SPARQL+, SPARQLScript, SPARQL Result Templates - SPARQL Extensions for the Mashup Developer

Benjamin Nowack
semsol
Bielefelder Str. 5
40468 Düsseldorf, Germany
+49.211.7316824

bnowack@semsol.com

ABSTRACT

SPARQL lowers the barrier to RDF-based mashup development. It does, however, not support write operations, useful constructs like aggregates, the ability to combine and post-process query results, or human-oriented result formats. This paper describes a set of extensions that largely reuse SPARQL's intuitive syntax to provide query aggregates and update functionality (SPARQL+), result processing and chained queries across multiple endpoints (SPARQLScript), and result templating. The combination of these extensions enables the creation of mashups not only from distributed data sources, but also from portable application components.

Categories and Subject Descriptors

D.1 [**Programming Techniques**].

D.2.13 [**Software Engineering**]: Reusable Software.

D.3.3 [**Programming Languages**]: Language Contructs and Features – *control structures, input/output, modules, packages, procedures, functions, and subroutines.*

General Terms

Algorithms, Experimentation, Human Factors, Standardization, Languages.

Keywords

SPARQL, SPARQL+, SPARQLScript, Templates, Mashups, RDF, Semantic Web

1. INTRODUCTION

Mashups are lightweight Web applications that combine data from multiple sources. They usually require custom code to retrieve and integrate results from different Application Programming Interfaces (APIs). Current "Web 2.0" mashups are often closed systems, even though they are based on open data. "Mashup chaining" (the ability to repurpose an augmented data set) is not possible.

The SPARQL W3C Recommendations [1][2] offer a standard way to serve and access structured Web data. For Mashup developers, this eliminates the need to locally implement handlers for the variuos methods provided by an API. SPARQL increases retrieval flexibiliy, and its SQL-like syntax guarantees a reduced learning curve[1]. Custom code is still needed for parts of the application logic and human-readable output, though. SPARQL also doesn't provide means to save intermediate or final data in a dedicated RDF store for repurposing.

The following sections describe SPARQL extensions for data updates, processing, and output, leading to a complementary set of tools for semantic mashup creation.

2. SPARQL+

SPARQL+[2] extends SPARQL with aggregates such as COUNT and SUM, GROUP BY, and SPARQL update operations like LOAD, INSERT and DELETE. The latter are aligned with HP's SPARQL/Update proposal [3]. With SPARQL+, developers don't have to fall back to custom code or local RDF API methods for certain calculations or RDF data storage and manipulation. SPARQL+ is a superset of SPARQL, a SPARQL+ endpoint transparently handles standard SPARQL queries, grammar extensions are kept to a minimum (see Table 1).

Table 1. SPARQL+ grammar changes

Query	Prologue (SelectQuery | ConstructQuery | DescribeQuery | AskQuery | LoadQuery | InsertQuery | DeleteQuery)
SelectQuery	'SELECT' ('DISTINCT' | 'REDUCED')? (Aggregate+ | Var+ | '*') DatasetClause* WhereClause SolutionModifier
Aggregate	('AVG' | 'COUNT' | 'MAX' | 'MIN' | 'SUM') '(' Var | '*' ')' 'AS' Var
LoadQuery	'LOAD' IRIref ('INTO' IRIref)?
InsertQuery	'INSERT' 'INTO' IRIref 'CONSTRUCT'? ConstructTemplate DatasetClause* WhereClause? SolutionModifier
DeleteQuery	'DELETE' ('FROM' IRIref)* 'CONSTRUCT'? ConstructTemplate? DatasetClause* WhereClause? SolutionModifier

[1] Comments by users of the ARC RDF toolkit suggest that SPARQL is possibly the most intuitive entry point to RDF development. Basic SQL know-how and a few initial example queries are often sufficient to get started.

[2] http://arc.semsol.org/docs/v2/sparql+

SolutionModifier	GroupClause? OrderClause? LimitOffsetClauses?
GroupClause	'GROUP' 'BY' Var (',' Var)*

3. SPARQLSCRIPT

SPARQL+ allows the manipulation of RDF data in a similar way to running standard SPARQL queries. The existing SPARQL protocol can be used to serve remixed data via a local endpoint for chained mashups. Still missing is a way to process and integrate data from multiple sources in a single operation, though. SPARQLScript[3] addresses this need with endpoint switching and the possibility to combine multiple SPARQL+ queries in a single request. Other features include placeholders (using the common '${variable}' notation), simple assignments, conditional branches and loops. Again, existing SPARQL syntax is reused where possible. Below is an example script and a list of the grammar changes (Table 2).

```
# The Prologue has a script-wide scope
BASE <http://localhost/my_mashup/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rss: <http://purl.org/rss/1.0/>

# set the current endpoint
ENDPOINT <sparql/public>

# feed still fresh?
$fresh = ASK FROM <graph-updates> WHERE {
  <http://planetrdf.com/index.rdf> dc:date ?date .
  FILTER (?date > "${NOW-60min}")
}

IF (!$fresh) {
  # refresh feed and update graph log
  LOAD <http://planetrdf.com/index.rdf>
  DELETE FROM <graph-updates> {
    <http://planetrdf.com/index.rdf> ?p ?o .
  }
  INSERT INTO <graph-updates> {
    <http://planetrdf.com/index.rdf> dc:date
    "${NOW}"
  }
}

# retrieve latest items
$items = SELECT * WHERE {
    ?item a rss:item ;
          rss:title ?title ;
          dc:date ?date .
} ORDER BY DESC(?date) LIMIT 3
```

Code snippet 1. SPARQLScript example

Table 2. SPARQLScript grammar changes (work in progress)

Script	(Query | EndpointDecl | PrefixDecl | Assignment | IFBlock | FORBlock | String)*
EndpointDecl	'ENDPOINT' IRI_REF
Assignment	Var (':=' | '=') (Query | String | Var) ';'?
IFBlock	'IF' BrackettedExpression '{' Script '}' ('ELSE' '{' Script '}')?
FORBlock	'FOR' '(' Var 'IN' Var ')' '{' Script '}'
Placeholder	('$' | '?') '{' [^}]* '}'

[3] http://arc.semsol.org/docs/v2/sparqlscript

4. SPARQL RESULT TEMPLATES

The missing piece for a fully SPARQL-oriented development workflow is output templating. SPARQL Result Templates are directly integrated with SPARQLScript, they can reuse loops and placeholders. Output can be generated at any step in the script, simply by adding stand-alone SPARQL/Turtle literals:

```
# retrieve latest items (see above)
$items = SELECT * WHERE ...

# template
$size = ${items.size}
IF ($size) {
  """ I found ${items.size} items: <ul> """
  FOR ($item in $items) {
    "<li>${item.title}</li>"
  }
  " </ul> "
}
ELSE { " no items found "}
```

Code snippet 2. SPARQL Result Template example

5. FUTURE WORK

The extensions presented in this paper are still evolving. They are, however, already part of the ARC toolkit and successfully used in deployed applications. It is now planned to further improve individual features based on feedback and experience. Especially SPARQLScript and the templating needs more work. One area that could be explored is the creation of semantic widgets and the practicability of simple, custom inference scripts as an alternative to ontology-based reasoning.

6. ACKNOWLEDGMENTS

This work is motivated by feedback and requests by the ARC[4] community and by contributors to SPARQLBot[5]. SPARQL+' update constructs are inspired by HP's SPARQL/Update proposal.

7. REFERENCES

[1] Prud'hommeaux, E., Seaborne, A. (editors) 2008. SPARQL Query Language for RDF. W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/

[2] Clark, K. G., Feigenbaum, L., Torres, E. (editors) 2008. SPARQL Protocol for RDF. W3C Recommendation. http://www.w3.org/TR/rdf-sparql-protocol/

[3] Seaborne, A., Manjunath, G. 2008. SPARQL/Update - A language for updating RDF graphs. http://jena.hpl.hp.com/~afs/SPARQL-Update.html

[4] http://arc.semsol.org/

[5] http://sparqlbot.semsol.org/