INRIA
SOPHIA ANTIPOLIS

SPARQL

W3C Simple Protocol And RDF Query Language

olivier.corby@inria.fr

SPARQL

W3C RDF Data Access

Use Case and Requirements :
`http://www.w3.org/TR/rdf-dawg-uc`

Query language :
`http://www.w3.org/TR/rdf-sparql-query`

XML Result Format:
`http://www.w3.org/TR-rdf-sparql-XML-res`

Protocol for Web Services :
`http://www.w3.org/TR/rdf-sparql-protocol`

SPARQL

Semantic Web SQL

Query an RDF triple store as graph(s)

SPARQL is neutral w.r.t inferences RDF/RDFS/OWL/RIF (e.g. subsumption, transitivity, etc.)

The triple store *may* process inferences

SPARQL 1.0 does not enable to tune inferences

SPARQL : example

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?mbox WHERE
  { ?x foaf:name 'Olivier' .
    ?x foaf:mbox ?mbox }
```

Return the mail box of resources whose name is 'Olivier'

SPARQL Syntax: Triple

```
subject property object

<http://www.inra.fr/john> foaf:name ?name

?x foaf:name 'Olivier'

?x ?p 'Olivier'

_:b1 foaf:age 49
```


Blank Node

```
?x c:speed [ rdf:value ?val ;
             c:unit 'km/h' ]
```

Equivalent to :

```
?x c:speed _:b
_:b rdf:value ?val
_:b c:unit 'km/h'
```

List Syntax

```
(1 ?x 'v')
⇔
_:a rdf:first 1 .
_:a rdf:rest _:b .
_:b rdf:first ?x .
_:b rdf:rest _:c .
_:c rdf:first 'v' .
_:c rdf:rest rdf:nil
```

Filter: Evaluable Expression

```
FILTER (?age >= 7 && ?age <= 77)
```

Comparison : `< <= = >= > !=`

Operation : `+ * / -`

Boolean : `&& (and) || (or) ! (not)`

Function : `isBlank(?x)  my:fun(?y)`

Filter: Evaluable Expression

```
?x c:age ?age
FILTER (?age >= 7 && ?age <= 77)
```

Variable must be bound by a triple

Tests

Match a regular expression

```
regex(?string, '.*inria.*')
```

Tests

Compare literals with and without language

```
"engineer" = str("engineer"@en)
```

Tests : functions

```
isURI(?x)

isLiteral(?y)

isBlank(?z)

bound(?t)
```

Tests : cast

```
xsd:integer(?x) >= 18

xsd:date("2012-10-30")

xsd:string("http://www.inria.fr/me")

xsd:double()

xsd:boolean()
```


Optional Pattern

```
SELECT * WHERE {
  ?x :hasCreated ?doc .
   OPTIONAL {
     ?x :isMemberOf ?org
   }
}
```

`hasCreated` is mandatory
`isMemberOf` is optional

Optional Pattern

```
SELECT * WHERE {
  ?x :hasCreated ?doc .
   OPTIONAL {
     ?x :age ?age .
     ?x :isMemberOf ?org
   }
}
```

`age` and `isMemberOf` must be present for the success of optional

Optional Pattern

```
SELECT * WHERE {
  ?x :hasCreated ?doc .
  OPTIONAL {?x :age ?age}
  OPTIONAL {?x :isMemberOf ?org}
}
```


Pattern UNION

```
SELECT * WHERE {
                                :Francis :apply :Presidency
   {?x :apply ?job}             :Nick :practice :Bike
   UNION                        :Nick :apply :Presidency
   {?x :practice ?act}
}
```

Results

```
(1) x = :Francis  ; job = :Presidency ; act = unbound
(2) x = :Nick     ; job = unbound     ; act = :Bike
(3) x = :Nick     ; job = :Presidency ; act = unbound
```


RDF Dataset

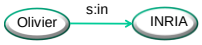
Query an RDF base with several graphs

Named graphs with URIs

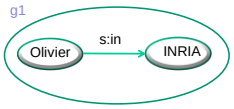
Default graph

Identify the graphs that are queried

SPARQL Graph Pattern



SPARQL Graph Pattern



SPARQL Graph Pattern

SPARQL Graph Pattern

```
select ?g
where { graph ?g {?x s:in ?org} }
```

SPARQL Graph Pattern

```
select ?g
where { graph ?g {?x s:in ?org} }
```

(1) ?g = g1
(2) ?g = g2

SPARQL Graph Pattern


```
select *
from named <g1>
from named <g2>
where {
   graph ?g {?x s:in ?org}
}
```

SPARQL Default Graph


```
select *
from <g1>
from <g2>
where {
    ?x s:in ?org
}
```


Result

- `SELECT * WHERE`
- `SELECT DISTINCT ?x ?y WHERE`
- `ORDER BY ?x DESC(?y)`
- `LIMIT 10`
- `OFFSET 10`

Distinct

```
select distinct ?x ?z
where {
  ?x :friend ?y
  ?y :friend ?z
}
```

Eliminate duplicate results (eliminate same variables with same values)

Order by

```
select ?pers ?date      :Jim  :author :d2
where {                 :Jack :author :d1
  ?pers :author ?doc    :d2 :date 2008-01-01
  ?doc :date ?date      :d1 :date 2007-12-31
}
order by ?date
```

Result

```
(1) pers = :Jim  ; date = 2007-12-31
(2) pers = :Jack ; date = 2008-01-01
```


Limit

```
select * where {
  PATTERN
}
LIMIT 20
```

Return 20 results at most

Limit/Offset

```
select * where {
  PATTERN
}
LIMIT 20
OFFSET 10
```

20 results (at most), after the 10th result
i.e. from 11th to 30th

Construct

Return new RDF graph

```
construct {
  ?girl :hasBrother ?boy
}
where {
  ?boy :hasSister ?girl
}
```

Describe

```
describe ?x where {
  ?x rdf:type :HotTopic
  ?x :date ?date
  filter (?date >= '2007-12-31'^^xsd:date)
}
```

Return a description of ?x

The **appropriate** description of a resource is determined by the RDF store

Ask

```
ask {:Olivier :teach :ensi}
```

yes/no answer

Negation as failure

Persons that are member of an organization and who have **not** created a document :

```
SELECT * WHERE {
  ?x c:isMemberOf ?org .
  OPTIONAL {
    ?x c:hasCreated ?doc
  }
  FILTER (! bound(?doc))
}
```

Extension Function (user defined)

```
PREFIX fun: <http://example.org/ext/>
SELECT * WHERE {
  ?x c:age ?age .

  FILTER fun:prime(?age)
}
```

XML Result Format

```
<?xml version="1.0"?>
<sparql
  xmlns="http://www.w3.org/2005/sparql-results#">

  <head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="age"/>
    <variable name="blurb"/>
  </head>
...
</sparql>
```

XML Result Format

```
<?xml version="1.0"?>
<sparql
  xmlns="http://www.w3.org/2005/sparql-results#">
  <head> ... </head>

  <results>
    <result>...
    </result>

    <result>...
    </result>
  </results>
</sparql>
```


XML Result Format

```
<result>

<binding name="x"><bnode>r2</bnode></binding>

<binding name="hpage"><uri>http://example.org</uri></binding>

<binding name="name">
  <literal xml:lang="en">Bob</literal>
</binding>

<binding name="age">
  <literal datatype="&rdfs;integer">30</literal>
</binding>


</result>
```


SPARQL 1.1 Query Language

W3C
http://www.w3.org/TR/sparql11-query/

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 {
  ?doc rdf:type c:Document
  ?doc 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')
```


New functions

```
strlen, concat, substr,
strstarts, strends, contains

year, month, day,
hours, minutes, seconds
```

43

Aggregates

Compute a global result over the list of results:

```
min, max, count
sum, avg
group_concat
sample
```

44

Aggregates

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

45

Aggregates

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

46

Aggregates: distinct

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

47

Having

Additional filter after aggregate

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

48

Aggregates

1. min
2. max
3. sum
4. avg
5. count
6. group_concat
7. sample

Aggregates

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

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

Aggregates

Return one result for each « group by » key

```
select ?class (min(?price) as ?min)
where {
  ?x a ?class ;
   ex:price ?price
}
group by ?class
```

Aggregates

Count the number of results

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

Aggregates

Count the number of distinct results

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

Property Path

Retrieve resources linked by a path in the graph

```
?x foaf:knows ?y
```

Property Path

Retrieve resources linked by a path in the graph

```
?x foaf:knows/foaf:knows ?y
```

Property Path

Retrieve resources linked by a path in the graph

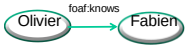
```
?x foaf:knows/foaf:knows/foaf:knows ?y
```


Property Path

```
prefix foaf: <http://xmlns.com/foaf/0.1/ >
select *
where {
        ?x foaf:knows+ ?y ;
            foaf:name 'Olivier'
}
```


Property Path

```
prefix foaf: <http://xmlns.com/foaf/0.1/ >
select *
where {
        ?x foaf:knows+ ?y ;
            foaf:name 'Olivier'
}
```



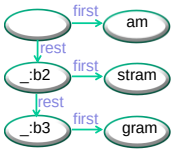
Property Path

```
prefix foaf: <http://xmlns.com/foaf/0.1/ >
select *
where {
        ?x foaf:knows+ ?y ;
            foaf:name 'Olivier'
}
```

Property Path

```
prefix foaf: <http://xmlns.com/foaf/0.1/ >
select *
where {
        ?x foaf:knows+ ?y ;
            foaf:name 'Olivier'
}
```

Property Path




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

Property Path Expression Operators

```
/ : sequence
| : alternative
+ : one or several
* : zero or several
? : optional
^ : reverse
! : negation
```

Negation : 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 may 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**

Negation : Not Exists

```
?x c:memberOf ?org .
filter(
  not 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 }
```

Sub 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
}
```


Bindings

```
select * where {
  ?x foaf:knows ?y
}
values ?x {
 ex:Robert
 ex:Jimmy
 ex:John
}
```


Service

```
select * where {
  ?x foaf:name 'Olivier'
  service <http://fr.dbpedia.org/sparql> {
     ?x ?p ?y
  }
}
```


SPARQL 1.1 Update

http://www.w3.org/TR/sparql11-update

Update language for RDF graphs

CRUD: Create Read Update Delete

Insert Data

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

INSERT DATA {

 <http://example/book3> dc:title    "A new book" ;
                        dc:creator  "A.N.Other" .
}
```


Delete Data

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

DELETE DATA {

 <http://example/book3> dc:title    "A new book" ;
                        dc:creator  "A.N.Other" .
}
```

Delete

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

DELETE { ?person foaf:firstName 'Bill' }

WHERE
  { ?person a foaf:Person .
    ?person foaf:firstName 'Bill'
  }
```


Insert

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

INSERT { ?person foaf:firstName 'William' }

WHERE
  { ?person a foaf:Person .
    ?person foaf:firstName 'Bill'
  }
```


Delete Insert

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

DELETE { ?person foaf:firstName 'Bill' }

INSERT { ?person foaf:firstName 'William' }

WHERE
  { ?person a foaf:Person .
    ?person foaf:firstName 'Bill'
  }
```


Insert Graph

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

INSERT {
  graph <g1> {
    ?person foaf:firstName 'William' }
}

WHERE
  { ?person a foaf:Person .
    ?person foaf:firstName 'Bill'
  }
```


Using: like From

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

INSERT {?person foaf:firstName 'William' }

USING <g1>

WHERE
  { ?person a foaf:Person .
    ?person foaf:firstName 'Bill'
  }
```


With

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

WITH <g1>

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 )

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

CREATE [ SILENT ] GRAPH <uri>
```