Introduction to a Web of Linked Data

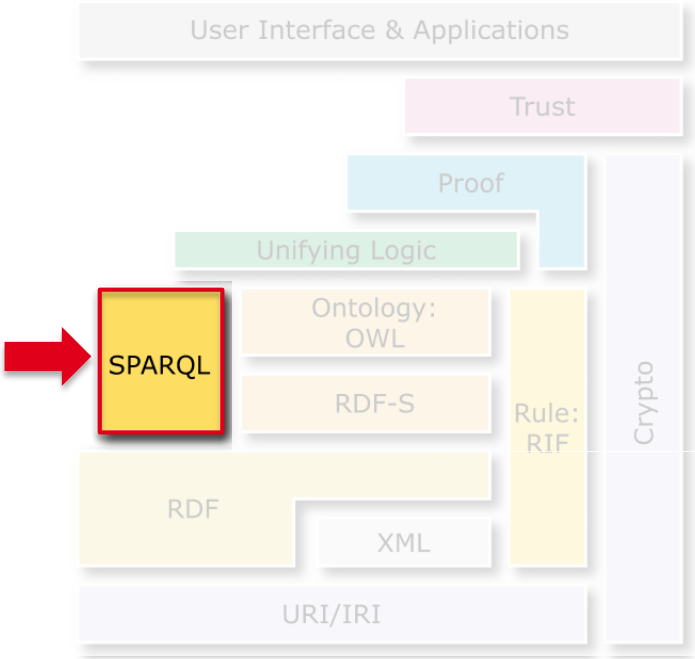
SPARQL Query Language

Access data sources on the Web

slides from Olivier Corby
presented by Catherine Faron

SPARQL Query Language

Query RDF triple stores published on the Web



SPARQL Query Language

1. RDF graph pattern matching
2. Statements
3. Filter, constraint and function
4. Pre and post processing
5. Several query forms
6. Results and Update

SPARQL Query Language

1. **RDF graph pattern matching**
2. Statements
3. Filter, constraint and function
4. Pre and post processing
5. Several query forms
6. Results and Update

SPARQL Protocol And RDF Query Language

1. Query Language (Turtle syntax)
   - SPARQL 1.1 Query Language - W3C REC 21 Mar. 2013

SPARQL Protocol And RDF Query Language

1. Query Language (Turtle syntax)
   - SPARQL 1.1 Query Language - W3C REC 21 Mar. 2013
   - SPARQL 1.1 Update - W3C REC 21 Mar. 2013

SPARQL Protocol And RDF Query Language

1. Query Language (Turtle syntax)
   - SPARQL 1.1 Query Language - W3C REC 21 Mar. 2013
   - SPARQL 1.1 Update - W3C REC 21 Mar. 2013
2. Result format
   - SPARQL Query Results XML Format - W3C REC 21 Mar. 2013

Query with SPARQL

```
SELECT   what you want
FROM     from where you want
WHERE    { as you want }
```

SPARQL triples

- Turtle syntax with question marks for variables:

```
?x    rdf:type    ex:Person
```

SPARQL triples

- Turtle syntax with question marks for variables:

```
?x    rdf:type    ex:Person
```

- Specify graph pattern to be found:

```
SELECT ?subject ?property ?value
WHERE { ?subject ?property ?value }
```

SPARQL triples

- Turtle syntax with question marks for variables:

```
?x    rdf:type    ex:Person
```

- Specify graph pattern to be found:

```
SELECT ?subject ?property ?value
WHERE { ?subject ?property ?value }
```

- A basic graph pattern is a conjunction of triples

```
SELECT ?x WHERE
{ ?x     rdf:type    ex:Person .
  ?x     ex:name     ?name . }
```

same abbreviations as Turtle

- triples with common subject:

```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

```
SELECT ?name ?fname
WHERE{?x rdf:type ex:Person.
      ?x ex:name ?name .
      ?x ex:firstname ?fname .
      ?x ex:author ?y . }
```

same abbreviations as Turtle

- triples with common subject:

```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

```
SELECT ?name ?fname
WHERE{?x rdf:type ex:Person.
      ?x ex:name ?name .
      ?x ex:firstname ?fname .
      ?x ex:author ?y . }
```

- several values:

```
?x ex:firstname "Fabien", "Lucien" .
```

same abbreviations as Turtle

- triples with common subject:

```
SELECT ?name ?fname                    SELECT ?name ?fname
WHERE {?x a ex:Person;                 WHERE{?x rdf:type ex:Person.
      ex:name ?name ;                        ?x ex:name ?name .
      ex:firstname ?fname ;                  ?x ex:firstname ?fname .
      ex:author ?y . }                       ?x ex:author ?y . }
```

- several values:

```
?x ex:firstname "Fabien", "Lucien" .
```

- blank nodes as anonymous variables:

```
[ ex:firstname "Fabien" ]
[] ex:firstname "Fabien" .
```

declare prefixes and namespaces

- declare prefixes for vocabularies used in the query:

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE {
 ?student mit:registeredAt ?x .
}
```

declare prefixes and namespaces

- declare prefixes for vocabularies used in the query:

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE {
 ?student <http://www.mit.edu#registeredAt> ?x .
}
```

declare prefixes and namespaces

- declare prefixes for vocabularies used in the query:

```
PREFIX mit:  <http://www.mit.edu#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student
WHERE {
 ?student mit:registeredAt ?x .
 ?x foaf:homepage <http://www.mit.edu> .
}
```

declare prefixes and namespaces

- declare prefixes for vocabularies used in the query:

```
PREFIX mit:  <http://www.mit.edu#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student
WHERE {
 ?student mit:registeredAt ?x .
 ?x foaf:homepage <http://www.mit.edu> .
}
```

- declare base namespace for relative URIs

```
BASE <http://ns.inria.fr/>
```

specify language and datatype of literals

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x ?f WHERE {
 ?x foaf:name "Fabien"@fr ; foaf:knows ?f .
}
```

specify language and datatype of literals

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x ?f WHERE {
 ?x foaf:name "Fabien"@fr ; foaf:knows ?f .
}
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x WHERE {
 ?x foaf:name "Fabien"@fr ;
    foaf:age "21"^^xsd:integer .
}
```

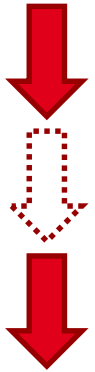
Week 03 : SPARQL Query Language

1. RDF graph pattern matching
2. **Statements**
3. Filter, constraint and function
4. Pre and post processing
5. Several query forms
6. Results and Update

optional pattern

when part of graph pattern is not mandatory.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:homepage <http://fabien.info> .
 OPTIONAL { ?person foaf:name ?name . }
}
```

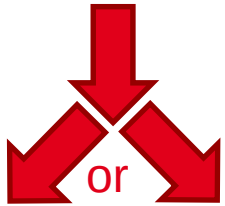


?name variable may be « unbound » in result

alternative patterns

union of results of graph patterns

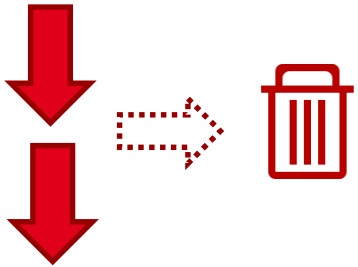
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:name ?name .
  {
   ?person foaf:homepage <http://fabien.info> .
  }
  UNION
  {
   ?person foaf:homepage <http://bafien.org> .
  }
}
```



negation

remove results that match a pattern

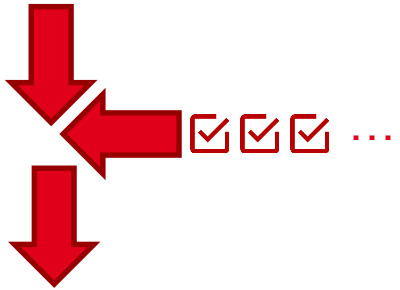
```
PREFIX ex: <http://www.example.abc#>
SELECT ?x
WHERE {
    ?x a ex:Person
    MINUS { ?x a ex:Man }
}
```



predefined variable values

results where part of the bindings are predefined.

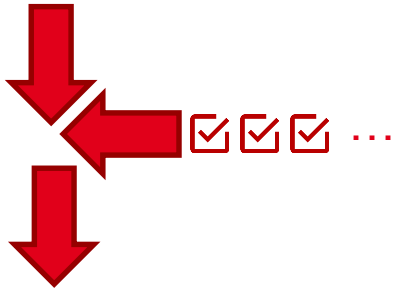
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:name ?name .
}
VALUES ?name  { "Peter" "Pedro" "Pierre" }
```



variable binding

results where part of the bindings are computed.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person ex:fname ?fname ;

   ex:lname ?lname .

 BIND (concat(?fname, ?lname) AS ?name)
}
```

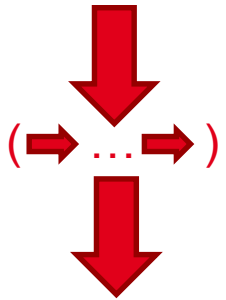


property path

Regular expressions on property path between resources

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

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?friend WHERE {
 ?x foaf:name "Fabien Gandon" ;
    foaf:knows+ ?friend .
}
```



keep distinct results

keep one occurrence of similar results with same values for same variables

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
WHERE { ?person foaf:name ?name . }
```

SPARQL Query Language

1. RDF graph pattern matching
2. Statements
3. **Filter, constraint and function**
4. Pre and post processing
5. Several query forms
6. Results and Update

filter results by values

declare constraints on variable values

- `select` = select values to be returned
- `where` = graph pattern
- **`filter`** = constraints in the `where` clause with XPath 2.0 functions or external functions

ex. person at least 18 years old

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name
WHERE {
 ?person    rdf:type ex:Person ;
            ex:name  ?name ;
            ex:age   ?age .
 FILTER (xsd:integer(?age) >= 18)
}
```

test values

test and compare constants, variables and expressions

- Comparators: `<, >, =, <=, >=, !=`
- Regular expressions: `regex(?x, "A.*")`
- Test variable values: `isURI(?x), isBlank(?x), isLiteral(?x), bound(?x)`

strings and literals

CONTAINS(lit$_1$,lit$_2$),STRSTARTS(lit1,lit2),STRENDS(lit$_1$,lit$_2$)	string inclusion
STRDT(value, type)	create literal with datatype
STRLANG(value, lang)	create literal with language
CONCAT(lit$_1$,…,lit$_n$)	concatenate strings
SUBSTR(lit, start [,length])	extract substring
ENCODE_FOR_URI(str)	encode string for URI
UCASE(str), LCASE(str)	change case
STRLEN(str)	string length

other functions

```
YEAR(Date), MONTH(Date), DAY(Date)
HOURS(Date), MINUTES(Date), SECONDS(Date)
NOW()


ABS(Val), CEIL(Val), FLOOR(Val), ROUND(Val)
isNumeric(Val)
RAND()


COALESCE(val1,…, valn)
IRI(str), URI(str)
BNODE(ID)
```

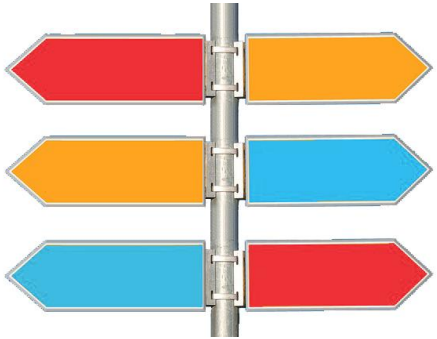
boolean connectors

- And: &&
- Or: ||
- Not: !
- ()

branching expression

usual test: if… then… else…

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * where {
 ?x foaf:name ?name ; foaf:age ?age .
 FILTER ( if (langMatches(lang(?name), "FR")
              ?age >= 18, ?age >= 21) )
}
```



verify presence/absence of a pattern

exists checks whether a pattern occur in the graph

not exists checks whether a pattern do not occur in the graph

```
SELECT ?name
WHERE {
 ?x foaf:name ?name .
 FILTER NOT EXISTS { ?x foaf:age -1 }
}
```

SPARQL Query Language

1. RDF graph pattern matching
2. Statements
3. Filter, constraint and function
4. **Pre and post processing**
5. Several query forms
6. Results and Update

specify default graph

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
FROM <http//www.mit.edu/data1.rdf>
FROM <http//www.mit.edu/data2.rdf>
WHERE { ?student mit:registeredAt ?x . }
```

specify named graphs

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?g ?student
FROM NAMED <http//www.mit.edu/data1.rdf>
FROM NAMED <http//www.mit.edu/data2.rdf>
WHERE {
 GRAPH ?g {
  ?student mit:registeredAt ?x .
 }
}
```

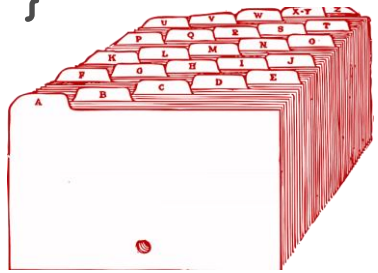
query remote SPARQL endpoint

```
SELECT ?x
WHERE {
 SERVICE <http://dbpedia.org/sparql> {
    ?x rdfs:label "Auguste"@fr .
 }
}
```

order and limit results

ex. sort results by *name* from n° 21 to n° 40

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name . }
ORDER BY ?name
LIMIT 20
OFFSET 20
```



aggregate results

group results by variable(s) values: `group by`
aggregate values: `count, sum, min, max, avg, group_concat, sample`
filter aggregated values: `having`

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE { ?student mit:score ?score . }
GROUP BY ?student
HAVING(AVG(?score) >= 10)
```

nested queries

use results of subquery in embedding query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name WHERE {
  { SELECT (max(?age) as ?max)
    WHERE { ?person foaf:age ?age } }
  ?senior foaf:age ?max .
  ?senior foaf:name ?name
}
```

select expressions

extend *select* clause with expressions

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x (month(?date) as ?month)
WHERE { ?x foaf:birthday ?date . }
```

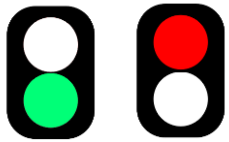
SPARQL Query Language

1. RDF graph pattern matching
2. Statements
3. Filter, constraint and function
4. Pre and post processing
5. **Several query forms**
6. Results and Update

check the existence of a solution

Do not enumerate all solutions, just answer *true* or *false*

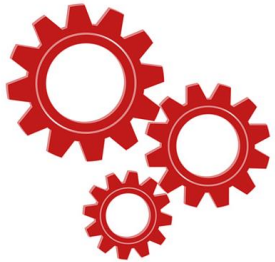
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?person foaf:age 111 . }
```



construct a result graph

Result of query is a fresh new RDF graph

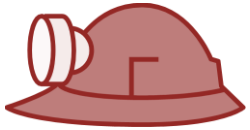
```
PREFIX mit:  <http://www.mit.edu#>
PREFIX corp: <http://mycorp.com/schema#>

CONSTRUCT { ?student a corp:FuturExecutive . }

WHERE     { ?student a mit:Student . }
```



describe a resource

discover unknown data

```
DESCRIBE <http://fabien.info>
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
DESCRIBE ?x WHERE { ?x foaf:name "Fabien" }
```

SPARQL Query Language

1. RDF graph pattern matching
2. Statements
3. Filter, constraint and function
4. Pre and post processing
5. Several query forms
6. **Results and Update**

SPARQL query result

- Select, Ask: XML Results format
- Construct, Describe: RDF/XML
- JSON

XML SPARQL Query Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>  <variable name="student"/> </head>
 <results>
  <result>
   <binding name="student">
    <uri>http//www.mit.edu/data.rdf#ndieng</uri>
   </binding>
  </result>
  <result>
   <binding name="student">
    <uri>http//www.mit.edu/data.rdf#jdoe</uri>
   </binding>
  </result>
 </results>
</sparql>
```

SPARQL Update

Manage the content of triple store

- Load
- Delete
- Insert
- Copy
- Move
- Add

…