Building Ontologies by using Re-engineering Patterns and R2RML Mappings

Freddy Priyatna[1] and Boris Villazón-Terrazas[1]

[1]OEG-DIA, FI, Universidad Politécnica de Madrid, Spain
fpriyatna@delicias.dia.fi.upm.es,bvillazon@fi.upm.es

Abstract. The ontologization of non-ontological resources has led to the design of several specific methods, techniques and tools. Among those techniques, we have the Re-engineering Patterns. More specifically, we have the Patterns for re-engineering NORs (PR-NOR) that define a procedure that transforms the NOR terms into ontology representational primitives. Currently, the W3C RDB2RDF Working Group is at the final stage of formalizing R2RML, a language for describing mappings among RDB elements and RDF. In this paper we claim that it is possible to combine PR-NORs with R2RML mappings for building ontologies from relational database content, i.e., transforming the database content into an ontology schema by using Re-engineering Patterns and R2RML mappings.

Key words: Re-engineering patterns, RDB2RDF, R2RML, Ontologies

1 Introduction

During the last decade, specific methods, techniques and tools were proposed for building ontologies from existing knowledge resources. When we are transforming non-ontological resources (NORs) [4] into ontologies, the transformation process may follow one of the following approaches: (1) ***ABox transformation*** [4], which transforms the resource schema into an ontology schema, and the resource content, into ontology instances; (2) ***TBox transformation*** [4], which transforms the resource content into an ontology schema; or (3) ***Population***, which transforms the resource content into instances of an available ontology. The ABox transformation leaves the informal semantics of the transformed resources mostly untouched, whereas, the TBox transformation tries to enforce a formal semantics into them. Figure. 1 depicts the three types of transformation.

According to the survey described in [9], most of the available methods and tools deal with ABox transformation and Population. However there are some cases when it is useful to follow the TBox transformation [4], for example when we have a taxonomy stored in a particular NOR. The ontologization of non-ontological resources has led to the design of several specific methods, techniques and tools [4]. Among those techniques, we have the Re-engineering Patterns,

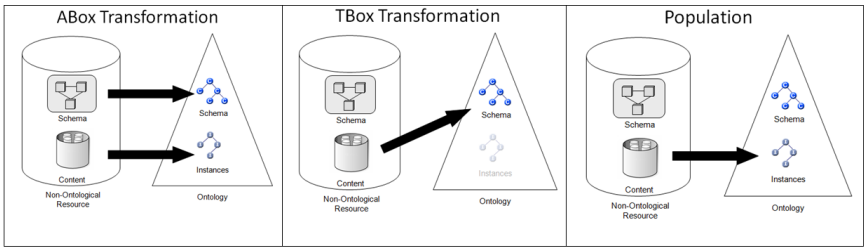

Fig. 1. NOR Transformation into ontologies

within the context of the ODP[1]. More specifically, we have the Patterns for re-engineering NORs (PR-NOR) that define a procedure that transforms the NOR terms into ontology representational primitives. Nevertheless, PR-NORs do not consider the implementation of the NOR, they just provide a general algorithm for the transformation.

The majority of non-ontological resources underpinning the Web are implementing in Relational Databases (RDB) [6]. RDB systems host a vast amount of structured data in relational tables augmented with integrity constraints [5]. When we are transforming RDB content into ontologies, we can follow two approaches (1) procedural/imperative approach, or (2) declarative approach, by defining mappings between RDB and ontology elements. There are several RDB2RDF mapping languages for describing transformation among RDB elements and ontologies [9, 7]. The RDB2RDF working group[2] is at the final stage of formalizing R2RML[3], a standard language for expressing mappings from relational databases to RDF datasets.

As we mentioned above, we provided a general algorithm to do the transformation for each of PR-NORs. However, in order to actually do the transformation, a user has to implement this algorithm in his choice of programming language (Java, Scala, etc). On the other hand, we observed that although R2RML mappings are normally used to generate ontology instances from database content (Population transformation), we figured out that when the database content follows specific patterns such as PR-NORs, then R2RML mappings can be useful in this situation. In this paper we propose to combine PR-NORs with R2RML mappings for building ontologies from relational database content, i.e., transforming the database content into an ontology schema by using Re-engineering Patterns and R2RML mappings.

The rest of the paper is organized as follows. Section 2 provides the background knowledge, by describing PR-NORs and R2RML. Then, Section 3 presents how we combine the PR-NORs with R2RML mappings for building the ontolo-

[1] `http://ontologydesignpatterns.org`
[2] `http://www.w3.org/2001/sw/rdb2rdf/`
[3] `http://www.w3.org/TR/r2rml/`

gies from relational database content, and includes two examples. Finally, Section 4 presents the conclusions and future work.

2 Background Knowledge

In this section we provide a brief description of the PR-NORs and R2RML.

2.1 Patterns for Re-engineering Non-Ontological Resources

The Patterns for Re-engineering Non-Ontological Resources (PR-NOR) [12] define a procedure that transforms the NOR terms into ontology elements. The patterns describe the transformation of classification schemes, thesauri, and lexicons into ontologies. The patterns rely on the data model[4] of the NORs. The patterns define, for every data model of the NORs, a process (expressed as an algorithm) with a well-defined sequence of activities in order to extract the NORs terms, and then to map these terms to a conceptual model of an ontology. Nevertheless, the patterns do not consider the implementation of the NOR, they just provide a general procedure for the transformation. It is worth noting that these patterns are included in the ODP Portal[5]. Table 1 lists the set of PR-NORs that perform the TBox transformation approach.

Table 1. Set of patterns for re-engineering NORs that perform the TBox transformation approach.

N	Identifier	Type of NOR	NOR Data Model	Target
1	PR-NOR-CLTX-01	Classification Scheme	Path Enumeration	Ontology Schema (TBox)
2	PR-NOR-CLTX-02	Classification Scheme	Adjacency List	Ontology Schema (TBox)
3	PR-NOR-CLTX-03	Classification Scheme	Snowflake	Ontology Schema (TBox)
4	PR-NOR-CLTX-04	Classification Scheme	Flattened	Ontology Schema (TBox)
5	PR-NOR-TSTX-01	Thesaurus	Record-based	Ontology Schema (TBox)
6	PR-NOR-TSTX-02	Thesaurus	Relation-based	Ontology Schema (TBox)
7	PR-NOR-LXTX-01	Lexicon	Record-based	Ontology Schema (TBox)
8	PR-NOR-LXTX-02	Lexicon	Relation-based	Ontology Schema (TBox)

In a nutshell, the main fields of a PR-NOR are:

[4]The data model[12] is the abstract model that describes how data is represented and accessed. The data model can be different even for the same type of non-ontological resource.

[5]`http://ontologydesignpatterns.org/wiki/Submissions:ReengineeringODPs`

– Name of the pattern
– Identifier of the pattern
– Use case, description in natural language of the re-engineering problem addressed by the pattern
– Input, description in natural language of the NOR, and its graphical representation
– Output, description in natural language of the ontology created after applying the pattern, and its graphical representation
– Process, algorithm for the re-engineering process

2.2 R2RML

R2RML[6] is a language for expressing mappings from relational databases to RDF datasets. These mappings provide the ability to view existing relational data in the RDF data model, expressed in a target ontology. The input to an R2RML mapping is a relational database. The output is an RDF dataset that uses predicates and types from the target ontology. It is worth mentioning that R2RML mappings are themselves expressed as RDF graphs and written down in Turtle syntax [2]. Figure 2 shows the elements of R2RML language

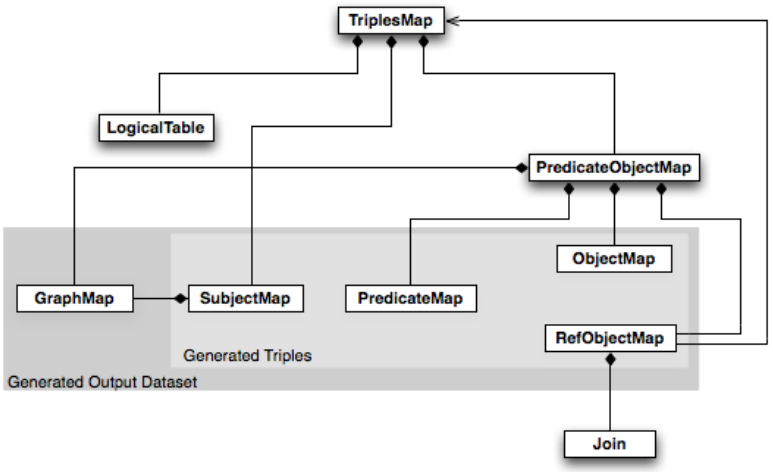

Fig. 2. An overview of R2RML

In a nutshell, an R2RML mapping points to logical tables to get data from the database. A logical table can be (1) a database table, (2) a view, or (3) a valid SQL query. Each logical table is mapped to RDF using a triples map. A

[6] `http://www.w3.org/TR/r2rml/`

triples map defines rules that map each row in the logical table to a set of RDF triples. Those rules have two main parts (1) a subject map, which generates the subject of all RDF triples that will be created from a logical table row; and (2) multiple predicate-object maps that consist of predicate maps and object maps (or referencing object maps). Triples are produced by combining the subject map with a predicate map and object map, and applying these three to each logical table row. It is possible that a triples map can contain graph maps that place some or all of the triples into named graphs, but by default, all RDF triples are in the default graph of the output dataset.

Next, we present a basic example in order to illustrate how to specify R2RML mappings[7]. Let us consider the database depicted in Figure 3. This database contains a table, one primary key, two columns, and one row.

Student

ID (PK) INTEGER	Name VARCHAR(50)
10	Venus

Fig. 3. Example database

For the that database we have the following R2RML mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

<TriplesMap1> a rr:TriplesMap;
rr:logicalTable [ rr:tableName "Student" ];
rr:subjectMap [ rr:class foaf:Person;
      rr:template "http://example.com/Student/{ID}/{Name}"; ];
rr:predicateObjectMap [ rr:predicate      ex:id ;
      rr:objectMap  [ rr:column "ID";  ] ];
rr:predicateObjectMap [  rr:predicate  foaf:name ;
      rr:objectMap  [ rr:column "Name" ] ].
```

Finally, it is worth mentioning that neither the RDB2RDF Use Cases and Requirements [8] nor the R2RML mappings in the R2RML Test Cases document[9] provide or say anything about generating ontologies from database content.

[7] Please refer to the R2RML specification and its testcases to check a detailed list of R2RML mapping examples.

[8] http://www.w3.org/TR/rdb2rdf-ucr/

[9] http://www.w3.org/2001/sw/rdb2rdf/test-cases/

3 Combining PR-NORs and R2RML mappings

In this section we present how to generate ontologies from database content by using PR-NORs and R2RML mappings. Our combination of PR-NORs and R2MRL mappings is two-fold, first (1) we want to specialize the PR-NORs that perform a TBox transformation by specifying Relational databases as NOR, (2) we want to show that it is possible to generate ontology schema triples using R2RML mappings.

Using R2RML mappings for transforming PR-NOR patterns brings two benefits. The first benefit is since R2RML mappings are expressed in RDF, we can store and reuse them. The second is, there are already several R2RML engines[10], therefore, it will be possible to execute the mappings and generate the ontologies in a short time.

Using our approach, each PR-NOR has the corresponding R2RML mapping. The mappings will be executed by an R2RML engine and the result of that execution will generate an ontology represented as an RDF document.

All the patterns, mappings, and some other files (sql dump, result in graphical representation etc) are available here at:

- Pattern Description : mappingpedia:pattern/SchemaPatterns/{PatternID}
- Mapping Algorithm : mappingpedia:pattern/SchemaPatterns/{PatternID}/algorithm.txt
- Mapping Example : mappingpedia:pattern/SchemaPatterns/{PatternID}/mapping-example.ttl
- SQL General : mappingpedia:pattern/SchemaPatterns/{PatternID}/sql-general.sql
- SQL Example : mappingpedia:pattern/SchemaPatterns/{PatternID}/sql-example.sql
- RDF Result Example : mappingpedia:pattern/SchemaPatterns/{PatternID}/result-example.nt

Where mappingpedia represents `http://mappingpedia.linkeddata.es/`

Next, we show examples of how to build R2RML mappings that correspond to the PN-NORs patterns. We use snowflake model as the representation of classification schema pattern and term-based relational model as the representation of thesaurus pattern.

PR-NOR-CLTX-03 : Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema A classification scheme is a rooted tree of terms, in which each term groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parent and children terms may vary depending on the context. The snowflake data model [8] is a normalized structure for hierarchy representations. In this case, the classification scheme items are grouped by levels or entities. There are as many groups as levels the classification scheme has. Snowflake models are widely used on data warehouses to build hierarchical classifications on structures known as dimensions. Some examples of dimension are Time, Product Category, Geography, Occupations, etc. An example of snowflake data model can be seen in Figure. 4. In this pattern the example is an occupation hierarchical

[10] `http://www.w3.org/2001/sw/rdb2rdf/implementation-report/`

Professione0

ID0	Desc0
01	Professioni specialistiche e tecniche
02	Professioni operative della gestione dimpresa

Professione1

ID1	Desc1	ID0
01.05	Specialist e tecnici delle scienze informatiche	01
02.05	Specialist e tecnici delle gestione dimpresa	02

Professione2

ID2	Desc2	ID1
01.05.01	Specialist delle scienze informatiche	01.05
01.05.02	Tecnici delle scienze informatiche	01.05
02.05.01	Specialist delle gestione dimpresa	02.05
02.05.02	Tecnici delle gestione dimpresa	02.05

Fig. 4. An example of Snowflake Data Model

classification hold on three different tables, one for each level (PROFESSIONI_0, PROFESSIONI_1, PROFESSIONI_2).

The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [11]. Each term in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent terms are made explicit by using an external resource. Figure. 5 illustrates the generated ontology from the example. Note that although the transformation copies the

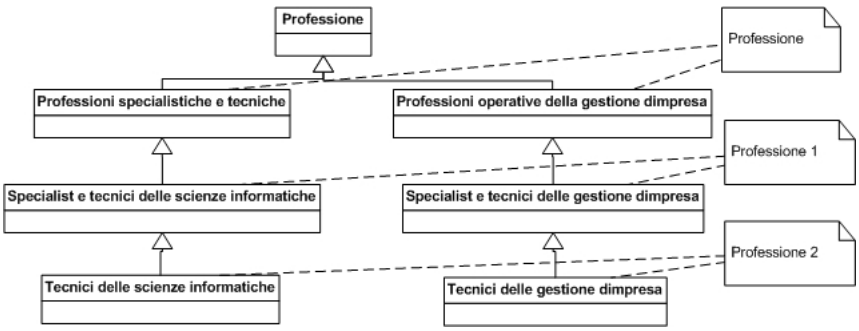

Fig. 5. Generated ontology from the example of Snowflake Data Model

the hierarchy expressed by the database content, the resulting ontologies consists only the schema (collection of classes and their labels) without their individuals. Hence, this is a T-Box transformation as we discussed in Section. 1.

Next we present the procedure that generates R2RML mapping corresponding to the pattern. First, create an rr:TriplesMap instance for every table with the table name as its rr:logicalTable value. Then create an rr:SubjectMap instance for the TriplesMap with rdfs:Class as its rr:class and concatenation of namespace base and primary key of the table as its rr:template value. Additionaly, the user may provide an rr:PredicateObjectMap instance that specifies the name of the class. If the table doesn't have a foreign key, it means that the table is mapped into the root class of the ontology. Otherwise, create an rr:PredicateObjectMap instance with rr:objectMap that joins the foreign key with the referenced primary key.

Input: The tables of database *tables*, the URI of the target class *classURI*
Output: R2RML Mapping Document MD

1: $MD \leftarrow$ createMappingDocument()
2: **for** $t \in tables$ **do**
3: $TM \leftarrow$ createTriplesMap()
4: $TM.logicalTable.tableName \leftarrow$ t.tableName
5: $TM.subjectMap.class \leftarrow$ rdfs:Class
6: $TM.subjectMap.template \leftarrow$ CONCAT(classURI, table.PKColumn)
7: $TM.predicateObjectMap[0].predicate \leftarrow$ rdfs:label
8: $TM.predicateObjectMap[0].objectMap.column \leftarrow$ t.LabelColumn
9: **if** $t.PK = \{\}$ **then**
10: $TM.predicateObjectMap[1].predicate \leftarrow$ rdfs:subClassOf
11: $TM.predicateObjectMap[1].objectMap.constant \leftarrow$ classURI
12: **else**
13: $TM.predicateObjectMap[1].predicate \leftarrow$ rdfs:subClassOf
14: $TM.predicateObjectMap[1].objectMap.constant \leftarrow$ classURI
15: **end if**
16: PUT(MD, TM)
17: **end for**
18: **return** MD

Listing 1.1. R2RML mappings corresponding to the example of Snowflake Data Model

```
<TriplesMapProfessione0> a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "Professione0"];
    rr:subjectMap [ rr:class rdfs:Class; rr:termType rr:IRI;
        rr:template
            "http://example.org/resource/Professione{id0}"; ];
    rr:predicateObjectMap [rr:predicate rdfs:label;
        rr:objectMap [rr:column "desc0"];];
    rr:predicateObjectMap [ rr:predicate rdfs:subClassOf;
        rr:objectMap [ rr:constant
            "http://example.org/resource/Professione"]; ];.

<TriplesMapProfessione1> a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "Professione1"];
    rr:subjectMap [ rr:class rdfs:Class; rr:termType rr:IRI;
        rr:template
            "http://example.org/resource/Professione{id1}"; ];
    rr:predicateObjectMap [ rr:predicate rdfs:label;
        rr:objectMap [rr:column "desc1"]; ];
    rr:predicateObjectMap [rr:predicate rdfs:subClassOf;
        rr:objectMap [rr:termType rr:IRI;
            rr:parentTriplesMap <TriplesMapProfessione0>;
                rr:joinCondition [
                    rr:child "id0" ;rr:parent "id0" ;]; ]; ];.

<TriplesMapProfessione2> a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "Professione2"];
    rr:subjectMap [rr:class rdfs:Class; rr:termType rr:IRI;
        rr:template
```

```
        "http://example.org/resource/Professione{id2}"; ];
    rr:predicateObjectMap [ rr:predicate rdfs:label;
        rr:objectMap [rr:column "desc2"]; ];
rr:predicateObjectMap [rr:predicate rdfs:subClassOf;
    rr:objectMap [rr:termType rr:IRI;
        rr:parentTriplesMap <TriplesMapProfessione1>;
            rr:joinCondition [
                rr:child "id1" ; rr:parent "id1" ;]; ]; ];.
```

PR-NOR-TSTX-02 : Pattern for re-engineering a thesaurus following the relation-based data model into an ontology schema A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them. The relation-based data model [10] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.

As an example, the AGROVOC Thesaurus is an structured and controlled vocabulary designed to cover the terminology of all subject fields in agriculture, forestry, fisheries, food and related domains. This thesaurus is available at `http://www.fao.org/agrovoc/`. See Figure. 6 for the graphical representation of the thesaurus.

agrovocterm

TermCode	Term
1328	Paddy
1474	Cereals
3354	Poaceae
5435	Oryza
6599	Rice

termlink

TermCode1	TermCode2	LinkTypeID
5435	6599	90
5435	3354	50
6599	5435	90
6599	1474	50
6599	1328	20

linktype

LinkTypeID	LinkDesc	LinkAbr
50	Broader Term	BT
90	Related Term	RT
60	Narrower Term	NT
20	Used For	UF

Fig. 6. An example of Thesaurus Relational-based Data Model

The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01)[11]. Each thesaurus term is mapped to a class. For the disambiguation of the semantics of the BT, NT, RT and UF relations among thesaurus terms the pattern relies on an external resource. In our case, the semantics of the BT, NT, RT and UF relations are encoded in rr:sqlQuery of the R2RML mappings. Figure. 7 illustrates the generated ontology from the example.

Next, we present the procedure to generate R2RML mappings corresponding to this pattern. First, create an rr:TriplesMap instance whose rr:logicalTable is a view of join result of Terminology table, Terms Relationship table, and Relationship Type table. We let the user decide whether the join type is INNER or LEFT OUTER. In the view, specify also the translation from Relationship Type table values into skos properties using SQL CASE. Then, create an instance of

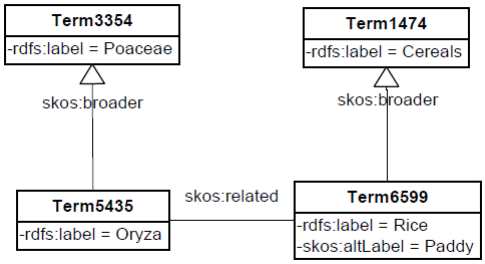

Fig. 7. The generated ontology from the example of Thesaurus Relational-based Data Model

rr:SubjectMap with rdfs:Class as its rr:class. The URI values is specified through rr:template as the concatenation of Term class namespace with the value of the table's primary key. Additionaly, an rr:PredicateObjectMap instance can be provided to specify the class name. Then, create another rr:PredicateObjectMap instance that maps a property to the the relationship between one term with another specified in the SQL CASE value provided in the view.

Input: Table Term $TblTerm$, Table Relationship $TblRel$, Table Relationship type $TblRelType$, Term Class URI $classURI$
Output: R2RML Mapping Document MD

1: $MD \leftarrow$ createMappingDocument()
2: $TM \leftarrow$ createTriplesMap()
3: $V = TblTerm \bowtie TblRel \bowtie TblRelType$
4: $TM.logicalTable.sqlQuery \leftarrow V$
5: $TM.subjectMap.class \leftarrow rdfs:Class$
6: $TM.subjectMap.template \leftarrow$ CONCAT(classURI, TblRel.TermCode1)
7: $TM.predicateObjectMap[0].predicate \leftarrow rdfs:label$
8: $TM.predicateObjectMap[0].objectMap.column \leftarrow TblTerm.Term$
9: $TM.predicateObjectMap[1].predicateMap.column \leftarrow linkURIObject$
10: $TM.predicateObjectMap[1].objectMap.template \leftarrow$
$CONCAT(classURI, TblRel.TermCode2)$
11: $TM.predicateObjectMap[2].predicateMap.column \leftarrow linkURIData$
12: $TM.predicateObjectMap[2].objectMap.template \leftarrow$
$CONCAT(classURI, TblRel.Term)$
13: PUT(MD, TM)
14: **return** MD

Listing 1.2. R2RML mappings corresponding to the example of Thesaurus Relation-based Data Model

```
<TriplesMapTerm> a rr:TriplesMap;
    rr:logicalTable [ rr:sqlQuery """
    SELECT t.TermCode, t.Term, tl.TermCode2
        , lt.LinkDesc , lt.LinkTypeID
        , CASE lt.LinkAbr
            WHEN 'BT' THEN 'skos:broader'
            WHEN 'NT' THEN 'skos:narrower'
            WHEN 'RT' THEN 'skos:related'
        END AS linkURIObject
        , CASE lt.LinkAbr
            WHEN 'UF' THEN 'skos:altLabel'
        END AS linkURIData
    FROM agrovocterm t
    LEFT OUTER JOIN termlink tl
```

```
    ON t.TermCode = tl.TermCode1
  LEFT OUTER JOIN linktype lt
    ON tl.LinkTypeID = lt.LinkTypeID
  """];

  rr:subjectMap [rr:class rdfs:Class;rr:termType rr:IRI;
    rr:template
      "http://example.org/resource/Term{TermCode}";];

  rr:predicateObjectMap [rr:predicate rdfs:label;
    rr:objectMap [rr:column "Term"];];

  rr:predicateObjectMap [rr:predicateMap [
    rr:column "linkURIObject"; rr:termType rr:IRI];
      rr:objectMap [ rr:template
      "http://example.org/resource/Term{TermCode2}"]; ];

  rr:predicateObjectMap [rr:predicateMap [
    rr:column "linkURIData"; rr:termType rr:Literal];
      rr:objectMap [ rr:template
      "http://example.org/resource/Term{Term}"]; ];
.
```

We have seen two examples of using R2RML mappings in order to generate ontologies from PR-NORs. There are other approaches for this goal, although our approach is better for several reasons. For example, using other RDB2RDF languages instead of R2RML, such as R2O [1] or D2RQ [3] can be employed. Unlike R2RML, both R2O and D2RQ do not permit the use of arbitrary SQL queries as the logical table, which can be useful for joining multiple tables which some complex conditions. Using a standard mapping language also bring benefits on the practical side, as multiple implementations are available. Even D2R system, which initially implemented as D2RQ engine, will also give support to R2RML. Other possible approach is not to use R2RML or any RDB2RDF mapping languages, but using ad-hoc approach like creating a custom program for each of the pattern. However, this approach can be considered inferior as the the reusability aspect is lower than reusing mappings, not to mention the time has to be invested to create the custom program instead of just choosing one of the available R2RML implementations, such as Morph[11].

4 Conclusions and Future Work

In this paper we have presented an approach that combines PR-NORs with R2RML mappings for building ontologies from relational database content, i.e., transforming the database content into an ontology schema by using Re-engineering Patterns and R2RML mappings. Furthermore, we have seen that besides for generating data-triples (triples that describes instances of an ontology), R2RML mappings are useful also to generate schema-triples (triples that describe the schema of an ontology).

In the future, we will continue identifying other Re-engineering patterns that may take benefit from R2RML mappings. We will explore the possibility to combine R2RML mappings with other data-source type, such as XML, CSV or

[11] https://github.com/jpcik/morph

spreadsheets. Because R2RML mappings are RDF documents, it is possible to store those mappings in a triple store. Once those mappings have been stored in the triple store and annotated with meta-data properties, users can pose query the triple store in order to get all mappings correspond to a specific pattern. For example, user can pose a SPARQL query `SELECT * FROM {?m a mapping. m hasPattern ?p. FILTER REGEX(?p, "Snowflake")}`, and the answers will be all mappings corresponding to the snowflake data model pattern. The system is currently under development and being populated with mappings and patterns and we plan to report this system as future work.

Acknowledgments: This work has been supported by the PlanetData (FP7-257641), BabelData (TIN2010-17550), and myBigData (TIN2010-17060) projects. We would like to kindly thank all W3C RDB2RDF Working Group members.

References

1. J. Barrasa, Ó. Corcho, and A. Gómez-pérez. R2o, an extensible and semantically based database-to-ontology mapping language. In *in In Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB2004*, 2004.
2. D. Beckett and T. Berners-Lee. Turtle-terse rdf triple language. *Syntax*, 28(January), 2008.
3. C. Bizer and A. Seaborne. D2rq-treating non-rdf databases as virtual rdf graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, page 26, 2004.
4. C. Caracciolo, J. Heguiabehere, V. Presutti, and A. Gangemi. Initial network of fisheries ontologies. Technical report, NeOn project deliverable D7.2.3, 2009.
5. C. Date. *An introduction to database systems*, volume 2. Addison Wesley Publishing Company, 1983.
6. B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, May 2007.
7. M. Hert, G. Reif, and H. Gall. A comparison of rdb-to-rdf mapping languages. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 25–32. ACM, 2011.
8. E. Malinowski and E. Zim·nyi. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data and Knowledge Engineering*, 2006.
9. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau Jr, S. Auer, J. Sequeda, and A. Ezzat. A survey of current approaches for mapping of relational databases to rdf. *W3C RDB2RDF XG Incubator Report*, page W3C, 2009.
10. D. Soergel. Data models for an integrated thesaurus database. *Comatibility and Integration of Order Systems*, 24(3):47–57, 1995.
11. M. C. Suárez-Figueroa, S. Brockmans, A. Gangemi, A. Gómez-Pérez, J. Lehmann, H. Lewen, V. Presutti, and M. Sabou. Neon modelling components. Technical report, NeOn project deliverable D5.1.1, 2007.
12. B. Villazón-Terrazas. *A Method for Reusing and Re-engineering Non-ontological Resources for Building Ontologies*. IOS Press, 2012.