# Towards Populating and Querying the Semantic Web

Dejing Dou[1], Jeff Z. Pan[2], Han Qin[1] and Paea LePendu[1]

[1] Computer and Information Science
University of Oregon, USA
{dou,qinhan,paea}@cs.uoregon.edu
[2] Department of Computing Science
University of Aberdeen, UK
jpan@csd.abdn.ac.uk

**Abstract.** There are two main issues on querying the Semantic Web: what data can be queried and how to query them. This paper tackles how to query existing data in relational databases over popular ontologies in the Semantic Web. It shows (i) how the mappings between Description Logic-based ontologies and relational schemas can be specified, and (ii) how queries against ontologies can be translated to SQL queries that can be answered by existing databases, with the help of an ontology-based information integration system called OntoGrate. Due to the obvious differences between the complexity of ontology-based query answering and that of database querying, this approach is incomplete. However, it can guarantee the correctness of query answering by a sound inferential query translation, which also benefits from DL reasoning with relationships among concepts that may be implicit in the databases. The paper also reports on preliminary, but encouraging, experimental results.

## 1 Introduction

How to support query answering on the Semantic Web [3] is among the most useful and important research problems in the Semantic Web. There are mainly two issues at stake: *what to query and how to query.* The issue of how to populate data on the Semantic Web seems to be quite obvious. There exist more and more RDF/OWL [3] ontologies, which can be used in annotations. However, the existing RDF/OWL files contain much less data than those stored in potentially related databases. We can either annotate all the data in databases using RDF/OWL, which is redundant and hard to scale for large size databases. Or we can provide some mechanism to directly relate ontologies with these databases. In this paper, we argue that the latter approach has a much promising future.

There are two main challenges in supporting current Semantic Web querying languages, such as OWL-QL [11], to query relational databases. Firstly, the semantics of an OWL-QL query are defined by ontologies, but the semantics of

[3] http://www.w3.org/TR/owl-ref/

relational data are defined by relational models. In general, ontologies are more expressive and have richer semantics than relational schemas, which only focus on the structure and integrity constraints of data tables. It is difficult to find exact mappings between the concepts of ontologies and those in the relational schemas. Secondly, we must overcome syntax differences, since most relational databases can only accept SQL queries.

In the rest of the paper, we first give some background (section 2) on query answering over OWL DL ontologies, e.g., using the OWL-QL query language [11]. The complexity of query answering by reasoning over OWL data instances is at least NEXPTIME hard and it is not very likely to be scalable to handle large size relational databases. In this paper, we argue that a sound but incomplete approach is a good way to solve this scalability problem. Firstly, we specify a way to specify the mappings between a domain ontology and a relational schema, and we introduce our inferential query translation model (section 3). We illustrate our approach to query existing databases with ontologies by using our ontology-based information integration system, OntoGrate (section 4). Due to the obvious differences between the complexity of ontology-based query answering and that of database querying, our approach is incomplete, but it can guarantee the correctness by sound inference. Besides the good performance indicated by our preliminary experiments, we show that ontology-based reasoning is both necessary and helpful to get answers regarding the relationships among concepts that may be implicit in the databases. We discuss some related work (section 5) before we conclude the paper (section 6).

## 2 Background

### 2.1 OWL-QL

The OWL-QL [11] specification is a language and protocol for query-answering dialogues using knowledge represented in OWL. It is a direct successor of the DAML Query Language (DQL) [12]. Both language specifications go beyond the aims of other current web query languages like XML Query [4], an XML query language, or RQL [14], an RDF query language, in that they support the use of inference and reasoning services for query answering. The OWL-QL specification suggests an independent reasoner, which helps agents (clients) query OWL knowledge bases on the Semantic Web in a more general way.

OWL-QL queries are conjunctive queries w.r.t. some knowledge bases (or simply, KBs). A query necessarily includes a query pattern that is a collection of OWL statements (axioms), where some URI references or literals are replaced by variables. In a query, the client can specify which variables the server has to provide a binding (must-bind variables, or distinguished variables, denoted as ?x) for, which variables the server may provide a binding (may-bind variables, denoted as ∼x) for, and which variables no binding (don't-bind variables, or non-distinguished variables, denoted as !x) should be returned. A client uses an answer KB pattern to specify which knowledge base(s) the server should use to answer the query. An answer KB pattern can either be a KB, a list of KB URI references, or a variable (of the above three kinds); in the last case, the server is allowed to decide which KB(s) to use. The use of may-bind and don't-bind variables is one of the features that clearly distinguishes OWL-QL from standard database query languages (such as SQL) and other web query languages (such as RQL and XML Query). Here is an example of a query pattern and an answer KB pattern.

```
queryPatten: {(name !x ?n),(mbox !x ?m),(homepage !x, ~h)}
answerKBPattern: {http://owlqlExample/people.owl}
```

Given the people KB, the above query asks for anything which has a name, a mbox, and a homepage. Note that ∼h is a may-bind variable; therefore, only known homepages are returned.

## 2.2 Reducing Ontology Query Answering to Instance Checking

Query answering over ontologies is generally a hard reasoning problem, at least as hard as instance checking w.r.t. an ontology. For example, if there are no distinguished variables in an acyclic query, there are two possibilities [17]: (i) For a query $q$ in which there exist some non-distinguished variables, query answering of $q$ can be reduced to concept satisfiability checking. (ii) For a query $q$ in which there exists no non-distinguished variables, query answering of $q$ can be reduced to instance checking. The possibility (ii) is more close to the scenario of database query answering. As instance checking in OWL DL is NEXPTIME, query answering over OWL DL ontologies is at least NEXPTIME. This high complexity suggests that it is a hard to provide a scalable ontology querying service for a huge amount of data. Applying sound but incomplete approach seems to be a convincing direction.

## 2.3 General Mappings between Ontologies and Relational Schemas

Currently, relational databases are some of the largest data resources in the world. The structure and integrity constraints of relational tables are defined by database schemas. Query answering techniques of relational DBs using table joins are mature. Query optimization also helps make these techniques more efficient, although database schemas do not have as rich semantics as those of ontologies.

When Semantic Web applications (which use ontologies) want to query relational databases, we first need to deal with both the semantic gap between ontologies and schema and syntax differences. We have developed an automatic translator, PDDSQL [8], between SQL and our internal Web ontology language, Web-PDDL (a strongly typed first-order logic language with Lisp-like syntax), to express ontologies, data instances (facts), queries, and mapping rules between different ontologies. Web-PDDL can be translated to and from RDF easily [16]. A recent syntax translator called PDDOWL [8] can translate OWL-QL to Web-PDDL as well.

In our initial findings [7], a few general mapping rules relating ontologies to schemas accomplished the majority of translations between ontology based queries and SQL queries:

$$\begin{aligned}\mathsf{Relation} &\leftrightarrow \mathsf{Class\ (Type)\ or\ ObjectProperty\ (Predicate)}\\ \mathsf{Attribute} &\leftrightarrow \mathsf{DatatypeProperty\ (Predicate)}\\ \mathsf{Integrity\ Constraint} &\leftrightarrow \mathsf{Rule\ (Axiom)}\\ \mathsf{Primary\ Key} &\leftrightarrow \mathsf{Fact\ (Statement)}\end{aligned}$$

These rules play an important part in developing the syntax translation functions in PDDSQL. In section 4, we will also show that these rules can be used to generate a *DB ontology* in Web-PDDL from a database schema in SQL.

## 3 Semantic Mappings and Inferential Query Translation

### 3.1 Semantic Mappings between domain ontologies and schemas

In simple scenarios, concepts in a domain ontology may have nearly the same meanings as those in a database schema. The domain ontology can play a similar role to the conceptual model of the database. The query translation mainly needs to handle the syntactic differences. For example, if the Foo university ontology has two “Professor” and “Student” classes and an “advising” (object) property between them, there can be three corresponding relations (i.e., “Professor”, “Student” and “Advising”) defined in a database. The mappings can be as simple as one to one correspondences. The “Student” class may have a “status” (datatype) property, and there can be an attribute in the corresponding “Student” relation, and so on. If we have an OWL-QL query in abstract syntax:

```
premise: {Professor(Anderson)}
queryPattern: {(advising Anderson ?s)
               (status ?s "Graduate"}
answerKBPattern: {http://universityOntos/FooUniversity.owl}
```

In Web-PDDL, it is:

```
(url "http://universityOntos/FooUniversity.owl" prefix "Foo")
(:objects Anderson - @Foo:Professor)
(and (advising Anderson ?s - @Foo:Student)
     (status ?s "Graduate"))
```

This Web-PDDL query can easily be translated to a SQL query by using PDDSQL with the general mappings in section 2.3. The SQL query looks like:

```
SELECT S.name
FROM Student S, Advising A, Professor P
WHERE S.status = "Graduate" AND P.name = "Anderson"
```

The query can be answered by the join of these three tables. However, in a slightly more complex scenario, the concepts in an ontology may not use the exact same meanings as those in a database schema, even if they are talking about the same domain. For example, a university ontology from the UK may use the "Lecturer" class, but the university database in the US may use the "Faculty" table and a "title" attribute. Experts familiar with the educational systems of the two countries need to specify a mapping that indicates the "Lecturer" class is equivalent to US "Faculty" that have the "Assistant Professor" title. The semantics of this mapping become complex. In Web-PDDL, it may look like:

```
(forall (x)
    (if (isa @UK:Lectuer x)
        (and (isa @US:Faculty x) (title x "Assistant Professor"))))
```

Some semi-automatic tools, such as [1, 2], can help domain experts find similar semantic mappings between database/XML schemas and ontologies. The focus of this paper is how to represent those mappings in a formal language, and how to use those mappings for query answering.

### 3.2 Inferential Query Translation

Before showing how to get the answers from real databases using semantic mappings, we want to formally define the problem of query translation and show how inference can solve the semantic differences, forming the basis of our *inferential query translation* model. It is different from general rewriting approaches in the database community, such as the approach in [18], which care less about semantics and reasoning than Semantic Web ontology-based queries.

The assumption here is that we already have the semantic mappings as mapping rules in a logic language, such as Web-PDDL, and leave the syntax issues to automatic translators, such as PDDSQL and PDDOWL.

**Definition** *Query Translation*: The process of extracting data expressed by a database schema to answer a query posed using a (different) ontology.

Suppose we already have mappings between the concepts in Ontology $Onto_s$ and those in Schema T from $DB_t$. Let the set of mappings be denoted $M_{s\_t}$. Let the symbol $\leadsto_Q$ indicate query translation. If $q_s$ is a query in Ontology $Onto_s$, its translation is a query $q_t$ in Schema T, such that any answer (set of bindings) to $q_t$ is also an answer to $q_s$. In other words:

$$(M_{s\_t}; q_s) \leadsto_Q q_t \ \textit{only if}\ (M_{s\_t}; \theta(q_t)) \models \theta(q_s)$$

for any substitution $\theta$, where $\theta(q_t)$ is the answer(s) from the target database $DB_t$. It means we use entailment ($\models$) to define the query translation: if all the mapping rules in $M_{s\_t}$ and all the answers in $\theta(q_t)$ are true, then all the answers in $\theta(q_s)$ should be true. Alternatively, we say that $\theta(q_s)$ is a logical (or semantic) consequence of $M_{s\_t}$ and $\theta(q_t)$. It is easy to get, for any substitution $\theta$,

$$\begin{aligned}(M_{s\_t}; q_s) \leadsto_Q q_t \ &\Leftrightarrow\ (M_{s\_t}; \theta(q_t)) \vdash \theta(q_s) \\ &\Rightarrow\ (M_{s\_t}; \theta(q_t)) \models \theta(q_s)\end{aligned}$$

where an (sound) inference ($\vdash$) can actually implement and guarantee the entailment. We claim that $q_t$ is the translation of $q_s$ if and only if, for every substitution $\theta$, $\theta(q_t)$ is the *weakest* statement in Schema T such that $\theta(q_t)$ is from $DB_t$,

$(M_{s\_t}; \theta(q_t)) \vdash \theta(q_s)$ and $M_{s\_t} \nvdash \theta(q_s)$. The weakest statement means that $q_t$ need not be (and seldom is) *equivalent* to $q_s$, in the sense that any answer to one is an answer to the other. All we need is that any answer to $q_t$ be an answer to $q_s$. In the literature this is also known as *query containment* [15]. Based on this model, we claim that the answers from the target database may not be complete but correct with regards to the source query as long as the inference is sound.

In order to use semantic mappings for inferential query translation, the special purpose inference engine OntoEngine [9] was used. OntoEngine has both forward chaining and backward chaining reasoners using generalized modus ponens [19], which is well known as a sound inference algorithm.

## 4 Direct Querying Relational DBs with Ontology-based Reasoning

OntoGrate is an ongoing research project [7, 8] to develop an ontology-based information integration framework that can integrate information that may be heterogenous in both semantics and syntax, such as databases, knowledge bases, XML files and Semantic Web data. However, in this paper, we mainly use OntoGrate as a system to support directly querying relational DBs with Semantic Web ontology-based reasoning. We will use real domain ontologies and databases to illustrate our approach.

### 4.1 System Architecture and an Illustrative Example

The system architecture to support OWL-QL queries to query relational databases is shown in Figure 1:

The process can be summarized as following:

1. *Semantic Mapping Representations:* Using a formal language (i.e., Web-PDDL) to represent semantic mappings between the *DB ontology* (generated from the database schema) and the domain ontology as logic rules.
2. *Syntax Translation, Phase One:* Translating an OWL-QL query into a Web-PDDL query using PDDOWL, with both queries defined by the domain ontology.
3. *Query Translation:* Translating a domain ontology-based query into a database schema-based query by inferential query translation with OntoEngine, with both of them in Web-PDDL syntax.
4. *Syntax Translation, Phase Two:* Translating a Web-PDDL query into a SQL query using PDDSQL.
5. *Querying Relational Databases:* Running a SQL query in the relational database and annotating answers (database tuples) with OWL using PDDSQL and PDDOWL.

To show that our approach works for real ontologies and databases in similar or related domains, we searched for OWL ontologies and commercial databases

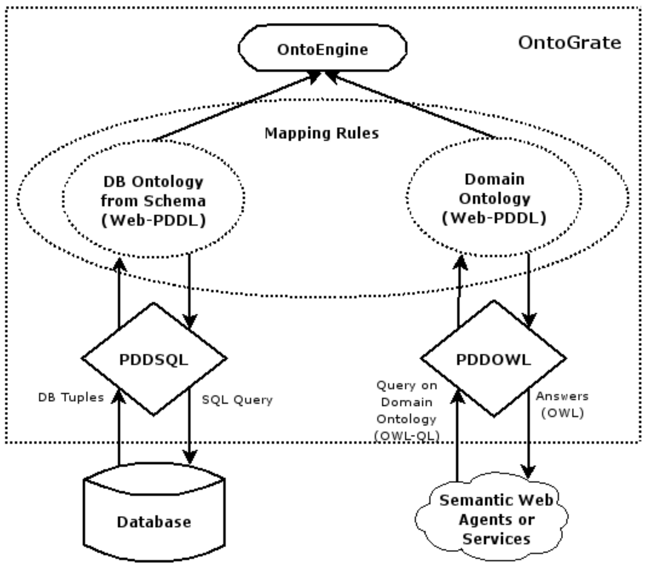
OntoGrate
OntoEngine
Mapping Rules
DB Ontology from Schema (Web-PDDL)
Domain Ontology (Web-PDDL)
PDDSQL
PDDOWL
DB Tuples
SQL Query
Query on Domain Ontology (OWL-QL)
Answers (OWL)
Database
Semantic Web Agents or Services

**Fig. 1.** Querying Relational Databases with Semantic Web Domain Ontologies.

on the Web. In this paper we consider an IBM Informix[4] database, Stores7, from the online sales domain, which we installed for the experiments. Stores7 defines relations such as customer, order and item, as well as the associated attributes. It was easy for us to find an Order OWL ontology[5] within a similar domain.

### 4.2 DB Ontology Generation and Semantic Mapping Representation

After we obtain the Stores7 schema[6] from its SQL representation and the Order OWL ontology, we first translate them into Web-PDDL syntax using PDDSQL and PDDOWL. The Web-PDDL representation of Stores7 is generated by using PDDSQL with the general mappings in section 2.3. We call the result a *DB ontology* for Stores7. The DB ontology can capture as much semantics as possible from the original SQL schema definitions. For example, there is a `Customer` table in Stores7, where `custmernumber` is the primary key of the table and `customerstatecode` is the foreign key that refers to the State table. By using PDDSQL, it can be translated to Web-PDDL as:

```
(:types Customer State - @sql:Relation)
(:predicates (customerfname x - Customer y - @sql:varchar)
             (customerlname x - Customer y - @sql:varchar)
             (customercity x - Customer y - @sql:varchar)
             (customerstatecode x - Customer y - @sql:varchar)   ...)
```

[4] http://www.ibm.com/software/data/informix/
[5] http://www.dayf.de/2004/owl/order.owl
[6] http://www.cs.uoregon.edu/~paea/research/stores7_schema.png

```
   (:axioms  (forall (c - Customer code - @sql:varchar)
                     (if (customerstatecode c code)
                         (exists (s - State) (statecode s code))))
   (:facts   (@sql:primarykey Customer "customernumber")...))
```

where we use `@sql:` as the prefix for concepts and datatypes defined in `SQL` language. The above Web-PDDL representation can be thought of as being part of the DB ontology of the `Stores7` database, which specifies the `Customer` and `State` types (classes) and their related predicates (properties) and axioms.

The Order ontology can also be represented in Web-PDDL using PDDOWL. Some properties related to the `Person` and `Address` classes can be defined as:

```
   (:types Agent Address
           Person - Agent)
   (:predicates (FirstName p - Person f - string)
                (LastName p - Person f - string)
                (hasAddress a - Agent ad - Address)
                (City a - Address c - string))
```

It is easy for humans to find mappings between concepts of Order and Stores7, maybe with the help of some mapping tools such as the work introduced in [1, 2]. Some example mappings look like:

```
(@stores7:Customer - @order:Person)

(forall (C - @stores7:Customer x - String)
   (iff (@stores7:customerfname C x) (@order:FirstName C x)))

(forall (C - @stores7:Customer y - String)
   (iff (@stores7:customerlname C y) (@order:LastName C y)))

(forall (C - @stores7:Customer z - String)
    (if (@stores7:customercity P z)
         (exists (A - @order:Address)
                 (and (@order:hasAddress P A)
                      (@order:City A z)))))
```

In the first mapping rule, `-` means that `Stores7`'s `Customer` is the sub class (type) of `Order`'s `Person`.

## 4.3 Querying with Explicit Mappings

The semantics of the mapping rules in section 4.2 are easy to understand. When we have those mappings, an OWL-QL query can first be translated into a Web-PDDL query. For example, a query in OWL-QL abstract syntax:

```
 premise: Person(C), Address(A), (City A "Eugene"), (hasAddress C A)
 queryPattern: {(FirstName C ?x)
                (LastName C ?y)}
 answerKBPattern: {http://www.dayf.de/2004/owl/order.owl}
```

can be translated into a Web-PDDL query by PDDOWL, which also uses the `Order` ontology:

```
(and (hasAddress ?C - Person ?A - Address)
     (City ?A "Eugene")
     (FirstName ?C - Person ?x - String)
     (LastName ?C - Person ?y - String))
```

OntoEngine uses backward chaining to do query translation from this query over `Order` ontology into a Web-PDDL query that uses the `Stores7` concepts:

```
(and
  (customercity ?C - Customer "Eugene")
  (customerfname ?C - Customer ?x - String)
  (customerlname ?C - Customer ?y - String))
```

PDDSQL then takes this Web-PDDL query and translates it into a SQL query:

```
SELECT C.customerfname, C.customerlname
FROM Customer C
WHERE C.customercity = "Eugene"
```

The SQL query returns a table, which is then transformed into Web-PDDL bindings (answers) by PDDSQL:

```
{?x/Dejing, ?y/Dou} {?x/Mike, ?y/Matloff} {?x/Han, ?y/Qin} ...
```

The OWL-QL answer bundle returned uses the `Order` ontology (i.e., answer KB):

```
<owl-ql:answer>
 ...
   <owl-ql:answerPatternInstance>
     <rdf:RDF>
       <rdf:Description rdf:about="#C">
         <FirstName rdf:resource="#Dejing"/>
         <LastName rdf:resource="#Dou"/>
       </rdf:Description>
     </rdf:RDF>
   <owl-ql:answerPatternInstance>
   ...
```

## 4.4 Testing Results for large data size and different queries

First we populated the Stores7 database (running on a local PC) with 100,000 data records.[7] Then we tested queries that returned result sets of varying sizes. The time that OntoGrate took to process each query is displayed in Figure 2. We have used around 22 mapping rules between Order and Stores7. The queries we used for testing have different complexities, although they are all conjunctive

[7] All experiments were performed on a 1.8Ghz Centrino processor with 1Gb of RAM and a MySQL database engine.

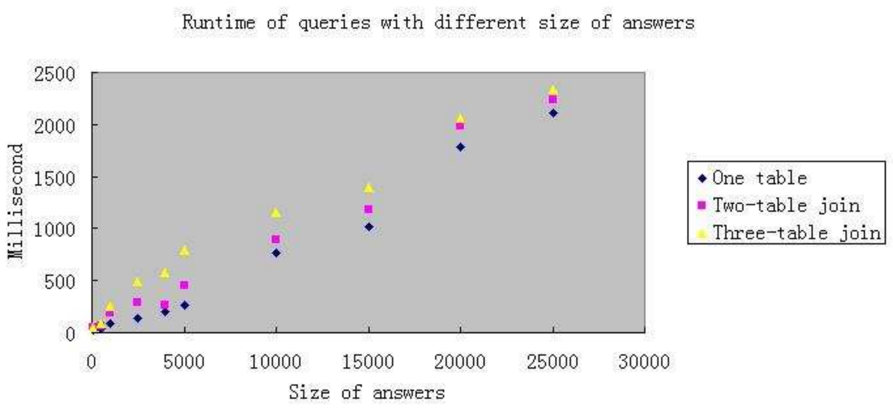

**Fig. 2.** The runtime of different size of answers.

queries. As Figure 2 shows, when the queries translated to SQL, they may need different number of table join (i.e., from one to three) and subgoals. For example, answering the query in Section 4.3 only needs `Customer` table in Stores7 and it has 2 subgoals. Answering the following query in OWL-QL abstract syntax:

```
queryPattern: { (billto !PurchaseOrder !Organization)
                (hasitems !PurchaseOrder !ItemsCollection)
                (hasitem !ItemsCollection !PurchasedItem)
                (partnumber !PurchasedItem ?x)
                (quantity !PurchasedItem "1")}
answerKBPattern: {http://www.dayf.de/2004/owl/order.owl}
```

needs to join `Customer`, `Order` and `Item` tables in Stores7 and it has 5 subgoals. However, all queries show query answering, which includes both query translation and running SQL Query in DB, can be finished in polynomial time. To get 25,000 answers from 100,000 records needs about 2500 milliseconds. The query translation (from `Order` to `Stores7`) with various queries always takes less than 100 milliseconds. It shows the query translation is answer size independent and it is fast enough with respect to the whole query answering process.

### 4.5 Querying with Implicit Relationships and Mappings

Since a DL-based ontology normally has richer semantics than a database schema, it may not be easy for mapping tools, or even a human being, to find all semantic mappings between them. Some concepts or relationships in an ontology may be implicit in a database. This makes it hard for a query answering system to get all of (completeness), and exactly (correctness), answers from database data.

As an example, we extend the `Personnel` concept in the current Order OWL ontology a bit using two axioms in DL syntax:

$$\texttt{Personnel} \sqsubseteq \exists\texttt{supervised.Consultant} \sqcap \forall\texttt{supervised.Manager}$$
$$\exists\texttt{supervised.(Consultant} \sqcap \texttt{Manager)} \sqsubseteq \texttt{Staff}$$

Basically, the above axioms say that each Personnel is supervised by some Managers, at least one of which is a Consultant, and that those supervised by someone who is both a Consultant and Manager are Staffs. Suppose an OWL-QL query can be defined using this extended Order ontology to list all Staffs:

```
queryPattern: { (Staff(?P)
                (FirstName ?P ?x)
                (LastName ?P ?y)}
answerKBPattern: {http://.../extendedOrder.owl}
```

However, it is difficult to exactly map this new Staff concept to concepts in the Stores7 schema or its DB ontology, even with mapping tools. Note that Stores7 has an Employee table. To show our idea, we add a supervisorid attribute into it: Employee(employeeid, employeelname, employeefname,... supervisorid), where supervisorid refers to some other Employee. Although domain experts or mapping tools may be able to guess that Staff can be mapped to Employee, it is not clear whether we should map Consultant or Manager to Employee as well. More naturally, domain experts can only be sure that Employee is one kind of Personnel. So the only mapping we are sure to get is: @stores7:Employee - @order:Personnel (in Web-PDDL syntax) or Employee $\sqsubseteq$ Personnel (in DL syntax). However, this mapping is not enough to directly answer the query related to Staff, whose mapping with Employee is not explicit. In the following, we will show that ontology-based reasoning can help the query answering system to get correct (but maybe not complete) answers.

By making use of the DL classification service, we have Personnel $\sqsubseteq$ Staff. Thus, if 1) Staff is used as the query, and 2) Staff has no corresponding table in the database, and 3) Personnel has a corresponding table in the database, then we rewrite Staff(?x) in the query as Personnel(?x). It actually uses the following conclusion after reasoning: *All Personnels are Staffs.*

With the above reasoning, the Staff related query can be reformed as:

```
queryPattern: { (Personnel(?P)
                (FirstName ?P ?x)
                (LastName ?P ?y)}
answerKBPattern: {http://.../extendedOrder.owl}
```

Therefore, the domain expert only needs to figure out simple mapping between Employee in the Stores7 schema and Personnel in the Order ontology and the mapping between the related supervisorid attribute and supervised property. It is not hard to do this, even if the domain expert is not sure about the relationships among Manager, Consultant and Staff, which are implicit in the database. The mappings in Web-PDDL look like:

```
(@stores7:Employee - @order:Personnel)

(forall (p - @stores7:Employee m - @order:Personnel)
     (if (@order:supervised p m)
         (exists (id - String)
                 (@stores7:supervisorid p id)))
```

In other words, an Employee is a kind of (subclassof) Personnel. If an Employee is supervised by another Personnel, he has a supervisorid.

OntoEngine can use the mappings above to translate the Staff related queries into queries using the Stores7 concepts.

```
(and (supervisorid ?S - Employee Sk_ID1 - String)
     (employeefname ?S ?x - String)
     (employeelname ?S ?y - String))
```

where Sk_ID1 is a skolem term generated by query translation; it means there exists a supervisorid for any personnel supervised by someone else. Finally, PDDSQL can transform this to a SQL query:

```
SELECT S.employeefname, S.employeelname
FROM Employee S
WHERE S.supervisorid != NULL
```

The SQL query can get all employees' names, whose supervisorid is not the NULL value, from the Stores7 database directly. The answers are correct according to what Staff means in the extended Order ontology.

**Discussion:** The test results for large-size data are similar from those of section 4.4, since the reasoning on two DL axioms does not increase the overhead much. However, some concepts in the ontologies may not have corresponding concepts in the schemas, or it could be difficult to specify the exact mappings. For example, if the OWL-QL query is Consultant(?P) in the extended Order ontology, we cannot get any answers from the Stores7 database because we cannot specify any mapping between the Stores7 concepts and Consultant.

## 5 Related Work

The primary goal of this paper is to show how the mappings between Description Logic-based ontologies and relational schemas can be specified and used to translate queries over ontologies to SQL queries that can be answered using existing databases. Related approaches are found in several disciplines:

*Mappings between Schemas and Ontologies*: The research in [2, 1] provides very interesting methods to construct complex mapping rules between relational tables or XML data and ontologies when given an initial set of correspondences between the concepts in the schemas and ontologies. They offer the mapping formalisms to capture the semantics of XML or relational schemas by constructing the semantic trees from them. Although their generated rules cannot be 100% correct, they will be useful to domain experts for further refinement, as well as to applications. Our research focuses on specifying the mapping rules as formal logic rules, which can be used for reasoning by an inference engine to translate queries over ontologies to relational schemas.

*Query answering by query rewriting*: In data integration and data exchange research [15, 10], the query based on one schema can be answered by rewriting it to the query based on another schema. Most of them are view-based query answering. For example, the MiniCon algorithm [18] rewrites queries expressed in a global view to a conjunction of queries over local ones so that a large set of (materialized) views can lead to efficient query answering. The query rewriting is defined by *query containment*[15]. Our query translation from the domain ontology to database schema is defined by the logic entailment and implemented as a sound inference.

*Bringing Relational Data into the Semantic Web*: A recent approach can transform relational schemas and data to OWL by using an ontology (i.e., Relational.OWL) defined for relational concepts [6]. It also can use SPARQL [8] to query the annotated data to simulate SQL query operations. Our research tries to annotate answers only and not all data from relational databases. More importantly, we focus on the scenario in which Semantic Web agents or services use already existing domain ontologies, which may need semantic mappings to be specified to connect the concepts in database schemas.

Last but not least, our approach is also similar to the DL Lite [5] approach, which argues that we should use a simple ontology language, so that query answering over ontologies can be completely reduced to query answering over databases. As OWL DL is the Semantic Web standard ontology language, we do not restrict users to only using DL Lite in our approach. The price we pay is that we may not provide complete results for some queries.

## 6 Conclusions and Future Work

Huge amounts of data stored in databases are potential candidates for populating the Semantic Web. The problem is how to query databases with related Semantic Web ontologies. In this paper, we have proposed to accomplish this by specifying the semantic mappings between ontologies and schemas in order to achieve this goal. Sound inference on mappings and ontology axioms are necessary to guarantee that the correctness of query answering. Our approach provides a sound but incomplete approximation for ontology-based query answering over databases. Initial experiments on OntoGrate have shown encouraging results.

Although we use OWL-QL to show our idea, we can easily switch it to the coming standard query language SPARQL by (1) using OWL entailment as the E-entailment regime and (2) developing a syntax translator between it and the internal language (Web-PDDL) of our system. Our focus is to solve semantic difference but not syntactic difference. In the immediate future, the scalability and efficiency of OntoGrate will be investigated in larger-size relational databases of various domains. We want to use some standard benchmarks to compare our system with other Semantic Web querying systems, such as the work reported in [13]. We also want to test OntoGrate with the scenario in which the answers for a query on the domain ontology may come from multiple databases. In that scenario, we can assume the database experts will specify the mappings between their local databases and a (popular) domain ontology, but we need to consider the consistency of the answers coming from different resources. This is similar to the LAV (local-as-view) scenario in data integration research [15].

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

## References

1. Y. An, A. Borgida, and J. Mylopoulos. Constructing complex semantic mappings between xml data and ontologies. In *International Semantic Web Conference*, pages 6–20, 2005.
2. Y. An, A. Borgida, and J. Mylopoulos. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *OTM Conferences (2), ODBASE*, pages 1152–1169, 2005.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5), May 2001.
4. XQuery 1.0: An XML query language. URL, Nov 12 2003. `http://www.w3.org/TR/2003/WD-xquery-20031112/`.
5. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.
6. C. P. de Laborda and S. Conrad. Bringing relational data into the SemanticWeb using SPARQL and Relational.OWL. In *SWDB'06*, page 55, 2006.
7. D. Dou and P. LePendu. Ontology-based Integration for Relational Databases. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, pages 461–466, 2006.
8. D. Dou, P. LePendu, S. Kim, and P. Qi. Integrating Databases into the Semantic Web through an Ontology-based Framework. In *Proceedings of the third International Workshop on Semantic Web and Databases (SWDB'06)*, page 54, 2006.
9. D. Dou, D. V. McDermott, and P. Qi. Ontology Translation on the Semantic Web. *Journal of Data Semantics*, 2:35–57, 2005.
10. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In *ICDT*, pages 207–224, 2003.
11. R. Fikes, P. Hayes, and I. Horrocks. OWL-QL—a language for deductive query answering on the Semantic Web. *J. of Web Semantics*, 2(1):19–29, 2004.
12. DAML Query Language (DQL) abstract specification. URL, Apr 2003. `http://www.daml.org/2003/04/dql/`.
13. Y. Guo, Z. Pan, and J. Heflin. An Evaluation of Knowledge Base Systems for Large OWL Datasets. In *International Semantic Web Conference*, pages 274–288, 2004.
14. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A declarative query language for RDF. In *Proceedings of the eleventh international conference on World Wide Web*, pages 592–603. ACM Press, New York, USA, May 7–11 2002.
15. M. Lenzerini. Data Integration: A Theoretical Perspective. In *Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.
16. D. V. McDermott and D. Dou. Representing Disjunction and Quantifiers in RDF. In *International Semantic Web Conference*, pages 250–263, 2002.
17. J. Z. Pan, E. Franconi, S. Tessaris, B. Glimm, W. Siberski, G. Stamou, V. Tzouvaras, I. Horrocks, L. Li, and H. Wache. Report on Query Language Design and Standardisation. Technical report, The Knowledge Web project, Dec 2004.
18. R. Pottinger and A. Levy. A Scalable Algorithm for Answering Queries Using Views. In *Proceedings of the 26th VLDB Conference*, pages 484–495, 2000.
19. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc, 1995.