morph-LDP: An R2RML-based Linked Data Platform implementation

Nandana Mihindukulasooriya[1,2], Freddy Priyatna[2], Oscar Corcho[2], Raúl García-Castro[1,2], and Miguel Esteban-Gutiérrez[1,2]

[1] Center for Open Middleware
[2] Ontology Engineering Group, Facultad de Informática
Universidad Politécnica de Madrid, Spain
`{nmihindu,fpriyatna,ocorcho,rgarcia,mesteban}@fi.upm.es`

Abstract. The W3C Linked Data Platform (LDP) candidate recommendation defines a standard HTTP-based protocol for read/write Linked Data. The W3C R2RML recommendation defines a language to map relational databases (RDBs) and RDF. This paper presents morph-LDP, a novel system that combines these two W3C standardization initiatives to expose relational data as read/write Linked Data for LDP-aware applications, whilst allowing legacy applications to continue using their relational databases.

1 Introduction

The W3C Linked Data Platform (LDP) candidate recommendation is focused on supporting read/write Linked Data by providing a standard HTTP-based RESTful protocol. An obvious way of generating an LDP implementation is by storing RDF data in a triple store and internally doing SPARQL SELECT and UPDATE queries. However, there are contexts in which organizations are interested in continuing with their existing relational databases instead of transforming data into an RDF format and managing it from a triple store. Hence, the work that has been done in the past at W3C on the definition of the R2RML language, as well as the implementations available that support such language (e.g., Ontop [1], morph-RDB [2], D2R [3]), may be useful as an alternative implementation for LDP, as shown in Figure 1.

In this demo we show how we can make use of the W3C R2RML recommendation as the underlying support for providing read/write Linked Data access to relational databases. Database administrators only need to generate a R2RML mapping document and morph-LDP exposes the relational data as Linked Data. Such Linked Data is not only dereferenceable and available through the HTTP GET operation, but can also be updated using write operations such as PUT, POST, PATCH, and DELETE.

2 Background

The Linked Data Platform (LDP)[4] is an initiative by the W3C LDP WG to provide a standard protocol and a set of best practices enabling read/write

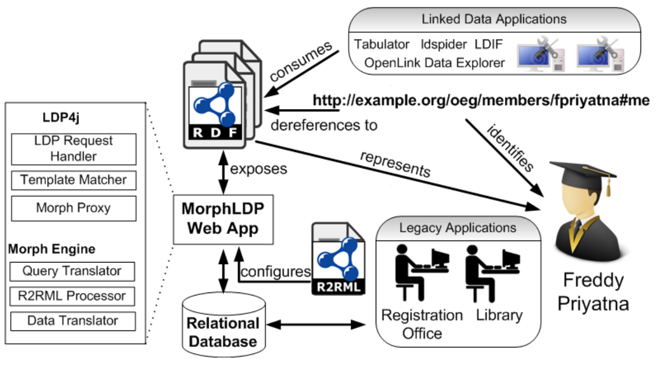


Fig. 1. The morph-LDP use case

Linked Data scenarios. The LDP specification extends the HTTP protocol with a set of new constraints, HTTP headers, and Link headers that are useful for read/write Linked Data applications. The two main types of concepts defined in LDP include:

– A Linked Data Platform Resource (LDPR) which is any HTTP resource that conforms to additional constraints defined in the LDP specification.
– A Linked Data Platform Container (LDPC) which is a specialization of an LDPR that acts as a collection resource that helps organizing LDPRs and creating new LDPRs as its members.

R2RML [5] is a W3C recommendation for the definition of a mapping language from relational databases to RDF. An R2RML mapping document contains a set of TriplesMap, used to specify the rules to generate RDF triples from database rows/values. A TriplesMap consists of a LogicalTable (either a base table or a SQL view), a SubjectMap (to generate the subject component of RDF triples), and a set of PredicateObjectMaps. A PredicateObjectMap is composed of a set of PredicateMaps and ObjectMaps (to generate the predicate and object components of RDF triples, respectively).

3 Related Works

There is some previous work on the provision of update functionalities to R2RML-based systems (e.g., [6]). However, so far Linked Data generated by such an approach cannot be dereferenced using HTTP GET but rather have to be queried through a SPARQL endpoint. This hinders the usage of the follow-your-nose approach[3] to traverse links found in Linked Data because it requires out-of-band information such as the SPARQL endpoint in addition to the link URI. In contrast, Linked Data generated by morph-LDP can be dereferenced and

[3] `http://patterns.dataincubator.org/book/follow-your-nose.html`

updated using URIs conforming to the HTTP-based LDP protocol. Moreover, morph-LDP does not require the clients to be SPARQL-aware, hence making the morph-LDP approach more attractive to those Web developers who are new to Semantic Web technologies. Finally, the usage of morph-RDB [2] underneath makes the query evaluation process more efficient.

Another approach consists in mapping relational data to Web resources following REST principles and making those resources accessible via the HTTP protocol. Some systems that follow this approach are: HTTP database connector (HDBC) [7], sqlREST[4], and restSQL[5]. Nevertheless, none of those systems provides RDF representations. HDBC generates data according to the Atom format, while both sqlREST and restSQL generate XML outputs using W3C XLink and custom SQL Resources. While following the same approach, morph-LDP provides RDF representations (Turtle, RDF/XML, N-Triples and JSON-LD utilizing HTTP content negotiation). In addition, another advantage of morph-LDP is the use of a standard mapping language.

4 morph-LDP

4.1 Mapping LDP components with R2RML

In this section, we explain how the concepts of LDP are mapped to R2RML so that the two standards can be combined to provide read/write Linked Data access to relational data.

The second Linked Data principle[6] mandates the use of HTTP URIs so that people can look up those names. By inspecting the rules specified in SubjectMaps, these URIs can be mapped to Linked Data resources (LDPR) enabling read/write access through the LDP protocol.

PredicateObjectMaps allow generating the information about a specific resource, so that *when someone looks up a URI*, morph-LDP can *provide useful information using RDF* following the third Linked Data principle. LDP not only allows to lookup those resources but also to update them using HTTP write operations.

In R2RML, TriplesMaps are used to generate RDF triples out of the rows of logical tables. In LDP, LDP Containers are used as collection resources to organize LDPRs. Thus logical tables and their corresponding TripleMaps can be mapped to LDP Containers.

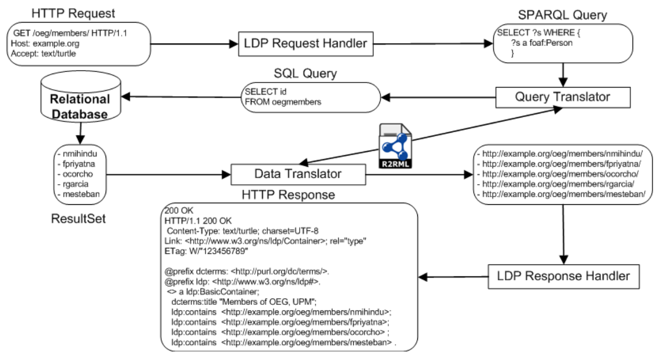
4.2 Implementation

Our implementation[7] is based on the result of two existing projects: **morph-RDB** and **LDP4j**. morph-RDB[2][8] is a Scala-based RDB2RDF engine that

[4] http://sqlrest.sourceforge.net/
[5] http://restsql.org/
[6] http://www.w3.org/DesignIssues/LinkedData.html
[7] https://github.com/fpriyatna/morph-LDP
[8] https://github.com/fpriyatna/morph

complies with the R2RML specification. Its query translator component is based on the algorithm defined in [8] with some optimizations, such as self-join, left-outer join, and subquery elimination. morph-RDB has been successfully applied in several projects: Répener [9], BizkaiSense[9], and Integrate[10]. LDP4j[11] is a Java-based middleware for the development of LDP-aware applications. LDP4j is being developed in the context of the ALM iStack project, where the LDP middleware is used to integrate Application Lifecycle Management tools [10].

morph-LDP extends morph-RDB with an LDP layer provided by LDP4j. The LDP layer extracts the metadata from the HTTP request and send it as an input for morph-RDB. morph-RDB has two modes of operation: API mode and SPARQL query mode and handles the transformation between RDF and relational data. Figure 4.2 illustrates the process when morph-LDP receives an HTTP Request when using the SPARQL query mode.



```
GET /oeg/members/ HTTP/1.1
Host: example.org
Accept: text/turtle
```

```
SELECT ?s WHERE {
    ?s a foaf:Person
    }
```

```
SELECT id
FROM oegmembers
```

```
- nmihindu
- fpriyatna
- ocorcho
- rgarcia
- mesteban
```

```
- http://example.org/oeg/members/nmihindu/
- http://example.org/oeg/members/fpriyatna/
- http://example.org/oeg/members/ocorcho/
- http://example.org/oeg/members/rgarcia/
- http://example.org/oeg/members/mesteban/
```

```
200 OK
HTTP/1.1 200 OK
 Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp/Container>; rel="type"
ETag: W/"123456789"

@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix ldp: <http://www.w3.org/ns/ldp#>.
 <> a ldp:BasicContainer;
   dcterms:title "Members of OEG, UPM";
   ldp:contains  <http://example.org/oeg/members/nmihindu>;
   ldp:contains  <http://example.org/oeg/members/fpriyatna>;
   ldp:contains  <http://example.org/oeg/members/ocorcho> ;
   ldp:contains  <http://example.org/oeg/members/mesteban> .
```

Fig. 2. An example of morph-LDP in action, using the SPARQL query mode.

5 An Example Scenario

Our demo scenario uses a relational database which manages information of members of a research group. morph-LDP exposes this data as Linked Data that can be managed using the LDP protocol. In the demo, we showcase the following user stories:

As a Linked Data application developer, I want to:

– retrieve the list of members of the research group (retrieve an LDP Container).
– retrieve details of a certain member of the group (retrieve an LDPR).
– update the details of a certain member of the group (update an LDPR).
– create a new member record of the group (create a new LDPR).

[9] http://www.tecnologico.deusto.es/projects/bizkaisense/
[10] http://www.fp7-integrate.eu
[11] http://ldp4j.org

– delete an outgoing member record of the group (delete a LDPR).

A video of the demo, together with the HTTP Requests/Responses, and SPARQL/SQL queries, can be found in the morph-LDP page[12].

6 Conclusion

W3C LDP is in the final stages of the standardization process and we believe that integrating it with relational databases using W3C R2RML will help a wider adoption in the industry. In this paper, we presented how LDP and R2RML can be combined to expose relational data as read/write Linked Data. Nevertheless, there is still some work to be done on providing support for Quality-of-Service requirements such as secure access and transactions. In addition to that, currently morph-LDP supports only simple R2RML mappings (no SQL view, no multiple mappings to a class/property) and we are working on giving support for more complex mappings.

Acknowledgments: The authors have been supported by the PlanetData (FP7-257641) , myBigData (TIN2010-17060), and ALM iStack (Center for Open Middleware) projects.

References

1. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontop at work. In: Proc. of the 10th OWL: Experiences and Directions Workshop (OWLED 2013). (2013)
2. Priyatna, F., Corcho, O., Sequeda, J.F.: Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. In: Proceedings of the 23rd International World Wide Web Conference. (2014)
3. Bizer, C., Cyganiak, R.: D2R server-publishing relational databases on the semantic web. In: Poster at the 5th International Semantic Web Conference. (2006)
4. Speicher, S., Arwe, J., Malhotra, A.: Linked Data Platform 1.0 (2014) W3C Last Call Draft, `http://www.w3.org/TR/ldp/`.
5. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language W3C Recommendation, `http://http://www.w3.org/TR/r2rml/`.
6. Garrote, A., García, M.: Restful writable APIs for the web of linked data using relational storage solutions. In: WWW 2011 Workshop: Linked Data on the Web (LDOW2011). (2011)
7. Marinos, A., Wilde, E., Lu, J.: HTTP database connector (HDBC): RESTful access to relational databases. In: Proceedings of the 19th international conference on World wide web, ACM (2010) 1157–1158
8. Chebotko, A., Lu, S., Fotouhi, F.: Semantics preserving SPARQL-to-SQL translation. Data & Knowledge Engineering **68**(10) (2009) 973–1000
9. Sicilia, A., Nemirovskij, G., Massetti, M., Madrazo, L.: RÉPENER's linked dataset (in revision). Semantic Web Journal (2013)
10. Mihindukulasooriya, N., García-Castro, R., Esteban-Gutiérrez, M.: Linked Data Platform as a novel approach for Enterprise Application Integration. (2013)

[12] http://oeg-dev.dia.fi.upm.es/morph-ldp/