SPARQL in a nutshell

fabien, gandon, inria

RDF triple model

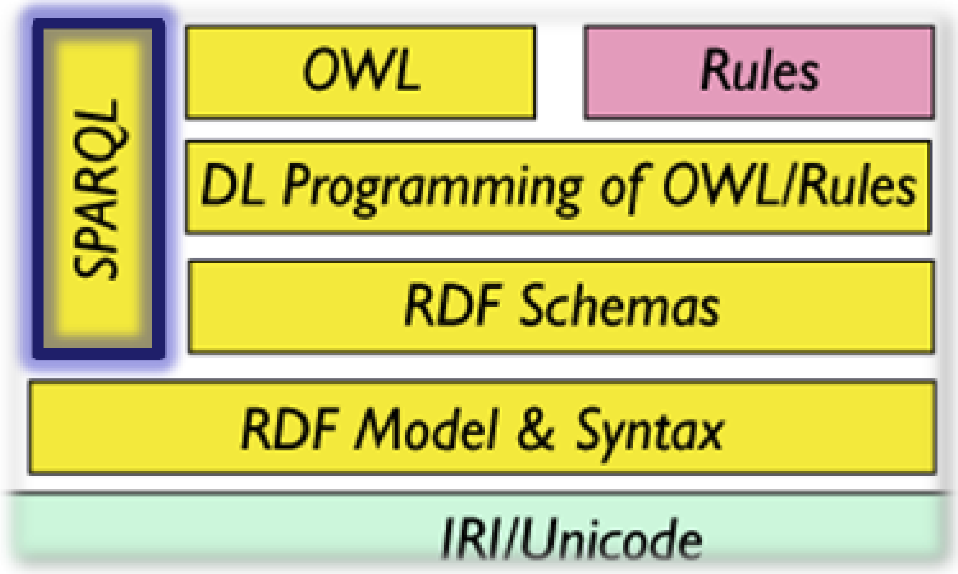
is the first layer of the semantic web standards

SPARQL on top...

an RDF query language and data access protocol

SPARQL stands for

SPARQL **P**rotocol and
RDF **Q**uery **L**anguage



SPARQL in 3 parts

part 1: query language

part 2: result format

part 3: access protocol

SPARQL query

```
SELECT ...
FROM ...
WHERE { ... }
```

SELECT clause

to identify the values to be returned

FROM clause

to identify the data sources to query

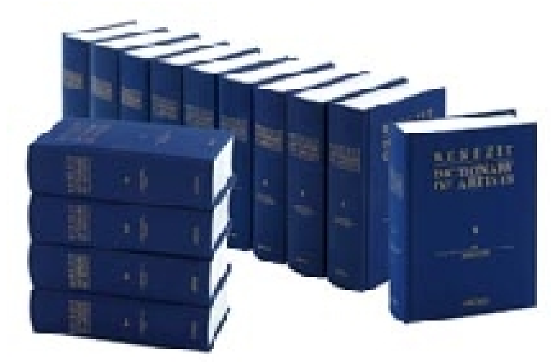
WHERE clause

the triple/graph pattern to be matched against the triples/graphs of RDF

WHERE clause

a conjunction of triples:

```
{ ?x rdf:type ex:Person
  ?x ex:name ?name }
```



PREFIX

to declare the schema used in the query

example persons and their names

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name
WHERE {
 ?person rdf:type ex:Person
 ?person ex:name ?name .
}
```

example of result

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#" >
 <head>
  <variable name="person"/>
  <variable name="name"/>
 </head>
 <results ordered="false" distinct="false">
  <result>
   <binding name="person">
    <uri>http://inria.fr/schema#fg</uri>
   </binding>
   <binding name="name">
    <literal>gandon</literal>
   </binding>
  </result>
  <result>  ...
```

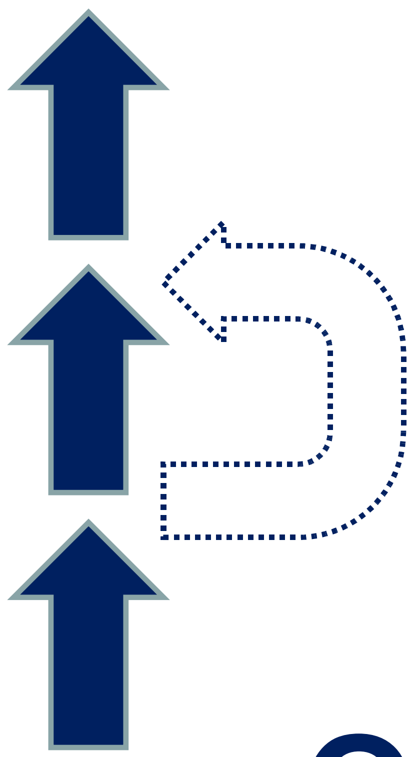
FILTER

to add constraints to the graph pattern (e.g., numerical like **`X>17`**)

example persons at least 18-year old

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name
WHERE {
 ?person rdf:type ex:Person
 ?person ex:name ?name .
 ?person ex:age ?age .
 FILTER (?age > 17)
}
```

FILTER can use many operators, functions (e.g., regular expressions), and even users' extensions



OPTIONAL

to make the matching of a part of the pattern optional

example

retrieve the age if available

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name ?age
WHERE {
 ?person rdf:type ex:Person
 ?person ex:name ?name .
 OPTIONAL { ?person ex:age ?age }
}
```

UNION

to give alternative patterns in a query

example explicit or implicit adults

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?name
WHERE {
 ?person ex:name ?name .
 {
  { ?person rdf:type ex:Adult }
  UNION
  { ?person ex:age ?age
    FILTER (?age > 17) }
 }
}
```

Sequence & modify

`ORDER BY` to sort

`LIMIT` result number

`OFFSET` rank of first resul

example

results 21 to 40 ordered by name

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name
WHERE {
 ?person rdf:type ex:Person
 ?person ex:name ?name .
}
ORDER BY ?name
LIMIT 20
OFFSET 20
```

UNBOUND

test a variable is not bound ; used for negation as failure

example

persons who are not known authors

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?name
WHERE {
 ?person ex:name ?name .
 OPTIONAL { ?person ex:author ?x }

 FILTER ( ! bound(?x))
}
```

negation

is tricky and errors can easily be made.

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?name
WHERE {
 ?person ex:name ?name .
 ?person ex:knows ?x
 FILTER ( ?x != "Java" )
}
```

? does this find persons who do not know "java" ?

NO! also persons who know something else !

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?name
WHERE {
 ?person ex:name ?name .
 ?person ex:knows ?x
 FILTER ( ?x != "Java" )
}
```

***fabien** ex:knows "Java"*

***fabien** ex:knows "C++"*

fabien is a answer...

YES! persons who are not known to know "java" ... negation of an option...

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?name
WHERE {
  ?person ex:name ?name .
  OPTIONAL { ?person ex:knows ?x
    FILTER ( ?x = "Java" ) }
  FILTER ( ! bound(?x) )
}
```

$\exists$

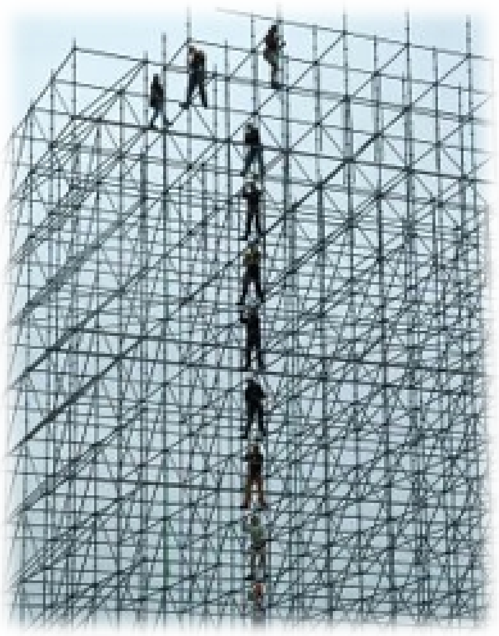
ASK

to check just if there is at least one answer ; result is "true" or "false"

example

is there a person older than 17 ?

```
PREFIX ex: <http://inria.fr/schema#>
ASK
 {
  ?person ex:age ?age
  FILTER (?age > 17)
 }
```



CONSTRUCT

return a specific RDF graph for each result

example

return instances of adults for persons older than 17

```
PREFIX ex: <http://inria.fr/schema#>
CONSTRUCT
{
 ?person rdf:type ex:Adult
}
WHERE
{
 ?person ex:age ?age
 FILTER (?age > 17)
}
```

SPARQL protocol

sending queries and their results accross the web

example
with HTTP Binding

```
GET /sparql/?query=<encoded query> HTTP/1.1
Host: www.inria.fr
User-agent: my-sparql-client/0.1
```

example

with SOAP Binding

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <query-request xmlns="http://www.w3.org/2005/09/sparql-
protocol-types/#">
   <query>SELECT ?x ?p ?y WHERE {?x ?p ?y}</query>
  </query-request>
 </soapenv:Body>
</soapenv:Envelope>
```

Take-away

summary of SPARQL

SPARQL is...

... a query language ...

... a result format ...

... an access protocol ...

... **for RDF**

SPARQL query language

based on the triple model ?x ?p ?y

filters to add constraints

optional parts and alternative parts

fabien, gandon