D2RQ/Update: Updating relational data via virtual RDF

Article · April 2012

DOI: 10.1145/2187980.2188095

2 authors:

Vadim Eisenberg
IBM
7 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Yaron Kanza
AT&T Labs - Research
92 PUBLICATIONS **1,465** CITATIONS
SEE PROFILE

D2RQ/Update: Updating Relational Data via Virtual RDF

Vadim Eisenberg*
Computer Science Department
Technion – Israel Institute of Technology
Haifa, Israel
eisenv@cs.technion.ac.il

Yaron Kanza*
Computer Science Department
Technion – Israel Institute of Technology
Haifa, Israel
kanza@cs.technion.ac.il

ABSTRACT

D2RQ is a popular RDB-to-RDF mapping platform that supports mapping relational databases to RDF and posing SPARQL queries to these relational databases. However, D2RQ merely provides a read-only RDF view on relational databases. Thus, we introduce D2RQ/Update—an extension of D2RQ to enable executing SPARQL/Update statements on the mapped data, and to facilitate the creation of a read-write Semantic Web.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Semantic Web*; D.2.12 [**Software Engineering**]: Interoperability—*Data mapping*

General Terms

Algorithms, Design, Languages

Keywords

Semantic Web, SPARQL, RDF, RDB-to-RDF mapping, Relational database, D2RQ, SPARQL/Update, Linked Data

1. INTRODUCTION

D2RQ is a popular mapping platform for publishing relational data as a virtual RDF graph [3]. It enables legacy relational databases to be exposed on the Web, according to the principles of Linked Data [1], and to be included in the Semantic Web. Since valuable data are frequently stored in relational databases, such functionality can be used to significantly enrich the Semantic Web. D2RQ exposes relational databases as *SPARQL endpoints* [6]. It translates SPARQL queries posed on a virtual RDF graph, to SQL queries posed to the underlying relational database. The translation can be based on an automatically generated mapping, according to the database schema, or on a manually defined one.

To realize the Semantic Web vision to its full potential, technologies for updating information on the Web are required. *Read-Write Linked Data* [2] extends the Linked Data principles with the requirement to allow applications to read, write and update data on the Semantic Web. SPARQL/Update is an extension of SPARQL to support update over RDF graphs [8]. D2RQ, however, merely provides read-only access to the relational data. So, extending D2RQ to support SPARQL/Update statements is an important step towards the creation of a read-write Semantic Web.

*The work of these authors was partially supported by The Israeli Ministry of Science and Technology (Grant 3-6472).

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

2. D2RQ/UPDATE

Before introducing D2RQ/Update, we provide a simple example of a mapping in D2RQ. Consider two relational tables, `Employee` and `Department`. The `Employee` table has a primary key `id` and additional attributes `name`, `salary` and `deptid`. Constraints are defined on the table: *(1)* a non-null constraint on `salary`, and *(2)* `deptid` is a foreign key referencing the attribute `id` of the table `Department`. The table `Department` has a primary key `id`, and another attribute `name`.

Listing 1 shows a snippet of a D2RQ mapping. In the example we use `acme` as a namespace (and as the base URI of D2RQ). According to the mapping, the `Employee` table is mapped to the `acme:Employee` class, while the URIs of the RDF individuals of the class are concatenations of `acme:emp` and the ids of the employees.

Listing 1: A snippet of a D2RQ mapping.

```
map:employees a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "emp@@Employee.id@@";
  d2rq:class acme:Employee.

map:salary a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:employees;
  d2rq:property acme:salary;
  d2rq:column "Employee.salary".
```

Given the following SPARQL query,

```
SELECT ?s WHERE {acme:emp7 acme:salary ?s}
```

it is translated by D2RQ to the SQL query

```
SELECT salary FROM Employee WHERE id = 7.
```

Listing 2: Inserting multiple triples

```
INSERT DATA { acme:emp7 acme:name "John" ;
                        acme:dept acme:dept17 .
       acme:dept17 acme:deptname "Sales" .
       acme:emp7  acme:salary 1000 . }
```

The goal of D2RQ/Update is to add SPARQL/Update capabilities to the D2RQ framework. For instance, we want D2RQ/Update to translate the SPARQL/Update statement

```
INSERT DATA {acme:emp7 acme:salary 1000 }
```
to the SQL statement
```
INSERT INTO Employee(id, salary) VALUES(7,1000) .
```

Some translations are not straightforward. Consider the SPARQL/Update statement in Listing 2. It should cause the tuples (7, "John", 1000, 17) and (17, "Sales") to be inserted into `Employee` and `Department`, respectively. Two challenges in translating such statements to SQL are (1) how to minimize the number of SQL statements, for optimization purposes, and (2) how to "bypass" integrity constraints, such as foreign keys and non-null constraints. (In theory, the constraints should be checked only at the end of the transaction, however, many existing RDBMS verify constraints after each SQL statement.)

A simple implementation would be to translate each insertion of a triple to a standalone statement. First, a row with primary key 7 can be inserted into the `Employee` table, using SQL INSERT, and then the attributes `salary`, `deptid` and `name` can be inserted using SQL UPDATE. In such approach, the number of SQL statements is equal to the number of attributes, and hence, it is not an optimal solution. More importantly, this approach may not work due to the constraints on the attributes `salary` and `deptid`. Since `salary` is defined as non-null, it has to be set in the first insert statement. Since `deptid` is a foreign key, it cannot be set before the referenced row of `Department` is created.

D2RQ++ [7] is an existing extension to D2RQ to enable updates of mapped relational data. It operates on a per-triple basis, missing the opportunity to minimize the number of translated SQL statements. When an update violates some constraint and is rejected by the underlying RDBMS, D2RQ++ adds the rejected triples to an auxiliary RDF store. The RDF store is periodically consolidated with the database, once the constraints in the database can be satisfied. Such approach, however, introduces inconsistencies between RDF applications working with the RDF view, and legacy applications working with the original data. Until the triples in the auxiliary RDF store are consolidated with the relational database, SPARQL queries evaluated by D2RQ++ may not be equivalent to their translation to SQL. The periodic consolidation solution may sometimes resolve inconsistencies caused by foreign-key constraints, by managing to insert the referenced data into the relational database first, and by inserting the referencing data in the next iteration. However, this solution does not handle non-null constrains. Tuples with a non-null constraint on an attribute other than the primary key will never be inserted into the relational database by the algorithm specified in [7], because the non-null constraints must be satisfied immediately, on the first INSERT statement.

Listing 3: Inserting multiple triples

```
SQL INSERT INTO Department
                VALUES(17, "Sales")
SQL INSERT INTO Employee
          VALUES(7, "John", 1000, 17)
```

In contrast to D2RQ++, D2RQ/Update considers groups of triples during translation to SQL. It groups together triples related to one subject and applies a bulk update. In the above example, D2RQ/Update will group the triples related to `acme:emp7` and `acme:dept17` and will create the statements in Listing 3.

D2RQ/Update reads the constraints from the schema of the mapped database. To preserve foreign key constraints during the update, D2RQ/Update performs a topological sort on the triples to be added, according to the order induced by the constraints. In our case, since `Department` is referenced by `Employee`, it is updated first.

D2RQ/Update performs delete and update statements in a similar way to the way insertion is being done—grouping triples by subject and sorting topologically the groups of triples according to the referential integrity constraints. If preserving the constraints is impossible, a descriptive error message is returned.

A different approach for mapping relational data to RDF is R2RML [4]. However, we are not aware of R2RML implementations that support SPARQL/Update. OntoAccess [5] is another mapping platform for exposing relational data as RDF. It introduces its own mapping language that allows specifying integrity constraints as part of the mapping. The drawback of OntoAccess is that, according to [5], it does not support SPARQL queries and has only basic support of SPARQL/Update. It is not a popular open-source framework such as D2RQ and it does not support all the features supported by D2RQ .

3. CONCLUSIONS

D2RQ/Update extends an existing popular framework for mapping relational data to RDF, while providing an RDF view that is always consistent with the underlying database. D2RQ/Update translates SPARQL/Update statements into SQL while preserving integrity constraints and while striving to minimize the number of the generated SQL statements.

The source code of D2RQ/Update, the algorithm and a tutorial are available on the D2RQ/Update website, via the link `http://d2rqupdate.cs.technion.ac.il`.

Future work includes optimizing the generated SQL statements and handling more complicated integrity constraints, such as cyclic references and cascading updates.

4. REFERENCES

[1] T. Berners-Lee. Linked data. `http://www.w3.org/DesignIssues/LinkedData.html`.

[2] T. Berners-Lee. Read-Write Linked Data. `http://www.w3.org/DesignIssues/ReadWriteLinkedData.html`.

[3] C. Bizer and A. Seaborne. D2RQ — treating non-RDF databases as virtual RDF graphs. In *ISWC*, 2004.

[4] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. Working draft, W3C, 2011.

[5] M. Hert, G. Reif, and H. C. Gall. Updating relational data via SPARQL/update. In *EDBT/ICDT Workshops*, 2010.

[6] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Recommendation, W3C, Jan. 2008.

[7] S. Ramanujam, V. Khadilkar, L. Khan, S. Seida, M. Kantarcioglu, and B. Thuraisingham. Bi-directional Translation of Relational Data into Virtual RDF Stores. In *ICSC*, 2010.

[8] A. Seaborne, G. Manjunath, C. Bizer, J. Breslin, S. Das, I. Davis, S. Harris, K. Idehen, O. Corby, K. Kjernsmo, and B. Nowack. SPARQL Update. A language for updating RDF graphs. Member submission, W3C, 2008.