Ultrawrap: Using SQL Views for RDB2RDF

Juan F. Sequeda
University of Texas at Austin
Department of Computer Sciences
Austin, Texas, USA
jsequeda@cs.utexas.edu

Rudy Depena
University of Texas at Austin
Department of Computer Sciences
Austin, Texas, USA
rdepena@mail.utexas.edu

Daniel P. Miranker
University of Texas at Austin
Department of Computer Sciences
Austin, Texas, USA
miranker@cs.utexas.edu

ABSTRACT

Ultrawrap is an automatic wrapping system that synthesizes an OWL ontology from the database's SQL schema and provides SPARQL query services for legacy relational databases. The system intentionally defines triples by using SQL view statements. The benefits of this organization include, the virtualization of the triple table assures real-time consistency between relational and semantic accesses to the database and the existing SQL optimizer implements the most challenging aspects of rewriting SPARQL to equivalent queries on the relational representation of the data. Initial experiments are auspicious.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems

General Terms

Management, Design, Experimentation.

Keywords

SQL Views, Relational Databases, OWL, RDF, RDB2RDF.

1. INTRODUCTION

Nearly 75% of the world's web sites are backed by SQL databases [1]. The proportion suggests that the success of the Semantic Web rests on methods for integrating those databases. The number of web sites imply that such methods be simple to deploy. In this paper we present the architecture of Ultrawrap, a relational database, semantic web wrapper system that speaks to three goals:

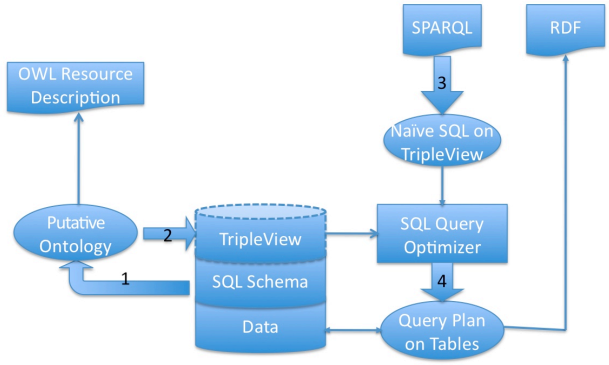
1. Fully automatic publication of legacy relational databases to the Semantic Web.
2. Assuring real-time consistency between the relational and RDF presentation of the data.
3. Making maximal use of existing SQL infrastructure

2. RDB2RDF

The literature abounds with proposed methods for integrating relational databases to the Semantic Web. We refer the reader to a survey prepared by the W3C RDB2RDF incubator group [2]. Existing approaches automatically generate RDF based on the relational schema [3] or map the relational data to an existing ontology [4,5,6]. However, to the best of our knowledge this paper represents the first exposition of an implemented architecture that fulfills all the aforementioned goals

3. ULTRAWRAP

Ultrawrap is comprised of four primary components.



Figrue. 1. Architecture of Ultrawrap.

3.1 Creating an OWL Putative Ontology

To create the OWL description of the legacy database Ultrawrap incorporates the mapping of Tirmizi et al. **[**7**].** Whether the results of a purely syntax driven translation of a SQL schema to OWL will result in an OWL file with all the properties necessary for an axiom system to be an ontology is controversial. Thus, we call the resulting OWL ontology a *putative ontology (PO)*. A generic definition of a putative ontology would be any syntactic transformation of a data source schema to an ontology.

3.2 Relational Database as RDF triples

The Ultrawrap PO becomes the basis for a user to develop SPARQL queries. The RDF triples of the database contents must be consistent with the PO. A central novelty of Ultrawrap is the definition of the triple representation intentionally as a SQL view. This view consists of the union of all the queries that define all the triples based on the PO. In other words, the view logically defines a three column view (subject, predicate and object), containing one row per triple.

```
CREATE VIEW TripleView(s,p,o) AS
SELECT “Product” +Product.id as s, “rdf:type” as p
“Product” as o FROM Product UNION
SELECT “Product” +Product.id as s, “label” as p
“ABC” as o FROM Product ….
```

3.3 SPARQL TO SQL

The SPARQL to SQL component of Ultrawrap naively translates a SPARQL query to a SQL query that is issued on the *TripleView*. For example, the SPARQL query in (1) gets translated to a SQL query in (2).

```
SELECT ?product ?label
WHERE { ?product label ?label. ?product
propNum1 1. ?product propNum2 2.} .
```

(1)

```
SELECT t1.s as product, t1.o as label,
FROM tripleview t1, t2, t3 WHERE t1.p =
'label' AND t1.s = t2.s AND t2.p =
'propNum1' AND t2.o = 1 AND t1.s = t3.s AND
t3.p = 'propNum2' AND t3.o = 2.
```

(2)

3.4 THE SQL OPTIMIZER IS THE REWRITE SYSTEM

We employ of the query optimizer to rewrite the query to the native SQL query on the relational schema. Consider a Datalog syntax to represent the *TripleView* from a relational table.

```
Triple(1, label, ABC) :- Product(1,ABC, _, _)
Triple(1, propNum1, 1) :- Product(1,_, 1, _)
Triple(1, propNum1, 2) :- Product(1,_, _, 2).
```

(3)

Now consider the SPARQL query in (1) in a Datalog syntax:

```
Answer(X, Y):-Triple(X, label, Y), Triple(X,
propNum1, 1), Triple(X, propNum2, 2)
```

(4)

The native SQL query on the relational table would be:

```
SELECT id, label FROM product WHERE propNum1
= 1 and propNum2 = 2
```

(5)

In Datalog syntax, this query would be represented:

```
Answer(X, Y) :- Product(X, Y, 1, 2)
```

(6)

If we take the SPARQL query in Datalog (4) and substitute it with the definition of the *TripleView* (3), we have the following:

```
Answer(X,Y):-Product(X,Y,1,_),Product(X,Y,_,
2)
```

(7)

Finally, by unifying both predicates in (7), we get the same result as (6), which is the same native SQL query on the relation schema. In other words, the query optimizer should be able to compile out self joins. This means that without any index support, the query in (5) would execute in worst case in O(n). Therefore we can take advantage of existing SQL infrastructure for this task.

4. EMPERICAL RESULTS

In our initial experimental study, we compare the performance of Ultrawrap to other native triple stores and RDB2RDF systems. We use Microsoft SQL Server as the relational database back-end. We execute three queries derived from the Berlin SPARQL Benchmark (BSBM), which exercise three characteristics: a small number of triple patterns (Q1), a large number of triple patterns (Q2) and free text search (Q3). Our preliminary results are in Figure 2.

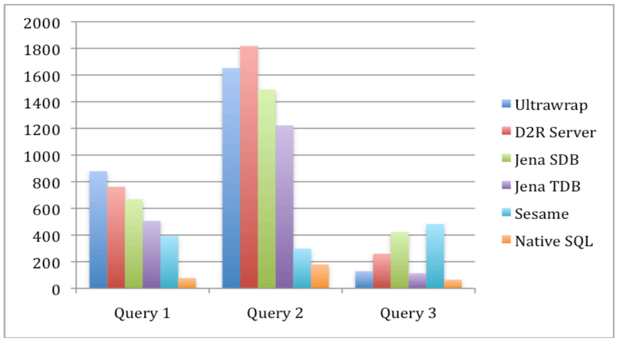


Figure. 2. Performance of Queries in sec on 1 Million triples.

5. CONCLUSIONS AND FUTURE WORK

In this work we have demonstrated a feasibility prototype of a system aiming for three goals. It is completely automatic. Any RDF output reflects in real-time the current state of the underlying database. It makes maximal use of existing SQL infrastructure. We have learned several lessons. Databases schemas created with contemporary data engineering methods (e.g. model driven architecture) can generate putative ontologies nearly as expressive as related domain ontologies. However, our current experiment with BSBM is not completely suitable because the relational schema apparently is derived from the domain ontology (or vice-versa). Furthermore, we realized that the current SQL Server optimizer does not compile out self joins. Instead of having just one SELECT statement with all the attributes, it would generate a SELECT statement for each attribute and then join them together. Nevertheless, our current results are promising. If the query optimizer is able to compile out the self joins, Ultrawrap could be a competitive option. On that basis, we will be moving forward by fully implementing Ultrawrap to its full capacity. We will implement specific SQL Server optimization techniques in order to make the query optimizer compile out self joins. Furthermore, we will formalize this optimization and do a complexity analysis. Consequently, we will evaluate with the complete BSBM and other benchmarks. Finally, we foresee the possibility of creating a Linked Data layer on top of Ultrawrap in order to expose the relational data following the linked data principles.

6. ACKNOWLEDGMENTS

This research has been funded partially by the NSF grant IIS-0531767 and the University of Texas Graduate School Diversity fellowship. We also thank the students of CS386 Database Management Systems, Spring 2009 for their many investigations of the optimization of databases for SPARQL queries per their term projects in that class.

7. REFERENCES

[1] He, B., Patel, M., Zhang, Z., Chang, K.C. Accessing the deep web. Communications of the ACM 50(5), 94–101 (2007).

[2] Sahoo, S. Halb, W. Hellmann, S. Idehen, K. Thibodeau, T. Auer, S. Sequeda, J F. Ezzat, A. A Survey of Current Approaches for Mapping of Relational Databases to RDF. W3C RDB2RDF Incubator Report, 2009.

[3] Sequeda, J.F., Tirmizi, S.H., Corcho, O., Miranker, D.P. Direct Mapping SQL Databases to the Semantic Web: a survey. Univeristy of Texas, Department of Computer Sciecnces Technical Report TR-09-04. 2009.

[4] C. Bizer and A. Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In Poster Proceedings of ISWC , 2004.

[5] O. Erling and I. Mikhailov. RDF support in the Virtuoso DBMS. In CSSW, 2007.

[6] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., and Aumueller, D. 2009. Triplify: light-weight linked data publication from relational databases. In WWW, 2009.

[7] Tirmizi, S.H., Sequeda, J.F., Miranker, D.P. Translating SQL applications to the semantic web. In DEXA, 2008