HAL
open science

RDF and XML: Towards a unified query layer

Nuno Lopes, Axel Polleres, Alexandre Passant, Stefan Decker, Stefan Bischof, Diego Berrueta, Antonio Campos, Stéphane Corlosquet, Jérôme Euzenat, Orri Erling, et al.

To cite this version:

Nuno Lopes, Axel Polleres, Alexandre Passant, Stefan Decker, Stefan Bischof, et al.. RDF and XML: Towards a unified query layer. Proc. W3C workshop on RDF next steps, Jun 2010, Stanford, United States. No pagination. hal-00822901

HAL Id: hal-00822901

https://inria.hal.science/hal-00822901v1

Submitted on 15 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RDF and XML: Towards a Unified Query Layer

Nuno Lopes[1] Axel Polleres[1] Alexandre Passant[1] Stefan Decker[1]
Stefan Bischof[2] Diego Berrueta[3] Antonio Campos[3] Stéphane Corlosquet[4]
Orri Erling[5] Jérôme Euzenat[6] Kingsley Idehen[5] Jacek Kopecký[7]
Thomas Krennwallner[2] Davide Palmisano[8] Janne Saarela[9]
Michal Zaremba[10]

Abstract

One of the requirements of current Semantic Web applications is to deal with heterogeneous data. The Resource Description Framework (RDF) is the W3C recommended standard for data representation, yet data represented and stored using the Extensible Markup Language (XML) is almost ubiquitous and remains the standard for data exchange. While RDF has a standard XML representation, XML Query languages are of limited use for transformations between natively stored RDF data and XML. Being able to work with both XML and RDF data using a common framework would be a great advantage and eliminate unnecessary intermediate steps that are currently used when handling both formats.

1 XML and RDF transformations

In many application domains combining and converting between XML and RDF data is a useful but often not trivial task, relying mostly on XSLT, or – more recently – XQuery. The importance of this issue is acknowledged within the W3C, for instance the Gleaning Resource Descriptions from Dialects of Languages (GRDDL) [6] and Semantic Annotations for WSDL (SAWSDL) [8] recommendations stress the problem of *lifting* arbitrary XML formats to RDF and vice versa (*lowering*). The W3C GRDDL working group has the goal to complement the concrete RDF/XML syntax with a mechanism to relate it to other XML dialects (especially XHTML or "microformats") [6]. GRDDL focuses on transformations to RDF from XML, typically XSLT is the language of choice to describe such transformations. In SAWSDL there is also a strong need for translations from RDF to arbitrary XML. The automatic communication between software agents or services requires transformations in both directions. Transformations from XML to RDF are used to extract RDF from incoming response messages, while transformations from RDF to XML are used to create the request XML messages from the RDF data available to the client.

We acknowledge the efforts of these working groups, however, we argue that writing XSLT can be cumbersome, since it is a general-purpose language for producing XML without special support for creating RDF. Also, although the approach proposed by the SAWSDL Working Group provides a good starting point, we argue that it can still be improved on several points: SAWSDL suggests a two-step approach, combining SPARQL queries and XSLT on the SPARQL query results format when it comes to lowering from RDF to XML. To us, the detour through the SPARQL query results XML format seems to be an unnecessary burden. Also, many use cases

[1]Digital Enterprise Research Institute, National University of Ireland, Galway
[2]Institut für Informationssysteme, Technische Universität Wien, Favoritenstrae 9-11, A-1040 Vienna, Austria
[3]Fundación CTIC (Centro Tecnológico para el Desarrollo en Asturias de las Tecnologías de la Información y la Comunicación)
[4]MIND, Massachusetts General Hospital, Boston, MA, USA
[5]OpenLink Software Inc.
[6]INRIA & LIG, Montbonnot, France
[7]The Open University, Milton Keynes, UK
[8]Fondazione Bruno Kessler, Web of Data Research Unit, Via S.Croce 77 - 38122 Trento - Italy
[9]Profium
[10]seekda GmbH, Innsbruck, Austria

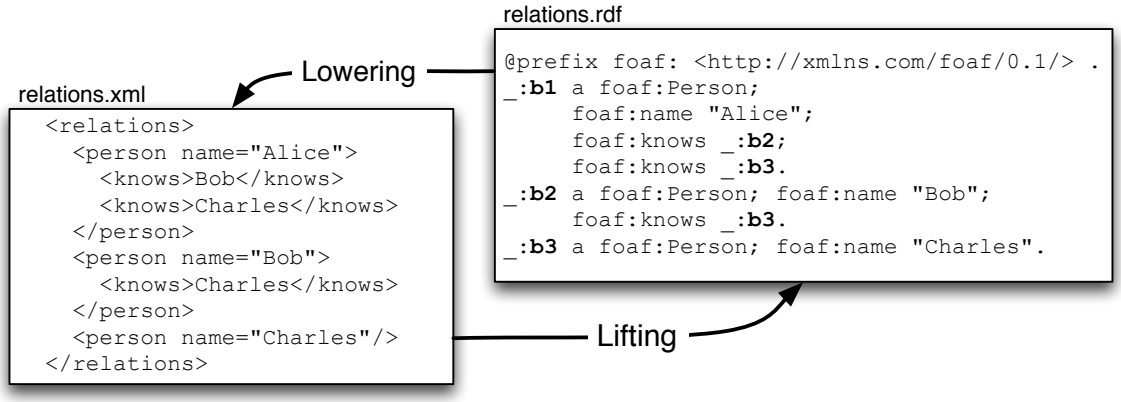

Figure 1: From XML to RDF and back: "lifting" and "lowering"

require some form of workflow management over SPARQL, that typically requires an implementation using an external programming/scripting framework. Moreover, features such as RDFS and OWL entailment which reveal the the potential of query expressiveness on the RDF side are currently being defined for SPARQL's next version [9]. A unified approach that builds up directly on SPARQL makes such extensions directly amenable to RDF/XML transformations. In contrast, incorporating such entailments in a pure XQuery/XSLT based approach, would require the implementation of e.g. an RDFS inference engine based on RDF/XML within XSLT which clearly looses the advantages of the declarative models of RDF, RDFS and OWL.

As an example for illustrating the need for a uniform language, we present the problem of mapping between FOAF data and a customised XML format as shown in Figure 1. The task here in either direction is to extract for all persons the names of people they know. Since data in RDF is on a higher level of abstraction than semi-structured XML data, the translation from XML to RDF is often called *lifting*, while the opposite direction is called *lowering*.

2 Lifting and Lowering

In the Semantic Web Services community, RDF-based client software needs to communicate with XML-based Web services, thus it needs to perform transformations between its RDF data and the XML messages that are exchanged with the Web services. In our example scenario, lifting transforms from the relations.xml format into RDF, for example to merge the data with other existing FOAF data. We can implement a straightforward lifting XSLT to generate the RDF/XML form in Figure 1. Lifting is analogous to the GRDDL situation – the client or an intermediate mediation service receives XML and needs to extract RDF from it.

RDF/XML [3] is the recommended syntax for RDF, using XML as the underlying representation model. This theoretically enables the use of XPath/XSLT/XQuery and other XML tools to translate between RDF/XML and other XML formats. This, however, is greatly complicated by the flexibility which the RDF/XML format allows RDF graphs to be serialised. Thus, tools that handle RDF/XML as XML data (and not as a set of triples) need to take different possible representations into account. Let us consider the example in Figure 3. It shows four versions of the same FOAF data.[11] Version (a) uses Turtle [2], a simple and readable textual format for RDF, inaccessible to pure XML processing tools though; the other three versions are all RDF/XML, ranging from concise (b) to verbose (d). For the lowering example, we can assume a social network site with a Web service for querying and updating the list of a users friends. The service accepts an XML format à la relations.xml (Figure 1) as the message format for updating a users list of friends. Assuming the client stores its FOAF data

[11] In listings and figures we often abbreviate well-known namespace URIs (`http://www.w3.org/1999/02/22-rdf-syntax-ns#`, `http://xmlns.com/foaf/0.1/`, etc.) with "...".

```
<xsl:stylesheet version="1.0" xmlns:rdf="...rdf-syntax-ns#"
  xmlns:foaf="...foaf/0.1/" xmlns:xsl="...XSL/Transform">
<xsl:template match="/rdf:RDF">
  <relations><xsl:apply-templates select=".//foaf:Person"/></relations>
</xsl:template>
<xsl:template match="foaf:Person"><person name="{./@foaf:name}">
  <xsl:apply-templates select="./foaf:knows"/>
</person></xsl:template>
<xsl:template match="foaf:knows[@rdf:nodeID]"><knows>
  <xsl:value-of select="//foaf:Person[@rdf:nodeID=./@rdf:nodeID]/@foaf:name"/>
</knows></xsl:template>
<xsl:template match="foaf:knows[foaf:Person]">
  <knows><xsl:value-of select="./foaf:Person/@foaf:name"/></knows>
</xsl:template>
</xsl:stylesheet>
```

Figure 2: Lowering attempt by XSLT (mylowering.xsl)

```
@prefix alice: <alice/> .
@prefix foaf: <...foaf/0.1/> .

alice:me a foaf:Person.
alice:me foaf:knows _:c.
_:c a foaf:Person.
_:c foaf:name "Charles".
```

(a)

```
<rdf:RDF xmlns:foaf="...foaf/0.1/"
   xmlns:rdf="...rdf-syntax-ns#">
 <foaf:Person rdf:about="alice/me">
   <foaf:knows>
     <foaf:Person foaf:name="Charles"/>
   </foaf:knows>
  </foaf:Person>
</rdf:RDF>
```

(b)

```
<rdf:RDF xmlns:foaf="...foaf/0.1/"
  xmlns:rdf="...rdf-syntax-ns#">
  <rdf:Description rdf:nodeID="x">
    <rdf:type
        rdf:resource=".../Person"/>
    <foaf:name>Charles</foaf:name>
  </rdf:Description>
  <rdf:Description
      rdf:about="alice/me">
    <rdf:type
      rdf:resource=".../Person"/>
    <foaf:knows rdf:nodeID="x"/>
  </rdf:Description>
</rdf:RDF>
```

(c)

```
<rdf:RDF xmlns:foaf="...foaf/0.1/"
       xmlns:rdf="...rdf-syntax-ns#">
    <rdf:Description rdf:about="alice/me">
      <foaf:knows rdf:nodeID="x"/>
  </rdf:Description>
  <rdf:Description rdf:about="alice/me">
    <rdf:type rdf:resource=".../Person"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="x">
    <foaf:name>Charles</foaf:name>
  </rdf:Description>
  <rdf:Description rdf:nodeID="x">
    <rdf:type rdf:resource=".../Person"/>
  </rdf:Description>
</rdf:RDF>
```

(d)

Figure 3: Different representations of the same RDF graph

(relations.rdf in Figure 1) in RDF/XML in the style of Figure 3(b), the simple XSLT stylesheet mylowering.xsl in Figure 2 would perform the lowering task. However, this XSLT will break if the RDF input is in any of the other variants shown in Figure 3. We could create a specific stylesheet for each of the presented variants, but creating one that handles all the possible RDF/XML forms would be much more complicated. We thus argue that RDF data should be handled through SPARQL's graph patterns, rather than through XPath/XSLT/XQuery.

In recognition of this problem, SAWSDL contains a non-normative example which performs a lowering transformation as a sequence of a SPARQL query followed by an XSLT transformation on the SPARQL query results XML format [5]. The advantage of this approach is that since SPARQL works on the RDF data model, all the input data from Figure 3 are considered to be equivalent.

3 XSPARQL

To address the problems of transforming between RDF and XML, and combining RDF data with XML data, we have developed the language XSPARQL, a combination of XQuery for accessing, manipulating and returning XML data, and SPARQL for accessing and returning RDF data. A more tightly coupled integration of SPARQL and XSLT/XQuery promises to provide a more expressive language, beyond the capabilities of using SPARQL and XSLT/XQuery sequentially. While both XQuery and SPARQL languages operate on different data models,

respectively XML and RDF, we defend that the merge of both in a common query language has the potential to finally bring XML and RDF closer together. XSPARQL provides concise and intuitive solutions for mapping between XML and RDF in either direction. An additional use for SPARQL is the construction of new RDF graphs by, for instance, converting RDF data between two different representations. XSPARQL also provides extended expressiveness for such transformations by allowing to use XPath functions in the construction step and providing the desired workflow/scripting facilities directly in the language. The XSPARQL language has been published as a W3C Member Submission [10] in January of 2009.[12]

Both in XQuery and in SPARQL, the processing of a query begins with selecting data of interest. XQuery uses XPath expressions to select XML nodes, and SPARQL uses graph patterns to select sets of variable bindings. XSPARQL combines these data access capabilities: a single XSPARQL query can access any number of RDF data sources using graph patters, and any number of XML data sources using XPath expressions; the results of both are stored in variables.

A query in XQuery usually returns an XML document, building it from elements and variable values. SPARQL can return an RDF graph by constructing it from triples that include variable values. In XSPARQL, a query can either return an XML document or it can construct an RDF graph.

Like XQuery, XSPARQL supports nested expressions and sub-queries, and XPath/XQuery operators and functions on all variable values. In effect, XSPARQL can even be used in an RDF-only environment as a more powerful alternative to SPARQL, which does not normally allow query nesting or rich data manipulation.

In summary, XSPARQL enables combining XML and RDF data using their native access mechanisms, and by returning either XML or RDF, it can support rich transformations between the two worlds. More details are available in the W3C submission document [10].

4 Even more formats

More recently, the RDFa [1] working group has introduced another standard representation format for RDF – embedded in X(HT)ML. We believe that a unified transformation language, along the lines of XSPARQL, could serve not only for extracting RDF from RDFa, but also and in particular for merging RDFa into existing X(HT)ML. Similarly, N3 and Turtle[13], both W3C team submissions and de facto standards being part of SPARQL, are widely used RDF representations not amenable to pure XML Transformation languages per se, but fit well with approaches such as XSPARQL.

The inclusion of other input formats such as Relational Databases and JSON potentially increases the scope and uses for the language to other items on W3C's agenda, such as the recently started RDB2RDF working group.[14] There are several projects that enable exposing RDB data as RDF data, most of these assume a fixed translation schema where, for instance, database tables are translated into RDFS classes and table columns are represented as properties. We feel that a generic/unified query and transformation language, e.g. extending XSPARQL, that allows access to relational data and allowing to expose the data in several other formats, can be a more flexible method to represent data transformations for all relational, XML and native RDF data.

5 Conclusion

Currently standardised approaches for transforming between RDF and other formats, particularly XML, are limited to pure XML transformation languages (such as XSLT/XQuery), which run into problems as RDF/XML representations are ambiguous, losing the declarative nature of RDF. Alternatives for realising such transformations are:

- sequential combinations of SPARQL and XSLT/XQuery going through an intermediate format such as the SPARQL query result XML format.

[12] The full submission, with documents on language semantics, implementation and test-cases, and use cases, can be found at `http://www.w3.org/Submission/2009/01/`.

[13] See `http://www.w3.org/TeamSubmission/n3/` and `http://www.w3.org/TeamSubmission/turtle/`

[14] `http://www.w3.org/2001/sw/rdb2rdf/`

- using custom RDF and XML APIs on top of scripting languages.
- creating a unified query/transformation language that encompasses both the declarative nature of SPARQL on the RDF side and the expressive power of XQuery/XSLT on the XML side, in order to realise arbitrary transformations.

Possible next steps in W3C

As more RDF data becomes available it is not going to replace custom application formats be it in XML or other formats (e.g., relational data or JSON). Thus, rather than touching RDF's core, we see a general need to discuss and address the problem of such transformations within W3C, particularly also how to exchange and process such transformations in a scalable manner. Based on our experiences with the unified language XSPARQL [10] – we argue for a unified approach to address this problem, potentially in an incubator or working group that spans the XML and Semantic Web activity and several working groups within (XQuery, SPARQL, RDB2RDF – to name the most prominent ones).

References

[1] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing. W3C recommendation, http://www.w3.org/TR/rdfa-syntax/, World Wide Web Consortium, October 2008.

[2] Dave Beckett and Tim Berners-Lee. Turtle - Terse RDF Triple Language, January 2008. W3C Team Submission, http://www.w3.org/TeamSubmission/turtle/.

[3] Dave Beckett and Brian McBride (eds.). RDF/XML Syntax Specification (Revised). Technical report, W3C, February 2004. W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax/.

[4] Don Chamberlin, Jonathan Robie, Scott Boag, Mary F. Fernández, Jérôme Siméon, and Daniela Florescu. XQuery 1.0: An XML Query Language. W3C recommendation, W3C, January 2007. W3C Recommendation, http://www.w3.org/TR/xquery/.

[5] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. SPARQL Protocol for RDF, November 2007. http://www.w3.org/TR/rdf-sparql-protocol/.

[6] Dan Connolly (ed.). Gleaning Resource Descriptions from Dialects of Languages (GRDDL). W3C recommendation, http://www.w3.org/TR/grddl/, W3C, September 2007.

[7] Michael Kay (ed.). XSL Transformations (XSLT) Version 2.0 , January 2007. W3C Recommendation, http://www.w3.org/TR/xslt20.

[8] Joel Farrell and Holger Lausen. Semantic Annotations for WSDL and XML Schema. W3C Recommendation, W3C, August 2007. Available at http://www.w3.org/TR/sawsdl/.

[9] Birte Glimm and Bijan Parsia. SPARQL 1.1 entailment regimes. W3C working draft, W3C, 2010. Available at http://www.w3.org/TR/2010/WD-sparql11-entailment-20100126/.

[10] Axel Polleres, Thomas Krennwallner, Nuno Lopes, Jacek Kopecký, and Stefan Decker. XSPARQL Language Specification, January 2009. W3C member submission, http://www.w3.org/Submission/xsparql-language-specification/.