Flexible RDF generation from RDF and heterogeneous data sources with SPARQL-Generate*

Maxime Lefrançois, Antoine Zimmermann, and Noorani Bakerally

Univ Lyon, MINES Saint-Étienne, CNRS, Laboratoire Hubert Curien UMR 5516,
F-42023 Saint-Étienne, France
firstname.lastname@emse.fr

Abstract. RDF aims at being the universal abstract data model for structured data on the Web. While there is effort to convert data in RDF, the vast majority of data available on the Web does not conform to RDF. Indeed, exposing data in RDF, either natively or through wrappers, can be very costly. In this context, transformation or mapping languages that define generation of RDF from non-RDF data represent an efficient solution. Furthermore, the declarative aspect of these solutions makes them easy to adapt to any change in the input data model, or in the output knowledge model. This paper introduces a novel such transformation language (SPARQL-Generate), an extension of SPARQL for querying not only RDF datasets but also documents in arbitrary formats. Its implementation on top of Apache Jena currently covers use cases from related work and more, and enables to query and transform web documents in XML, JSON, CSV, HTML, CBOR, and plain text with regular expressions.

Keywords: RDF, SPARQL, linked data, data transformation

1 Introduction

The vision of a Semantic Web where machines can more easily process web content is hampered by the coexistence of many, heterogeneous data formats and models available on the Web or *via* the Web. At first sight, this vision seems to require a worldwide adoption of a common universal data model as a document format (namely, RDF). Yet, in the emerging Web of Data, a multitude of formats are flourishing: XML (not RDF/XML) is still very present, open data portals tend to prefer CSV, web APIs rely more on JSON, and the Web of Things introduces new formats adapted to the resource constraints inherent to less powerful devices.

In this context, transformation or mapping languages that define generation of RDF from non-RDF data represent an efficient solution. Furthermore, the declarative aspect of these solutions makes them easy to adapt to any change in the input data model, or in the output knowledge model. In this paper, we use term *RDF lifting* to designate the process of generating RDF from non-RDF data.

* This paper has been partly funded by the ITEA2 12004 SEAS (Smart Energy Aware Systems) project, the ANR 14-CE24-0029 OpenSensingCity project, and a research contract with ENGIE R&D.

Section 2 first overviews existing initiatives and evaluates them with respect to a set of identified requirements. Then section 3 introduces SPARQL-Generate, a language that can be used to specify a mapping from distributed data sources in arbitrary data formats to the RDF data model. Section 4 concludes and identifies directions for future work.

2 Requirements for RDF Lifting Mechanisms

From the use cases faced by industrial partners in collaborative projects we are involved in, we identified the following set of requirements for a RDF lifting mechanisms. It should:

- be able to generate RDF from multiple sources in heterogeneous formats;
- be able to deal with text-based *and* binary representation formats;
- make it easy to combine non-RDF sources with RDF data;
- be extensible to account for new syntaxes;
- integrate seamlessly with existing standards for consuming Semantic Web data, such as SPARQL or Semantic Web programming frameworks.

Several systems exist for converting data to RDF. We are interested only in systems that provide a formal transformation language, potentially mapping-based.[1] Such languages include GRDDL [1], XSPARQL [5], R2RML [2], RML [3], and CSVW [6]. Table 1 overviews the comparison of these solutions with respect to the identified requirements. GRDDL and XSPARQL rely respectively on XSLT and XQuery, that have been proven to be Turing-complete. These languages are hence full-fledged procedural programming language with explicit algorithmic constructs to produce RDF. We argue that a procedural paradigm is less suited to semantic web engineers, who may be more familiar with a declarative paradigm such as SPARQL. Furthermore, only RML and XSPARQL are specifically dedicated to generate RDF from various formats, namely XML, CSV, HTML, and JSON. However, to our knowledge, they do not accept binary formats such as EXI, CBOR or BSON, which become of much importance in the emerging Web of Things.

3 SPARQL-Generate

SPARQL-Generate extends SPARQL 1.1 with only three new clauses, generate, source and iterator. It queries the combination of an RDF dataset and zero or more named documents. Each document is interpreted as a literal and is associated with an IRI, thus forming what we call a *literalset*. Below is an example of a SPARQL-Generate query,[2] more complex examples can be found online.[3]

[1] A lot of hardcoded transformation are available for many formats – `https://www.w3.org/wiki/ConverterToRdf`.

[2] Prefixes are omitted to save space.

[3] `http://w3id.org/sparql-generate/tests-reports.html`

Table 1: Features of related work compared to the SPARQL Generate language.

	GRDDL [1]	CSVW [6]	R2RML [2]	XSPARQL [5]	RML [3]	SPARQL-Generate
produces RDF	✓	✓	✓	✓	✓	✓
multiple source		✓	✓	✓	✓	✓
heterogeneous formats				✓	✓	✓
binary formats						✓
combines RDF data				✓	✓	✓
extensibility	XSLT			XQuery	*ad-hoc*	SPARQL

Document airport.csv

```
id,stop,latitude,longitude
6523,25,50.901389,4.484444
7000,40,56.901389,4.584444
```

Output (in turtle)

```
<http://ex.com/6523>
 a transit:Stop;
 transit:route 21;
 geo:lat 50.901389;
 geo:long 4.484444 .
```

SPARQL-Generate request

```
GENERATE {
  ?airport a transit:Stop;
        transit:route ?route;
        geo:lat ?lat;
        geo:long ?long . }
SOURCE <http://ex.com/airport.csv> AS ?source
ITERATOR iter:CSV(?source) AS ?busStop
WHERE {
    BIND( fn:CSV(?busStop, "id" ) AS ?id )
    BIND( xsd:int( fn:CSV(?busStop, "stop") ) AS ?route)
    BIND( fn:CSV(?busStop, "longitude" ) AS ?long )
    BIND( fn:CSV(?busStop, "latitude" ) AS ?lat )
    BIND (URI(CONCAT("http://ex.com/",?id)) AS ?airport)
    FILTER( ?route < 30 ) }
```

Implementation on top of Apache Jena SPARQL-Generate has been implemented on top of Apache Jena. It is available as open-source on GitHub.[4] It can be used as a Maven dependency, via a Web API, via a web form that itself uses the Web API, or as an executable jar. All of these tools may be found on the demonstration website:

`http://w3id.org/sparql-generate`

Supported data formats, and extensibility. Binding and iterator functions are available for the following formats: XML (exploiting XPath), CSV, TSV (conforming to the RFC 4180, or custom), HTML (exploiting CSS3 selectors), JSON and CBOR (exploiting JSONPath), and plain text (exploiting regular expressions). A complete documentation of the available binding and iterator functions is available at `http://w3id.org/sparql-generate/functions.html`.

The implementation relies on Jena's SPARQL binding function extension mechanism, and copies it for iterator functions. Therefore, covering a new data format in this implementation merely consists in implementing new binding and iterator functions in Jena. Even what is not covered by existing query languages can be implemented as an iterator function. For instance, iterator function `iter:JSONListKeys` iterates on key names of a JSON object, which is not feasible using JSONPath.

[4] `https://github.com/thesmartenergy/sparql-generate`

4 Future Work and Conclusion

Future plans consist of implementing more functions for more data formats, enabling on-the-fly function integration with an approach similar to [4], and adding some syntactic sugars that could strongly improve readability and conciseness of the queries. For instance one could use binding functions directly in the generate pattern, or use curly-bracket expressions instead of concatenating literals. Using such techniques, the example query from section 3 could be shortened as follows:

```
GENERATE {
  <http://ex.com/{?id}> a transit:Stop;
        transit:route ?route ;
        geo:lat xsd:decimal( fn:CSV(?busStop, "latitude" ) );
        geo:long xsd:decimal( fn:CSV(?busStop, "longitude" ) ). }
SOURCE <http://ex.com/airport.csv> AS ?source
ITERATOR iter:CSV(?source) AS ?busStop
WHERE {
    BIND( fn:CSV(?busStop, "id" ) AS ?id )
    BIND( xsd:int( fn:CSV(?busStop, "stop") ) as ?route )
    FILTER( ?route < 30 ) }
```

The problem of exploiting data from heterogeneous sources and formats is common on the Web, and Semantic Web technologies can help in this regard. We introduced SPARQL-Generate, that extends the SPARQL language in its ability to generate RDF graphs, such that any non-RDF data sources, as well as RDF sources, can be exploited to create an output RDF graph. Its syntax closely matches SPARQL with little additions, and hence combines the following advantages: (i) it may be implemented on existing SPARQL engines; (ii) it is modular since extensions to new formats do not require a redefinition of the language (thanks to the use of SPARQL custom functions); and (iii) it is easy to learn by Semantic Web specialists that know SPARQL 1.1. Our implementation on top of Apache Jena covers many use cases, as reported on the dedicated web site `http://w3id.org/sparql-generate`.

References

1. Dan Connolly. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). W3C Recommendation, 2007. `http://www.w3.org/TR/2007/REC-grddl-20070911/`.
2. Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF Mapping Language. W3C Recommendation, 2012. `http://www.w3.org/TR/2012/REC-r2rml-20120927/`.
3. Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Proceedings of the Workshop on Linked Data on the Web (LDOW 2014)*, 2014.
4. Maxime Lefrançois and Antoine Zimmermann. Supporting Arbitrary Custom Datatypes in RDF and SPARQL. In *Proceedings of the Extended Semantic Web Conference (ESWC 2016)*, 2016.
5. Axel Polleres, Thomas Krennwallner, Nuno Lopes, Jacek Kopecký, and Stephan Decker. XSPARQL Language Specification. W3C Member Submission. `http://www.w3.org/Submission/2009/SUBM-xsparql-language-specification-20090120/`.
6. Jeremy Tandy, Ivan Herman, and Greg Kellogg. Generating RDF from Tabular Data on the Web. W3C Recommendation. `http://www.w3.org/TR/2015/REC-csv2rdf-20151217/`.