluated defining the corresponding *idb* relations ($S_{A_{i_x}}$, $i : 1, \ldots, n$).
3. $\mathcal{F}_3$ is evaluated iteratively defining the corresponding *idb* relations, until no new tuples are generated, i.e. until a fixed-point is reached. ($S_{A_{i_{(x+1)}}} = S_{A_{i_x}}$, $i : 1, \ldots, n$).
4. $\mathcal{F}_4$ is evaluated defining the corresponding *idb* relations ($S_{P_{j_{(x+1)}}}$, $j : 1, \ldots, m$).
5. Steps from 1 to 4 are repeated until no new tuples are generated by step 4, i.e. until a fixed-point is reached ($S_{P_{j_{(x+1)}}} = S_{P_{j_x}}$, $j : 1, \ldots, m$).

$\mathcal{F}(P \sqsubseteq Q)$	A new *idb* relation S_{Q_i} is defined as $\pi_{subject,object}(S_{Q_{(i-1)}}) \cup \pi_{subject,object}(S_{P_j})$.
$\mathcal{F}(P \equiv Q)$	A new *idb* relation S_{Q_i} is defined as $\pi_{subject,object}(S_{Q_{(i-1)}}) \cup \pi_{subject,object}(S_{P_j})$.
$\mathcal{F}(P \equiv Q^-)$	A new *idb* relation S_{P_i} is defined as $\pi_{subject,object}(S_{P_{(i-1)}}) \cup \pi_{object,subject}(S_{Q_j})$.
$\mathcal{F}(P^+ \sqsubseteq P)$	If (x, y) is a tuple in $S_{P_{(i-1)}}$ and (y, z) is also a tuple in $S_{P_{(i-1)}}$, then a new *idb* relation S_{P_i} is defined as $\pi_{subject,object}(S_{P_{(i-1)}}) \cup (\pi_{subject,object}((S_{P_{(i-1)}}) \bowtie_{object=subject} (S_{P_{(i-1)}})))$.
$\mathcal{F}(P \equiv P^-)$	A new *idb* relation S_{P_i} is defined as $\pi_{subject,object}(S_{P_{(i-1)}}) \cup \pi_{object,subject}(S_{P_{(i-1)}})$.
$\mathcal{F}(\geq 1P \sqsubseteq A)$	A new *idb* relation S_{A_i} is defined as $\pi_{id}((S_{A_{(i-1)}})) \cup \pi_{subject}((S_{P_j}))$.
$\mathcal{F}(\top \sqsubseteq \forall P.A)$	A new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{object}(S_{P_j})$.
$\mathcal{F}(A \sqsubseteq B)$	A new *idb* relation S_{B_i} is defined as $\pi_{id}(S_{B_{(i-1)}}) \cup \pi_{id}(A_j)$.
$\mathcal{F}(A \equiv B)$	A new *idb* relation S_{B_i} is defined as $\pi_{id}(S_{B_{(i-1)}}) \cup \pi_{id}(A_j)$.
$\mathcal{F}(A \equiv C \sqcap D)$	A new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup (\pi_{id}(\mathcal{F}(B \equiv C)) \cap \pi_{id}(\mathcal{F}(B \equiv D)))$.
$\mathcal{F}(A \sqsubseteq C \sqcap D)$	$\mathcal{F}(A \equiv C)$, $\mathcal{F}(A \equiv D)$
$\mathcal{F}(A \equiv C \sqcup D)$	A new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{id}(\mathcal{F}(B \equiv C)) \cup \pi_{id}(\mathcal{F}(B \equiv D))$.
$\mathcal{F}(A \equiv B \sqcup C)$	If $I \equiv \neg B$, a new *idb* relation S_X is defined as $\pi_{id}(S_{A_i}) \cap \pi_{id}(S_{I_j})$, $\mathcal{F}(X \equiv C)$
$\mathcal{F}(A \sqsubseteq \forall P.C)$	A new *idb* relation S_X is defined as $\pi_{object}(S_{A_i} \bowtie_{id=subject} S_{P_j})$. $\mathcal{F}(X \equiv C)$
$\mathcal{F}(A \equiv \exists P.C)$	A new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{id}(\mathcal{F}(X \equiv C)) \bowtie_{id=object} S_{P_j})$.
$\mathcal{F}(A \sqsubseteq \exists P.C)$	If P is a functional property, a new *idb* relation S_X is defined as $\pi_{subject}(S_{A_i} \bowtie_{id=subject} S_{P_j})$. $\mathcal{F}(X \equiv C)$
$\mathcal{F}(A \equiv \neg B)$	If $B \equiv \neg I$, a new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{id}(S_{I_j})$
$\mathcal{F}(A \equiv \neg B)$	If $I \equiv B \sqcup C)$, a new *idb* relation S_X is defined as $\pi_{id}(S_{A_i}) \cap \pi_{id}(S_{I_j})$, $\mathcal{F}(X \equiv C)$
$\mathcal{F}(A \equiv \neg B)$	If $I \equiv \neg A$, a new *idb* relation S_{B_i} is defined as $\pi_{id}(S_{B_{(i-1)}}) \cup \pi_{id}(S_{I_j})$
$\mathcal{F}(A \equiv \{v_1, \ldots, v_n\})$	A new *edb* relation $T(id)$ is defined where $r(T) = \{t_1, \ldots, tn\}$ and $t_i = v_i$, $i : 1 \ldots n$. Then a new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{id}(T)$.
$\mathcal{F}(A \equiv\leq nP)$	if (x, y_i), $i : 1..n$ are instances of P, and the y_i are all different, a new *edb* relation $T(id)$ is defined where $r(T) = \{t\}$ and $t = x$. Then a new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{id}(T)$.
$\mathcal{F}(A \equiv \exists P.\{v\})$	A new *idb* relation S_{A_i} is defined as $\pi_{id}(S_{A_{(i-1)}}) \cup \pi_{subject}(\sigma_{object=v}(S_{P_j}))$.
$\mathcal{F}(A \equiv \exists P.\{v\})$	A new *idb* relation S_{P_j} is defined as $\pi_{subject,object}(S_{P_{(j-1)}}) \cup \pi_{id,v}(\pi_{id}(S_{A_j}))$.

Table 2. DBOWL Classification Function

4 DBOWL Advantages and Limitations

DBOWL is implemented using Oracle 10g as Relational Database Management Systems. *edb* relations are tables in the database while *idb* relations are SQL views. A view in SQL terminology is a single table that is derived from other tables [4]. A view does not necessarily exists in physical form; it is considered a *virtual table* (non-materialized). The query defining the view is evaluated when needed. Then, once the knowledge base is created, for each Named Class and Property in the ontology there is a SQL view which defines the set of tuples (asserted and inferred) belonging to such Named Class or Property.

In order to query the knowledge base, SPARQL queries are re-written in terms of the SQL views and evaluated on the database. The names of the Named Classes and Properties involved in the query are changed by the corresponding SQL view name. Note that the queries defining the views are evaluated when the SPARQL query is evaluated. Thus, the inferred instances are not materialized in the database. Only some intermediate results are physically stored in the database, like the results of the transitive function. This non-materialized approach allows us to deal with billions of instances without the need of very large storage repositories. However, the main advantage of this approach is regarding updates. The non-materialization of the inferred instances permits the support of low-cost updates, as well as the possibility of implementing incremental reasoning algorithms.

Another important feature of DBOWL is the management of the *owl:sameAs* statement. At the end of each loop of the algorithm for the knowledge base creation, those individuals related by the *owl:sameAs* statement are included in the SQL views. DBOWL obtains the individuals related by the *owl:sameAs* statement as: (1) Those individuals explicitly asserted as *(x sameAs y)*; (2) By means of functional and inverse functional properties; (3) By means of the maxCardinality to 1 restriction.

DBOWL is complete with respect to the DBOWL knowledge base and the implemented functions, classifying all instances in Named Classes and Properties correctly. However, it presents some limitations:

As DBOWL separates Tbox and Abox reasoning, some inferences with nominals are lost. Fortunately, this information is not relevant for DBOWL because the objective of DBOWL is to classify instances in Named Classes and these inferences do not generate additional information for classification of instances.

DBOWL presents a problem regarding the open-world semantics of a DL Abox, which implies that an Abox has several models. The problem of exploring all the possible models in DBOWL is not trivial, even so, as DBOWL supports a large number of instances, it is logical to think that it could be very inefficient. Nevertheless, we plan to study how to provide a (partial) solution to this problem in the future.

Currently updates are not efficiently supported in DBOWL.

Finally, consistency checking of the knowledge base is not completely supported. Currently only the inconsistency caused by the classification of the same instance into (or the assertion of the same instance as member of) two disjoint

		Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	Query 7	Query 8
20MG	DBOWL	32	2512	666	383	200	165	19	303
	UOB	32	2512	666	383	200	165	19	303
100MG	DBOWL	35	11305	666	414	200	772	145	344
	UOB	35	11305	666	414	200	772	145	344
200MG	DBOWL	26	22833	666	389	200	1668	8	340
	UOB	26	22833	666	389	200	1688	8	340

		Query 9	Query 10	Query 11	Query 12	Query 13	Query 14	Query 15
20MG	DBOWL	1057	25	1930	65	379	6893	61
	UOB	1057	25	1930	65	379	6893	61
100MG	DBOWL	1041	29	6230	37	416	6913	73
	UOB	1041	29	6225	37	10503	6913	72
200MG	DBOWL	1039	25	4353	43	408	7088	79
	UOB	1039	25	4346	43	37577	7088	79

Fig. 1. Number of instances for each UOBM query

classes, and by the classification of one instance in a unsatisfiable class are implemented.

5 DBOWL Evaluation

In order to demonstrate practically the completeness of DBOWL we use the UOBM (University Ontology Benchmark) [2], a well known benchmark to compare repositories in the Semantic Web. This benchmark is intended to evaluate the performance of OWL repositories with respect to extensional queries over a large data set that commits to a single realistic ontology. Furthermore, the benchmark evaluates the system completeness and soundness with respect to the queries defined. This benchmark provides three OWL-DL ontologies, i.e. a 20, 100 and 200 Megabytes ontologies and the query results for each one. This experiment is conducted on a VMWARE virtual machine (one for each tool) with 8192 MB memory, running on a Windows XP 64 bits professional and java runtime environment build $1.6.0\_14 - b08$.

We evaluated the UOBM-DL queries for the 20, 100 and 200 Megabytes ontologies in DBOWL and obtained the correct results for all queries. Figure 1 presents the results for each ontology and for each query. As we can see, some DBOWL results are marked in a different color. This is because DBOWL and UOBM return different results for queries 11, 13 and 15. We checked the UOBM results for these queries and we believe that they are incorrect. For query 11 DBOWL returns more results than UOBM. In the case of queries 11 and 15, it is because several owl:sameAs relationships between some UOBM individuals can be inferred. Therefore, these individuals should be in the result. In the case of query 13, it is because the UOBM result includes instances of all departments, but query 13 asks only for instances in department0. Figure 2 presents the response times for the UOBM-DL 200 Megabytes ontology.

We have also evaluated DBOWL using the standard datasets provided in the context of the ORE 2012 workshop. These datasets include a set of state

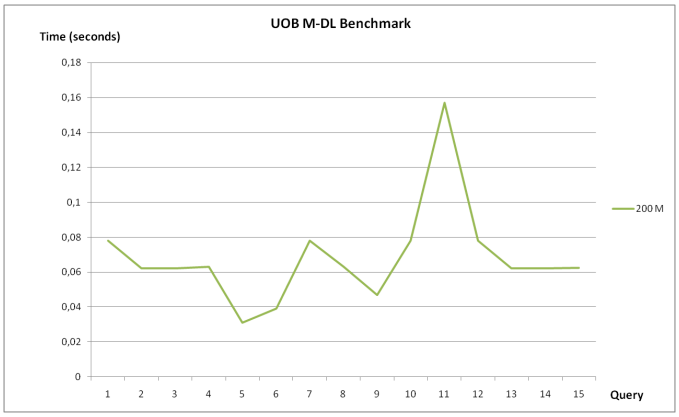

Fig. 2. Response times for UOBM-DL 200M ontology

of the art ontologies in OWL 2 language, both in RDF/XML and Functional syntax. and they are organised by reasoning services, i.e. Classification, Class satisfiability, Ontology satisfiability, Logical entailment and non entailment and Instance retrieval. DBOWL uses Pellet in order to classify the ontology Tbox and to check the class satisfiability. Therefore, datasets corresponding to these reasoning services are not included in our evaluation. As the main objective of DBOWL is to classify instances in Named Classes and Properties, we evaluate DBOWL using the Instance Retrieval test cases. Some of the ontologies in these datasets present unsatisfiable classes. We use these ontologies to test the behavior of DBOWL in such cases. However, the total time taken to load and test the satisfiability of one ontology and the satisfiability result is reported by Pellet. DBOWL only stores in the database the classes that Pellet returns as unsatisfiable. When DBOWL classifies an instance in a unsatisfiable class, it returns that the ontology is inconsistent, via a simple SQL query to the relational database. Thus, the performance of the ontology classification reasoning service falls on Pellet. Finally, as DBOWL is an OWL-DL reasoner, we use the OWL-DL Instance Retrieval test case for the evaluation.

This experiment has been carried out in two phases. In the first step, we loaded the nine ontologies in DBOWL. Most of the ontologies could not be loaded due to different problems: (1) Some ontologies are not valid DBOWL ontologies (see section 2.3). Information_397.owl, minswap.owl, and people.owl define complex property's domains or range (different from named Classes). Information_397.owl and people2.owl contain complex Abox assertions (different from Named Classes assertions). Finally, obi.owl cannot be classified by Pellet because of memory problems. (2) DBOWL presented some problems dealing with unsatisfiable classes, because the storage and management of the class *Nothing* was not completely implemented. (3) DBOWL presented some problems regarding the management of the namespaces.

Ontology Name	Class description	Load time (seconds)	Instance Retrieval time (seconds)	Number of Instances returned	Unsatisfiable classes found
artontology.owl	http://mr.teknowledge.com/DAML/artontology.daml#_anon0	6610	0.023	3	
food.owl	http://www.w3.org/TR/2003/CR-owl-guide-20030818/food#Grape	23541	0.06	17	
information_397.owl	http://www.loa-cnr.it/ontologies/ExtendedDnS#role	79876	0.1	10	
MGEDOntology.owl	http://mged.sourceforge.net/ontologies/MGEDOntology.owl#MethodologicalFactorCategory	45526	0.114	32	
mindswap.owl	http://xmlns.com/wordnet/1.6/Person	17718	0.17	38	
obi.owl	http://purl.obofoundry.org/obo/IAO_0000033	—	—	—	
people.owl	http://cohse.semanticweb.org/ontologies/people#newspaper	26702	0.037	4	&people; Mad+cod
people+pets.owl	http://cohse.semanticweb.org/ontologies/people#animal	27410	0.044	15	&people; Mad+cod
travel.owl	http://www.owl-ontologies.com/travel.owl#RuralArea	12877	0.037	4	&travel;Safari

Fig. 3. Results for OWL-DL Instance Retrieval dataset

In the second step, we solved the aforementioned problems and we loaded eight of the nine ontologies in DBOWL (obi.owl could not be loaded because it presented a problem with Pellet) and we obtained the corrected result for all of them. We followed the guidelines outlined in Section 2.3 in order to convert the ontologies into DBOWL ontologies. Figure 3 summarizes the results of the evaluation. Load time includes Tbox classification (Pellet), database creation, ontology storage and knowledge base creation (instances classification).

6 Conclusions

From the evaluation we extract some general conclusions. To the best of our knowledge, DBOWL is the only OWL reasoner able to deal with the three UOBM-DL ontologies obtaining the correct results for all queries in all cases. Furthermore, this allows us to check the UOBM results for queries 11, 13 and 15 and to conclude that they are incorrect. Finally, DBOWL response times are very good the highest one being 0.328 seconds for the UOBM 200MB ontology. The results obtained with both evaluations suggest that DBOWL is a real complement to current OWL reasoners. Currently, DBOWL supports ontologies with much bigger Aboxes than traditional systems based on description logic and satisfiability. This is especially important for some applications such as life sciences, where particularly large ontologies are used. The datasets provided in the context of the ORE 2012 workshops have allowed us to improve DBOWL in several ways. Thus, the latest version of DBOWL is able to deal with all types of namespaces, to control when a class is non-satisfiable and to check the ontology consistency in such a case. Furthermore, we empirically test that the restrictions imposed on the DBOWL ontologies are not a problem for developing real ontologies, because any ontology can be converted to a DBOWL ontology, keeping the ontology expressivity. With respect to instance retrieval, DBOWL is

able to obtain the same results as the expected result provided by the OWL-DL Instance Retrieval dataset, suggesting that the DBOWL classification functions and the algorithm for creating the knowledge base work well.

The use of a relational database to store the ontologies implies that the time for loading an ontology in DBOWL can be longer than the load time in main-memory reasoners. The advantage of our approach is that, once the knowledge base is created, the query time is really small. Furthermore, as the knowledge base is persistent, you can query it at any moment without creating it again. Although other approaches also provide solutions for instance retrieval, they present some problems regarding reasoning expressivity or response query times. SHER [6], is a platform developed by IBM which supports sound and complete reasoning for the fragment of OWL 1 DL without nominals. SHER adopts a modularisation-based approach in which the ontology breaks into small parts and is reasoned with a DL reasoner in the main memory. After the reasoning procedure is finished, the corresponding axioms are stored in the database. Reasoning with instances is performed at query time. Oracle 11g [7] is the laster version of the extremely well known RDMS Oracle. Oracle 11g includes a native inference engine able to handle a subset of OWL called OWLPrime which covers part of OWL Lite and a little part of OWL 1 DL. It also supports querying of RDF/OWL data using SPARQL-like graph patterns embedded in SQL.

As for future work, we are studying some optimisation techniques (such as database indexes, parallel computation and incremental reasoning) in order to improve the response times of the queries. We also are studying the possibility of incorporating other OWL reasoners different from Pellet, in DBOWL. The idea is to select the most convenient OWL reasoner depending on the ontology expressivity and size.

7 Acknowledgements

This work is supported by the Project Grant TIN2011-25840 (Spanish Ministry of Education and Science) and P11-TIC-7529 (Innovation, Science and Enterprise Ministry of the regional government of the Junta de Andalucía).

References

1. Ullrich Hustadt , Boris Motik , Ulrike Sattler. *Reasoning in Description Logics by a Reduction to Disjunctive Datalog*. Journal of Automated Reasoning, v.39 n.3, p.351-384, October 2007.
2. Ma, L; Yang, Y; Qiu, Z; Xie, G; Pan, Y. *Towards A Complete OWL Ontology Benchmark*. In. Proc. of the 3rd European Semantic Web Conference (ESWC 2006).
3. Codd, E. *A relational Model for Large Shared Data Banks*, CACM, 13:6, june 1970.
4. Abiteboul, S., Hull, R., Vianu, V. *Foundations of Databases*. Addison-Wesley Publishing Company. 1995.

[6] http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa.index.html
[7] http://www.oracle.com