RDF Mapping Rules Refinements according to Data Consumers' Feedback

Anastasia Dimou
anastasia.dimou@ugent.be

Miel Vander Sande
miel.vandersande@ugent.be

Tom De Nies
tom.denies@ugent.be

Erik Mannens
erik.mannens@ugent.be

Rik Van de Walle
rik.vandewalle@ugent.be

Ghent University – iMinds – Multimedia Lab
Gaston Crommenlaan 8/201
9050 Ghent, Belgium

ABSTRACT

The missing feedback loop is considered the reason for broken Data Cycles on current Linked Open Data ecosystems. Read-Write platforms are proposed, but they are restricted to capture modifications after the data is released as Linked Data. Triggering though a new iteration results in loosing the data consumers' modifications, as a new version of the source data is mapped, overwriting the currently published. We propose a prime solution that interprets the data consumers' feedback to update the mapping rules. This way, data publishers initiate a new iteration of the Data Cycle considering the data consumers' feedback when they map a new version of the published data.

1. INTRODUCTION

Nowadays, more and more data owners map their data to a corresponding RDF representation before they publish and interlink them with other Linked Data. After the data is published, the data owners or other data consumers modify the published datasets. While this is in compliance with the envisaged Read-Write Web and Government Linked Open Data life cycle[1] , Semantic Web compliant systems neither intuitively process data consumers' feedback, nor properly propagate the modifications over different serialisations of the original data. Most current solutions that publish Linked Data from other source formats focus on providing the RDF data model, dissociating the resulting representation from its original source. As a result, modifications over the triplified data are not propagated to the mapping rules and to the source data, as their RDF representation is handled independently of their source. Thus they are overwritten every time a new version of the source data is mapped to RDF.

We propose a solution that considers the data consumers feedback to refine the mapping rules. Provenance information is used to identify the mapping rules that generated the triples, and thus the data in their original serialisation. The modifications over the triplified data are captured as they are translated to refinements over the corresponding mappings definitions. This way, the next time a new version of the same data is mapped, namely a new Data Cycle is initiated, the modifications applied to the RDF representation of the data remain unaffected.

2. INCORPORATING USER FEEDBACK

In our approach, the mapping rules from different source files to their RDF representation are defined using the RDF Mapping Language (RML, [1]). RML is a generic language for mapping heterogeneous structured data formats to RDF[2]. An RML mapping consists of several Triples Maps. A *Triples Map* is a rule that defines a number of RDF triples to be generated from a source file and is identified by a URI. We output all the triples generated by a Triples Map as *quads* [3], extending the triple with an extra context element. This allows us to uniquely identify each triple, by adding a URI to it. With PROV[4], we describe the relation between each generated triple and its generating Triples Map, and thus the original file defined in its *Logical Source*. In the following example, the *GeneratedTriple* (3) is an entity generated from a non-RDF source file (7), mapped according to the *MappingProcedure* (4) and executed by the *dataOwner* (5). The *MappingProcedure* refers to the corresponding *TriplesMap*. The source file (12) is specified at the *Logical source* of the *Triples Map*.

```
1  @prefix prov:  <http://www.w3.org/ns/prov#>.
2  @prefix :  <http://example.com/>.
3  :GeneratedTriple a prov:Entity;
4      prov:wasGeneratedBy :MappingProcedure;
5      prov:wasAttributedTo :dataOwner;
6      prov:hadPrimarySource
7        <http://example.org/sourceData.xml>.
8  :MappingProcedure a prov:Activity;
9      prov:wasAssociatedWith :dataOwner;
10     prov:used   <http://example.org/sourceData.xml>;
11     prov:generated :GeneratedTriple.
12 <http://example.org/sourceData.xml> a prov:Entity.
13 :dataOwner a prov:Agent.
```

Next, R&Wbase [2] can track all the changes made to the generated triples. R&Wbase is an interpretation layer built on top of SPARQL endpoints and follows the principles of distributed version control. It stores different deltas in a single graph, allowing simple resolving of different versions and separate access. Every time a triple is modified, we can extract the mapping definition that generated it from

[1] https://dvcs.w3.org/hg/gld/raw-file/default/bp/index.html

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2577332.

[2] http://semweb.mmlab.be/rml/spec.html
[3] http://www.w3.org/TR/n-quads/
[4] http://www.w3.org/TR/prov-o/

the provenance information. A modification may refer to the data context (e.g., a modification over a data value), or it may refer to the semantics of the data. The former requires the data corrections to be propagated. The latter updates the corresponding *Triples Map* to confront with the modifications in the data's RDF representation. Thus, the Triples Maps are *refined* according to the modifications. The next time an updated version of the source data is mapped, these refinements are kept into account and are applied to the updated data as to the already published ones.

3. MAPPING RULES REFINEMENTS

Consider the following *source file*(1–8), its *Triples Maps* (10–22) and the initially generated output(24–25).

```
1  <Dep>...<Person>
2    <Name>Bill</Name>
3    <Surname>Gates</Name>
4    <Work>
5      <company>Microsoft<company>
6      <turnover>10000</turnover>
7    </Work>
8   </Person>...</Dep>
9
10 @prefix rr:  <http://www.w3.org/ns/r2rml#>.
11 @prefix rml:  <http://semweb.mmlab.be/rml#>.
12 @prefix ex:  <http://ex.com/>.
13 <#PersonMap>
14   rr:subjectMap [
15   rr:template "http://ex.com/{/Dep/Person/Name}"
16   rr:class ex:Person ].
17 <#Company> rr:subjectMap [
18   rr:template "http://ex.com/{/Dep/Person/Work/company}"]
19     rr:predicateObjectMap [
20       rr:predicate ex:profit;
21       rr:objectMap [ rml:reference
22         "/Dep/Person/Work/turnover" ]].
23
24 <http://ex.com/Bill> a ex:Person .
25 ex:Microsoft ex:profit "100000" .
```

The following ways to submit feedback and their interpretation to mapping rules' refinements are examined.

1. Update an RDF Term. RML *Term Maps* are used to generate the RDF *Terms*, namely the subjects, predicates, objects and graphs. A data consumer may modify the term's URI. In this case, the corresponding *template-valued term map* is updated to include the modification. For example if the subject at (24) is changed to `<http://ex.be/Bill>`, the *rr:template* of (15) is updated respectively to `[] rr:template "http://ex.be/{/Dep/Person/Name}"`. Alternatively, the data consumer may change the RDF Term to a constant value, e.g., `ex:SoftwareCompany`. Then, the *template-valued term map* is modified to a *constant-valued term map*; e.g. (18) is updated to `[] rr:constant ex:SoftwareCompany`. Finally, the unique identifier of the URI may be modified; e.g., the surname is prefered and (24) becomes `<http://ex.com/Gates>`. The element containing the value is identified at the source file and replaces the existing one; (15) becomes `"http://ex.com/{/Dep/Person/Surname}"`.

2. Update the class of an instance. A data consumer can modify the class of an instance. In this case, the property *rr:class* of the corresponding *Subject Map* is updated. For example, the data consumer modifies (24) to `ex:Bill a foaf:Person`. The corresponding *Subject Map* (16) is modified to `[] rr:class foaf:Person`. If an additional class is added besides the existing one, the *Subject Map* will be adjust accordingly, e.g. `[] rr:class ex:Person , schema:Person`. The mapping rule is accordingly modified if the class annotation is removed by the data consumer.

3. Update a datatype property. A data consumer can add a new property that did not exist before and thus associate an existing individual with a certain value. A new *Predicate-Object Map* is added to the *Triples Map*. The referring element of the source file needs to be identified considering the modified triples. The element whose values fits accurately to the objects' value is chosen. For example, a data consumer adds the property `ex:income` that interlinks a *Person* with the turnover of his company, resulting in commiting the triple `ex:Bill ex:income "100000"`, the following *Predicate-Object Map* will be generated:

```
[] rr:predicateObjectMap [ rr:predicate ex:income;
  rr:objectMap [ rml:reference "/Dep/Person/Work/turnover"]]
```

If a datatype property is updated or removed, the *Predicate-Object Map* will be modified or removed, respectively.

4. New object property. The data consumer can add a new property that did not exist before and interlink two existing individuals. A *Predicate-Object Map* is again added to the *Triples Map*. If the object is generated based on a certain *Triples Map*, a *Referencing Object Map* is used, else, an *Object Map* with a *template* definition is added. For example, there are two unlinked individuals, `ex:Bill` and `ex:Microsoft`, and a data consumer adds the property `ex:owns` to interlink them. The new triple `ex:Bill ex:owns ex:Microsoft` is committed. The following *Predicate-Object Map* is added to the *Triples Map*:

```
[] rr:predicateObjectMap [ rr:predicate ex:owns;
    rr:objectMap [ rr:parentTriplesMap <#Company> ]]
```

Similarly, the *Predicate-Object Map* is updated or removed if a data consumer updates an existing link or deletes it.

5. Update datatype or language property. The data consumer can add a new datatype or language property. For example, he specifies the datatype, e.g., `ex:Microsoft ex:profit "100000"^^xsd:integer .` The *rr:datatype* property is added to the *Object Map* (21) as follows: `[] rml:reference "/Dep/Person/Work/turnover";rr:datatype xsd:integer.` If the data consumer updates the datatype or language annotation, the *rr:datatype* and *rr:language* property of the *Triples Map* is updated or removed respectively.

4. CONCLUSIONS AND FUTURE WORK

We presented a solution to incorporate each Data Cycle's feedback loop to its successive iterations. Data consumers' feedback is adjust to the mapping rules. Thus, we achieve to align intermediate mappings with the data consumers' feedback resulting in smooth iteration over consecutive cycles, as the execution of the mappings does not reset the interaction between the published Linked Data and the data consumers. In the future, a modification over the mapping rules will be proposed as a recommendation to the data publishers, considering the extent of a certain modification over the total number of the triples generated by a certain mapping rule.

5. REFERENCES

[1] A. Dimou, M. Vander Sande, P. Colpaert, E. Mannens, and R. Van de Walle. Extending R2RML to a source-independent mapping language for RDF. In *International Semantic Web Conference (Posters and Demos)*, 2013.

[2] M. Vander Sande, P. Colpaert, R. Verborgh, S. Coppens, E. Mannens, and R. Van de Walle. R&Wbase: git for triples. In *Workshop on Linked Data on the Web*, 2013.