SPARQL1.1

olivier.corby@inria.fr

SPARQL 1.1

1. Overview
2. Query
3. Update
4. Protocol (WSDL 2.0)
5. Service Description (as an RDF Graph)
6. Uniform HTTP Protocol for Managing RDF Graphs
7. Entailment Regimes
8. Test Cases

Update

Defines an update language for RDF graphs.

Update

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT DATA
{ <http://example/book3> dc:title    "A new book" ;
                         dc:creator  "A.N.Other" .
}
```

Update

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

WITH <http://example/addresses>
DELETE { ?person foaf:firstName 'Bill' }
INSERT { ?person foaf:firstName 'William' }
WHERE
  { ?person a foaf:Person .
    ?person foaf:firstName 'Bill'
  }
```

Update

```
LOAD <documentURI> [ INTO GRAPH <uri> ]

CLEAR  [ SILENT ] (GRAPH <uri> | DEFAULT
   | NAMED | ALL )

CREATE [ SILENT ] GRAPH <uri>

DROP  [ SILENT ] (GRAPH <uri> | DEFAULT |
   NAMED | ALL )
```

Protocol: WSDL 2.0

Defines an abstract interface and HTTP bindings for a protocol to issue SPARQL Query and SPARQL Update statements against a SPARQL endpoint.

It uses **WSDL 2.0** to describe a means for conveying SPARQL queries to an SPARQL query processing service and returning the query results to the entity that requested them

Service Description: an RDF Graph

Defines a vocabulary and discovery mechanism for describing the capabilities of a SPARQL endpoint accessible by SPARQL Protocol.

SPARQL services made available via the SPARQL Protocol SHOULD return a service description document at the service URL.

Service Description

This service description SHOULD be made available in an RDF serialization, and MAY be provided embedded in HTML by RDFa

It is an RDF graph that describes the service (e.g. it's default graph, named graphs, support for 1.0, 1.1, Update, etc.)

HTTP Protocol

Describes the use of the HTTP protocol for managing named RDF graphs on an HTTP server.

Binding of PUT, GET, POST, etc.

Entailment Regime

Defines conditions under which SPARQL queries can be used with entailment regimes such as RDF, RDF Schema, OWL, or RIF.

SPARQL 1.1 Query Language

Project Expression

Aggregates

Property Path

New statements

Minus, Exists

Subquery

Project Expression

Return the result of an expression

```
select * (ext:price(?doc) as ?price)
where {
  ?x rdf:type c:Document
  ?x c:author ?a
}
```

New filters

`coalesce(?x, ?y, 10):` return first value that is not an error (such as unbound)

`if(?x>10, ?y , ?x+10)`

`?x in (“alpha”, ?beta, ‘gamma’)`

`uri(), iri():` **`create an uri`**

New functions

`strdt(“12”, xsd:integer):`
create a literal with a datatype

`strlang(“human”, “en”):`
create a literal with a language tag

`bnode() bnode(“id314”) :`
create a blank node

Aggregates

```
select ?x (count(?doc) as ?count) where {
  ?x c:hasCreated ?doc
}
group by ?x
```

Group by

Several arguments

```
select ?x ?date (count(?doc) as ?count)
  where {
  ?x c:hasCreated ?doc
  ?doc c:date ?date
}
group by ?x ?date
```

Group by + count + order

```
select ?x (count(?doc) as ?count)
  where {
  ?x c:hasCreated ?doc
}
group by ?x
order by desc (count(?doc))
```

Having

Additional filter after aggregate

```
select ?x
  (count(?doc) as ?count) where {
  ?x c:hasCreated ?doc
}
group by ?x
having (count(?doc) >= 10)
```

Aggregates

min, max, count

sum, avg

group_concat, sample

Aggregates

Return **one result** when there is no group by

```
select (min(?price) as ?min) where {
  ?x ex:price ?price
}
```

Aggregates

Count the number of results

```
select (count(*) as ?count) where {
  ?x ex:price ?price
}
```

Aggregates

Count number of distinct values

```
select (count(distinct ?x) as ?count)
  where {
  ?x ex:price ?price
}
```

Exercise

Find the number of persons member of an organization and who are not author of any document

Find in which organization there is the most persons that are not author of any document

Exercise

```
select * (count(?x) as ?count)
where {
  ?x ex:member ?org
  filter(?org = <O>)
  optional {?x ex:author ?doc}
 filter(!bound(?doc))
}
group by ?org
```

Exercise

```
select * (count(?x) as ?count)
where {
  ?x ex:member ?org
  optional {?x ex:author ?doc}
 filter(!bound(?doc))
}
group by ?org
order by desc(?count)
limit 1
```

Property Path

Path of length more than one between resources

```
xxx member yyy include zzz author ttt
```

```
select * where {
  xxx member/include/author ttt
}
```

Property Path

```
xxx rdf:first aaa
xxx rdf:rest yyy rdf:first bbb
xxx rdf:rest yyy rdf:rest zzz rdf:first ccc
```

Zero or more *rest* followed by one *first :* `rest* first`

```
select ?val where {
  xxx rdf:rest*/rdf:first ?val
}
```

Property Path Expression Operators

```
/ : sequence
| : alternative
+ : one or several
* : zero or several
? : optional
^ : inverse
! : negation
{min,max} : variable length path
```

Property Path: Reverse

```
?x ^ex:prop ?y ::=
?y  ex:prop ?x


?x ^EXP ?y ::=
?y  EXP ?x
```

Property Path: Negation

```
?x ! ex:prop ?y
```

All properties **but** ex:prop

Property Path: Variable length

```
?x rdfs:subClassOf{1,5} ?y

?x rdfs:subClassOf{,5} ?y

?x rdfs:subClassOf{1,} ?y

?x rdfs:subClassOf{5} ?y
```

Negation

Two patterns:

- Minus
- Not Exists

MINUS

Remove the results of PAT2 from the results of PAT1

```
PAT1 minus {PAT2}
```

MINUS

Remove from the member of org the resources whose name is ‘Olivier’

```
select * where {
 ?x c:memberOf ?org
 minus {?x c:name ‘Olivier’}
}
```

MINUS

```
PAT1 minus {PAT2}
```

Remove results

- that are compatible: same variables have same values
- when there is **at least one common variable**

MINUS: remove nothing

```
select * where {
 ?x c:memberOf ?org
  minus {?y c:name ‘Olivier’}
}
```

Remove results that are compatible (same variables have same values) when there is **at least one common variable**

(NOT) EXISTS

Test (absence) presence of pattern in RDF Graph

```
PAT1 . filter(! exists {PAT2})
```

NOT EXISTS

```
?x c:memberOf ?org .
filter(! exists {?x c:author ?doc })
```

Minus vs Exists

Same results:

```
?x c:memberOf ?org .
  filter(! exists {?x c:author ?doc })
```

```
?x c:memberOf ?org .
  minus {?x c:author ?doc }
```

Minus vs Exists

Different results:

```
?x c:memberOf ?org .
  filter(! exists {?y c:author ?doc })
```

```
?x c:memberOf ?org .
  minus {?y c:author ?doc }
```

Quiz

```
?x c:memberOf ?org .
  minus {ex:a c:memberOf ex:b}
```

Does it return:

```
 ex:a c:memberOf ex:b
```

Sub Query: Nested Query

Find properties of the cheapest car

```
select * where {
  {select (min(?price) as ?min) where {
     ?car ex:hasPrice ?price}
  }
  ?car ex:hasPrice ?min
  ?car ?p ?val
}
```