Integrating Databases into the Semantic Web through an Ontology-based Framework

Dejing Dou, Paea LePendu, and Shiwoong Kim
Computer and Information Science
University of Oregon
Eugene, Oregon 97403, USA
dou, paea, shkim@cs.uoregon.edu

Peishen Qi
Computer Science Department
Yale University
New Haven, Connecticut 06520, USA
peishen.qi@yale.edu

Abstract

To realize the Semantic Web, it will be necessary to make existing database content available for emerging Semantic Web applications, such as web agents and services, which use ontologies to formally define the semantics of their data. Our research in the design and implementation of an ontology-based system, OntoGrate, addresses the critical and challenging problem of supporting human experts in multiple domains to interactively integrate information that is heterogenous in both structure and semantics. Databases, knowledge bases, the World Wide Web, and the emerging Semantic Web are some of the resources for which scalable integration remains a challenge. To integrate databases into the Semantic Web, we use Semantic Web ontologies to incorporate database schemas. An expressive first order ontology language, Web-PDDL, is used to define the structure, semantics, and mappings of data resources. A powerful inference engine, OntoEngine, can be used for query answering and data translation. In this paper, besides introducing new ideas in the OntoGrate system, we will elaborate on two case studies for which our system works well.

1 Introduction

During the last several years, we have seen many achievements towards realizing the Semantic Web [5], where ontologies play a key role in defining the semantics of data. Besides developing Semantic Web ontology languages (e.g., OWL [2]) and applications (e.g., web agents and services [20]), it is necessary to make existing data resources, such as databases, available for Semantic Web applications to access and share. Currently, relational databases are some of the largest data resources in the world, but the structure and integrity constraints of relational tables are defined by schemas, which are not as expressive as ontologies when representing the semantics of data. To deal with both the expressivity gap between schemas and ontologies and the syntax difference between different definition languages, we apply our ontology-based information integration system, OntoGrate, and thereby integrate relational databases into the Semantic Web.

Despite considerable progress in research on information integration, as well as several commercial implementations in this field [15, 17], current information integration systems suffer from several major drawbacks. First, in the foreseeable future, databases, knowledge bases, the World Wide Web, and the Semantic Web will coexist. It is still a challenge for current systems to accommodate their contents, which are either defined by schemas based on structure or defined by ontologies based on semantics. Second, automatic tools cannot generate 100% accurate *matchings* and executable *mappings* based on the semantics of data: only domain experts can perform judicious validation. As such, integration systems must be interactive and usable by *domain experts who are not necessarily computer experts*. Third, the mapping tools and data translation or query answering systems may use different theoretical models to interpret the semantics of mappings. Syntax translators alone are inadequate, since semantics cannot be ignored.

As a result, the maturity of current research and practical applications has demonstrated the need for a broader and more formal approach to information integration. We have several goals to achieve in our ongoing research project, OntoGrate. First, we are extending the expressivity of Semantic Web ontologies to incorporate database schemas, defining both structure and semantics. Second, we are using machine learning and data mining techniques to provide more meaningful mapping rules to users, and stimulate further interaction with domain experts to justify the rules and make them executable. Third, our inference engine, OntoEngine [12], is being extended not only to help check the consistency and redundancy of mapping rules, but also to conduct optimized query answering and data translation. Once users in multiple domains generate large amounts of metadata (e.g., mapping rules), we expect to provide online services that make available to the public the storage and efficient access to large collections of metadata.

In this paper, we will focus on how to integrate databases into the Semantic Web by incorporating database schemas with Semantic Web ontologies. We will first summarize our previous work on ontologies, Web-PDDL, and our inferential data integration model (section 2). We will then briefly describe our new ideas for *interactive* information integration by introducing the OntoGrate architecture (section 3). In section 4, we will elaborate on two scenarios in which we integrate relational databases into the Semantic Web using the OntoGrate system. We will report the testing results based on these scenarios. We will finalize this paper with a section on related work (section 5) and present our future work and conclusions in section 6.

2 Previous work

2.1 Schemas, Ontologies, Web-PDDL

In previous work, we have described in some detail how to represent database schemas as ontologies and how to use logic axioms to merge two database schemas [11]. We use Web-PDDL [19], a strongly typed first order logic language with Lisp-like syntax, to internally describe and process these representations. We now briefly review and summarize this work.

Consider an Informix database schema, Stores7, from the online sales domain, in Figure 1. It defines the relations, such as customer, order and item, and associated attributes.

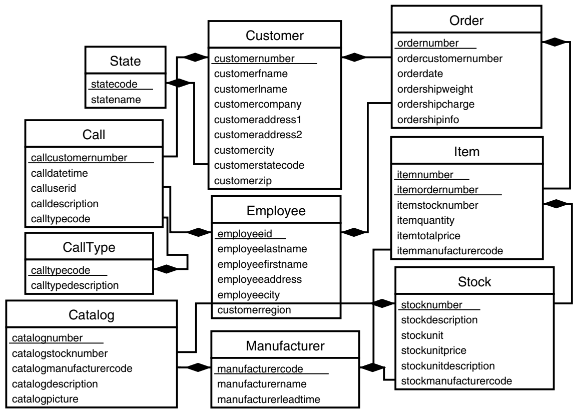

Figure 1. The database schema of Stores7**.**

Although Web-PDDL was originally designed as a language for representing ontologies, their mappings, data instances, and queries on the Semantic Web, it also can be used to represent Stores7 as an ontology as shown in Figure 2.

```
(define (domain stores7-ont)
   (:extends
      (uri "http://www.cs.uoregon.edu/~paea/research/sql.pddl#"
         :prefix sql) ...)
   (:types
      Customer State - @sql:Relation
      String - @sql:Varchar ...)
   (:predicates
      (customerfname x - Customer y - @sql:varchar)
      (customerlname x - Customer y - @sql:varchar)
      (customerstatecode x - Customer y - @sql:varchar) ...)
   (:axioms
      (forall (c - Customer code - @sql:varchar)
         (if (customerstatecode c code)
            (exists (s - State) (statecode s code)))) ...)
   (:facts
      (@sql:primarykey Customer "customernumber") ...))
```

Figure 2. The Stores7 **schema as an ontology.**

Some of the highlights of the Web-PDDL syntax include:

- *Inheritance*: The `:extends` declaration in Figure 2 expresses that Stores7 inherits features from the "sql" domain, which defines concepts for relational databases such as relations, data types, and aggregate functions. Therefore, the inheritance capability of Web-PDDL allows us to incorporate some of the more common and desirable features of SQL.
- *Namespaces*: To avoid symbol clashes, symbols imported from other ontologies are given prefixes, such as `@sql:Relation`. These correspond to XML namespaces, and when Web-PDDL is translated into RDF, that is exactly what they become [19].
- *Types*: Types (also known as "classes") start with capital letters. A type T_1 is declared to be a subtype of a type T_0 by writing "T_1 - T_0" in the `:types` field of a domain definition. In other contexts, the hyphen notation is used to declare a constant or variable to be of a type T, by writing "x - T."
- *Predicates*: Predicates correspond roughly to properties in the Web Ontology Language (OWL [2], which is based on Description Logics), but they can take any number of arguments.
- *Axioms*: The axiom in Figure 2 is the Web-PDDL expression for the foreign key relationship between Customer and State, where customerstatecode is the foreign key.
- *Functions*: There are also functions in Web-PDDL, including Skolem functions and built-in functions such as + and -, that can be evaluated when appropriate.

- *Facts*: Assertions (facts) are written in the usual Lisp style: e.g. (@sql:primarykey Customer "customernumber") states that the primary key attribute of the Customer relation is "customernumber."

In our initial findings, a few simple rules relating schemas to ontologies can accomplish the majority of transformations. These rules also play an important part in developing the SQL wrappers (PDDSQL) for OntoGrate architecture:

Relation	$\leftrightarrow$	Type
Attribute	$\leftrightarrow$	Predicate
Integrity Constraint	$\leftrightarrow$	Axiom
Primary Key	$\leftrightarrow$	Fact

Although these simple rules appear to work well for rudimentary transformation, we believe further work is required to capture more subtle database semantics.

2.2 Merging Ontologies with Bridging Axioms

A merged ontology consists of common elements from a source and target ontology, but also defines the semantic mappings between them as *bridging axioms* [12]. A merged ontology allows all the relevant symbols in a domain to interact with each other, so that facts can be translated from one ontology to another using inference over the bridging axioms.

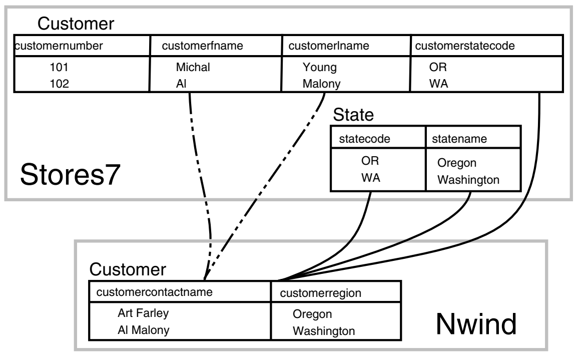

Figure 3. Portions of two database schemas and the mappings between them.

Suppose we have a different sales schema, such as Nwind from Microsoft. Although Stores7 and Nwind are similar schemas, they do have some differences. We can write bridging axioms in Web-PDDL to express mappings between Stores7 and Nwind. Consider, for example, the mappings depicted in Figure 3. The 3-way mapping from Nwind's *region* to Stores7's *statename* and *statecode* information, can be expressed in Web-PDDL as:

```
(forall (x - @nwind:Customer y - @sql:varchar)
 (if (@nwind:customerregion x y)
 (exists (z - @stores7:State t - @sql:varchar)
     (and (@stores7:customerstatecode x t)
          (@stores7:statename z y)
          (@stores7:statecode z t)))))
```

We can finally put all bridging axioms into the new, merged ontology, called Stores7-Nwind.

2.3 Inferential Data Integration

Given a merged ontology between two sources expressed in the first order ontology language, Web-PDDL, we can then perform integration using first order theory. In this section, we specify the problem of integration and show how inference can solve it, forming the basis of our *inferential data integration* model.

- *Query Translation*: The process of extracting data expressed by one schema to answer a query posed using another schema, also known as *query answering*.
- *Data Translation*: Translating data from a source schema to a target (or integrated) schema for the purpose of information exchange.

The problem of query answering has been the focus of most recent work, but data translation is also important [9]. We argued that both of these problems can be addressed seamlessly by using sound inference in [11]. We repeat that argument here for the sake of completeness.

Suppose we have already used bridging axioms to describe the mappings between two schemas, Schema S from DB_s and Schema T from DB_t (such as the 3-way mapping for Stores7 and Nwind). Let the set of bridging axioms be denoted M_{s_t}. Let the symbol $\rightsquigarrow_D$ indicate data translation between two schemas (such as S and T) so that: $\alpha_s \rightsquigarrow_D \beta_t$ means β_t is the translation of α_s, where α_s and β_t are assertions (facts) corresponding to data instances in DB_s and DB_t. For example, the following is an assertion from Stores7 and its equivalent expressed in Web-PDDL:

The statecode of a Customer identified as 101 is OR.
(customerstatecode Customer#101, "OR")

For a set of assertions (or "dataset"), we stipulate that the translation of α_s is simply the largest set of assertions, β_t, entailed by α_s through the mapping rules M_{s_t}. A consequence of this stipulation is that

$$(M_{s\_t}; \alpha_s) \rightsquigarrow_D \beta_t \textit{ only if } (M_{s\_t}; \alpha_s) \vDash {}^1\beta_t$$

[1]The logic symbol, $\vDash$, can be read as "entails."

To guarantee this requires sound inference. In other words, "$\leadsto_D$" entails soundness, so we can use $\vdash$[2] to represent data translation with our algorithm:

$$(M_{s\_t}; \alpha_s) \leadsto_D \beta_t \;\Leftrightarrow\; (M_{s\_t}; \alpha_s) \vdash \beta_t \;\Rightarrow\; (M_{s\_t}; \alpha_s) \vDash \beta_t$$

This definition means that β_t can be derived from the mappings M_{s_t} and assertions α_s using inference.

One may assume that translating a dataset means finding an *equivalent* dataset in a different vocabulary. As justification of our approach, we point out that our standard has been taken for granted in the case of another main problem in data integration: query translation, the process of extracting data expressed using one schema to answer a query in another schema. We will use the symbol $\leadsto_Q$ to indicate the query translation. If α_s is a query in Schema S, its translation is a query β_t in Schema T such that any answer (set of bindings) to β_t is also an answer to α_s. In other words:

$$(M_{s\_t}; \alpha_s) \leadsto_Q \beta_t \textit{ only if } (M_{s\_t}; \theta(\beta_t)) \vDash \theta(\alpha_s)$$

for any substitution θ, where $\theta(\beta_t)$ is from the target database DB_t. It also means, for any substitution θ,

$$\begin{aligned}(M_{s\_t}; \alpha_s) \leadsto_Q \beta_t \;&\Leftrightarrow\; (M_{s\_t}; \theta(\beta_t)) \vdash \theta(\alpha_s)\\ &\Rightarrow\; (M_{s\_t}; \theta(\beta_t)) \vDash \theta(\alpha_s)\end{aligned}$$

We claim that β_t is the translation of α_s if and only if, for every substitution θ, $\theta(\beta_t)$ is the *weakest* statement in Schema T such that $\theta(\beta_t)$ is from DB_t, $(M_{s\_t}; \theta(\beta_t)) \vdash \theta(\alpha_s)$ and $M_{s\_t} \nvdash \theta(\alpha_s)$. The weakest statement means that β_t need not be (and seldom is) *equivalent* to α_s, in the sense that any answer to one is an answer to the other. All we need is that any answer to β_t be an answer to α_s. In the literature this is also known as *query containment* [17].

Since both data translation and query translation can be defined as an inference, we can call our data integration approach *inferential data integration*.

Summary of previous results

Our previous work either focused on the ontology translation for Semantic Web documents [12] or focused on the integration of relational databases themselves [11]. However, our new approach tries to integrate databases and Semantic Web resources together. In our previous work, we have built a first order inference engine, OntoEngine, to first evaluate our approach on several real ontology translation tasks for the Semantic Web documents. Some of them need OntoEngine to process large sets of data. The results of our experiments show that the translation works efficiently. For example, OntoEngine can translate 21,164 facts (about 3,010 individuals and 1,422 families from European royalty in a daml file[3]) from one genealogy ontology to 26,956 facts in another genealogy ontology. This takes less than 1 minute on a typical PC (the details were reported in [12]). Recently, we also tested OntoEngine on query translation and answering in an early version of OntoGrate framework [11] for integrating relational databases. The system can reformulate conjunctive queries and retrieves over 100,000 answers from a target database in under 30 seconds. In addition to query answering, the system can translate 40,000 database facts from source to target in under 30 seconds.

[2]The logic symbol, $\vdash$, can be read as "infers."

[3]http://www.daml.org/2001/01/gedcom/royal92.daml

3 OntoGrate Architecture

We are developing OntoGrate, a system that, given different but related ontologies or schemas and their associated data, will be able to learn or mine a set of first order mapping rules that accurately describes how the input ontologies or schemas relate to each other. These rules will be used by OntoEngine to perform data integration. For example, different genomic databases, which are used to study NIH model organisms such as the zebrafish, mouse and fruit fly, and different genomic ontologies (e.g., the Gene Ontology [22] in OWL) could benefit from integration when used to answer more interesting questions geneticists may have about the human genome.

To explain how OntoGrate can help domain experts (e.g., geneticists) integrate their databases and Web resources, we explain the architecture of OntoGrate, shown in Figure 4. It is composed of six modules and interfaces with four different kinds of data resources related to our specific aims.

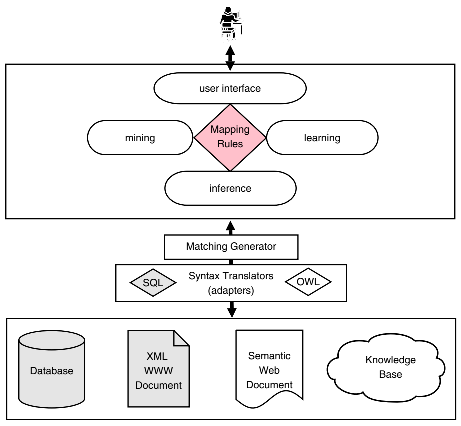

Figure 4. Architecture of OntoGrate The system is composed mainly of six modules: the *inference module*, the *learning module*, the *mining module*, the *matching generation* module, the *syntax translators*, and the *user interface module*. The input can consist of four different kinds of data resources, along with their ontologies (or schemas).

In between OntoGrate and the data resources exist some syntax translators (adapters). For example, we have built an automatic translator between Web-PDDL and OWL (PDDOWL) and an automatic translator between Web-PDDL and SQL (PDDSQL). When all ontologies, schemas, and data are represented in Web-PDDL, the matching generation module will produce correspondences as suggestions to present to the users. We will briefly introduce our aims and new ideas as follows.

Integration of schemas and ontologies: Since databases are defined by schemas, which focus more on structure than semantics, we first need to build an automatic translator to represent schemas as ontologies. We have discovered some basic heuristics for transforming schemas into ontologies. These heuristics can be embedded into an automatic tool to perform most schema transformations, and it has worked well so far in our preliminary database experiments [11]. While it appears that this task can be automated, we anticipate some theoretical challenges, especially given some application-oriented database schema designs with complex constraints, such as biomedical databases. User interaction may be required to capture subtle semantics.

Matching (correspondence) generation: To derive candidate matchings for users, we are building a matching generator based on the names, structure, and relationships of concepts in ontologies (schemas). A user interface will depict the ontologies and candidate matchings based on the input ontologies and data. The system will then enter into an interactive loop with the learning and mining modules, during which the user will make further refinements.

Learning mappings from the knowledge of domain experts: Based on the matching suggestions, domain experts may be able to specify simple relationships, such as "equal," "subclass" or "subproperty." However, more complex relationships may be too subtle for a domain expert to specify accurately at first. We expect machine learning to be helpful in this case: all a user has to do is to provide specific examples and let the system generalize them into formal rules expressed in first order logic. In general, we expect that this learning module will benefit from the extensive literature that deals with learning in first order logic (Inductive Logic Programming) [21, 4]. Well-suited learning algorithms will have to be developed, depending on (i) the amount of information available, (ii) the quality of user input, (iii) the complexity of mapping rules needed, and (iv) the complexity of the input ontologies.

Mining large data sets to find candidate mappings: We are using association rule mining in the OntoGrate system to discover candidate mappings, which the user can then select or refine to generate final executable mapping rules. Association rule mining [3] is a data mining technique that has been successfully used to find frequent patterns in large data sets. For instance, this technique might help discover patterns in different databases that share some partial data, such as different gene databases that share the same GenBank [1] ID. Similarly, medical databases use term IDs to point to medical terms in UMLS [18].

User Interface: Interaction between the system and domain experts through a user interface is important to realizing our vision of an *interactive* integration system. Our goal is to have the user interface module present information about input ontologies or data resources, the current candidate mappings generated by data mining and machine learning, and any intermediate and final data translation or query answering results in a clean, concise, and accessible fashion. Also, the user interface will be designed to facilitate user feedback, such as having the user select nodes in the ontologies or facts in the databases, choose mapping rules mined by the data mining module or learned by the learning module to refine or verify them, or select suggestions from the matching generation module to further explore and incorporate.

Inference Engine: The inference module is in charge of using mapping rules to answer queries and exchange (i.e., translate) data among available data sources. It will also be used for rule refinement, that is, to check the consistency and redundancy of the generated rule set. OntoEngine is a special purpose first order theorem prover developed for ontology translation on the Semantic Web [12]. We are extending it for the new purposes mentioned in this paper. In particular, the output from the inference engine can be fed back into the learning and mining processes for better results.

4 Case Study

Although OntoGrate is an ongoing research project, we have applied this framework to integrate relational databases and Semantic Web resources. In this section, we will elaborate on our approach in two different scenarios.

4.1 Integrating DBs into the Semantic Web without an existing domain ontology

After we get the Stores7 ontology in Web-PDDL (as in Figure 2), we can call our syntax wrapper PDDOWL to transform it into OWL syntax like:

```
...
<owl:Class rdf:ID="Customer">
 <rdfs:subClassOf
   rdf:resource="...sql#Relation"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="customercity">
 <rdfs:domain rdf:resource="#Customer"/>
 <rdfs:range rdf:resource="...#String"/>
</owl:DatatypeProperty>
...
```

Besides classes and property definitions, PDDOWL can also translate axioms (rules) into RDF syntax using our approach in [19]. We can put the Stores7 OWL ontology on the web.

Suppose some Semantic Web agents or services wish to access (e.g., query) the data in the Stores7 database using the new Stores7 OWL ontology. One way to do this is to use the Stores7 OWL ontology to mark up (annotate) data tuples from the database into an OWL document, but it is obvious that the annotation of all data tuples is not efficient and will create much redundancy on the Semantic Web. It would be better to let web agents or services access relational databases directly.

However, web agents or services may want to use some Semantic Web query languages to send queries to the Stores7 database, after they locate the new Stores7 OWL ontology. OWL-QL[4] is a promising Semantic Web query language and it may become W3C standard in the future. If that, we just need to provide functions in PDDOWL to translate OWL-QL queries into Web-PDDL queries. Then we call the PDDSQL translator to translate Web-PDDL queries into SQL queries and get relational answers (i.e., tables) directly from the Stores7 database. PDDSQL can again translate those answers to bindings in Web-PDDL. Finally, the bindings can be filled in to provide answer bundles in OWL-QL by PDDOWL. The process is illustrated in Figure 5.

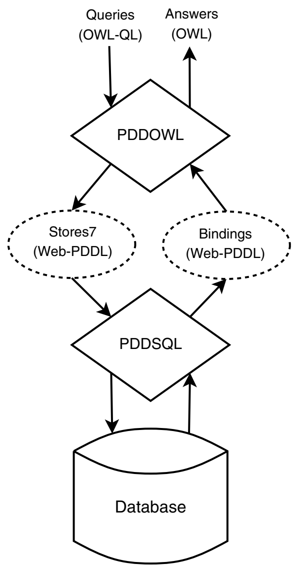

Figure 5. Integrating databases into the Semantic Web without an existing domain ontology.

[4]http://ksl.stanford.edu/projects/owl-ql/

For example, a simple query one might pose to the system: “What are the names of customers in the Stores7 database living in the city of Eugene?” Suppose the query is using the Stores7 OWL ontology.

```
<owl-ql:query
      xmlns:owl-ql=".../owl-ql-syntax#"...>
 <owl-ql:premise>
   <rdf:RDF>
     <rdf:Description rdf:about="#C">
       <rdf:type rdf:resource="#Customer"/>
       <customercity rdf:resource="#Eugene"/>
     </rdf:Description>
   </rdf:RDF>
 </owl-ql:premise>
 <owl-ql:queryPattern>
   <rdf:RDF>
     <rdf:Description rdf:about="#C">
       <customerfname rdf:resource="...#x"/>
       <customerlname rdf:resource="...#y"/>
     </rdf:Description>
   </rdf:RDF>
 </owl-ql:queryPattern>
 <owl-ql:answerKBPattern>
  <owl-ql:kbRef rdf:resource="...stores7.owl"/>
  ...
```

PDDOWL translates the OWL-QL query into a query in Web-PDDL, described by the Stores7 ontology:

```
 (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 query in SQL:

```
 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 by PDDSQL:

```
{?x/Paea, ?y/LePendu}
{?x/Dejing, ?y/Dou}
{?x/Shiwoong, ?y/Kim}
...
```

The Web-PDDL bindings are used to generate an OWL-QL answer bundle by PDDOWL:

```
 <owl-ql:answerBundle
    xmlns:owl-ql="...owl-ql-syntax#" ...>
  <owl-ql:answer>
   <owl-ql:binding-set>
     <var:x rdf:resource="#Paea"/>
     <var:y rdf:resource="#LePendu"/>
   </owl-ql:binding-set>
   <owl-ql:answerPatternInstance>
```

```
    <rdf:RDF>
     <rdf:Description rdf:about="#C">
      <customerfname rdf:resource="#Paea"/>
      <customerlname rdf:resource="#LePendu"/>
     </rdf:Description>
    </rdf:RDF>
   <owl-ql:answerPatternInstance>
  </owl-ql:answer>
  <owl-ql:answer>
   <owl-ql:binding-set>
     <var:x rdf:resource="#Dejing"/>
     <var:y rdf:resource="#Dou"/>
   </owl-ql:binding-set>
  ...
```

Testing results in this scenario

We have put more than 100,000 data records in a Stores7 database running in a local PC.[5] Both the translation of the query from OWL-QL to Web-PDDL using PDDOWL and the translation from Web-PDDL to SQL using PDDLSQL are less than 1 second. The speed of getting an answer table from the Stores7 database by MySQL database engine depends on the size of table, for example, 1000 data records (e.g., 1000 customers are living in Eugene) takes 3 seconds. Next, PDDSQL takes less than 1 second to transform the table with 1000 facts into Web-PDDL bindings. Finally, PDDOWL can translate those 1000 Web-PDDL bindings to OWL answers in 3 seconds.

4.2 Integrating DBs into the Semantic Web with an existing domain ontology

It is possible for an existing ontology to be applied by some Semantic Web applications within the same domain. For example, for the domain of sales and orders, it was easy for us to find that there is an Order ontology written in OWL and published on the Web[6]. Suppose that some Semantic Web applications have already used this ontology. Once the Stores7 schema has been represented in Web-PDDL, we can also run PDDOWL to transform the Order ontology into Web-PDDL syntax. These two ontologies can then be merged using bridging axioms. Unlike the process of transforming schemas to ontologies, defining semantic relationships between concepts is often too subtle for full automation. Some tools do exist [24, 10, 26, 8, 30] that facilitate the process, but human interaction is invariably required. In OntoGrate, although we are developing both machine learning and data mining tools to aid this process, it still can not be fully automated and it needs human involvement. Based on our experience, an ontology representation can help us to define bridging axioms manually. For example, the following are some bridging axioms between Stores7 and Order:

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

[6]http://www.dayf.de/2004/owl/order.owl

```
(T-> @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 (P - @order:Person A - @order:Address
                                      z - String)
    (if (and (@order:hasAddress P A)
             (@order:City A z))
        (@stores7:customercity P z)))

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

where order is the prefix of Order ontology.

After we get the merged ontology of Stores7 and Order, we can use OntoGrate to integrate the Stores7 database with other Semantic Web applications that use the Order ontology. The process is shown in Figure 6:

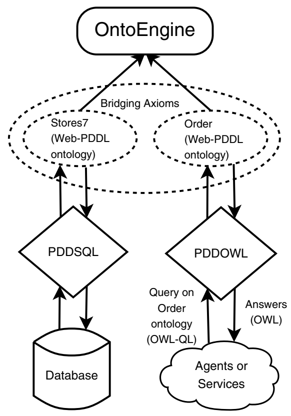
OntoEngine
Bridging Axioms
Stores7 (Web-PDDL ontology)
Order (Web-PDDL ontology)
PDDSQL
PDDOWL
Query on Order ontology (OWL-QL)
Answers (OWL)
Database
Agents or Services

Figure 6. Integrating databases into the Semantic Web with an existing domain ontology.

If a Semantic Web agent has an OWL-QL query described by the Order ontology, PDDOWL first translates it into a Web-PDDL query. OntoEngine can then do backward chaining to translate this query into a query in Stores7 with the aforementioned bridging axioms. Afterwards, PDDSQL can translate the query into a SQL query and get the relational data from the Stores7 database. The answers (facts) can be translated back into the Order ontology by OntoEngine, and then finally get marked up in OWL-QL.

We will use a similar query as the one from Section 4.1 to illustrate the process. The only difference is that the OWL-QL query is described by the Order ontology:

```
<owl-ql:query
   xmlns:owl-ql=".../owl-ql-syntax#"...>

<owl-ql:premise>
  <rdf:RDF>
    <rdf:Description rdf:about="#C">
      <rdf:type rdf:resource="#Person"/>
      <hasAddress rdf:resource="#A"/>
    </rdf:Description>
    <rdf:Description rdf:about="#A">
       <rdf:type rdf:resource="#Address"/>
         <City rdf:resource="#Eugene"/>
       </rdf:Description>
    </rdf:Description>
  </rdf:RDF>
</owl-ql:premise>
<owl-ql:queryPattern>
    <rdf:RDF>
      <rdf:Description rdf:about="#C">
        <FirstName rdf:resource="...#x"/>
        <LastName rdf:resource="...#y"/>
      </rdf:Description>
    </rdf:RDF>
</owl-ql:queryPattern>
<owl-ql:answerKBPattern>
 <owl-ql:kbRef rdf:resource="...order.owl"/>
   ...
```

PDDOWL translates this into a Web-PDDL query, 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 translate this query into a Web-PDDL query that uses the Stores7 ontology:

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

The rest of the process is the same as that shown in Section 4.1. However, the OWL-QL answer bundle returned uses the Order ontology, not the Stores7 ontology:

```
<owl-ql:answerBundle
  xmlns:owl-ql="...owl-ql-syntax#"...>

<owl-ql:answer>
 <owl-ql:binding-set>
   <var:x rdf:resource="#Paea"/>
   <var:y rdf:resource="#LePendu"/>
 </owl-ql:binding-set>
   <owl-ql:answerPatternInstance>
     <rdf:RDF>
       <rdf:Description rdf:about="#C">
         <FirstName rdf:resource="#Paea"/>
         <LastName rdf:resource="#LePendu"/>
       </rdf:Description>
     </rdf:RDF>
   <owl-ql:answerPatternInstance>
 </owl-ql:answer>
 ...
```

Testing results in this scenario

The testing results are similar to those in the last scenario. The only overhead added to this scenario is that OntoEngine needs to translate the query in the Order ontology into a query in the Stores7 ontology. The speed of translation is determined by the complexity of queries. We have tested the queries with 4 joined tables. The query translation takes less than 1 second. We also tested translating the data facts (in Web-PDDL) from Stores7 ontology to Order ontology (the speed is similar as we reported in [12]: 10,000 facts in 30 seconds). After the facts are semantically translated into the Order ontology, PDDOWL can mark up 10,000 facts in OWL syntax with the Order ontology in 11 seconds.

5 Related Work

The primary goal of this paper is to introduce the OntoGrate framework and our approach to use OntoGrate to integrate relational databases into the Semantic Web in two different scenarios. The related approaches can be found in several disciplines:

Semantic Annotation for Schemas: There are very few approaches investigating the transformation of relational schemas into ontologies. The most similar approach to ours is [28], in which a relational model is mapped to frame logic, which can then be represented in RDF. The two approaches share the same process of semantic annotation. However, our focus is not just on the representation of, but more importantly on the integration of, relational databases and Semantic Web applications. Another approach described in the DOGMA ontology framework also talked about how to translate a query from an ontology language to SQL [29], but it did not mention anything about the translation problem between different ontologies or schemas.

Schema and ontology mapping: Although the focus of this paper is not on finding mappings, automatic or semi-automatic schema (ontology) matching and mapping tools [24, 10, 26, 8] can be helpful in providing suggestions to a domain expert who uses the OntoGrate system.

Clio [30, 14] is a semi-automatic tool that can generate mappings in SQL and XQuery. COMA [13] combines schemas with a reference ontology and uses composition to build mappings. Our approach is to define schema and ontology mappings as first order logic axioms (bridging axioms).

Data integration and query answering: General integration models, such as federated databases [27], data warehouses [6], and peer-to-peer data management [16], exist. The integration process is always implemented with a data mediator and an integrated schema represented as view definitions. The MiniCon algorithm [23] 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. Our approach uses an inference engine and syntax translators between SQL, OWL, and first order logic (Web-PDDL) to translate data and answer queries. Although Datalog systems also use inference to process the queries and data, they are not designed for data integration.

Logic and inference: Approaches based on a declarative model (as opposed to a procedural one) often use a logical framework from the area of knowledge representation and reasoning. Raymond Reiter's reconstructions of the relational data model in first order logic [25] is an early example of such approaches. The Carnot, SIMS, and Information Manifold systems are brilliantly summarized and compared in [7]. While very similar to our approach, these approaches tend to be more constricted, depending on fixed global ontologies such as CYC or LOOMS, or a less expressive logic such as Description Logic or Datalog.

6 Conclusion and Future Work

We have applied an ontology-based information integration framework to make relational databases accessible to emerging Semantic Web applications. After defining different schemas as ontologies, we can define mapping rules as bridging axioms and merge the schemas (ontologies) together. The data integration can then be implemented as an inference process by our special purpose theorem prover, OntoEngine, with the help of syntax wrappers (i.e., PDDSQL and PDDOWL). Our new OntoGrate framework not only has the advantage of the rich expressiveness of first order logic for conceptual modeling, but also exploits both the desirable features of SQL and OWL. We elaborate on our two approaches to query sales databases from the Semantic Web applications in two scenarios.

Our testing results so far demonstrate that OntoGrate is promising for integrating databases into the Semantic Web. In the immediate future, besides implementing other research aims described in section 3, the scalability and efficiency of OntoGrate needs to be investigated in larger-sized relational databases of various domains. Also, another interesting integration task is to integrate current XML documents in the world-wide web with the Semantic Web in the OntoGrate framework. We believe this will be more difficult than integrating relational databases into the Semantic Web, since not all XML documents have XML schemas (which we can treat similarly as relational schemas) or DTD definitions. We plan to tackle these challenging problems in our immediate future work while making progress on the other important aspects of our system outlined in section 3.

References

[1] GenBank Database. `http://www.ncbi.nlm.nih.gov/Genbank/`.

[2] OWL Web Ontology Language. `http://www.w3.org/TR/owl-ref/`.

[3] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Very Large Data Bases (VLDB) Conference*, pages 487–499. Morgan Kaufmann, 1994.

[4] M. Arias and R. Khardon. Complexity Parameters for First Order Structures. *To appear in the Machine Learning Journal*, 2005.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5), May 2001.

[6] P. A. Bernstein and E. Rahm. Data warehouse scenarios for model management. In *ER 2000*, pages 1–15, 2000.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Knowledge representation approach to information integration. In *Proc. of AAAI Workshop on AI and Information Integration*, pages 58–65. AAAI Press/The MIT Press, 1998.

[8] R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos. iMAP: Discovering Complex Mappings between Database Schemas. In *Proceedings of the ACM Conference on Management of Data*, pages 383–394, 2004.

[9] A. Doan and A. Y. Halevy. Semantic-integration research in the database community: a brief survey. *AI Magazine*, 26(1):83–94, 2005.

[10] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to Map Between Ontologies on the Semantic Web. In *International World Wide Web Conferences (WWW)*, pages 662–673, 2002.

[11] D. Dou and P. LePendu. Ontology-based Data Integration for Relational Databases. 2006. To appear in *ACM SAC'06 DTTA Track*.

[12] D. Dou, D. V. McDermott, and P. Qi. Ontology Translation on the Semantic Web. *Journal of Data Semantics*, 2:35–57, 2005.

[13] E. Dragut and R. Lawrence. Composing mappings between schemas using a reference ontology. In *Proceedings of International Conference on Ontologies, Databases and Application of SEmantics (ODBASE)*, 2004.

[14] L. M. Haas, M. A. Hernandez, H. Ho, L. Popa, and M. Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 805–810, 2005.

[15] A. Y. Halevy, N. Ashish, D. Bitton, M. J. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka. Enterprise Information Integration: Successes, Challenges and Controversies. In *Proceedings of the ACM Conference on Management of Data*, pages 778–787, 2005.

[16] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *International Conference on Data Engineering*, pages 505–516, 2003.

[17] M. Lenzerini. Data Integration: A Theoretical Perspective. In *Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[18] D. Lindberg, B. Humphries, and A. McCray. The Unified Medical Language System. *Methods of Information in Medicine*, 32(4):281–291, 1993.

[19] D. V. McDermott and D. Dou. Representing Disjunction and Quantifiers in RDF. In *International Semantic Web Conference*, pages 250–263, 2002.

[20] S. A. McIlraith and D. L. Martin. Bringing Semantics to Web Services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.

[21] S. Nienhuys-Cheng and R. De Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag, 1997. LNAI 1228.

[22] G. Ontology Consortium. Creating the Gene Ontology Resource: Design and Implementation. *Genome Research*, 11(8):1425–1433, 2001.

[23] R. Pottinger and A. Levy. A Scalable Algorithm for Answering Queries Using Views. In *Proceedings of the 26th VLDB Conference*, pages 484–495, 2000.

[24] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *Very Large Data Bases (VLDB) Conference*, 10(4):334–350, 2001.

[25] R. Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Topics in Information Systems, pages 191–233. Springer, 1984.

[26] L. Serafini, P. Bouquet, B. Magnini, and S. Zanobini. An algorithm for matching contextualized schemas via sat. In *Proceedings of CONTEXT'03*, 2003.

[27] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[28] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 1100–1107, New York, NY, USA, 2002. ACM Press.

[29] P. Verheyden, J. D. Bo, and R. Meersman. Semantically unlocking database content through ontology-based mediation. In *SWDB*, pages 109–126, 2004.

[30] L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-Driven Understanding and Refinement of Schema Mappings. In *Proceedings of the ACM Conference on Management of Data*, 2001.