Efficient Inference-aware RDB2RDF Query Rewriting

Jose Mora[1] and Oscar Corcho[1]

Ontology Engineering Group, Departamento de Inteligencia Artificial,
Facultad de Informática,Universidad Politécnica de Madrid, Spain
{jmora,ocorcho}@fi.upm.es

Abstract. RDB2RDF systems generate RDF from relational databases, operating in two different manners: materializing the database content into RDF or acting as virtual RDF datastores that transform SPARQL queries into SQL. In the former, inferences on the RDF data (taking into account the ontologies that they are related to) are normally done by the RDF triple store where the RDF data is materialised and hence the results of the query answering process depend on the store. In the latter, existing RDB2RDF systems do not normally perform such inferences at query time. This paper shows how the algorithm used in the REQUIEM system, focused on handling run-time inferences for query answering, can be adapted to handle such inferences for query answering in combination with RDB2RDF systems.

Keywords: RDB2RDF, query rewriting, reasoning

1 Introduction

Organizations store ever growing amounts of information and a big part of that information is available in relational databases (RDBs). In the context of the Semantic Web, and specially related to the Linked Data initiative, RDB2RDF tools are being used to generate RDF from a large number of these databases. The importance of offering the information available in relational databases (RDBs) as RDF is reflected in the number of approaches taken to produce RDF data from relational databases [1].

The representation of the correspondence between the information in the DB and the code generated in RDF is done by means of mappings, which are specified in languages like D2R[2], R2O [3], etc. There are important differences in the expressiveness of these mappings [3], for instance in how the selection of the data from the DB is performed, in whether it is possible to select elements just from tables or whether filter conditions can be applied to rows and columns, etc. The transformation operations to generate URIs and to obtain derived data from raw data are also a significant source of differences among systems. Due to the similarities among RDB2RDF tools a standardization process has started in the context of the W3C and R2RML [4] is being developed as a standard language to define these mappings for these tools.

The execution of these mappings may be performed in a static Extract Transform Load (ETL) manner or may be query-driven, depending on how the generation of the RDF code is performed, either in a batch process generating an RDF materialization of part of the database that is normally stored afterwards in an RDF triple store for consumption, or as an on-demand process for the answering of each query posed to the system [1].

RDB2RDF tools have attracted more attention recently due to the emergence of the Linked Data initiative, where efforts are mainly focused on the publication of RDF data according to the Linked Data guidelines. In this context, the materialization approach is the one most commonly used. However, some of these tools originated earlier as a means to create wrappers in ontology-based information integration architectures [5], normally focused on the creation of virtual RDF datastores with run-time transformations. This usage will be the focus of this paper without constraining the approach to any specific mapping language.

In such ontology-based information integration scenarios [5,6] description logic (DL) has been applied extensively. The TBox normally takes the role of the global schema [7] in the description of the contents of the integrated ABoxes, which are constructed from the distributed data sources. When a TBox is used as the global schema, the complexity of the queries that can be posed to the system and the use of reasoning for query answering depend on the expressiveness of the knowledge base that is the TBox [8].

Finding answers in these scenarios requires a logical and physical path search. The former consists on finding rewritings of the query that keep the semantics. The latter consists in finding the physical information sources that provide the relevant information for the rewritten query [9]. Thus in information integration the rewriting consists on these two aspects: the concepts involved in the query can be first replaced by the combination of other concepts according to the inference enabled by the TBox, and once the relevant concepts have been found it is necessary to pose the queries to the original sources of information, for which the mappings are used. These mappings may follow different approaches: GaV, LaV or GLaV [10].

During the first step, the query answering procedure uses the TBox to rewrite the original query according to the semantics of the ontology language used. In general, the more expressive the ontology language, the more complex this rewriting. As a result of rewriting, a number of normally simpler queries are generated, whose combination will provide answers for the original query. Hence, performing this reasoning introduces an efficiency penalty in the query answering process, and thus most RDB2RDF systems with a run-time query answering approach do not use any reasoning in the process[1] providing less results, and leaving the task of reasoning with the generated RDF data to standard ontology reasoners on triple stores afterwards. A compromise solution consists in performing some efficient reasoning, in polynomial time, with limited expressiveness and reasoning capabilities, as explained in the next section.

Moreover, when the ABox is generated from a RDB through RDB2RDF mappings and a TBox is added for the provision of reasoning capabilities, some

concepts and relations in the TBox may not be covered in the RDB2RDF mappings. This is the case of the mappings generated for the Food and Agriculture Organization of the United Nations (FAO)[1]. The test cases were used for populating the fisheries ontologies [11]. These ontologies were developed for use within the Fish Stock Depletion Assessment System (FSDAS) in the context of the NeOn project[2]. All ontologies are publicly available from the FAO website[3]. Table 1 shows the ontology coverage provided by R2O mappings. Each row contains the information regarding a FAO test case, as a group composed by an ontology and a set of mappings. The columns show the number of elements, the number of mappings and the percentage of elements covered by the mappings with respect to concepts, object properties and datatype properties on the ontology respectively. The last column displays the percentage of the ontology that is covered. As can be seen, the elements mapped with R2O are less than one third of those present in the ontologies.

	Ontology Concepts			Object Properties			Datatype Properties			
Group	Num.	Mapped	Coverage	Num.	Map.	Coverage	Num.	Map.	Coverage	Total
1	7	5	28.57%	15	14	6.67%	126	100	20.63%	19.59%
2	3	1	66.67%	1	0	100.00%	8	2	75.00%	75.00%
3	6	0	100.00%	25	9	64.00%	30	10	66.67%	68.85%
4	5	0	100.00%	20	8	60.00%	90	32	64.44%	65.22%
5	7	3	57.14%	0	0	0.00%	104	70	32.69%	34.23%
6	5	0	100.00%	20	8	60.00%	90	32	64.44%	65.22%
7	3	1	66.67%	1	0	100.00%	8	2	75.00%	75.00%
8	16	15	6.25%	0	0	0.00%	201	198	1.49%	1.84%
9	16	15	6.25%	0	0	0.00%	201	198	1.49%	1.84%
Total	68	40	41.18%	82	39	52.44%	858	644	24.94%	28.27%

Table 1: Test cases in FAO, ontology coverage by R2O mappings. Each group consists of an ontology and a set of mappings.

After query rewriting, if a predicate is the head in several clauses in the TBox, all of them will provide correct answers for this predicate, but when only some of them lead to RDB2RDF mappings and the remaining clauses do not provide answers, these remaining clauses can be removed from the obtained query plan.

Our claim is that pruning those clauses that do not provide any solution leads to a reduction in the number of clauses that the system has to handle, what may result in an increase in the efficiency of the whole query answering process. The sooner the pruning is performed, the better the load reduction for the following steps of the process. With this rationale, a method to prune the number of clauses generated during the process of query rewriting is presented in this paper.

This paper is structured as follows: section 2 will provide some background about RDB2RDF tools and their mapping languages, as well as the use of de-

[1] The test cases enumerating the specific elements in table 1 can be found at `http://delicias.dia.fi.upm.es/~jmora/qr4rdb2rdf`

[2] `http://www.neon-project.org`

[3] `http://www.fao.org/aims/neon.jsp`

scription logics languages to describe the global schema in information integration scenarios and the reasoning methods used with these logics for query rewriting. Section 3 will explain the method used to prune clauses during the rewriting process so as to avoid the generation of queries rendered as futile because of the mismatch between existing RDB2RDF mappings and the global schema. Section 4 presents the evaluation and comparison of the results obtained, where the efficiency gain with respect to other existing approaches, namely REQUIEM [12], can be checked. Finally, section 5 contains the conclusions of the paper.

2 Background

The DL family in which the ontology TBox is implemented in these information integration scenarios defines the rules to be used for query rewriting. Besides, the efficiency of the query rewriting process depends on the number of axioms in the ontology and the way in which these are combined to describe the global schema. In this section we summarise the current state of the art in the relationship between query rewriting algorithms and DL families.

Several alternatives have been explored in the past to address the trade off between the expressiveness of ontology languages and the complexity of the query rewriting process, as described next.

One approach considers the use of the ***DL-Lite* family** [13]. The complexity of reasoning in *DL-Lite* is polynomial in time and size of the TBox and LogSpace with respect to the size of the ABox [8]. The *DL-Lite* family includes *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, the former includes the possibility to express functional restrictions on roles while the latter includes ISA and disjointness assertions between roles. Moreover, ABox assertions in *DL-Lite* can be expressed as relations in a database.

Further to the *DL-Lite* work, the ***QL* profile** in *OWL 2* has been proposed [14]. This profile is tailored for the query rewriting to SQL storages for ABoxes. In order to ensure that a query can be rewritten into a union of conjunctive queries, *OWL 2 QL* forbids the use of some knowledge representation primitives from *OWL 2*, such as disjunctions and universal property restrictions, as well as certain other features that require recursive query evaluation. The ***EL* profile** in *OWL 2* has also a special relevance, since it is based on the $\mathcal{EL}++$ family of description logics and it is designed to reason with large terminologies. The central modeling features of this profile are class conjunction and existential property restrictions. In order to achieve tractability, the use of negation, disjunction, universal property restrictions, and cardinality restrictions is disallowed.

Another logic that has been proposed in this context is $\boldsymbol{\mathcal{ELHIO}}^{\neg}$, which is one of the most expressive Horn logics for which query answering is polynomial with respect to data complexity [12]. Besides $\mathcal{EL}$, as in the previously described *EL* profile of *OWL 2*, it provides basic concepts of the form $\{a\}$, inverse roles and role inclusions. The reasoning in this logic can be carried out by the REQUIEM algorithm [15], which performs saturation on the query and the relevant part of

the *DL-Lite* ontology to produce all the rewritings. The whole process can be divided in four main phases performed over the ontology and query:

- In the first phase, the ontology and the query are parsed and converted into clauses (the conversion into clauses for $\mathcal{ELHIO}^{\neg}$ *KBs* is explained in [12]). The statements that fall out of the expressiveness of $\mathcal{ELHIO}^{\neg}$ are discarded.
- In the second phase, the ontology is pruned and the part of it that is useful for the generation of rewritings is taken for the following phases, since saturation performs all the possible inferences. This is how the inferences are limited to those useful for the current query.
- In the third phase, saturation is performed over the ontology and query, generating a datalog program as query rewriting. The datalog program is pruned, removing clauses that contain equality, functional terms or nominal terms.
- In the fourth phase, the datalog program is resolved, generating the set of conjunctive queries that is the rewriting of the original query and that can be posed to the database or set of databases that are abstracted with the schema used by the algorithm.

After that process the queries that should be posed to the DB according to the ontology and the original query are generated, without considering the possible query containment (and therefore redundancy) among them or the availability of the information in the DB that is to be queried.

3 Improvements over the REQUIEM query rewriting process

The REQUIEM query rewriting process, described in section 2, performs a pruning of the search space at two specific points. First, the ontology is pruned so as to reduce the number of clauses and thus the computational load for the inference; second, the datalog program generated by the saturation process that continues with this pruned ontology is pruned again, after its generation, so that it only contains predicates that can be obtained from the declared RDB2RDF mappings. The resulting queries generated from the datalog program are noticeably reduced after both prunes. Our proposal consists on improving the pruning steps done by the REQUIEM process at these two specific points and according to two different methods that will be useful in different situations. As a consequence of the improved pruning, only a subset of the original union of conjunctive queries will be generated. Since the union is disjunctive, the correctness of the remaining queries is unaltered. The completeness of the remaining set of queries will be proved in the next sections.

The first improvement is in the pruning process after transforming the ontology into clauses, which can be more restrictive, including only the clauses that are relevant and can provide results according to the declared RDB2RDF mappings and those necessary for the later inference, that will need to be pruned after the inference is done. The prune at this stage can be performed according

to two methods, depending on the inference that the RDB2RDF system may perform. If the system is designed to be able to perform some inferences during query answering the prune can be more restrictive, otherwise, this inference will be performed by our algorithm and thus some additional clauses will be required. This will be further explained in section 3.1.

The second improvement comes immediately after the generation of the datalog program. The generation of the datalog program may require the use of some clauses for the inference; however, since the datalog program may serve as an input for other processes and algorithms, it is pruned for a better efficiency, by removing the clauses that contain predicates that are not included in the declared RDB2RDF mappings, without losing relevant answers. This is performed with an additional saturation step, as explained in section 3.2.

3.1 Pruning the ontology

The processes of parsing and conversion to clauses, which define the ontology expressiveness that can be handled, have been reused from REQUIEM [12]. The maximum expressiveness covered by REQUIEM corresponds with $\mathcal{ELHIO}^{\neg}$: axioms not covered by this expressiveness are simply discarded and not used in the later stages.

We start from the assumption that our global schema will be described in the $\mathcal{ELHIO}^{\neg}$ *DL* [12]. In the usual case that our schema is described in OWL, we will consider only the axioms that fall under the $\mathcal{ELHIO}^{\neg}$ *DL*, as it is done in other works (e.g, [12]). It is important to note that we are aware of the fact that this axiom exclusion may have some relevant implications in the results obtained from the query rewriting process, and as a part of our future work we will aim at providing user support tools to inform about those consequences.

This $\mathcal{ELHIO}^{\neg}$ projection of the original ontology is transformed into clauses as described in [12]. Once the ontology has been converted into clauses, finding the rewritings is analogous to searching through a tree in this context. In this case the head of the query would act as the root of the tree, the children of the root would be the predicates in the body of the query, and as far as these predicates are in the head of other clauses the process can be repeated, conforming a tree. The leaves of the tree will be a set of predicates that are not in the head of any clause. This tree, avoiding loops, is the result of the prune process in REQUIEM, containing only clauses relevant for the current query. Table 2 contains a fragment of the *hydrOntology*[4] ontology and its translation to clauses that will be used as an example for further clarification.

Let us imagine that the query posed to the ontology in table 2 is $Q(x) \leftarrow Water(x)$. In this case, the clauses relevant for the query will provide information about $Water$ and thus the starting point are clauses whose head is the "Water" predicate, this is clauses 1 and 5. The predicates referred in the body of clause

[4] Ontology URL: `http://mayor2.dia.fi.upm.es/oeg-upm/files/hydrontology/hydrOntology_GeoLinkedData.owl`. Description URL: `http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/107-hydrontology`

$\mathcal{ELHIO}$ clause	$\mathcal{ELHIO}$ axiom
1 $Water(x) \leftarrow DrainsAt(x, y)$	$\exists drainsAt \sqsubseteq Water$
1 $DrainsAt(x, drains(x)) \leftarrow RunningWater(x)$	$RunningWater \sqsubseteq \exists drainsAt$
2 $RunningWater(x) \leftarrow River(x)$	$River \sqsubseteq RunningWater$
3 $River(x) \leftarrow Tributary(x)$	$Tributary \sqsubseteq River$
4 $RunningWater(x) \leftarrow Stream(x)$	$Stream \sqsubseteq RunningWater$
5 $Water(x) \leftarrow StillWater(x)$	$StillWater \sqsubseteq Water$
6 $StillWater(x) \leftarrow Enclosure(x)$	$Enclosure \sqsubseteq StillWater$
7 $Enclosure(x) \leftarrow SaltMarsh(x)$	$SaltMarsh \sqsubseteq Enclosure$
8 $SalineGround(x) \leftarrow SaltMarsh(x)$	$SalineGround \sqsubseteq SaltMarsh$
9 $Morphology(x) \leftarrow FloodableArea(x)$	$FloodableArea \sqsubseteq Morphology$

Table 2: Axioms in the example ontology and the corresponding clauses

1, DrainsAt lead to clause 2, and so we can continue the process, conforming the tree of the clauses that may be relevant for the rewriting of the query. In this case the only clause pruned is the last one, since it is not connected to any other by this procedure, and since the head present in this clause is not present in the tree it cannot be unified in the resolution. Notice that if some clause could be unified with a predicate in the body of this discarded clause that would still be irrelevant for the resolution of the query, in that case either directly or transitively Q could imply $Morphology$, but the implication of Q by $Morphology$ is impossible even transitively given the set of clauses in the example.

Notice that a simple fragment of the ontology has been chosen for the example, most of the predicates are unary and all the bodies in the rules are composed by one single atom, the focus in the example is on the additional prune that can be done when considering that only some predicates are actually mapped by a RDB2RDF tool, but $\mathcal{ELHIO}$ *DL* has a much greater expressiveness as shown in [12]. That expressiveness has no impact on the prune presented here, though.

When considering that only some predicates are mapped by some RDB2RDF mappings and thus only some predicates can provide valid answers, the prune can be more strict. This prune can be performed according to two different methods, depending on whether the RDB2RDF system performs any inference (e.g. the answers of subconcepts are included in their respective concepts) or not. These two methods replace the original pruning step performed in REQUIEM and generate a number of clauses that is smaller or equal, depending on the mapped predicates and the method used. In the example we consider that the only RDB2RDF-mapped predicates are $River, Enclosure$ and $SalineGround$.

Definition 1. *Retrievable predicate. A predicate 'A' is retrievable if individuals can be obtained for that specific predicate. In the context of with RDB2RDF, retrievable predicates are those for which a mapping with the database exists and allows retrieving their instances.*

$$\exists(A \mapsto Q_A) \Rightarrow retrievable(A) \forall A \tag{1}$$

$$(A \mapsto Q_A) \Rightarrow \forall \boldsymbol{x} \in ans(Q_A).A(\boldsymbol{x}) \tag{2}$$

Where $(A \mapsto Q_A)$ *is a mapping from the predicate* A *to the query* Q_A *in the DB as in [15] and* $ans(Q_A)$ *are the answers to the query* Q_A *when posed to the database.*

Corollary 1. *If a predicate 'A' is not retrievable this means the values for that predicate cannot be directly retrieved, thus the origin must be other predicates that imply the predicate 'A'.*

$$\neg retrievable(A) \Rightarrow \forall \boldsymbol{x}/A(\boldsymbol{x}).\exists\{\boldsymbol{y_i}\}/(A(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i})) \tag{3}$$

Definition 2. *Instantiable predicate. A predicate 'A' is instantiable if there may be individuals that make that predicate true.*

$$instantiable(A) \Leftrightarrow retrievable(A) \vee \exists(A(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i}))/instantiable(B_i)\forall B_i \tag{4}$$

Proof. In datalog, individuals are assigned to a predicate when the predicate is the head of some clause, even with a possibly empty body (a fact). This means that there must be some clause for which the head is the instantiable predicate, either with an empty body, thus a fact and therefore requiring a mapping to obtain those instances from the DB, or in a clause that allows to infer the instances.

Corollary 2. *If a predicate is not instantiable then all clauses containing that predicate can be removed from the datalog program since there are no individuals that make that predicate true.*

$$\neg instantiable(A) \Rightarrow \nexists \boldsymbol{x}/A(\boldsymbol{x}) \tag{5}$$

Proof. If the predicate is in the body of the clause then the values for the head are the empty set, since the clauses are conjunctive and at least one element in the conjunction, the non-instantiable predicate, will have no values. If the element is in the head then the body of the clause cannot be used for inference, or the predicate would be instantiable.

If the inclusion relies on inference and retrieving the instances from one concept requires using the mappings for the subconcepts, then the prune cannot be stopped as soon as a retrievable concept is found and has to continue to the lowest retrievable concept in this search tree, possibly including not retrievable gaps that will have to be pruned in the next pruning step. Since the taxonomy is known at query time these relations are not reflected in the mappings, hence it is also important when there are no underlying systems that would take care of this inference. In this case, *Enclosure* would be relevant and so would *SalineGround*, since the information of *SalineGround* being included in *Enclosure* is only known at query time, with the ontology.

On the contrary, if querying a concept or property retrieves all the values from the subconcepts or subproperties, then all the lower concepts or properties, the

datalog predicates that imply the current one, can be pruned, since they will not provide additional values. In the previous example, once the mapping with *Enclosure* has been found, the predicates in its body would not be inspected, since the RDB2RDF mappings return all the instances for *Enclosure*, including *SalineGround*, whose mapping is rendered superfluous in this case. This can be summarised in the following lemma:

Lemma 1. *For any mapped predicate and any clause in a datalog program whose head is composed by that predicate. If the instances provided by inference on the clause are contained in the instances provided by the mappings of the predicate, then the clause can be removed without losing any instance in the program result.*

Proof. Given the condition in the lemma

$$\exists\{Q_A/A \mapsto Q_A\} \wedge \bigcup ans(Q_A) \supseteq \{\boldsymbol{x}/A(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i})\} \tag{6}$$

If (6) then given any two programs, Q_C and Q'_C such that $Q_C = \{Q(\boldsymbol{x}) \leftarrow A(\boldsymbol{x})\}$ and $Q'_C = Q_C \cup \{A(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i})\}$ which can be unfolded into $Q'_C = Q_C \cup \{Q(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i})\}$ and with $pred(Q_C)$ as the set of predicates in Q_C then $(\{\boldsymbol{x}/Q \in pred(Q'_C) \wedge Q(\boldsymbol{x})\} \subseteq \{(\boldsymbol{x}/Q \in pred(Q_C)) \wedge Q(\boldsymbol{x})\}) \vee (\exists \boldsymbol{x}/(\boldsymbol{x} \in \{\boldsymbol{x}/Q \in pred(Q'_C) \wedge Q(x)\} \wedge \boldsymbol{x} \notin \{\boldsymbol{x}/Q(\boldsymbol{x}) \leftarrow A(\boldsymbol{x})\}))$ the second part of the disjunction in this context would imply that $\exists \boldsymbol{x}/\boldsymbol{x} \in \{\boldsymbol{x}/A(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i})\}$ which is absurd, conflicting with (6) thus the first part of the disjunction must be true, the answers obtained the same and hence the clause, which is the difference between both programs, can be removed without losing any answer to the query Q.

This defines respectively two pruning methods: "global", for all retrievable concepts and properties, and "first", which prunes the clauses that imply a specific concept or property if this concept or property is retrievable, since that predicate does provide all the information and no alternative rewritings are necessary. In table 3 the results of applying the original prune method 'N', the global 'G' or the first 'F' on the example are presented.

	'N'	'G'	'F'
1 $Water(x) \leftarrow DrainsAt(x, y)$	Yes	Yes	Yes
2 $DrainsAt(x, drains(x)) \leftarrow RunningWater(x)$	Yes	Yes	Yes
3 $RunningWater(x) \leftarrow River(x)$	Yes	Yes	Yes
4 $River(x) \leftarrow Tributary(x)$	Yes	No	No
5 $RunningWater(x) \leftarrow Stream(x)$	Yes	No	No
6 $Water(x) \leftarrow StillWater(x)$	Yes	Yes	Yes
7 $StillWater(x) \leftarrow Enclosure(x)$	Yes	Yes	Yes
8 $Enclosure(x) \leftarrow SaltMarsh(x)$	Yes	Yes	No
9 $SaltMarsh(x) \leftarrow SalineGround(x)$	Yes	Yes	No
10 $Morphology(x) \leftarrow FloodableArea(x)$	Yes	Yes	No

Table 3: Clauses kept after pruning ('Yes' means that the clause is kept and 'No' means that it can be discarded).

3.2 Pruning the datalog program

The generation of the datalog program in REQUIEM is performed through saturation. In the saturation phase some clauses that will need to be pruned later are still kept so that they can be used in the inferences. For instance, this happens to the clauses containing functions. Similarly, in our case, non retrievable predicates are still kept for the inference, and are removed after the useful clauses depending on them have been generated. In order to get a datalog program that does not contain non retrievable predicates, but keeps the capability of retrieving all the answers, some resolution steps that would rely on non-mapped predicates have to be performed.

Once that the saturated datalog program is obtained as described in [15], we propose the execution of a new stage, focused on the prune of clauses containing predicates that are not mapped and hence cannot provide answers. To achieve this, we apply resolution to the clauses that contain non retrievable predicates, so that the resolution that would be done with clauses that contain non-mapped predicates in the next phase is done in this one. This way, all the inferences enabled by these clauses are already done at this phase and the corresponding clauses can be safely removed from the datalog program, pruning it without loss of answers.

The selection function in this case will select the unmapped atoms in the body of the clauses, if there are any. If not, the head will be selected in case that it is not mapped. This way, every inference made will remove one unmapped atom in the body of a clause, unifying it with the head of another clause, whose body will contain only mapped predicates, thus generating a clause that has at most one unmapped atom less than the base clause of the inference. After saturation has been performed using this selection function and no new clauses can be generated the clauses that contain unmapped predicates can be removed safely, without removing any valid answer with them.

Lemma 2. *If there is some non-mapped predicate in a datalog program then there is another datalog program that provides the same answers and does not contain that predicate.*

$$\forall Q_C/(A \in pred(Q_C) \wedge \nexists(A \mapsto Q_A)).\exists Q'_C/(A \notin pred(Q'_C) \wedge ans(Q_C) = ans(Q'_C)) \tag{7}$$

Where Q_C is a datalog program as in [12], $A \mapsto Q_A$ is a mapping from the predicate A to the query Q_A in the DB as in [15], $pred(Q_C)$ is the set of predicates in the program Q_C and $ans(Q_C)$ are the answers obtained from the program Q_C.

Proof. If there is some predicate that is not mapped, and thus non-retrievable, that means the values for that predicate are obtained by inference from other predicates, as stated in definition 1, this inference can be used to replace the non-retrievable predicate with the predicates that provide its values. If this is done for every clause that contains this non-retrievable predicate with every conjunction

of predicates that implies the non-mapped predicate then the predicate can be safely removed from the program.

$$\nexists Q_A/A \mapsto Q_A \tag{8}$$

$$\nexists \boldsymbol{x} \in \{\boldsymbol{x}/A(\boldsymbol{x})\}/\boldsymbol{x} \in ans(Q_A) \tag{9}$$

$$\forall \boldsymbol{x} \in \{\boldsymbol{x}/A(\boldsymbol{x})\} \exists (\bigwedge B_i(\boldsymbol{y_i}))/(A(\boldsymbol{x}) \leftarrow \bigwedge B_i(\boldsymbol{y_i})) \tag{10}$$

$$\{\boldsymbol{u}/D(\boldsymbol{u}) \leftarrow (\bigwedge C_j(\boldsymbol{z_j}) \wedge A(\boldsymbol{x}))\} \subseteq \{\boldsymbol{u'}/D(\boldsymbol{u'}) \leftarrow (\bigwedge C_j(\boldsymbol{z_j}) \bigwedge B_i(\boldsymbol{y_i}))\} \tag{11}$$

Corollary 3. *If a non-mapped predicate can be removed without consequences, as proved in lemma 2, with the only condition of being a non-mapped predicate; then all non-mapped predicates can be removed in the same way, without losing answers in the resulting datalog program.*

Resuming our example the saturation process would execute iterations and generate new clauses used for the inference in the next iteration, until there are no new clauses to generate and the saturation process finishes. For example, using the global strategy, the contents after the first prune would be:

$$Q(x) \leftarrow Water(x) \tag{12}$$

$$Water(x) \leftarrow DrainsAt(x, y) \tag{13}$$

$$DrainsAt(x, drains(x)) \leftarrow RunningWater(x) \tag{14}$$

$$RunningWater(x) \leftarrow River(x) \tag{15}$$

$$Water(x) \leftarrow StillWater(x) \tag{16}$$

$$StillWater(x) \leftarrow Enclosure(x) \tag{17}$$

$$Enclosure(x) \leftarrow SaltMarsh(x) \tag{18}$$

$$SaltMarsh(x) \leftarrow SalineGround(x) \tag{19}$$

$$Morphology(x) \leftarrow FloodableArea(x) \tag{20}$$

In this case the saturation process does not perform any inference, so these are the clauses that serve as input in the added stage for the removal of non-mapped predicates. The resolution performed takes clauses two by two, first trying with all the combinations of the initial knowledge base and then, as new clauses are generated, with those clauses and all the previous clauses. This procedure always takes a non-mapped predicate in the body of a clause and unifies this predicate with the head of other clause that does not contain any non-mapped predicates in the body, thus every inference step generates a clause with at least one non-mapped predicate less than the clauses used to infer it, proving convergence. In our example the mapped predicates are $River, Enclosure$ and $SalineGround$. In the first step of the resolution we get:

$$from(14) and (15) : DrainsAt(x, drains(x)) \leftarrow River(x) \tag{21}$$

$$from(16) and(17) : Water(x) \leftarrow Enclosure(x) \tag{22}$$

$$from(18) and(19) : Enclosure(x) \leftarrow SalineGround(x) \tag{23}$$

In the second step of the resolution we get:

$$from(13) and(21) : Water(x) \leftarrow River(x) \tag{24}$$

$$from(12) and(22) : Q(x) \leftarrow Enclosure(x) \tag{25}$$

Similarly, in the third step we get:

$$from(13) and(21) : Water(x) \leftarrow River(x) \tag{26}$$

And finally in the fourth step we get:

$$from(13) and(21) : Q(x) \leftarrow River(x) \tag{27}$$

After pruning the clauses that contain non-mapped predicates we get the following datalog program:

$$Q(x) \leftarrow Enclosure(x) \tag{28}$$

$$Q(x) \leftarrow River(x) \tag{29}$$

$$Enclosure(x) \leftarrow SalineGround(x) \tag{30}$$

This datalog program generated with the 'G' variant of the algorithm contains all the mapped predicates that can provide useful answers, with a reduced number of clauses. In the next section the evaluation and quantification of this reduction is performed.

4 Evaluation

For the evaluation of the system two geographical ontologies have been used, both developed and used in the context of independent projects. The first ontology is *hydrOntology* [16], with 155 concepts and expressiveness $\mathcal{SHIN}(\mathcal{D})$. *HydrOntology* has been used in several projects and mapped with several geographical databases of the Instituto Geográfico Nacional (IGN)[5], generating several RDB2RDF mapping files, which, being available, have been used for the tests. More precisely, the databases mapped here are three:

- The National Atlas data source, with 32 mappings that provide 1,100 hydrographical instances for an overview of Spain's human and physical environment.
- The Numerical Cartographic Database (BCN200) with 57 mappings that provide 60,000 toponyms related to hydrographical instances.

[5] `http://www.ign.es`

– EuroGlobalMap (EGM), produced in cooperation with the National Mapping Agencies of Europe, with 32 mappings that provide 3,500 Spanish hydrographical toponyms.

The second ontology is *PhenomenOntology* [17]. In this case among the phenomena that could be covered only the module regarding transportation networks has been used, which provides 66 concepts, having a common ancestor in *"Red" ("Network")* which is the concept used for the test queries whose results are shown in table 4. In the case of *PhenomenOntology*, only one file with 18 mappings is used.

Both of these two ontologies are expressed in OWL, without imposing any kind of restriction, thus some of the axioms have to be discarded in the beginning of the process, as described in section 3.1, as they do not fall into the expressiveness of $\mathcal{ELHIO}^{\neg}$ *DL*. The mapping files are R2O mappings [3].

Ontologies		HydrOntology							Phenomenontology		
Information		686 clauses, 405 ignored statements							66 c., 114 i.s.		
Mappings		None	BCN200		Atlas		EGM		None	Unique	
Mappings #		0	57		32		32		0	18	
Prune method		A	H	L	H	L	H	L	A	H	L
Mode	Phase	Number of clauses generated after each phase									
N	Prune	535	323	445	287	415	336	429	66	66	66
	Saturation	412	25	50	17	24	19	31	66	60	60
	Unfolding	2333	25	46	17	22	19	28	64	14	14
	Prune	2333	25	46	17	22	19	28	64	14	14
G	Prune	535	323	445	287	415	336	429	66	66	66
	Saturation	407	25	49	15	24	19	30	66	60	60
	Unfolding	894	25	45	15	22	19	27	64	14	14
	Prune	688	25	45	15	22	19	27	64	14	14
F	Prune	535	323	445	287	415	336	429	66	66	66
	Saturation	407	25	49	15	24	19	30	66	60	60
	Unfolding	2245	25	45	15	22	19	27	64	14	14
	Prune	911	25	45	15	22	19	27	64	14	14
N	Total time (ms)	2735	172	625	156	359	250	625	17	15	15
G	Total time (ms)	2250	172	512	156	360	218	532	31	16	16
F	Total time (ms)	3719	172	531	156	344	235	531	31	16	16

Table 4: Results of the evaluation

The results[6] are summarised in table 4. They vary depending on the method used. The first letter is for the methods in the original REQUIEM, 'N' for "naive", 'F' for "full forwarding" and 'G' for "greedy", the second letter stands for the modification applied as described in this paper, in this case 'A' stands for "all", since all predicates are kept independently of the mappings. 'H' will stop the prune on the first predicate that is mapped (the highest), as all the values that are correct for that predicate are assumed to be retrievable from the

[6] Available at `http://delicias.dia.fi.upm.es/~jmora/qr4rdb2rdf/`

RDB2RDF mappings, as described in section 3.1. Finally 'L' will be used for the "global" approach, which will keep all the mapped predicates in the ontology after pruning it, since they could be complementary to some extent, again as described in section 3.1.

The query posed to the system for comparison purposes is a general query covering a good part of the ontology, in the case of *hydrOntology* the query is simply $Q(x) \leftarrow Aguas(x)$, "Water" covers as a superclass most of the taxonomy present in *hydrOntology*. Similarly in the case of *PhenomenOntology* the query used is $Q(x) \leftarrow Red(x)$. Being a module about transportation networks, the concept "Red" (Network) is the most general one. Both queries cover most of the mappings and ontologies used for the tests, providing a good approximation of what could be expected in a different context.

As we can see in the table, the reduction in the number of classes (and hence in the number of queries submitted) is very important, especially in the case of Atlas, which has the same number of mappings than EGM but these allow less combinations in the rewritings. It is also noteworthy that the number of clauses is unaltered after the last prune phase, which has less clauses to prune due to the prunes performed previously.

5 Conclusions

In this paper we have presented an extension of the REQUIEM algorithm, used for query rewriting, where we propose to reduce the number of queries finally generated by adding two additional pruning steps. As a first consequence of the work performed, the number of queries can be reduced when considering the information provided by the RDB2RDF mappings. Because of this reduction there may be a gain in efficiency, taking less time to rewrite the query and, most importantly, to execute it. Therefore, an algorithm that takes into account the capabilities of the source of information can make a better use of this queried source.

There is still room for improvement in our work, especially in information integration scenarios involving heterogeneous data sources with overlapping data. The intersection among the individuals from several concepts may not be null and some may not be required to obtain the same result, specially among subclasses. This redundancy of information in related concepts is not considered, and it is the responsibility of the user to specify whether there is overlap in the individuals provided by some classes and their respective subclasses. However, when the overlap is partial and multiple, pruning to the minimum set of predicates to retrieve the maximum number of answers to the original query is not trivial. Thus, the description of the information provided by the source, with respect to the concepts provided, should be improved in this sense.

Similarly, the information regarding the sources of information available and the information that they provide is not considered here. When the same concept is provided by several sources of information the individuals that these contain may overlap in many different ways. The same way that concept inclusion can

be considered, these concepts could have information about their provenance, allowing the algorithm to differentiate among the concepts that are provided by different sources and again the overlapping that may exist among them, allowing better decisions on the selection of sources.

It must be noted that the improvement presented here depends on the mappings that are provided. In an extreme case, if all the concepts in the ontology were mapped to the database, the results provided on the H (prune to highest) mode would simply be the query provided with no inference on it, while those provided on the L (to the lowest) mode of this modification would be those provided by the original REQUIEM, with some computational overhead, that could be prevented by switching to the A (keep all) mode, which is the original unmodified REQUIEM algorithm.

Finally, the unfolding process performed after saturation preserves all the possible inferences by anticipating some of them, those related with the clauses that contain non-mapped predicates and thus are going to be removed. This partial unfolding may result in an increase of the number of clauses despite of the prune performed. For instance, when several non-mapped predicates are present in the body of a clause and these predicates are the head of several clauses, the possible combinations of unifications of the latter in the former may exceed the original number of clauses.

Acknowledgements. This work has been supported by an R&D grant from UPM. We would like to kindly thank Luis Manuel Vilches-Blázquez, Freddy Priyatna, Miguel Ángel García and Boris Villazón-Terrazas for the mappings provided as well as Héctor Pérez-Urbina for sharing the code, which allowed this work.

References

1. S. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau Jr, S. Auer, J. Sequeda, and A. Ezzat, "A survey of current approaches for mapping of relational databases to RDF," W3C RDB2RDF incubator group report, W3C, Jan. 2009.
2. C. Bizer, "D2R MAP - a database to RDF mapping language.," in *The Twelfth International World Wide Web Conference*, (Budapest, Hungary), May 2003.
3. J. Barrasa, O. Corcho, and A. Gómez-Pérez, "R2O, an extensible and semantically based database-to-ontology mapping language," *2nd Workshop on Semantic Web and Databases (SWDB2004)*, vol. 3372, 2004.
4. S. Das, S. Sundara, and R. Cyganiak, "R2RML: RDB to RDF mapping language." http://www.w3.org/TR/2010/WD-r2rml-20101028/, Oct. 2010.
5. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner, "Ontology-based integration of information-a survey of existing approaches," in *Workshop: Ontologies and Information Sharing*, vol. 2001, pp. 108–117, 2001.
6. N. W. Paton, C. A. Goble, and S. Bechhofer, "Knowledge based information integration systems," *Information and Software Technology*, vol. 42, pp. 299–312, Apr. 2000.
7. A. Halevy, A. Rajaraman, and J. Ordille, "Data integration: the teenage years," in *Proceedings of the 32nd international conference on Very large data bases*, (Seoul, Korea), pp. 9–16, VLDB Endowment, 2006.

8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Tractable reasoning and efficient query answering in description logics: The DL-Lite family," *Journal of Automated Reasoning*, vol. 39, pp. 385–429, Oct. 2007.
9. J. Bleiholder, S. Khuller, F. Naumann, L. Raschid, and Y. Wu, "Query planning in the presence of overlapping sources," in *Advances in Database Technology - EDBT 2006*, Lecture Notes in Computer Science, pp. 811–828, Springer, 2006.
10. M. Friedman, A. Levy, and T. Millstein, "Navigational plans for data integration," in *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference*, (Orlando, Florida, United States), pp. 67–73, 1999.
11. C. Caracciolo, J. Heguiabehere, A. Gangemi, W. Peters, and A. Stellato, "Second network of fisheries ontologies," deliverable D7.2.4, NeOn project, 2010.
12. H. Pérez-Urbina, B. Motik, and I. Horrocks, "Tractable query answering and rewriting under description logic constraints," *Journal of Applied Logic*, vol. 8, pp. 186–209, June 2010.
13. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev, "The DL-lite family and relations," *J. Artif. Int. Res.*, vol. 36, no. 1, pp. 1–69, 2009.
14. B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel Schneider, and U. Sattler, "OWL 2: The next step for OWL," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, pp. 309–322, Nov. 2008.
15. H. Pérez-Urbina, I. Horrocks, and B. Motik, "Efficient query answering for OWL 2," in *The Semantic Web - ISWC 2009*, vol. 5823 of *Lecture Notes in Computer Science*, pp. 489–504, Springer, 2009.
16. L. M. Vilches-Blázquez, M. Á. Bernabé-Poveda, M. C. Suárez-Figueroa, A. Gómez-Pérez, and A. F. Rodríguez-Pascual, "Towntology & hydrOntology: relationship between urban and hydrographic features in the geographic information domain," *Ontologies for Urban Development*, vol. 61, pp. 73–84, 2007.
17. A. Gómez-Pérez, J. Ramos, A. Rodríguez-Pascual, and L. Vilches-Blázquez, "The IGN-E case: Integrating through a hidden ontology," in *Headway in Spatial Data Handling* (A. Ruas and C. M. Gold, eds.), Lecture Notes in Geoinformation and Cartography, pp. 417–435, Springer, 2008.

Note: This paper was submitted to ESWC2011 as it is here. The only differences are this note and minor changes to references [6] and [14] which contained errors that have been corrected in a referred file. Everything else is left unaltered for reference purposes.