Extensions of SPARQL towards Heterogeneous Sources and Domain Annotations

Nuno Lopes

Digital Enterprise Research Institute,
National University of Ireland Galway, Ireland
nuno.lopes@deri.org

Abstract. SPARQL is the W3C Recommended query language for RDF. My current work aims at extending SPARQL in two distinct ways: (i) to allow a better integration of RDF and XML; and (ii) to define a query language for RDF extended with domain specific annotations. Transforming data between XML and RDF is a much required, but not so simple, task in the Semantic Web. The aim of (i) is to enable transparent transformations between these two representation formats, relying on a new query language called XSPARQL. On a different aspect, representing and reasoning with meta-information about RDF triples has been addressed by several proposals for representing time, trust and provenance. A common extension of RDF (and RDFS inferencing rules), capable of encompassing all these proposals, with a clearly defined semantics is much desirable. Building on top of Annotated RDF, we present such an extension and an extension of the SPARQL language capable of querying triples with annotations. An open research issue remains the possibility of unifying the two, currently independent, extensions of SPARQL.

1 Overview

XML and RDF are the underlying representation and storage formats for the Semantic Web. For instance, in the Semantic Web Services domain, data represented in RDF needs to be converted to specific formats of XML (*lowering*) in order to be transmitted and the received data needs to be converted back to RDF (*lifting*). However, it is not easy to convert data between the two formats. These transformations, mainly the *lowering* traditionally have been done in a two step approach, first performing a SPARQL query to retrieve the RDF data and then using XSLT or XQuery on the SPARQL XML results format. One focus of my PhD is to improve these procedures by defining a single step approach relying on a combination of SPARQL and XQuery, called XSPARQL. This language allows to easily convert between the XML and RDF formats thus improving the tasks of *lifting* and *lowering*. The merge of XQuery and SPARQL allows to interchangeably use XQuery *return* clauses and SPARQL *construct* clauses for the generation of XML and RDF data respectively. The motivations and initial sketch of the XSPARQL language were presented in [1].

Another extension of SPARQL involves querying Annotated RDF, an extension of RDF towards domain specific annotations. Several extensions to RDF

have been presented in order to deal with temporal information [9,13,16], uncertainty or fuzzy information [14], trust [10] and provenance [7]. An extension of RDF towards domain specific annotations, called Annotated RDF, was presented by Udrea *et al.* [17]. Some of my previous work [15], was to define an extension of Annotated RDF towards supporting RDFS [5] inferencing rules, along with the definition of the Temporal and Fuzzy annotation domains. The presented RDFS reasoning procedure which can be formulated independently of the specific annotation domain by being parameterised with operations any domain needs to provide. In order to support querying Annotated RDF, an extension of the SPARQL query language towards domain specific annotations, called AnQL, is currently under submission. An in depth technical report [11] describing our framework is available.

2 XSPARQL

The main contributions of the XSPARQL language can be summarised as: i) Seamless integration of XML and RDF data, allowing for the easy transformation of data between the two representation languages; ii) Allow to perform more expressive SPARQL queries with the addition of e.g. nested queries; iii) Extend SPARQL with all the readily available XQuery functions. These extensions allow to perform several queries that are outside of the expressivity of SPARQL and, as the following query example presents, provides an easy way to perform queries over Linked Data. The following query was originally stated in [12] as "given input about publications taken from DBLP, create an RDF graph containing, for each co-author pair, the number of publications they co-authored". This query can be expressed using XSPARQL as presented below,[1] making use of some features not available in SPARQL:

```
1 prefix foaf: <http://xmlns.com/foaf/0.1/>
2 prefix dc:   <http://purl.org/dc/elements/1.1/>
3
4 let $ds := for * from <http://dblp.l3s.de/d2r/resource/authors/Axel_Polleres>
5            where { $pub dc:creator [] }
6            construct { { for * from $pub where { $p dc:creator $o . }
7                          construct {$p dc:creator <{$o}>} } }
8
9 let $allauthors := distinct-values(for $o from $ds where {$p dc:creator $o}
10 order by $o return concat("<",$o,">"))
11
12 for $auth at $auth_pos in $allauthors
13  for $coauth in $allauthors[position() > $auth_pos]
14     let $commonPubs := count({ for $pub from $ds
15                                where { $pub dc:creator $auth, $coauth }
16                                return $pub })
17 where ($commonPubs > 0)
18 construct { [ :author1 $auth; :author2 $coauth;  :commonPubs $commonPubs ] }
```

This query starts by collecting all the co-author list of Axel Polleres (lines 4-7), constructing an RDF dataset with a publication and its authors (`$ds` variable).

[1] The presented query is restricted to co-author list of Axel Polleres

The query continues by retrieving all the distinct authors into the XQuery sequence `$allauthors` (lines 9-10). The nested loop on lines 12-16, for each pair of authors, performs a SPARQL query to the constructed dataset (`$ds`) to count the number of shared publications of the authors, and constructs the final dataset with the required information in line 18. For further information on the syntax and semantics of XSPARQL we refer the reader to [1] and to the published W3C Member Submission at `http://www.w3.org/Submission/2009/01/`.

Most of the existing proposals to merge XML and RDF rely on translating the data from different formats and/or translating the queries from different languages. Deursen et al. [6] present an approach for transforming between XML and RDF in a ontology dependant manner. In [3] is presented a framework that allows to perform SPARQL queries from XSLT: XSLT+SPARQL. SPARQL2XQuery [4] translates each SPARQL query into an XQuery using a previously defined mapping from OWL to XML Schema. Groppe *et al.* [8] proposes to embed SPARQL into XSLT or XQuery, presenting extensions to these languages to enable SPARQL querying.

One of the next steps is to research possible optimisations for the XSPARQL query language. So far, I am focusing on optimisations for specific types of queries, namely queries using nested loops. Such optimisations are commonly referred to as *Dependent Join Optimisation*. Currently under development are two forms of optimisation for dependent joins: one is based on reordering the loops in the resulting XQuery in order to minimise the number of calls to the SPARQL endpoint. The other is based on storing intermediate results of the outer loop in a named graph in the SPARQL endpoint and performing a more complex SPARQL query that takes care of joining the variables. Another of the next goals is to include the ability to query relational databases from within XSPARQL, adding more input types to the language. This poses several problems ranging from syntax design to efficient implementation of the integration.

3 AnQL – Annotated SPARQL

Annotated RDF, first presented by Udrea *et al.* [17], consists of extending an RDF triple (s, p, o) with an annotation, where the annotation is taken from a specific domain. For instance, in the temporal domain [9], a triple:

$$:nuno\ :worksFor\ :DERI\ : [2008, 2010]$$

has intended meaning "Nuno has worked for DERI from 2008 to 2010", while in the fuzzy domain [14] the triple:

$$:nuno\ :locatedIn\ :room103\ : 0.9$$

has the intended meaning "Nuno is located in Room 103 to a degree of at least 0.9". The annotation domain must define the elements of the annotations, a *partial order* between elements and operations for combining elements of the domain. Based on this, it is possible to define an extension of the RDFS rules where the inferences take into account the annotations of the triples by using these domain specific operations. Further details about our prototype implementation are available in our project webpage at `http://anql.deri.org`.

For demonstrating the practical application of this work, we present the use case of exposing sensor data readings as Annotated RDF. With freely available sensor readings,[2] it is possible to generate an annotated dataset with triples of the form:

$$(:tag1, :locatedIn, :room103) : ([09{:}23, 11{:}56], 0.9)$$

where the annotation domain is a complex domain combining the temporal[3] and fuzzy domains. The annotation indicates the time interval when a tag was located in a specific place and the confidence value of such statement (triple). Using the extension of SPARQL to query Annotated RDF, AnQL, it is possible to state queries like "where was Nuno between 10:00 and 11:00 and what is the confidence value of the answer":

```
SELECT ?room ?conf
WHERE { ?tag :assignedTo :nuno .
        ?tag :locatedIn ?room :([10:00, 11:00],?conf) }
```

The evaluation of AnQL queries can work in a similarly to evaluating SPARQL queries, retrieving the bindings for the variables by graph pattern matching.

SPARQL extensions towards querying domain specific RDF were presented previously. Tappolet and Bernstein [16] present the τ-SPARQL query language that uses an extended SPARQL syntax to match the graph pattern against a temporal graph at any given time point. Hartig [10] introduces a trust aware query language, tSPARQL, that includes a new constructor to access the trust value of a graph pattern. Dividino *et al.* [7] extend the syntax and semantics of SPARQL to consider an additional expression that enables querying the named graphs where they store meta-information regarding provenance and uncertainty.

For AnQL, the most important next steps are the definition and implementation of use-cases. One use-case may consist of the presented scenario of exposing sensor readings as Annotated RDF. Annotated RDF provides a concise and expressive enough representation language for exposing such data. A different use-case can be the scenario of providing an enhanced version of DBPedia[4] with trust values for the triples. These trust values can be deduced by looking at the versioning information in Wikipedia. The semantics of complex domains need to be further researched, as well as improving the definition of other domains such as provenance and trust. Considering more expressive regimes than RDFS is another possible research direction.

4 Summary

In this paper I presented a short overview of the two extensions of SPARQL I am working on in the course of my PhD studies. These extensions tackle known problems with converting data from XML to RDF (and vice-versa) and allow to perform SPARQL queries over Annotated RDF data. Along side the future work presented for each of the extensions, another open research question is how to combine the two extensions.

[2] `http://openbeacon.org/`

[3] For presentation I omit the date from the temporal interval and use only hours.

[4] `http://dbpedia.org/`

References

1. W. Akhtar, J. Kopecký, T. Krennwallner, and A. Polleres. XSPARQL: Traveling between the XML and RDF Worlds - and Avoiding the XSLT Pilgrimage. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021, pages 432–447. Springer, 2008.
2. L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. P. B. Simperl, editors. *The Semantic Web: Research and Applications, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proc.*, volume 5554. Springer, 2009.
3. D. Berrueta, J. E. Labra, and I. Herman. XSLT+SPARQL : Scripting the Semantic Web with SPARQL embedded into XSLT stylesheets. In C. Bizer, S. Auer, G. A. Grimmes, and T. Heath, editors, *4th Workshop on Scripting for the Semantic Web*, Tenerife, June 2008.
4. N. Bikakis, N. Gioldasis, C. Tsinaraki, and S. Christodoulakis. Querying XML Data with SPARQL. In S. S. Bhowmick, J. Küng, and R. Wagner, editors, *DEXA*, volume 5690, pages 372–381. Springer, 2009.
5. D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Rec, `http://www.w3.org/TR/rdf-schema/`, W3C, February 2004.
6. D. V. Deursen, C. Poppe, G. Martens, E. Mannens, and R. V. d. Walle. Xml to rdf conversion: A generic approach. In *Proc. of AXMEDIS'08*, pages 138–144, Washington, DC, USA, 2008. IEEE Computer Society.
7. R. Dividino, S. Sizov, S. Staab, and B. Schueler. Querying for provenance, trust, uncertainty and other meta knowledge in rdf. *Web Semant.*, 7(3):204–219, 2009.
8. S. Groppe, J. Groppe, V. Linnemann, D. Kukulenz, N. Hoeller, and C. Reinke. Embedding SPARQL into XQuery/XSLT. In R. L. Wainwright and H. Haddad, editors, *SAC*, pages 2271–2278. ACM, 2008.
9. C. Gutierrez, C. A. Hurtado, and A. A. Vaisman. Introducing Time into RDF. *IEEE TKDE*, 19(2):207–218, February 2007.
10. O. Hartig. Querying Trust in RDF Data with tSPARQL. In Aroyo et al. [2], pages 5–20.
11. N. Lopes, G. Lukácsy, A. Polleres, U. Straccia, and A. Zimmermann. A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data. Technical report, DERI, 2010. `http://www.deri.ie/fileadmin/documents/DERI-TR-2010-03-29.pdf`.
12. M. S. Martín and C. Gutierrez. Representing, Querying and Transforming Social Networks with RDF/SPARQL. In Aroyo et al. [2], pages 293–307.
13. A. Pugliese, O. Udrea, and V. S. Subrahmanian. Scaling RDF with Time. In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *WWW*, pages 605–614. ACM, 2008.
14. U. Straccia. A Minimal Deductive System for General Fuzzy RDF. In S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset, and R. A. Schmidt, editors, *RR*, volume 5689, pages 166–181. Springer, 2009.
15. U. Straccia, N. Lopes, G. Lukacsy, and A. Polleres. A General Framework for Representing and Reasoning with Annotated Semantic Web Data. In M. Fox and D. Poole, editors, *AAAI*. AAAI Press, 2010.
16. J. Tappolet and A. Bernstein. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In Aroyo et al. [2], pages 308–322.
17. O. Udrea, D. R. Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Trans. Comput. Logic*, 11(2):1–41, 2010.