Use of OWL and SWRL for Semantic Relational Database Translation

Matthew Fisher, Mike Dean, Greg Joiner

BBN Technologies, 1300 N. 17th Street, Suite 400, Arlington, VA 22209
{mfisher, mdean, gjoiner}@bbn.com

Abstract. General purpose query interfaces to relational databases can expose vast amounts of content to the Semantic Web. In this paper, we discuss Automapper, a tool that automatically generates data source and mapping ontologies using OWL and SWRL. We also describe the use of these ontologies in our Semantic Distributed Query architecture, an implementation for mapping RDF queries to disparate data sources, including SQL-compliant databases, using SPARQL as the query language. This paper covers Automapper functionality that exploits some of the expressiveness of OWL to produce more accurate translations. A comparison with related work on Semantic Web access to relational databases is also provided as well as an investigation into the use of OWL 1.1.

Keywords: Semantic, Database, Mapping, OWL, SWRL

1 Introduction

A wealth of information resides in relational databases, which are highly engineered for scalability, transaction management, security and access by existing applications [1]. Access to this information significantly increases the utility of the Semantic Web. Dynamic access is preferred, to accommodate high data volumes and change rates. Custom servlets or other programs can export high-quality semantic representations, but the development cost is often prohibitive and can limit reusability. This motivates the development of application-independent tools that can generate a basic ontology from a database schema and dynamically produce instance data using that ontology. This method quickly exposes the data to the Semantic Web, where a variety of tools and applications are available to support translation, integration, reasoning, and visualization [2].

The paper presents one such tool, Automapper, and is organized as follows: Sections 2 and 3 describe both Automapper and the overall architecture in which Automapper operates. Section 4 discusses the current use of OWL and Section 5 details the current use of SWRL. Section 6 provides a simple application of Automapper and the Semantic Distributed Query architecture. Section 7 covers related work. Section 8 explores how the new features in OWL 1.1 can be used to enhance Automapper. Finally, Section 9 concludes with future directions.

2 Architecture

To understand Automapper's utility, a description of its subsuming architecture is

provided. While that is not the focus of this paper, a general understanding is necessary to appreciate Automapper's role in the system as a whole. As shown in Figure 1, Automapper is part of a larger system for decomposing a SPARQL query, expressed using a domain ontology, over multiple data sources and merging the corresponding data into a single result set [3]. Specifically, Automapper uses the database schema to create an OWL data source ontology and mapping instance data (the mapping ontology is discussed in Section 3) to support two layers of processing: the higher-level Semantic Query Decomposition component (SQD) and the lower level SPARQL-to-SQL translation component, also known as a Semantic Bridge for Relational Databases (SBRD). Each RDBMS has its own Semantic Bridge instantiation that can be either collocated or hosted remotely. SQD relies on a set of SWRL data source-

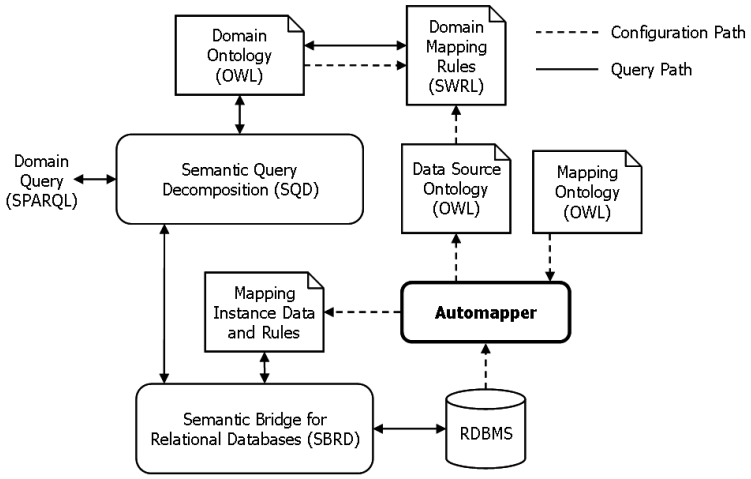

Fig. 1. Overview of the Semantic Distributed Query Architecture

to-domain mapping rules and optional domain-to-domain mapping rules to properly translate inbound queries into data source queries and vice-versa. The use of a domain ontology allows queries to be independent of any particular data source. In addition, different communities can each adopt their own domain ontology while reusing the same data sources. Mapping rules need only cover the data of interest thereby minimizing integration costs.

SBRD uses both Automapper artifacts to correctly map a SPARQL query expressed using the data source ontology into SQL SELECT statements. Database query result sets are mapped back into the data source ontology and returned to SQD. As a final step, SQD recombines the various result sets, maps the data into the domain ontology and returns this data to the user. Automapper was developed in Java and needs to be run only once against a given relational database to automatically generate both the data source ontology and the mapping data. Using the method *java.sql.Connection.getMetaData()*, we are able to mine the necessary table definitions such as column names, primary keys, foreign keys, remarks and type declarations to generate the data source ontology. Automapper constructs these artifacts based on a configuration file, which contains instance data based

on a simple configuration ontology. This ontology permits additional primary and foreign key information, the option to include or exclude comments in the mapping instance data, the capacity to limit the visibility of specific tables and the ability to override a declared column datatype with another XML Schema datatype [4].

3 Utilizing Automapper Mapping Data

While SQD relies on SWRL rules for mapping, SBRD depends on Automapper's generated mapping data. SBRD employs these mappings for transforming SPARQL data source queries into SQL queries for data retrieval. Our mapping ontology, based on the mapping ontology used by D2RQ [5], defines a *ClassMap* OWL class whose instances represent each table in a given schema. A table has a corresponding OWL class in the data source ontology, an owning schema, a name, a *uriPattern* and one or more datatype property bridges and object property bridges. The *uriPattern* is an OWL datatype property used to identify each row in a table (instance) with a unique URI by concatenating its table name with the value of each primary key column in the table. The datatype and object property bridges correspond to columns containing literal values and foreign keys, respectively.

4 Use of OWL

The following class descriptions, axioms and restrictions are currently generated by Automapper:

1. `maxCardinality` is set to 1 for all nullable columns
2. `cardinality` is set to 1 for all non-nullable columns
3. All datatype and object properties that represent columns are marked as `FunctionalProperties`. To ensure global uniqueness and class specificity, these properties are given URIs based on concatenating the table and column names
4. `allValuesFrom` restrictions reflect the datatype or class associated with each column

5 Use of SWRL

Automapper generates SWRL rules to identify identical individuals based on their primary key values. These rules use `swrl:SameIndividualAtom` statements to express class-specific inverse functional relationships, including those involving multiple properties (primary key columns), neither of which is directly supported by OWL [6]. The inclusion of these rules can reduce the number of SPARQL variables and statements used by both SQD during the query decomposition process and Semantic Bridges during translation. The resulting SQL queries are more concise and, in certain cases, execute in a shorter period of time.

6 Example

Assume a simple OWL domain ontology of personnel information used by a human resources department. This ontology defines classes *Person* and *Dept* and datatype properties *gender*, *name*, *code* and *projectName*[1]. It also defines an object property, *department*, used to associate a *Person* with a *Dept*. A *Dept* is uniquely identified by its code. Below is a fragment represented in Turtle:

```
:Person a owl:Class;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty :name ;
        owl:maxCardinality
         "1"^^xsd:nonNegativeInteger ],
      [ a owl:Restriction ;
        owl:onProperty :name ;
        owl:allValuesFrom xsd:string ],
      [ a owl:Restriction ;
        owl:onProperty :department ;
        owl:maxCardinality
        "1"^^xsd:nonNegativeInteger ],
      [ a owl:Restriction ;
        owl:onProperty :department ;
        owl:allValuesFrom :Dept ] .

:Dept a owl:Class;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty :code ;
        owl:maxCardinality
        "1"^^xsd:nonNegativeInteger ],
      [ a owl:Restriction ;
        owl:onProperty :code ;
        owl:allValuesFrom xsd:int ] .
```

A domain model will often incorporate data from multiple databases but not all terms defined in a data source ontology will map to the domain ontology. For the sake of brevity, we limit our example to a single database (hresources) that holds staffing information; Tables 1 and 2 list the contents of the Staffing and Departments tables, respectively. Note that the Department ID column is only partially dependent on the primary key (an employee belongs to one department, independent of a project) and therefore the table is not in second normal form [7]. Unfortunately, such usage is not uncommon in practice and is included here as a real-world example.

[1] The association between Person and a Project could also be appropriately modeled using an object property.

Table 1. Staffing Table. Name and Project together form a primary key and Department ID is a foreign key.

Name	Project	DeptID	Hours	Role
MattF	Alpha	1	100.5	Developer
MikeD	Alpha	2	50.2	Tech Lead
MattG	Beta	1	92.0	Architect
DaveK	Beta	1	120.0	Developer
MikeD	Beta	2	30.8	Consultant
DaveK	Alpha	1	87.8	Indagator

Table 2. Department Table. ID is a primary key.

ID	Name
1	System Solutions
2	Research and Development
3	Management

From this schema, Automapper creates the data source ontology and class-specific inverse functional rules, of which fragments are listed below:

```
dsont:Hresources.Departments a owl:Class ;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty
        dsont:hresources.departments.id ;
        owl:allValuesFrom xsd:decimal ],
      [ a owl:Restriction ;
        owl:onProperty
        dsont:hresources.departments.id ;
        owl:cardinality
        "1"^^xsd:nonNegativeInteger ] .

dsont:Hresources.Staffing a owl:Class ;
   rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty
        dsont:hresources.staffing.name ;
        owl:cardinality
        "1"^^xsd:nonNegativeInteger ],
      [ a owl:Restriction ;
        owl:onProperty
        dsont:hresources.staffing.name ;
        owl:allValuesFrom xsd:string ],
      [ a owl:Restriction ;
        owl:onProperty
        dsont:hresources.staffing.deptid.Object ;
        owl:cardinality
        "1"^^xsd:nonNegativeInteger ] .

dsont:Hresources.DeptsSame a ruleml:Imp ;
   ruleml:body
      ( [ a swrl:ClassAtom ;
```

```
            swrl:argument1 :A ;
            swrl:classPredicate
            dsont:Hresources.Departments ]
         [ a swrl:ClassAtom ;
            swrl:argument1 :B ;
            swrl:classPredicate
            dsont:Hresources.Departments ]
         [ a swrl:DatavaluedPropertyAtom ;
            swrl:argument1 :A ;
            swrl:argument2 :Var0 ;
            swrl:propertyPredicate
            dsont:hresources.departments.id ]
         [ a swrl:DatavaluedPropertyAtom ;
            swrl:argument1 :B ;
            swrl:argument2 :Var0 ;
            swrl:propertyPredicate
            dsont:hresources.departments.id ] ) ;
   ruleml:head
      ( [ a swrl:SameIndividualAtom ;
            swrl:argument1 :A ;
            swrl:argument2 :B ] ) .
```

The corresponding mapping data is also generated. Below are two datatype property bridges, an object property bridge (representing a foreign key) and a class map all relating to the Departments table:

```
:HRESOURCES.DEPARTMENTS.ID a
map:DatatypePropertyBridge;
   map:column "ID";
   map:datatype xsd:decimal;
   map:language "en";
   map:property
      dsont:hresources.departments.id .

:HRESOURCES.STAFFING.DEPTID.OBJ a
map:ObjectPropertyBridge;
   map:constraint
      [ a map:KeyConstraint;
        map:objectColumnOperand "ID";
        map:operator :EqualsOperator;
        map:subjectColumnOperand "DEPTID" ];
   map:objectClassMap
      dsont:Hresources.Departments;
   map:property
      dsont:hresources.staffing.deptid.Object .

:Hresources.Departments a map:ClassMap;
   map:datatypePropertyBridge
      :HRESOURCES.DEPARTMENTS.ID,
      :HRESOURCES.DEPARTMENTS.NAME;
   map:table "DEPARTMENTS";
```

```
   map:type dsont:Hresources.Departments;
   map:uriPattern
      "http://example.org/2007/08/ds-ont#
      Hresources.Departments@@ID@@" .
```

In the final step, we create SWRL rules to map between the data source and domain ontologies to enable SPARQL query decomposition and reconstitution. Below are sample rules used for mapping the data source *Departments* class to the domain ontology *Dept* class including the department ID:

```
:RuleDeptClass a ruleml:Imp ;
   ruleml:body
      ( [ a swrl:ClassAtom ;
          swrl:classPredicate
             dsont:Hresources.Departments ;
          swrl:argument1 :d ] ) ;
   ruleml:head
      ( [ a swrl:ClassAtom ;
           swrl:classPredicate domont:Dept ;
           swrl:argument1 :d ] ) .

:RuleDepartmentCode
   ruleml:body
      ( [ a swrl:ClassAtom;
          swrl:classPredicate
             dsont:Hresources.Departments ;
          swrl:argument1 :d ]
        [ a swrl:DataValuedPropertyAtom ;
          swrl:propertyPredicate
             dsont:hresources.departments.id ;
          swrl:argument1 :d ;
          swrl:argument2 :c ] ) ;
   ruleml:head
      ( [ a swrl:DataValuedPropertyAtom ;
           swrl:propertyPredicate domont:code ;
           swrl:argument1 :d;
           swrl:argument2 :c ] ) .
```

Running the following query for the names of all people in department 1 and their associated project names:

```
PREFIX : <http://example.org/2007/08/domain-ont#>
SELECT ?pName ?name
WHERE {
   [ a :Person ;
     :projectName ?pName ;
     :name ?name ;
     :department ?d ] .
    ?d :code 1 }
```

yields: `Alpha MattF, Alpha DaveK, Beta MattG, Beta DaveK`.

7 Related Work

Various tools have been developed to provide Semantic Web interfaces to relational databases, including D2RQ, Gnowsis [8], ISENS [9], Relational.OWL [10] and OntoGrate [11].

We initially used D2RQ and incorporated several modifications which we submitted to the D2RQ development team[2]. The changes involved query optimizations such as eliminating duplicate table prefixing, increased selectivity of property bridges (to limit the desired number of tables in a query), and SQL SELECT query partitioning for smaller queries. As SQD became more sophisticated, we determined that SBRD and other Semantic Bridges did not require the full power of D2RQ.

Automapper corresponds to D2RQ's `generate-mapping`[3] script. It continues the D2RQ use of instance data for SPARQL-to-SQL mappings, entity concepts (e.g. *ClassMap*, *uriPattern*) and separate datatype and object property bridges. Object property bridges more precisely model foreign key relationships, although our model is not the only possibility [12]. The `d2rq:join` property has been replaced with a `Constraint` class in the mapping ontology. A Constraint is modeled as a binary operation with a single operator and two operands. Further precision is captured in `KeyConstraint`, a subclass of `Constraint`, which limits the operator to equality. Thus, `KeyConstraint` conceptually combines `d2rq:refersToClassMap` and `d2rq:join`. `d2rq:AdditionalProperty` and `d2rq:additionalProperty` are not used since the data source ontology is a straight-forward model of the RDBMS schema. This simplified representation makes Automapper-generated artifacts easier to create and understand.

8 OWL 1.1

The continued progress of of the W3C OWL Working Group provides an exciting preview of the new features in OWL 1.1, many of which can be used to enhance the functionality and expressivity of Automapper. The submission [13] divided the OWL extensions into four broad categories: syntactic sugar, new Description Logic constructs, expanded datatype expressiveness, and metamodeling constructs. This section will focus on the extension categories that are relevant to the relational database space, detailing how each can be used to enhance Automapper, and then discuss the impact of the new DL-Lite sub-species [14].

8.1 New Description Logic Constructs

While many of the new description logic constructs are generally useful, only one can be automatically derived from relational database schemas, the `IrreflexiveObjectProperty`. Reflexive relationships involve the same instance as both the subject and object of the relation. Consider the example found above in Table 1. Suppose a new column was added, “Manager”, that was a foreign key to the “Name” column. The referential integrity constraints of a

[2]http://sf.net/mailarchive/message.php?msg_id=1143734217.12135.8.camel@localhost
[3]http://sites.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/spec/#commandline

relational database system would guarantee that someone's manager must already be defined in the same table. Often, an additional constraint would be placed on this foreign key to prevent someone from being their own manager. However, OWL 1.0 cannot express this concept with a simple object (or functional object) property. The `IrreflexiveObjectProperty` in OWL 1.1 provides the ability to state that any given instance cannot be related to itself. In many cases, the irreflexivity of a database relation can be determined in an automated fashion and therefore this would be a valuable enhancement to Automapper. However, in cases where the irreflexivity cannot be automatically deduced, manual intervention can augment the generated OWL.

It is worth noting that many of the concepts represented by the other new property constructs are used in relational database systems in functions and stored procedures. While, these database concepts cannot be readily mapped in an automated fashion, an analyst using Automapper with knowledge of the existing database could take advantage of the new property constructs after the automated process has completed.

8.2 Expanded Datatype Expressiveness

OWL 1.1 allows the definition of user-defined datatypes. Specifically, the working group's specification document [15] outlines three new capabilities with two of them being relevant: `dataOneOf`, the ability to restrict a datatype's values to an enumerated list; and `datatypeRestriction`, the ability to restrict a datatype's values to a range or pattern. These two new capabilities are very commonly used in relational database system and would be a very beneficial enhancement to Automapper by enforcement through the configuration settings.

8.3 New OWL-DL Sub-Species

While not directly affecting Automapper, as an application in the relational database OWL space, it is worthwhile to briefly call out DL-Lite which is specifically designed to provide the minimum expressivity required to meet the needs of modeling a relational database system. DL-Lite provides several performance gains over complete OWL DL; most notably it reduces *data complexity*, the complexity measured with respect to the number of facts in the ontology, from an NP-Hard problem to a LOGSPACE problem. Thus, Automapper should produce OWL that is restricted to DL-Lite.

9 Future Work

We are currently applying Automapper's approach to other Semantic Bridges. Specifically, we are exploring its use for both SOAP and RESTful services in our Semantic Bridge for Web Services (SBWS).

Currently, URIs returned by SBRD are unique but generally not resolvable. We intend to address this issue in future versions by generating resolvable URIs and incorporating the best practices of the Linking Open Data initiative [16].

To the best of our knowledge, we believe that our rules and their usage are

consistent with the design goals of the DL Safe SWRL Rules task force[4]. Decidability is a critical aspect of our architecture and is therefore focused on features such as the use of Horn rules with unary and binary predicates. We will continue to monitor the task force's progress and incorporate necessary modifications. The advantages of SWRL built-ins have also proven essential. It is our hope that they are addressed in the DL Safe task force and will be comparable to the built-ins provided by SWRL.

References

1. C. Ritchie, *Relational Database Principles*. London, England: Int'l Thomson Business Press, pp. 5-52, 2002.
2. M. Dean, "Suggestions for Semantic Web Interfaces to Relational Databases," W3C Workshop on RDF Access to Relational Databases, March 2007. http://www.w3.org/2007/03/RdfRDB/papers/dean.html
3. D. Kolas, "Query Rewriting for Semantic Web Information Integration," Sixth International Workshop on Information Integration on the Web (IIWeb-07), *Twenty-Second AAAI Conference on Artificial Intelligence*, 2007.
4. P. Biron and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition," *W3C Recommendation 28 October 2004,* http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.
5. C. Bizer and A. Seaborne, "D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs," *Proc. Third International Semantic Web Conference*, Nov. 2004.
6. M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider and L. Stein, "OWL Web Ontology Language Reference," *W3C Recommendation 10 February 2004,* http://www.w3.org/TR/2004/REC-owl-ref-20040210/.
7. Codd, E.F. "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM 13*, pp. 377-387, June 1970.
8. L. Sauermann and S. Schwarz, "Gnowsis Adapter Framework: Treating Structured Data Sources as Virtual RDF Graphs," *Proc. Fourth International Semantic Web Conf.*, pp. 1016-1028, Nov. 2005.
9. D. Dimitrov, J. Heflin, A. Qasem, and N. Wang, "Information Integration via an End-to-End Distributed Semantic Web System," *Proc. Fifth International Semantic Web Conf.*, pp. 764-777, Nov. 2006.
10. C.P. de Laborda and S. Conrad, "Database to Semantic Web Mapping Using RDF Query Languages," *Lecture Notes in Computer Science: Conceptual Modeling – ER 2006*, pp. 241-254, 2006.
11. D. Dou, P. LePendu, S. Kim and P. Qi, "Integrating Databases into the Semantic Web through an Ontology-Based Framework," *Proc. Twenty-Second International Conf. on Data Engineering Workshops (ICDEW'06)*, pp. 54, 2006.
12. C.P. de Laborda and S. Conrad, "Relational.OWL: a Data and Schema Representation Format Based on OWL," *Second Asia-Pacific Conf. on Conceptual Modelling*, pp.89-96, 2005.
13. I. Horrocks,,and P.F. Patel-Schneider, OWL 1.1 Web Ontology Language Overview. W3C Submission 19 December 2006. http://www.w3.org/Submission/2006/SUBM-owl11-overview-20061219
14. B. Grau, OWL 1.1 Web Ontology Language Tractable Fragments. W3C Submission 19 December 2006. http://www.w3.org/Submission/2006/SUBM-owl11-tractable-20061219
15. B. Motik, I. Horrocks, and P.F. Patel-Schneider, OWL 1.1 Web Ontology Language Structural Specification and Functional-Style Syntax W3C Working Draft 8 January 2008. http://www.w3.org/TR/2008/WD-owl11-syntax-20080108
16. D. Berrueta and J. Phipps, Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Draft 23 January 2008. http://www.w3.org/TR/2008/WD-swbp-vocab-pub-20080123/

[4] http://code.google.com/p/owl1-1/wiki/SafeRules