Relational Database to RDF Mapping Patterns

Juan Sequeda[1], Freddy Priyatna[2], and Boris Villazón-Terrazas[2]

[1]Department of Computer Science, The University of Texas at Austin
`jsequeda@cs.utexas.edu`
[2]OEG-DIA, FI, Universidad Politécnica de Madrid, Spain
{`fpriyatna,bvillazon`}`@fi.upm.es`

Abstract. In order to integrate relational databases into Semantic Web applications, relational databases need to be mapped to RDF. The W3C RDB2RDF Working Group is in the process of ratifying two standards to map relational databases to RDF: Direct Mapping and R2RML mapping language. Through our experience as implementors of two RDB2RDF systems: Ultrawrap and Morph, and as authors of R2RML mappings, we have observed mappings that are reusable in order to solve a commonly occurring problem. In this paper, we have compiled these mappings and present a non-exhaustive list of RDB2RDF Mapping Patterns. We aspire that the mapping patterns in this paper are considered as a starting point for new mapping patterns.

Key words: RDB2RDF, Mapping Patterns, Mapping Language, R2RML, Relational Databases, SPARQL, SQL

1 Introduction

In order to integrate relational databases into Semantic Web applications, relational databases need to be mapped to RDF. The W3C RDB2RDF Working Group is in the process of ratifying two standards to map relational databases to RDF: Direct Mapping[1] and R2RML (Relational Database to RDF Mapping Language)[2]. Direct Mapping is the default way of representing a relational database as RDF based on the structure of the database schema. R2RML is a language for expressing customized mappings from relational databases to RDF.

As implementors of Ultrawrap[1][8] and Morph[2], two RDB2RDF systems, that support the Direct Mapping and R2RML standards; and as authors of several R2RML mappings that have been the basis of several projects including the W3C RDB2RDF Test Cases[9], we have observed mappings that are reusable in order to solve a commonly occurring problem. In this paper, we present a series of reusable mappings, which we define as RDB2RDF Mapping Patterns. The mappings are represented in R2RML.

We would like to point out that this is not an exhaustive list of mapping patterns. The mappings patterns that we present are based on our experience.

[1]http://www.capsenta.com/
[2]https://github.com/jpcik/morph/

Assuming the RDB2RDF standards are widely adopted, we expect the mapping patterns to increase. We aspire that the mapping patterns in this paper are considered as a starting point for new mapping patterns.

2 A Motivating Example

We present an example that motivates the need of RDB2RDF mapping patterns. Due to lack of space, we do not present an overview of R2RML. We refer the reader to the R2RML spec [2]. Assume you have a table Student with attributes id, name, and homephone. An application would like to map the table person to foaf:Person, create URIs based on the attribute id, map the attribute name to foaf:name and homephone to foaf:phone. The following R2RML mapping will produce the desired output:

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{id}"; ];
  rr:predicateObjectMap [ rr:predicate foaf:name;
    rr:objectMap [rr:column "name"];
  rr:predicateObjectMap [ rr:predicate foaf:phone;
    rr:objectMap [rr:column "homephone"]; ] .
```

We observe a One to One mapping between tables and ontology classes and a One to One mapping between attributes and ontology properties.

Now assume that the table Student has a new attribute, mobilephone, which we would also like to map to foaf:phone. This means that we would need to have a Many to One mapping between attributes and an ontology property. The previous mapping could be augmented by adding rr:objectMap [rr:column "mobilephone"]; to the existing rr:predicateObjectMap that has foaf:phone as a predicate. Another solution would be to repeat the entire rr:predicateObjectMap, but with a rr:column of mobilephone. This type of pattern impacts query performance. The following SPARQL query: `SELECT ?s ?o WHERE {?s foaf:phone ?o}` would get translated to the following SQL query: `SELECT id, homephone FROM Student UNION SELECT id, mobilephone FROM Student`. If we were to increase the amount of attributes mapped to the same ontology property, the size of the SQL query would increase. This example suggests that there is a tradeoff between mapping patterns and query performance. In order to further study tradeoffs and design decisions of RDB2RDF mappings, it is important to understand the different types of mapping patterns. In the next section, we present fourteen mapping patterns which we have observed as implementors of RDB2RDF systems and authors of R2RML mappings.

3 R2RML Mapping Patterns

A RDB2RDF mapping pattern is a reusable mapping that solves a commonly occurring problem. We present four type of mapping patterns: Attribute Mapping Patterns, Table Mapping Patterns, Join Mapping Patterns and Value Translation Patterns. Each pattern consists of a name, a question that defines the problem that is being addressed, description of the context, description of the solution in R2RML, an example R2RML mapping, a discussion and related patterns. Some mapping patterns may consist of different R2RML solutions.

3.1 Table Mapping Patterns

Tables in a relational database are (usually) mapped to ontology classes[3]. Each record of the table is mapped to an instance of the ontology class. In R2RML, every TripleMap must have exactly one rr:logicalTable and one rr:subjectMap. The rr:logicalTable defines the table (or SQL query) that is being mapped. The rr:subjectMap defines how to generate the subjects for the RDF triples. The following patterns define different ways that a table can be mapped to an ontology class. Patterns to generate the URIs for the subjects are out of the scope of this work. We refer the reader to [3].

Pattern 1: One to One Table Mapping

How to map a table to an ontology class?
Context: An application would like to map a tables to an ontology class. Moreover, every record of the tables is mapped to an instance of the corresponding ontology classes. For example, the table student is mapped to foaf:Person.
Solution: Create a TriplesMap for the table and specify the rr:logicalTable whose value corresponds to the table name. In the TriplesMap, create a rr:subjectMap with a rr:template to define the URI template for each row. Finally, the rr:subjectMap will have a rr:class corresponding to the ontology class for that table.
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
     rr:subjectMap [ rr:class foaf:Person;
     rr:template "http://example.com/resource/Student/{sid}" ] .
```

Discussion: This is the simplest pattern for table mapping. This is the case of the Direct Mapping, which automatically generates a unique ontology class for each table. However, a user has the option to specify a particular ontology class.
Related Patterns: N/A

Pattern 2: One to Many Table Mapping

How to map a table to several ontology classes?
Context: An application would like to map a table to many ontology classes. Moreover, every record of the table is mapped to an instance of the corresponding ontology classes. For example, the table student is mapped to foaf:Person and ex:Student.
Solution: Create a TriplesMap for the table and specify the rr:logicalTable whose value corresponds to the table name. In the TriplesMap, create a rr:subjectMap with a rr:template to define the URI template for each row. Finally, the rr:subjectMap will have multiple rr:class that correspond to the ontology classes for that table.
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
   rr:subjectMap [ rr:template
     "http://example.com/resource/Student/{sid}";
     rr:class foaf:Person; rr:class ex:Student ].
```

Discussion: This pattern extends Pattern 1 by adding multiple rr:class.
Related Patterns: Pattern 1

[3]Except for the case when the table represents a many-to-many relationship

Pattern 3: Many to One Table Mapping

How to map several tables to an ontology class?
Context: An application would like to map many tables to an ontology class. Moreover, every record of these table are mapped to an instance of the corresponding ontology class. For example, the tables student and professor are both mapped to foaf:Person.
Solution: Repeat the solution in pattern 1 for each table to be mapped.
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName  "student" ];
  rr:subjectMap [ rr:class foaf:Person;
    rr:template "http://example.com/resource/Student/{sid}" ] .

<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName  "professor" ];
  rr:subjectMap [ rr:class foaf:Person;
    rr:template "http://example.com/resource/Professor/{sid}" ] .
```

Discussion: This pattern extends Pattern 1 and it is used when instances of an ontology class may be distributed over several tables.
Related Patterns: Pattern 1

Pattern 4: Many to Many Table Mapping

How to map several tables to several ontology classes?
Context: An application would like to map a table to several ontology classes. Additionally, the application would like to map an ontology class to several tables.
Solution: Repeat solution of pattern 2 for each table to be mapped.
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName  "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}";
                  rr:class foaf:Person; rr:class ex:Academic ] .

<TriplesMapProf> a rr:TriplesMap; rr:logicalTable [ rr:tableName  "professor" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Professor/{sid}";
                  rr:class foaf:Person; rr:class ex:Academic ] .
```

Discussion: This pattern extends Pattern 3. In addition to instances of an ontology classes are distributed over several database tables, each row of the database table produces multiple ontology instances.
Related Patterns: Pattern 2, Pattern 3

3.2 Attribute Mapping Patterns

Attributes of tables are mapped to ontology properties. In R2RML, a TripleMap can have zero or more rr:predicteObjectMap, which in turn specifies a predicate-object pair. The following patterns define different ways that an attribute can be mapped to an ontology property.

Pattern 5: One to One Attribute Mapping

How to map an attribute to an ontology property?
Context: An application would like to map an attribute to an ontology property. For example, the attribute firstname is mapped to foaf:givenName.
Solution: Given a TripleMap, create a rr:predicateObjectMap for the attribute, which has only one rr:predicate for the ontology property and a rr:objectMap for the attribute.
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [
    rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate foaf:givenName;
    rr:objectMap [rr:column "firstname"]; ] .
```

Discussion: This is the simplest pattern for attribute mapping. This is the case of the Direct Mapping, which automatically generates a unique ontology property for each attribute.
Related Patterns: N/A

Pattern 6: One to Many Attribute Mapping

How to map an attribute to several ontology properties?
Context: An application would like to map an attribute to several ontology properties. For example, the attribute lastname is mapped to foaf:familyName and ex:apellido.
Solution 1: Given a TripleMap, create a rr:predicateObjectMap for the attribute, which has a rr:predicate for each ontology property and a rr:objectMap for the attribute.
Example R2RML Mapping for Solution 1

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate foaf:familyName; rr:predicate ex:apellido;
    rr:objectMap [rr:column "lastname"]; ]
```

Solution 2: Repeat the solution for Pattern 5 for the attribute to be mapped but having the rr:predicate specific to each ontology property
Example R2RML Mapping for Solution 2

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [
    rr:predicate foaf:familyName; rr:objectMap [rr:column "lastname"]; ];
  rr:predicateObjectMap [
    rr:predicate ex:apellido; rr:objectMap [rr:column "lastname"]; ].
```

Discussion: Solution 1 is a short cut for Solution 2.
Related Patterns: Pattern 5

Pattern 7: Many to One Attribute Mapping

How to map several attributes to an ontology property?
Context: An application would like to map several attributes to an ontology property. For example, the attribute homephone and mobilephone is mapped to foaf:phone.

Solution 1: Given a TripleMap, create a rr:predicateObjectMap, which has only a rr:predicate for the ontological property and a rr:objectMap for each attribute.
Example R2RML Mapping for Solution 1

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate foaf:phone;
    rr:objectMap [rr:column "homephone"]; rr:objectMap [rr:column "mobilephone"]; ].
```

Solution 2: Repeat the solution for Pattern 5 for each attribute to be mapped
Example R2RML Mapping for Solution 2

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate foaf:phone;
    rr:objectMap [rr:column "homephone"]; ];
  rr:predicateObjectMap [ rr:predicate foaf:phone;
    rr:objectMap [rr:column "mobilephone"]; ].
```

Discussion: Solution 1 is a short cut for Solution 2. As described in the motivating example of Section 2, this pattern impacts performance on queries that select on the ontology property.
Related Patterns: Pattern 5

Pattern 8: Many to Many Attribute Mapping

How to map several attributes to several ontology properties?
Context: An application would like to map an attribute to several ontology predicates. Additionally, the application would like to map an ontology predicate to several attributes.
Solution 1: Combine Solution 1 of Pattern 6 with Solution 1 of Pattern 7.
Example R2RML Mapping for Solution 1

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate foaf:phone; rr:predicate ex:telefono;
    rr:objectMap [rr:column "homephone"]; rr:objectMap [rr:column "mobilephone"]; ].
```

Solution 2: Combine Solution 2 of Pattern 6 with Solution 2 of Pattern 7.
Example R2RML Mapping for Solution 2

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}" ];
  rr:predicateObjectMap [ rr:predicate foaf:phone;
    rr:objectMap [rr:column "homephone"]; ];
  rr:predicateObjectMap [ rr:predicate foaf:telefono;
    rr:objectMap [rr:column "homephone"]; ];
  rr:predicateObjectMap [ rr:predicate foaf:phone;
    rr:objectMap [rr:column "mobilephone"]; ];
  rr:predicateObjectMap [ rr:predicate foaf:telefono;
    rr:objectMap [rr:column "mobilephone"]; ].
```

Discussion: Solution 1 is a shortcut for Solution 2.
Related Patterns: Pattern 6 and Pattern 7.

Pattern 9: Concatenate Attributes

How to concatenate attributes and map it to an ontology property?
Context: An application would like to concatenate several attributes and map the result to an ontology property. For example, concatenate the attributes firstname and lastname and map it to foaf:name.

Solution 1: Given a TripleMap, create a rr:predicateObjectMap which has a rr:predicate for the ontology property and the rr:objectMap as a rr:template. The concatenation is represented as a template.
Example R2RML Mapping for Solution 1

```
<TriplesMapStudent1> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}" ];
  rr:predicateObjectMap [
    rr:predicate foaf:name; rr:objectMap [ rr:template "{firstname} {lastname}" ]; ].
```

Solution 2: Create a new TripleMap with an R2RML view which consists of a rr:logicalTable that has a rr:sqlQuery which includes the concatenation. Additionally, create a rr:predicateObjectMap which has a rr:predicate for the ontology property and the rr:objectMap for the attribute which represents the concatenation in the SQL query.
Example R2RML Mapping for Solution 2

```
<TriplesMapStudent1> a rr:TriplesMap; rr:logicalTable [
    rr:sqlQuery "SELECT sid, firstname || ' ' || lastname AS fullname FROM student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}" ];
  rr:predicateObjectMap [
    rr:predicate foaf:name; rr:objectMap [rr:column "fullname"]; ].
```

Discussion: Consider the SPARQL query `SELECT ?s ?fullname WHERE {?s foaf:name ?fullname }`. Solution 1 would produce a SQL query and concat the name in the SELECT clause. Solution 2 does the concatenation operation as a SQL query as specified in rr:sqlQuery and then use this query as a subquery.

```
Solution 1 SQL : SELECT sid, firstname || '' || lastname as fullname FROM student
Solution 2 SQL : SELECT sid, fullname
                 FROM (SELECT sid, firstname || ' ' || lastname AS fullname
                       FROM student)
```

Note that these queries are semantically equivalent. From a query performance perspective, they should be equal unless the RDBMS does not have optimizations for subqueries.
Related Patterns: Pattern 5

3.3 Join Mapping Patterns

Foreign Key relationships among tables can be mapped to ontology properties. The following patterns define different ways that foreign key relationships can be mapped to an ontology property.

Pattern 10: Foreign Key between Two Tables

How to represent the relationship between two tables?
Context: An application would like to map a table to an ontology class. However, some of the property values are stored in another table. Therefore, it is necessary to perform a join to get those values.
Solution: Given two tables, one table will be considered the child and the other the parent. Create a TripleMap for each table. Given the child TripleMap, create a rr:predicateObjectMap which will have, in addition to the rr:predicate, a rr:objectMap which has a rr:parentTripleMap and a rr:joinCondition. The rr:parentTripleMap will point to the parent TripleMap and the rr:joinCondition

will have a rr:child and rr:parent which represent the join attributes in the child and parent table respectively.
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.org/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate ex:countryOfBirth;
    rr:objectMap [ rr:parentTriplesMap <TriplesMapCountry>;
      rr:joinCondition [ rr:child "country_of_birth" ;rr:parent "cid" ; ];];].

<TriplesMapCountry> a rr:TriplesMap; rr:logicalTable [ rr:tableName "country" ];
  rr:subjectMap [ rr:template "http://example.org/resource/Country/{cid}"; ].
```

Discussion: This pattern describes an R2RML mapping that joins two tables. This mapping can also be represented using Pattern 11. However, if a join involves more than two tables, then Pattern 11 must be used. The addition of another property that involves a join in Pattern 10 means that the user has to specify a new parent TriplesMap and then refer this parent TriplesMap in a rr:objectMap for the new property. On the other hand, the rr:logicalTable value stays the same and no changes needed.
Related Patterns: Pattern 11

Pattern 11: Foreign Keys between Two or more Tables

How to represent the relationship between two or more tables?
Context: An application would like to map a table to an ontology class. However, some of the property values are stored in other tables. Therefore, it is necessary to perform a joins to get those values.
Solution: Create a TripleMap with an R2RML view which consists of a rr:logicalTable that has a rr:sqlQuery which includes a SQL query that represent explictly the join(s).
Example R2RML Mapping

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:sqlQuery """
    SELECT s.sid AS sid, c.country_code AS country_code FROM student s, country c
    WHERE s.country_of_birth = c.country_id """];
  rr:subjectMap [ rr:template "http://example.org/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate ex:countryOfBirth;
    rr:objectMap [ rr:template "http://example.org/resource/Country/{cid}"; ]; ] .
```

Discussion: If the join is between two tables, then Pattern 10 can be used. However, if the join is between more than two tables, then Pattern 10 can not be used and the SQL query must be made explicit. Unlike Pattern 10, Pattern 11 does not require to have additional TriplesMap needed in order to map the property that needs a join. An addition of a new property that involves a join does not require to create additional TriplesMap instance, but the user has to modify the SQL query.
Related Patterns: Pattern 10

Pattern 12: Many to Many Table

How to map a table that represents a many-to-many relationship between two other tables to an ontology property?
Context: A many-to-many table represents a relationship between two entities. For example, the table StudentSport records the relationship of which Students

play a specific Sport. Several students can play a sport and several sports can be played by a student. An application would like to map the many-to-many table to an ontology property.
Solution: Create a TriplesMap for the many-to-many table. Specify the rr:logicalTable whose value corresponds to the table name of the many-to many table. In the TriplesMap, create a rr:subjectMap with a rr:template to define the URI template for one of the tables of the many-to-many relationship. Create an instance of rr:predicateObjectMap which has a rr:predicate for the ontology property. Finally, create a rr:objectMap with a rr:template to define the URI template for the other table of the many-to-many relationship.
Example R2RML Mapping

```
<TriplesMapStudentSport> a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "StudentSport" ];
    rr:subjectMap [ rr:template "http://example.org/resource/Student/{studentid}"; ];
    rr:predicateObjectMap [ rr:predicate ex:plays;
    rr:objectMap [ rr:template "http://example.org/resource/Sport/{sportid}" ]; ] .
```

Discussion: This mapping can also be represented through Pattern 11 given that it consists of a join between three tables (Student, StudentSport and Sport).
Related Patterns: Pattern 11

3.4 Value Translation Patterns

It is common that specific values in the database are code values which need to be translated to URIs. R2RML relies on SQL's `CASE` statement for the translation. The following patterns define different ways that values can be translated.

Pattern 13: Translate Values

How to map values in a table to URIs?
Context: An application would like to map values in the data to IRIs. For example, if the value in a job attribute is engineer, then a special IRI needs to be generated. A translation using rr:template is not possible because templates can only use the same values from the database.
Solution: Create a new TripleMap with an R2RML view which consists of a rr:logicalTable that has a rr:sqlQuery. Represents the translation in the SQL query by using SQL CASE statement.
Example R2RML Mapping

```
<#TriplesMap1> a rr:TriplesMap; rr:logicalTable [ rr:sqlQuery """
    SELECT EMP.*, (CASE JOB
         WHEN 'CLERK' THEN 'general-office' WHEN 'NIGHTGUARD' THEN 'security'
         WHEN 'ENGINEER' THEN 'engineering' END) ROLE FROM EMP  """ ];
  rr:subjectMap [ rr:template "http://data.example.com/employee/{EMPNO}"; ];
  rr:predicateObjectMap [ rr:predicate ex:role;
    rr:objectMap [ rr:template "http://data.example.com/roles/{ROLE}";
        rr:termType rr:IRI; ]; ] .
```

Discussion: The R2RML language does not have the expressivity to represent such value translation. Therefore, translating values has been pushed into SQL using the CASE statement. Query performance depends on the optimizations that a RDBMS has for the CASE statement.
Related Patterns: N/A

Pattern 14: Translate Values between Tables

How to map values in a referenced table

Context: An application would like to map a table to an ontology class. The table has a foreign key that references another table. The referenced table contains columns whose values need to be translated into URIs.

Solution 1: Combine Pattern 11 and Pattern 13.

Example R2RML Mapping for Solution 1

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:sqlQuery """
    SELECT s.sid AS sid, a.aid AS aid, (CASE a.article_type
        WHEN 'ppr' THEN 'Paper' WHEN 'ths' THEN 'Thesis') AS ArticleType
    FROM student s, article a WHERE s.sid=a.author """ ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate ex:isAuthorOf;
    rr:objectMap [ rr:template "http://example.com/resource/{ArticleType}/{aid}";];].
```

Solution 2: Use Pattern 10

Example R2RML Mapping for Solution 2

```
<TriplesMapStudent> a rr:TriplesMap; rr:logicalTable [ rr:tableName "student" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Student/{sid}"; ];
  rr:predicateObjectMap [ rr:predicate ex:isAuthorOf; rr:objectMap [
    rr:parentTriplesMap <TriplesMapPaper>;
    rr:joinCondition [ rr:child "sid" ; rr:parent "author" ; ]; ]; ];
  rr:predicateObjectMap [ rr:predicate ex:isAuthorOf; rr:objectMap [
    rr:parentTriplesMap <TriplesMapThesis>;
    rr:joinCondition [ rr:child "sid" ; rr:parent "author" ; ]; ]; ].

<TriplesMapPaper> a rr:TriplesMap; rr:logicalTable [
    rr:sqlQuery """SELECT author, aid FROM article WHERE article_type='ppr'""" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Paper/{aid}"; ].

<TriplesMapThesis> a rr:TriplesMap; rr:logicalTable [
    rr:sqlQuery """SELECT author, aid FROM article WHERE article_type='ths'""" ];
  rr:subjectMap [ rr:template "http://example.com/resource/Thesis/{aid}"; ].
```

Discussion: The selection of the possible solutions affect the way a user add a new article type, and also the SQL needed to execute the mappings.

In solution 1, to add a new article type, a user just need to modify the SQL query, appending the corresponding WHEN THEN pair statement. In solution 2, a new (parent) TriplesMap instance is needed for each article type. Then, in the child TriplesMap, every article type needs a predicateObjectMap property.

Now consider the following SPARQL query:
`SELECT ?s ?o WHERE { ?s ex:isAuthorOf ?o }`. The resulting SQL query for Solution 1 is the following:

```
SELECT sid, ArticleType, aid FROM (
    SELECT s.sid AS sid, a.aid AS aid, (CASE a.article_type
        WHEN 'ppr' THEN 'Paper' WHEN 'ths' THEN 'Thesis') AS ArticleType
    FROM student s, article a WHERE s.sid=a.author)
```

The resulting SQL query for Solution 2 is the following:

```
SELECT sid, aid
    FROM student S, (SELECT author, aid FROM article WHERE article_type='ppr') P
    WHERE S.sid = P.author UNION
    SELECT sid, aid
    FROM student S, (SELECT author, aid FROM article WHERE article_type='ths') T
    WHERE S.sid = T.author
```

Note that the generated SQL queries are very different and may have an impact on query performance depending on the RDBMS.

Related Patterns: Pattern 10, Pattern 11, Pattern 13

4 Related Work

In the (Object-Oriented) software community, patterns are used to describe software design structures that can be used over and over again in different systems. They provide a general solution that has to be applied in a particular context, in which the design considerations serve to decide whether the pattern is useful and how it could be implemented best [4]. A kind of software patterns are the re-engineering software patterns [6]. These patterns describe how to change a legacy system into a new, refactored system that fits current conditions and requirements. Their main goal is to offer a solution for re-engineering problems. They are also on a specific level of abstraction, that describes a process of re-engineering without proposing a complete methodology, and sometimes can suggest which type of tool to use. Therefore RDB2RDF mappings can be seen as re-engineering patterns because they map a legacy system (RDBMS) into a new system (RDF).

In the Semantic Web community, the Ontology Design Pattern portal[4] has been created in order to help in the design and quality of ontologies. Additionally, Dodds and Davis presents a pattern catalogue for modeling, publishing and consuming Linked Data [3]. These Linked Data patterns do not include RDB2RDF mapping patterns. Therefore our work is complemented by the Linked Data patterns of Dodds and Davis. Furthermore, Hert et al. presents a comparison of the expressivity of RDB2RDF mapping languages [5]. A framework for comparison is introduced, which consists of a set of fifteen features such as support for different types of mappings, datatypes, named graphs, blank nodes etc. R2RML supports all but one feature (write access) . Moreover, Rivero et al. presents fifteen RDF to RDF mapping patterns [7]. Some of these patterns are specific to RDF to RDF mappings such as Remove Language Tag while others are not applicable to RDB2RDF mappings such as Rename Class or Rename Property.

5 Conclusions and Future Work

In this paper, we have introduced fourteen RDB2RDF mapping patterns[5], which we have observed as reusable mapping throughout our experience as RDB2RDF system developers and R2RML mapping authors. As previously mentioned, this is a non-exhaustive list of mapping patterns and we aspire that the mapping patterns presented in this paper serve as a starting point. We foresee new mapping patterns in areas such as Named Graphs, Blank Nodes for anonymous or sensitive data, Metadata, Languages, Datatypes. We hope that this work encourages the Semantic Web community to further extend the RDB2RDF mapping patterns.

In certain patterns, we have identified two R2RML mapping solutions. These solutions may or may not have performance issues. As future work, we will

[4]http://ontologydesignpatterns.org/

[5]The mappings are stored at `http://mappingpedia.linkeddata.es/pattern/DataPatterns/`

thoroughly study the tradeoff between mapping patterns and query performance. Additionally, we will investigate the overlap between ontology design patterns, linked data patterns, the feature set of Hert et al. and the RDF to RDF mapping patterns of Rivero et al. with RDB2RDF mappings in general.

Finally, we have to see how to align the proposed RDB2RDF mapping patterns with Re-engineering Patterns category in the ODP Portal.

Acknowledgments: This work has been supported by the PlanetData, BabelData, and myBigData projects. Juan F. Sequeda was supported by the NSF Graduate Research Fellowship. We would like to kindly thank all W3C RDB2RDF Working Group members. We thank Richard Cyganiak for discussions.

References

1. M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda. Direct mapping of relational data to RDF. W3C Working Draft 29 May 2012, `http://www.w3.org/TR/2012/WD-rdb-direct-mapping-20120529/`.
2. S. Das, S. Sundara, and R. Cyganiak. R2rml: Rdb to rdf mapping language. W3C Working Draft 29 May 2012, `http://www.w3.org/TR/2012/WD-r2rml-20120529/`.
3. L. Dodds and I. Davis. Linked data patterns-a pattern catalogue for modelling, publishing, and consuming linked data. http://patterns.dataincubator.org/book/, 2011.
4. H. Edwards, R. Puckett, and A. Jolly. Analyzing Communication Patterns in Software Engineering Projects. In *Software Engineering Research and Practice*, pages 310–315, 2006.
5. M. Hert, G. Reif, and H. C. Gall. A comparison of rdb-to-rdf mapping languages. In *Proceedings of the 7th International Conference on Semantic Systems*, 2011.
6. R. Pooley and P. Stevens. Software Reengineering Patterns. Technical report, 1998.
7. C. R. Rivero, A. Schultz, C. Bizer, and D. Ruiz. Benchmarking the performance of linked data translation systems. In *Proceedings of the 5th Linked Data on the Web Workshop (LDOW)*, 2012.
8. J. F. Sequeda and D. P. Miranker. Ultrawrap: Sparql execution on relational data. Technical Report TR-12-10, The University of Texas at Austin, Department of Computer Sciences, 2012.
9. B. Villazón-Terrazas and M. Hausenblas. R2rml and direct mapping test cases. W3C Editor's Draft 24 July 2012, `http://www.w3.org/2001/sw/rdb2rdf/test-cases/`.