RDB2RDF Plugin: Relational Databases to RDF plugin for Eclipse

Percy E. Salas    Edgard Marx    Alexander Mera    Karin K. Breitman
Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de S. Vicente, 225 - Rio de Janeiro, Brazil - CEP 22451-900
{psalas, emarx, acaraballo, karin}@inf.puc-rio.br

ABSTRACT
RDB2RDF is the process by which a relational database schema (RDB) is transformed into a set of RDF triples. A major step in this process is deciding how to represent database schema concepts in terms of RDF classes and properties. This is done by the RDB2RDF mapping file, which is used as the base for the RDF triples generation. Most RDB2RDF engines today provide support to the mechanical process of transforming RDB to RDF each with its own mapping language. Due to this fact, the W3C RDB2RDF Working Group has been working to standardize a language to map relational data to RDF called R2RML [1]. Part of their effort is directed to fostering the development of methods, tools and techniques to support the standard RDB2RDF mapping, that is currently under development. In this paper, we introduce an Eclipse plug-in that supports the standard RDB to RDF Mapping Language (R2RML) to produce Direct Mappings in RDF [2].

Keywords
RDB2RDF, Relational Databases, RDF, Eclipse

1. INTRODUCTION
In the last few years the Semantic Web has showed significant growth, as industry, government and academia are gradually adopting RDF [3] and OWL [4] standards. Best Buy and Data.gov are well known business cases where the adoption of semantic standards, RDF in particular, served to increase relevance and facilitate data interoperability [5][6]. Nevertheless, there is still a significant gap, if one compares the growth rates of the Web to that of the Semantic Web. The main reason is that of the existing websites, over 70% derive their data from relational databases (RDB) according to ACM [7]. Given the astounding amount of data stored in relational databases, a critical requirement for the evolution of the Semantic Web is the inclusion of such data.

In order to make relational data more easily available to the Semantic Web, we need reliable strategies in which to map data from RDBs to the Resource Description Framework (RDF) format. There are tools that do this mapping, also called RDB2RDF engines. The most noteworthy are Triplify [8], Virtuoso RDF views [9] and D2RQ [10]. It is important to note, however, that each engine provides a different, proprietary, language to do so. Due to this fact, some initiatives are being taken to establish standards to govern this process and provide more consistent (interoperable) results. That is the case of the RDB to RDF Mapping Language (R2RML) that provides a language to map relational database schema into RDF, which allows the implementations of views and real-time conversions, and Direct Mapping, a technique to convert relational data into RDF statements. That are some W3C initiatives whose main goal is to provide a default behavior for relational to RDF mappings.

Most existing RDB2RDF engines provide support to the mechanical process of transforming RDB to RDF. However, experience demonstrated that most have scalability, extensibility and, to a certain extent, usability problems.

We believe that most of these shortcomings could be alleviated by the adoption of a plug-in approach. Among the advantages, plug-in architectures provide support to fine grained modularity, an import software quality aspect [11][12], if one wants to encourage code reuse and facilitate the development of extensions. In the case of the RDB2RDF mechanism proposed in this paper, that is exactly the case because, while the RDF specification is unique, relational databases come in many flavors.

Plug-in frameworks also help to handle complexity, simplify application configuration, deployment, and enable users (or third parties) to easily extend functionality of existing applications with self-developed modules, without having access to the whole source code.

In this paper we propose a plug-in for Eclipse¹ which supports the conversion of relational data to RDF triples using either R2RML or Direct Mapping. The remainder of this paper is organized as follows. In Section 2, we discuss the basic concepts involved in the plug-in. The architecture chosen is explained in Section 3. In Section 4, we present implementation details of the map process. Related Works is given in Section 5 and finally, Section 6 concludes with an outlook of our approach and ideas for future work.

2. BASIC CONCEPTS
Mapping is the process by which a database schema and its instances are transformed into a RDF dataset [20]. A major step in this process is deciding how to represent database schema concepts in terms of RDF classes and properties. This is done by mapping database concepts to a vocabulary, to be used as the base in which to generate the RDF triples from.

Every R2RML mapping is tailored to a specific database schema and target vocabulary [1]. The input to an R2RML mapping is a relational database that conforms to that schema. The output is an RDF dataset in which the RDF statements are formed using concepts from the target vocabularies. This mapping itself is represented as an RDF graph, called mapping graph. When a mapping graph is encoded in the Turtle RDF syntax [13] it becomes an R2RML mapping document.

¹ The plugin is available for download at http://www.lod2go.com/downloads,
2.1 R2RML : RDB to RDF Mapping Language in a Nutshell
R2RML is a language for expressing customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice.

2.1.1 Mapping Overview
An R2RML mapping document consists of one or more structures called TriplesMaps. Each TriplesMap contains a reference to a logical table in the input database. A logical table can be one of the following:

- A base table that exists in the input SQL schema.
- A view that exists in the input SQL schema.
- A valid SQL query against the input schema.

Furthermore, a TriplesMap contains the rules for mapping a logical table row to a set of RDF triples. It consists of one SubjectMap structure and one or more PredicateObjectMap structure(s).

The RDF triples generated from one row all share the same subject. The SubjectMap structure in a TriplesMap contains the rules for generating the subject for a row.

Each PredicateObjectMap structure in a TriplesMap contains the rules for generating a (predicate, object) pair from the values in the table row. It consists of a PredicateMap structure and an ObjectMap structure.

A TriplesMap is used to generate RDF triples from a row in the logical table by combining the single subject for the row, generated using the SubjectMap, with the (predicate, object) pair(s) generated from the row using the PredicateObjectMap(s).

2.2 A Direct Mapping of Relational Data to RDF
The Direct Mapping is intended to provide a default behavior for R2RML, when no customization is needed. The structure of the resulting RDF graph directly reflects the structure of the database, the target RDF vocabulary directly reflects the names of database schema elements, and neither structure nor target vocabulary can be changed.

2.2.1 Types and IRIs elaboration
In the process of translating relational data into RDF, the direct mapping must create IRIs for identifying tables, the columns in a table, and each row in a table. The types and IRIs construction is given by the Table 1.

<table>
<thead>
<tr>
<th>IRIs</th>
<th>Base IRI</th>
<th>Table Name</th>
<th>Column(s) Name</th>
<th>Value(s)</th>
<th>Blank Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table IRI</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column IRI</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Single-column)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column IRI</td>
<td>x</td>
<td>x</td>
<td>x(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Multi-column)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.2.2 Mapping Rules
Each row in the database produces a set of RDF triples with a subject, predicate, and object. Some mapping rules for the database schema concepts in terms of RDF triples (subject, predicate and object) are given by the Table 2.

Table 2. Mapping Rules to set RDF triples

<table>
<thead>
<tr>
<th></th>
<th>Shared Subject (SubjectMap)</th>
<th>Predicate (PredicateMap)</th>
<th>Object (ObjectMap)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Triples</td>
<td>Row RDF Node</td>
<td>rdf:type property</td>
<td>Table IRI (Table)</td>
</tr>
<tr>
<td>Literal Triples</td>
<td>Row RDF Node</td>
<td>Column IRI</td>
<td>RDF Literal with XML Schema datatype</td>
</tr>
<tr>
<td>Reference Triples</td>
<td>Row RDF Node</td>
<td>Column IRI (foreign key)</td>
<td>Row RDF Node</td>
</tr>
</tbody>
</table>

3. ARCHITECTURE OF THE RDB2RDF PLUGIN
The architecture of the proposed plugin is centered around its main functionalities, metadata acquisition and mapping generation, organized in two separate modules, as illustrated by Figure 1. The first module is an Eclipse plug-in that imports metadata and generates the mappings using the algorithm selected by the users; the second module implements several mapping algorithms, such as RDB2RDF detailed in section 2 and illustrated in section 4. This architecture separates the main concerns and facilitates extensions and the inclusion of new mapping algorithms.

Figure 1 - Plug-in Architecture

3.1 Database Connection and Metadata Extraction
Both database connection and metadata extraction are performed through the JDBC driver. Implementations of this driver for several different relational databases are available [21,22,23,24]. In our implementation we already included drivers for major commercial and open source databases, including SQLServer, Postgres, MySQL and Oracle.

To perform metadata extraction it is necessary that the driver corresponding to the database is properly installed, and that the user supplies necessary connection parameters, e.g., driver, server address (host), the connection port (port), user, password and the
3.1.1 Extending the Mapping Algorithm Plug-in
To add a new mapping algorithm the user must implement the MapGen interface (Figure 2). Following the user must generate the Jar file and install it in the same directory as the Eclipse plug-in. If the process is successful, the new algorithm option will appear in the GUI, and the method process will be invoked receiving the metadata extracted from the RDB, processing and returning the resulting mappings.

4. RDB2RDF IMPLEMENTATION
The process to generate the mapping file using R2RML is given by the following algorithm.

```java
/* The Metadata is obtained via JDBC*/
Get the metadata from a data source;
For each table
 |
/* this function that allows specification of a
mapping of the rows in a logical table, represented
by a SQL query, or the name of a table or view */
TriplesMap();
For each column
 |
/* this function defines the mapping from the
values in a column of a relational table to a collection of
RDF (property, object) pairs*/
RDFTermMap();
/* this function allows specification of the mapping
that describes a foreign key relationship*/
ForeignKeyMap();
Return R2RML mapping file;
```

As described in section 2, R2RML is a language for expressing customized mappings from relational databases to RDF datasets. Thus, the input to an R2RML mapping is a relational database that conforms to that schema. The output is an R2RML mapping file; the mapping is itself also represented as an RDF graph. In other words, RDF is used not just as the target data model of the mapping, but also as the formalism in which to represent the R2RML mapping, as illustrated by the following example. Let us assume that we have two tables with a single-column primary keys and one foreign key between them. The result of R2RML mapping process is presented in listings 1 and 2.

<table>
<thead>
<tr>
<th>Table 3. DEPT Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>deptno</td>
</tr>
<tr>
<td>dname</td>
</tr>
<tr>
<td>loc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4. EMP Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>empno</td>
</tr>
<tr>
<td>ename</td>
</tr>
<tr>
<td>job</td>
</tr>
<tr>
<td>deptno</td>
</tr>
<tr>
<td>etype</td>
</tr>
</tbody>
</table>

Listing 1. Mapping Specification for the DEPT Table

```xml
<?xml version="1.0" encoding="UTF-8"?>
<#TriplesMap1>
  a rr:TriplesMap;
  rr:logicalTable "Select ('_:Department' || deptno) AS deptId ,
                   deptno , dname ,
                   loc from dept ";
  rr:tableGraphIRI xyz:DeptGraph;
  rr:subjectMap [ a rr:BlankNodeMap; rr:column "deptId" ];
  rr:InverseExpression "{alias.}deptno = substr({alias.}deptId,length('_:Department')+1)";";
  rr:propertyObjectMap [ rr:property dept:name; rr:column "dname" ];
  rr:propertyObjectMap [ rr:property dept:location; rr:column "loc" ];
  rr:propertyObjectMap [ rr:property dept:COMPANY; rr:constantValue "XYZ Corporation" ];
</#TriplesMap1>
```

Listing 2. Mapping Specification for the EMP Table

```xml
<?xml version="1.0" encoding="UTF-8"?>
<#TriplesMap2>
  a rr:TriplesMap;
  rr:logicalTable "Select ('xyz.com/emp/' || empno||') AS empURI ,
                  empno ,
                  name ,
                  ('<xyz.com/emp/job/'|| job||') AS jobTypeURI ,
                  job ,
                  deptno ,
                  ('<xyz.com/emp/etype/'|| etype||') AS empTypeURI ,
                  etype ,
                  ('<xyz.com/graph/'|| job || '/|| etype||') AS graphURI from emp ";
  rr:tableGraphIRI "graphURI"
</#TriplesMap2>
```
We firmly believe that the development of plugins such a the one proposed here represent an important advance in the popularization of Semantic Web tools. However there is still much to be done, if we aim at non-expert adoption. The most critical issues, in our opinion, are:

- Facilitate data selection: It is not always the case that all relational data needs to be mapped to RDF. Advanced user interfaces, that help identify sensitive data and exclude it from the process are on demand.
- Choice of vocabulary: it's still important to encourage the creation of algorithms for automatic vocabulary matching, thus reducing the overhead and need for manual intervention in the process.
- Provide support to relational data sets in the cloud. In particular in respect to Open Government Data [15] and data stored in open access repositories, such as Azure Marketplace [16].

5. RELATED WORK
There is a large variety of RDB2RDF engines that provide support to the mechanical process of transforming relational data to RDF triples, such as Triplify [8], Virtuoso RDF views [9], StdTrip [20] and D2RQ [10]. We condensed the comparison of these tools to the proposed plug-in in Table 1 because of space constraints.

Table 2. Mapping Rules to set RDF triples

<table>
<thead>
<tr>
<th>Tool</th>
<th>Interface</th>
<th>Automation</th>
<th>Mapping Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplify</td>
<td>Text editor</td>
<td>Manual</td>
<td>SQL</td>
</tr>
<tr>
<td>D2RQ</td>
<td>Text editor</td>
<td>Manual/Auto</td>
<td>RDF Based</td>
</tr>
<tr>
<td>StdTrip</td>
<td>Visual Tool</td>
<td>Manual/Auto</td>
<td>SQL, RDF Based,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R2RML</td>
</tr>
<tr>
<td>Virtuoso RDF views</td>
<td>Text editor</td>
<td>Manual</td>
<td>RDF Based</td>
</tr>
<tr>
<td>Proposed tool:</td>
<td>Visual Tool</td>
<td>Auto</td>
<td>R2RML (current</td>
</tr>
<tr>
<td>Mapping Plug-in</td>
<td></td>
<td></td>
<td>version)</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS
We believe that the use of the Semantic Web standards, as RDF, in schema design is the only viable way to guarantee future interoperability [17, 20]. However existing solutions that aim to use these standards, are still lacking in what concerns user usability and extensibility.

In this paper we introduced an Eclipse plug-in that enables experts as well as non-experts to automatic produce RDF mapings from information stored in relational databases using the R2RML format, a W3C standard in the making [1]. The tool is extensible, i.e., it offers the possibility of including, additional mapping algorithms. That is particularly interesting to secure interoperability with other types of standards, e.g., Open Data [14], REST [25].

7. ACKNOWLEDGMENTS
This research was made possible by grants E-26/170028/2008 from FAPERJ and 557.128/2009-9 from CNPq, at the Brazilian Web Science Institute.

8. REFERENCES


