SPARQL

Hala Skaf-Molli
Associate Professor, HDR
Nantes University
Hala.Skaf@univ-nantes.fr

http://pagesperso.ls2n.fr/~skaf-h/pmwiki/pmwiki.php/Main/SemanticWeb

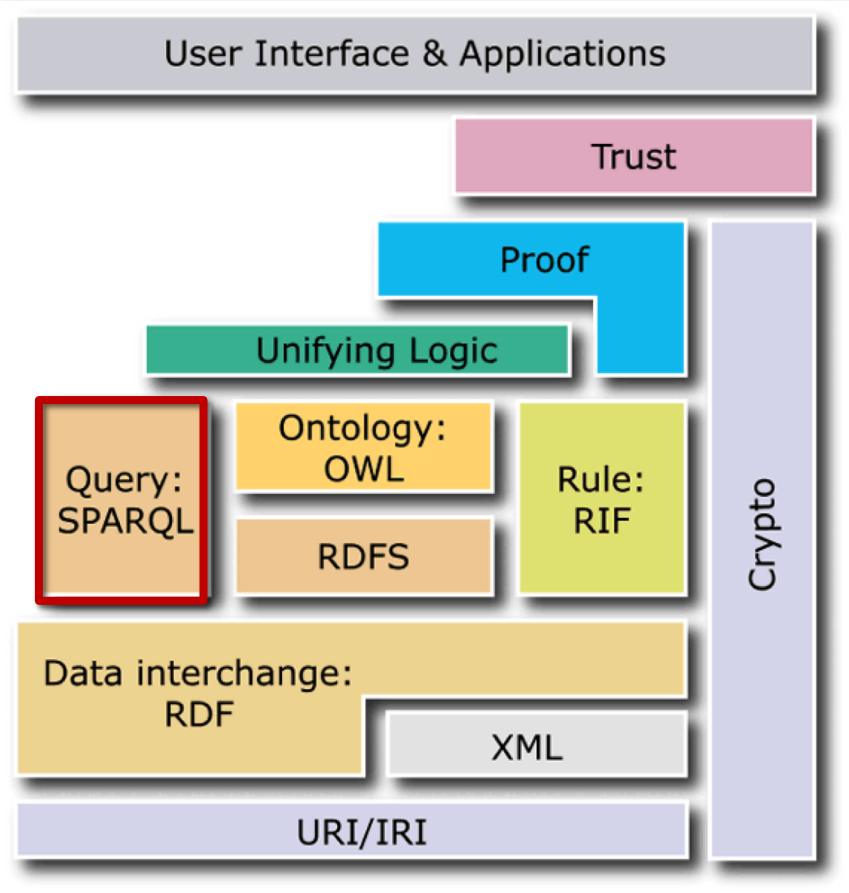
The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Linked data

SPARQL (https://www.w3.org/TR/sparql11-query/)

- RDF **query language + access protocol**
- SPARQL Protocol for RDF
  - Transmission of SPARQL queries and receiving the results
  - **SPARQL endpoint** : implements SPARQL protocol



SPARQL Endpoints

- Service that can
  - receive SPARQL queries sent by a machine
  - receive SPARQL queries typed by a human in a Web interface

http://esw.w3.org/SparqlEndpoints

Project	status	SPARQL endpoint	Webform	comment
BBC Programmes and Music	(2010-06-29) alive	endpoint	Ajax based Visual Query Builder	Powered by OpenLink Virtuoso; also supports Faceted Browsing and Exploration
Bio2RDF	(2010-01-07) alive	List of 40 SPARQL endpoints	n/a	uses OpenLink Virtuoso
BioGateway	(2010-01-07) timeout	endpoint	webform	BioGateway provides many parameterizable SPARQL queries, both biological as ontological, on RDF graphs that were optimized for querying. The graphs have relational closures. Empowered by OpenLink Virtuoso.
BBC Backstage (HP Labs)	(2010-01-07) server not responding	endpoint	webform	uses joseki 3
BBC John Peel sessions from DBTune (Centre for Digital Music, Queen Mary, University of London)	(2010-01-07) alive	endpoint	n/a	dbtune aims to gather and interlink music-related information.
BBC playcount data from DBTune (Centre for Digital Music, Queen Mary, University of London)	(2010-01-07) alive	endpoint	n/a	dbtune aims to gather and interlink music-related information.
DailyMed	(2010-01-07) alive	endpoint		a D2R endpoint
data.gov	(2010-05-22) alive	endpoint	webform	uses OpenLink Virtuoso
data.gov.uk	(2010-02-04) alive	endpoint	webform	The data.gov.uk endpoint
DBLP Bibliography Database published through D2R Server (Freie Universität Berlin)	(2010-01-07) alive	endpoint	webform (Maybe only Firefox)	The DBLP database provides bibliographic information on major computer science journals and conference proceedings.
DBpedia (Freie Universität Berlin, Universität Leipzig, OpenLink Software)	(2010-01-07) alive	endpoint	SNORQL webform (Firefox/Safari/Opera); Ajax based Visual Query Builder	dbpedia.org is a community effort to extract structured information from periodic Wikipedia dumps and to make this information available on the Web. It is served to the public via a live instance of OpenLink Virtuoso, and also offers Faceted Browsing and Exploration

SPARQL Example

Example at http://dbpedia.org/sparql:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/property/>
SELECT ?name ?birth ?death ?person
WHERE {  ?person dbo:birthPlace :Nantes .
          ?person dbo:birthDate ?birth .
          ?person foaf:name ?name .
         ?person dbo:deathDate ?death .
         FILTER (?birth < "1900-01-01"^^xsd:date) . }
ORDER BY ?name
```

SPARQL results:

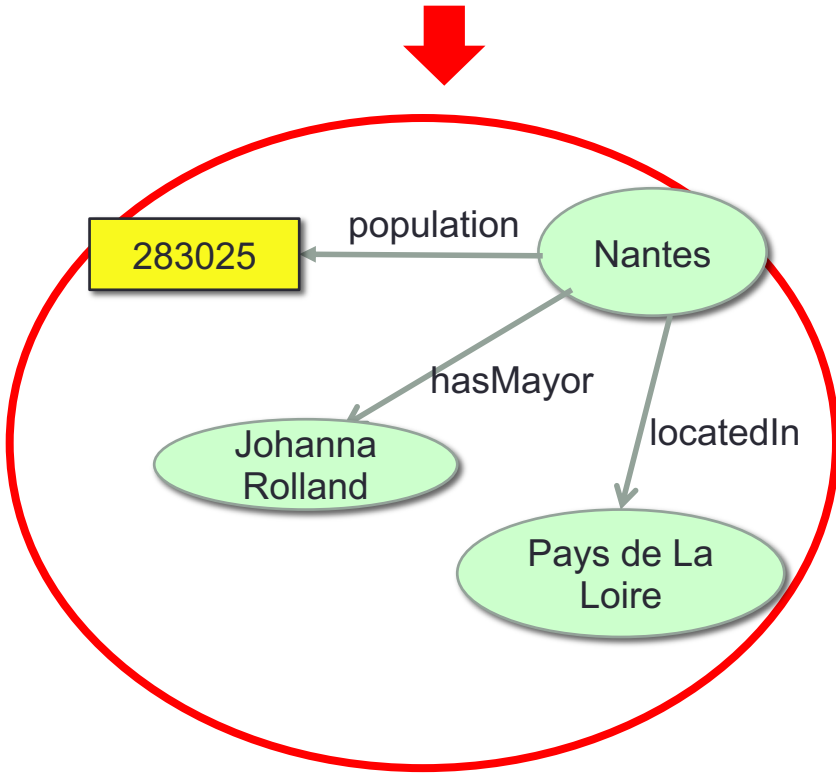
name	birth	death	person
"50pxAlfred Marie-Joseph Heurtaux"@en	"1893-05-19+02:00"^^xsd:date	"1985-12-29+02:00"^^xsd:date	:Alfred_Heurtaux
"Adelaide Dufrenoy"@en	"1765-12-02+02:00"^^xsd:date	"1825-03-07+02:00"^^xsd:date	:Ad%C3%A9la%C3%AFde_Dufr%C3%A9noy
"Adélaïde-Gillette Dufrénoy"@en	"1765-12-02+02:00"^^xsd:date	"1825-03-07+02:00"^^xsd:date	:Ad%C3%A9la%C3%AFde_Dufr%C3%A9noy
"Alfred Heurtaux"@en	"1893-05-19+02:00"^^xsd:date	"1985-12-29+02:00"^^xsd:date	:Alfred_Heurtaux
"Anacharsis Baizeau"@en	"1821-06-02+02:00"^^xsd:date	"1910-02-05+02:00"^^xsd:date	:Anacharsis_Baizeau
"Anne"@en	"1477-01-24+02:00"^^xsd:date	"1514-01-08+02:00"^^xsd:date	:Anne_of_Brittany
"Anne Of Brittany"@en	"1477-01-24+02:00"^^xsd:date	"1514-01-08+02:00"^^xsd:date	:Anne_of_Brittany
"Aristide Briand"@en	"1862-03-27+02:00"^^xsd:date	"1932-03-06+02:00"^^xsd:date	:Aristide_Briand
"Aristide Hignard"@en	"1822-05-19+02:00"^^xsd:date	"1898-03-19+02:00"^^xsd:date	:Aristide_Hignard
"Arthur I"@en	"1187-03-28+02:00"^^xsd:date	"1203-04-02+02:00"^^xsd:date	:Arthur_I,_Duke_of_Brittany
"Auguste Toulmouche"@en	"1829-09-20+02:00"^^xsd:date	"1890-10-15+02:00"^^xsd:date	:Auguste_Toulmouche
"Bl. Mary of the Passion, F.M.M."@en	"1839-05-20+02:00"^^xsd:date	"1904-11-14+02:00"^^xsd:date	:Mary_of_the_Passion
"Charles Lory"@en	"1823-07-29+02:00"^^xsd:date	"1889-05-02+02:00"^^xsd:date	:Charles_Lory
"Charles Monselet"@en	"1825-04-29+02:00"^^xsd:date	"1888-05-18+02:00"^^xsd:date	:Charles_Monselet
"Claude Cahun"@en	"1894-10-24+02:00"^^xsd:date	"1954-12-07+02:00"^^xsd:date	:Claude_Cahun
"Clemence Royer"@en	"1830-04-20+02:00"^^xsd:date	"1902-02-05+02:00"^^xsd:date	:Cl%C3%A9mence_Royer
"Clémence Royer"@en	"1830-04-20+02:00"^^xsd:date	"1902-02-05+02:00"^^xsd:date	:Cl%C3%A9mence_Royer
"Didier Lecour Grandmaison"@en	"1889-05-17+02:00"^^xsd:date	"1917-05-09+02:00"^^xsd:date	:Didier_Lecour_Grandmaison
"Didier Louis Marie Charles Lecour Grandmaison"@en	"1889-05-17+02:00"^^xsd:date	"1917-05-09+02:00"^^xsd:date	:Didier_Lecour_Grandmaison
"Duke of Brittany Peter II"@en	"1418-07-06+02:00"^^xsd:date	"1457-09-21+02:00"^^xsd:date	:Peter_II,_Duke_of_Brittany
"Gaston Serpelle"@en	"1846-11-03+02:00"^^xsd:date	"1904-11-02+02:00"^^xsd:date	:Gaston_Serpette
"Germain Boffrand"@en	"1667-05-15+02:00"^^xsd:date	"1754-03-18+02:00"^^xsd:date	:Germain_Boffrand
"James Tissot"@en	"1836-10-14+02:00"^^xsd:date	"1902-08-07+02:00"^^xsd:date	:James_Tissot
"Jean Alexandre Barre"@en	"1880-05-24+02:00"^^xsd:date	"1967-04-25+02:00"^^xsd:date	:Jean_Alexandre_Barr%C3%A9
"Jean Brochard"@en	"1893-03-11+02:00"^^xsd:date	"1972-06-16+02:00"^^xsd:date	:Jean_Brochard
"Jean Ferdinand Rozier"@en	"1777-11-08+02:00"^^xsd:date	"1863-12-31+02:00"^^xsd:date	:Jean_Ferdinand_Rozier
"Joseph Perotaux"@en	"1883-01-07+02:00"^^xsd:date	"1967-04-22+02:00"^^xsd:date	:Joseph_Perotaux
"Joseph Peroteaux"@en	"1883-01-07+02:00"^^xsd:date	"1967-04-22+02:00"^^xsd:date	:Joseph_Perotaux
"Jules Verne"@en	"1828-02-07+02:00"^^xsd:date	"1905-03-23+02:00"^^xsd:date	:Jules_Verne
"Louis-Albert Bourgault-Ducoudray"@en	"1840-02-01+02:00"^^xsd:date	"1910-07-03+02:00"^^xsd:date	:Louis-Albert_Bourgault-Ducoudray
"Mary Of The Passion"@en	"1839-05-20+02:00"^^xsd:date	"1904-11-14+02:00"^^xsd:date	:Mary_of_the_Passion
"Maxime Maufra"@en	"1861-05-16+02:00"^^xsd:date	"1918-05-22+02:00"^^xsd:date	:Maxime_Maufra
"Michel Coiffard"@en	"1892-07-15+02:00"^^xsd:date	"1918-10-28+02:00"^^xsd:date	:Michel_Coiffard
"Michel Joseph Callixte Marie Coiffard"@en	"1892-07-15+02:00"^^xsd:date	"1918-10-28+02:00"^^xsd:date	:Michel_Coiffard
"Peter II"@en	"1418-07-06+02:00"^^xsd:date	"1457-09-21+02:00"^^xsd:date	:Peter_II,_Duke_of_Brittany
"Pierre Jacques Etienne Cambronne"@en	"1770-12-25+02:00"^^xsd:date	"1842-01-28+02:00"^^xsd:date	:Pierre_Cambronne
"Pierre Jacques Étienne Cambronne"@en	"1770-12-25+02:00"^^xsd:date	"1842-01-28+02:00"^^xsd:date	:Pierre_Cambronne
"Pierre Roy"@en	"1880-08-09+02:00"^^xsd:date	"1950-09-25+02:00"^^xsd:date	:Pierre_Roy_(painter)
"Pierre Waldeck-Rousseau"@en	"1846-12-01+02:00"^^xsd:date	"1904-08-09+02:00"^^xsd:date	:Pierre_Waldeck-Rousseau
"Rene Waldeck-Rousseau"@en	"1846-12-01+02:00"^^xsd:date	"1904-08-09+02:00"^^xsd:date	:Pierre_Waldeck-Rousseau
"Suzanne Malherbe"@en	"1892-07-18+02:00"^^xsd:date	"1972-02-18+02:00"^^xsd:date	:Suzanne_Malherbe

SPARQL

```
PREFIX p: <http://lodpaddle.or/>

SELECT ?mayor
WHERE {
   p:Nantes   p:hasMayor   ?mayor
}
```

Find me all the values for ?mayor such that the triple is true.

Who is the mayor of Nantes?

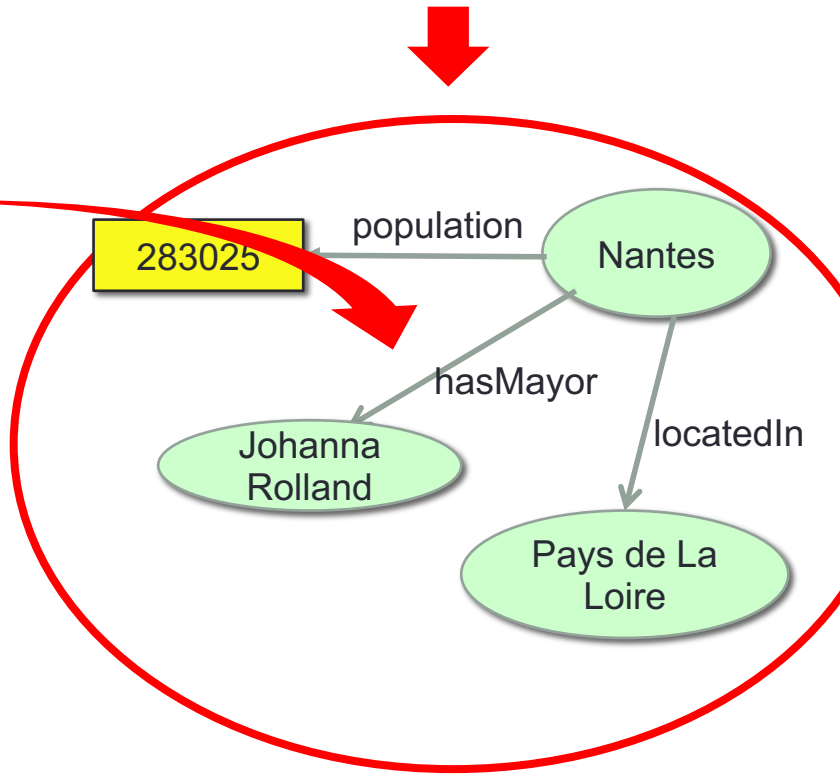


SPARQL: Triple Pattern

SPARQL is based on matching graph patterns

```
PREFIX p: <http://lodpaddle.or/>

SELECT ?mayor
WHERE {
   p:Nantes   p:hasMayor   ?mayor
}
```

Who is the mayor of Nantes?



p:Nantes p:hasMayor ?mayor

mayor

---

p:Joanna Rolland

SPARQL Matching

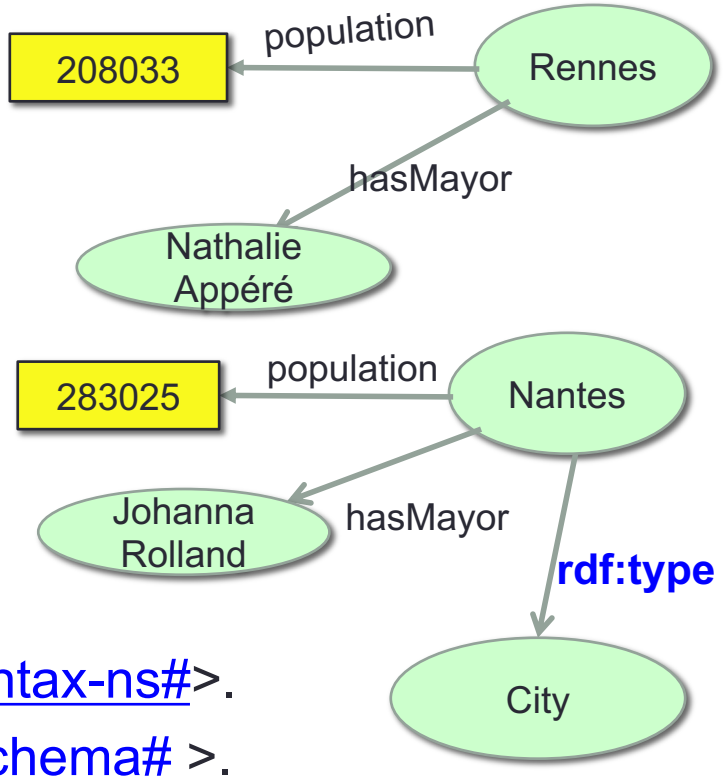
- A triple pattern is *matched* against the RDF dataset
- Each way a pattern can be matched yields a solution
- *Matches the graph*
  - find a set of bindings such that the substitution of variables for values creates a triple that is in the set of triples making up the graph.

Basic Graph Patterns: set of triple patterns

```
PREFIX p: <http://lodpaddle.or/>
PREFIX rdf: <http://w3c.og/..  />

SELECT ?x ?mayor
WHERE {
   ?x   p:hasMayor   ?mayor .
   ?x    rdf:type     p:City
}
```

x	mayor
p:Nantes	p:Johanna Roland

RDF Data



@prefix p: <http://lodpaddle.or/> .
@prefix rdf: <http://w3c.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd: <http://www.w3c.org/2001/XMLSchema# >.

p:Nantes p:poplulation “283025”^^xsd:integer ;
p:hasMoyer p:JohannaRolland ;
rdf:type p:City .
P:Rennes p:population “208022”^^xsd:integer;
p:hasMayor p:NatalieAppéré .

Data Set:

@prefix p: <http://lodpaddle.org/> .
@prefix rdf: http://w3c.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd: http://www.w3c.org/2001/XMLSchema# .


```
p:Nantes        p:poplulation     “283025”^^xsd:integer ;
                p:hasMoyer        p:JohannaRolland ;
                rdf:type          p:City .
P:Rennes        p:population      “208022”^^xsd:integer;
                p:hasMayor        p:NatalieAppéré .
```

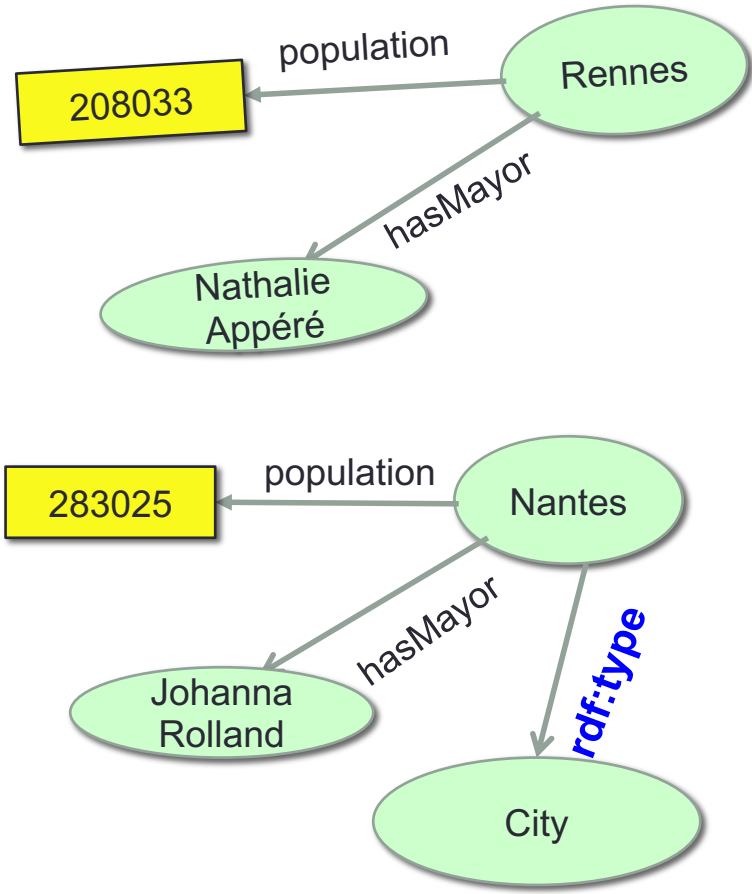
Query:

```
PREFIX p: <http://lodpaddle.org/ >
PREFIX rdf: <http://w3c.org/1999/02/22-rdf-syntax-ns#>

SELECT ?x ?mayor
WHERE {
   ?x   p:hasMayor   ?mayor ;
        rdf:type     p:City
        }
```

x	mayor
p:Nantes	p:Johanna Roland

SPARQL Filter

```
PREFIX p: <http://lodpaddle.org/>
PREFIX rdf: <http://w3c.og/..  />

SELECT ?x
WHERE {
   ?x   p:population   ?nbpop  .
   Filter (?nbpop > 250000)
}
```



x

P:Nantes

SPARQL Filter (2)

Retrieve titles starts with SPARQL .

Data:

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

Query:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "^SPARQL")
        }
```

Query Result:

title
"SPARQL Tutorial"

Scope of Filters (3)

A constraint, expressed by the keyword FILTER, is a restriction on solutions over the whole group in which the filter appears.

Data:

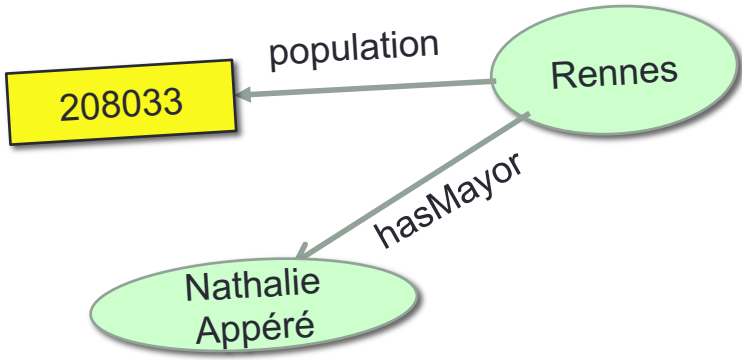
```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

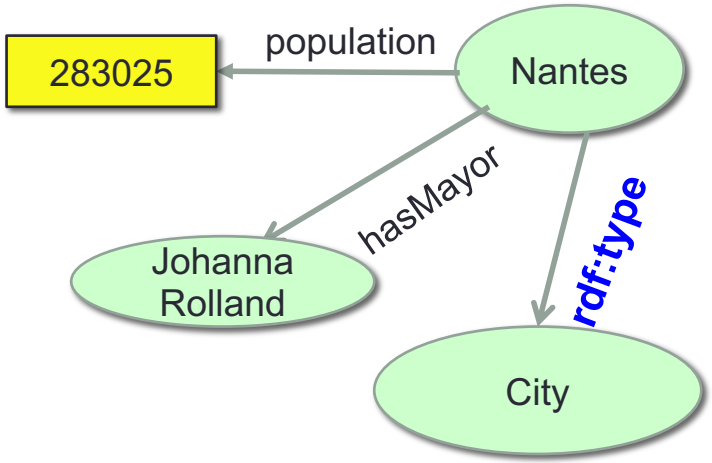
```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

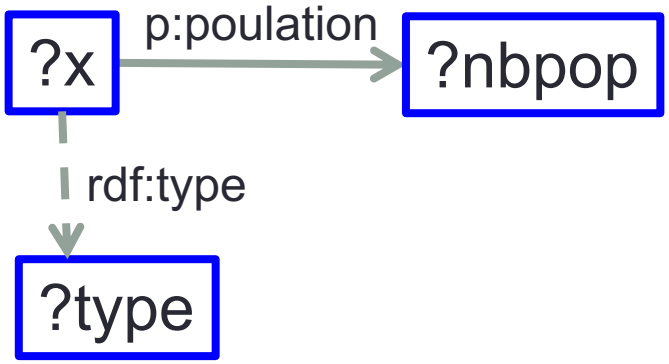
title	price
"The Semantic Web"	23

SPARQL OPTIONAL

```
PREFIX p: <http://lodpaddle.org/>
PREFIX rdf: <http://w3c.og/..  />

SELECT ?x ?type
WHERE {
   ?x   p:population   ?nbpop  .
   OPTIONAL {?x  rdf:type ?type }
}
```





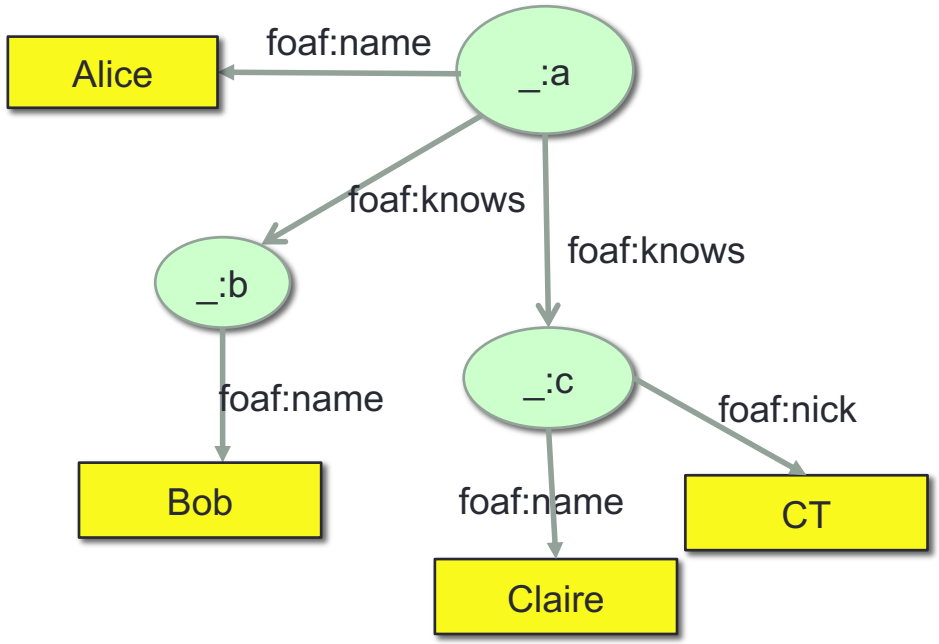


x	type
p:Nantes	p:City
p:Rennes	

SPARQL OPTIONAL

Retrieve the name of a person and the name of her friends and their nickname, if it exist..

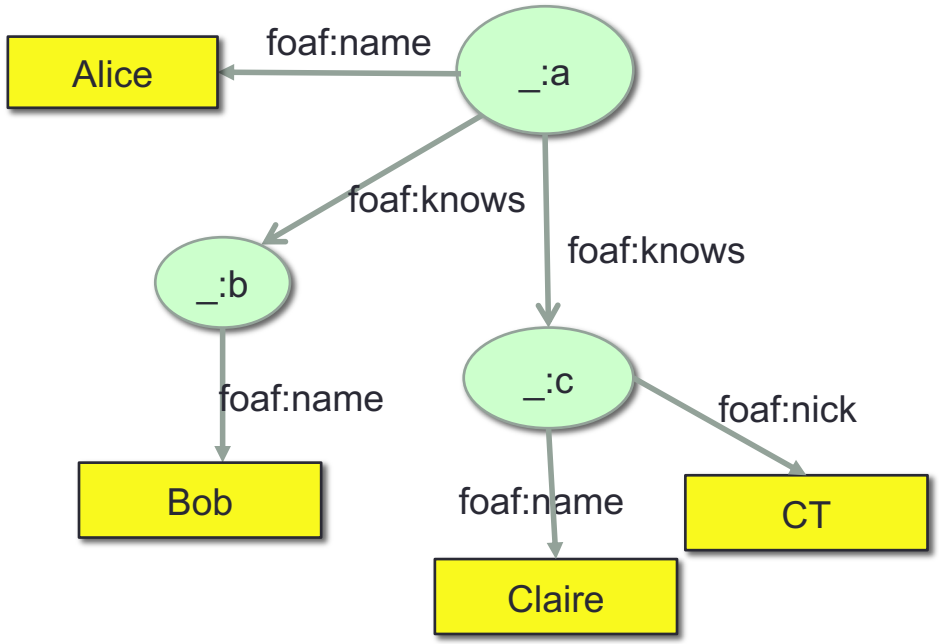
```
PREFIX foaf :http://xmlns.com/foaf/0.1/

SELECT ?nameX ?nameY ?nickY
WHERE {
   ?x     foaf:kowns       ?y;
          foaf:name        ?nameX .
   ?y     foaf:name        ?nameY .
   OPTIONAL {?y foaf:nick ?nickY }
   }
```



SPARQL OPTIONAL

Retrieve the name of a person and the name of her friends and their nickname, if it exist..

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

SELECT ?nameX ?nameY ?nickY
WHERE {
   ?x    foaf:kowns      ?y;
         foaf:name       ?nameX .
   ?y    foaf:name       ?nameY .
   OPTIONAL {?y foaf:nick ?nickY }
   }
```



nameX	nameY	nickY
"Alice"	"Bob"	
"Alice"	"Clare"	"CT"

SPARQL Filter and Optional

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x dc:title ?title .
          OPTIONAL { ?x ns:price ?price . FILTER (?price < 30) }
        }
```

title	price
"SPARQL Tutorial"	
"The Semantic Web"	23

Negation

- Two styles of negation
  - **Filtering results : Not Exists**
  - **Minus: removing solutions related to another pattern**

Filtering Using Graph Patterns Negation: not exists: absence of pattern

Data:

```
@prefix : <http://example/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
 @prefix foaf: <http://xmlns.com/foaf/0.1/> .

:alice rdf:type foaf:Person .
:alice foaf:name "Alice" .
 :bob rdf:type foaf:Person .
```

Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE {
      ?person rdf:type foaf:Person .
      FILTER NOT EXISTS { ?person foaf:name ?name }
}
```

Filtering Using Graph Patterns exists: Presence of a Pattern

Data:

```
@prefix : <http://example/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
 @prefix foaf: <http://xmlns.com/foaf/0.1/> .

:alice rdf:type foaf:Person .
:alice foaf:name "Alice" .
 :bob rdf:type foaf:Person .
```

Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE {
      ?person rdf:type foaf:Person .
      FILTER  EXISTS { ?person foaf:name ?name }
}
```

Negation with MINUS: removing possible solutions

Data:

```
@prefix : <http://example/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

 :alice foaf:givenName "Alice" ;
        foaf:familyName "Smith" .
 :bob foaf:givenName "Bob" ;
         foaf:familyName "Jones" .
  :carol foaf:givenName "Carol" ;
        foaf:familyName "Smith" .
```

Query:

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

SELECT DISTINCT ?s
WHERE { ?s ?p ?o .
MINUS {
?s foaf:givenName "Bob" .
   }
 }
```

- *NOT EXISTS in FILTERs*
  - *detect non-existence*
- *(P1 MINUS P2) as a new binary operator*
  - *“Remove rows with matching bindings”*
  - *only effective when P1 and P2* ***share variables***

Alternative Graph Pattern

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .

_:a  dc10:title     "SPARQL Query Language Tutorial" .
_:a  dc10:creator   "Alice" .

_:b  dc11:title     "SPARQL Protocol Tutorial" .
_:b  dc11:creator   "Bob" .

_:c  dc10:title     "SPARQL" .
_:c  dc11:title     "SPARQL (updated)" .
```

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title  ?title } }
```

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Summary of Graph Patterns

- Different types of graph patterns for the query pattern
- (WHERE clause):
  - Basic graph pattern (BGP)
  - Group graph pattern
  - Optional graph pattern – keyword OPTIONAL
  - Alternative graph pattern – keyword UNION
  - Constraints – keyword FILTER

Solution Modifiers

- **Order by**: put the solutions in order
- **Distinct** : removes duplicates from the result set
- **Offset** : control where the solutions start from in the overall sequence of solutions
- **Limit :** puts an upper bound on the number of solutions returned

Examples

Q1: PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
        ?x foaf:name ?name }
ORDER BY ?name

Q2: PREFIX : <http://example.org/ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: http://www.w3.org/2001/XMLSchema#
SELECT ?name
WHERE { ?x foaf:name ?name ;
                :empId ?emp }
ORDER BY DESC(?emp)
Limit 5

Examples

```
_:x foaf:name "Alice" .
_:x foaf:mbox <mailto:alice@example.com> .
_:y foaf:name "Alice" .
_:y foaf:mbox  <mailto:asmith@example.com> .
_:z foaf:name "Alice" .
_:z foaf:mbox <mailto:alice.smith@example.com> .
```

Q1: PREFIX foaf: <http://xmlns.com/foaf/0.1/>

```
SELECT ?name
WHERE
{
 ?x foaf:name ?name }
```

Q2: PREFIX foaf: <http://xmlns.com/foaf/0.1/>

```
SELECT DISTINCT ?name
WHERE
 {
 ?x foaf:name ?name }
```

SPARQL Query forms

- **Select**
  - Sequence of results (i.e. sets of variable bindings)
  - Selected variables separated by space (not by comma!)
- **Construct**
  - Returns an RDF graph created from a template
  - Template: graph pattern with variables from the query pattern
- **Describe**
  - Returns an RDF graph with data about resources
  - Nondeterministic (i.e. query processor determines the actual structure of the returned RDF graph).
- **Ask**
  - Check whether there is at least one answer

Construct

Question : construct the *foaf* graph containing the name of employees

```
@prefix org:     <http://example.com/ns#> .

_:a   org:employeeName    "Alice" .
_:a   org:employeeId      12345 .

_:b   org:employeeName    "Bob" .
_:b   org:employeeId      67890 .
```

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX org:    <http://example.com/ns#>

CONSTRUCT { ?x foaf:name ?name }
WHERE  { ?x org:employeeName ?name }
```

```
@prefix org: <http://example.com/ns#> .

_:x foaf:name "Alice" .
_:y foaf:name "Bob" .
```

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    >
  <rdf:Description>
    <foaf:name>Alice</foaf:name>
  </rdf:Description>
  <rdf:Description>
    <foaf:name>Bob</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

ASK

- Test whether or not a query pattern has a solution.
- No information is returned about the possible query solutions, just whether or not a solution exists.

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name       "Alice" .
_:a  foaf:homepage   <http://work.example.org/alice/> .

_:b  foaf:name       "Bob" .
_:b  foaf:mbox       <mailto:bob@work.example> .
```

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" }
```

```
yes
```

Does Alice has a mailbox ?

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" ;
          foaf:mbox  <mailto:alice@work.example> }
```

```
no
```

DESCRIBE

- The DESCRIBE form returns a single result RDF graph containing RDF data about resources

```
PREFIX ent:  <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
```

might return a description of the employee and some other potentially useful details:

```
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix vcard:  <http://www.w3.org/2001/vcard-rdf/3.0> .
@prefix exOrg:  <http://org.example.com/employees#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#>

_:a     exOrg:employeeId    "1234" ;

        foaf:mbox_shalsum   "ABCD1234" ;
        vcard:N
         [ vcard:Family       "Smith" ;
           vcard:Given        "John"  ] .

foaf:mbox_shalsum  rdf:type  owl:InverseFunctionalProperty .
```

RDF Data Sets

- A SPARQL query is executed against an *RDF Dataset*
- An **RDF Dataset** comprises
  - One **default graph**
  - and zero or more **named graphs**(identified by an IRI).
- Keyword GRAPH make one of the named graph the **active graph** used for patterns matching

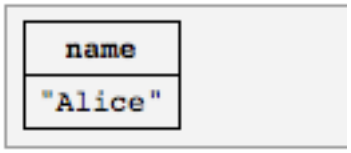
Using Select-From-Where

#Default graph (storedt at http://example.org/foaf/aliceFoaf)

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

_:a foaf:name ‘’Alice’’ .
_:a foaf:mbox mailto:alice@work.example

PREFIX foaf : <http://xmlns.com/foaf/0.1/> .
SELECT ?name
FROM <http://example.org/foaf/aliceFoaf>
WHERE { ?x foaf:name ?name }



Named Graph

Defaut graph (stored at http://example.org/dft.ttl)
@prefix dc: <http://pur.org/dc/elements/1.1>.

<http://example.org/bob> dc:publisher "Bob Hacker".
<http://example.oeg/alice> dc:publisher "Alice Hacker".

Named graph :http://example.org/bob
@prefix foaf : <http://xmlns.com/foaf/0.1/> .

_:a foaf:name ''Bob'' .
_:a foaf:mbox <mailto:bob@work.example.org>

Named graph :http://example.org/alice
@prefix foaf : <http://xmlns.com/foaf/0.1/> .

_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org>

GRAPH FROM NAMED

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?who ?src ?mbox
FROM <http://example.org/dft.ttl>
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE
{
   ?g dc:publisher ?who .
   GRAPH ?src { ?x foaf:mbox ?mbox }
}
```

```
# Defaut graph (stored at http://example.org/dft.ttl)
@prefix dc: <http://pur.org/dc/elements/1.1>.

<http://example.org/bob>   dc:publisher  “Bob Hacker”.
<http://example.oeg/alice> dc:publisher  “Alice Hacker”.

 # Named graph :http://example.org/bob
@prefix foaf : <http://xmlns.com/foaf/0.1/> .

_:a        foaf:name          ‘’Bob’’ .
_:a        foaf:mbox
           <mailto:bob@work.example.org>

# Named graph :http://example.org/alice
@prefix foaf : <http://xmlns.com/foaf/0.1/> .

_:a        foaf:name          ‘’Alice’’ .
_:a        foaf:mbox
           <mailto:alice@work.example.org>
```

```
SELECT ?who ?src ?mbox
FROM <http://example.org/dft.ttl>
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE
{
   ?src dc:publisher ?who .
   GRAPH ?src { ?x foaf:mbox ?mbox }
}
```

DataSet:

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix g:  <tag:example.org,2005-06-06:> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .

g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph1
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph2
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

Find email addresses, detailing the name of the person and the date the information was discovered.

DataSet:


```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix g:  <tag:example.org,2005-06-06:> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .

g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph1
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph2
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>

SELECT ?name ?mbox ?date
WHERE
  {  ?g dc:publisher ?name ;
        dc:date ?date .
    GRAPH ?g
      { ?person foaf:name ?name ; foaf:mbox ?mbox }
  }
```

name	mbox	date
"Bob"	<mailto:bob@oldcorp.example.org>	"2004-12-06"^^xsd:date
"Bob"	<mailto:bob@newcorp.example.org>	"2005-01-10"^^xsd:date

SPARQL 1.1 Federated Query

- Remote source

http://people.example.org

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:people15 foaf:name "Alice" .
:people16 foaf:name "Bob" .
:people17 foaf:name "Charles" .
:people18 foaf:name "Daisy" .
```

- Local data:

http://example.org/myfoaf.rdf

```
<http://example.org/myfoaf/I>
  <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/people15> .
```

SPARQL 1.1 Federated Query

Remote source
<http://people.example.org/sparql>

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

 :people15 foaf:name "Alice" .
 :people16 foaf:name "Bob" .
 :people17 foaf:name "Charles" .
 :people18 foaf:name "Daisy" .
```

Local data: http://example.org/myfoaf.rdf

```
<http://example.org/myfoaf/I>
    <http://xmlns.com/foaf/0.1/knows>
        <http://example.org/people15> .
```

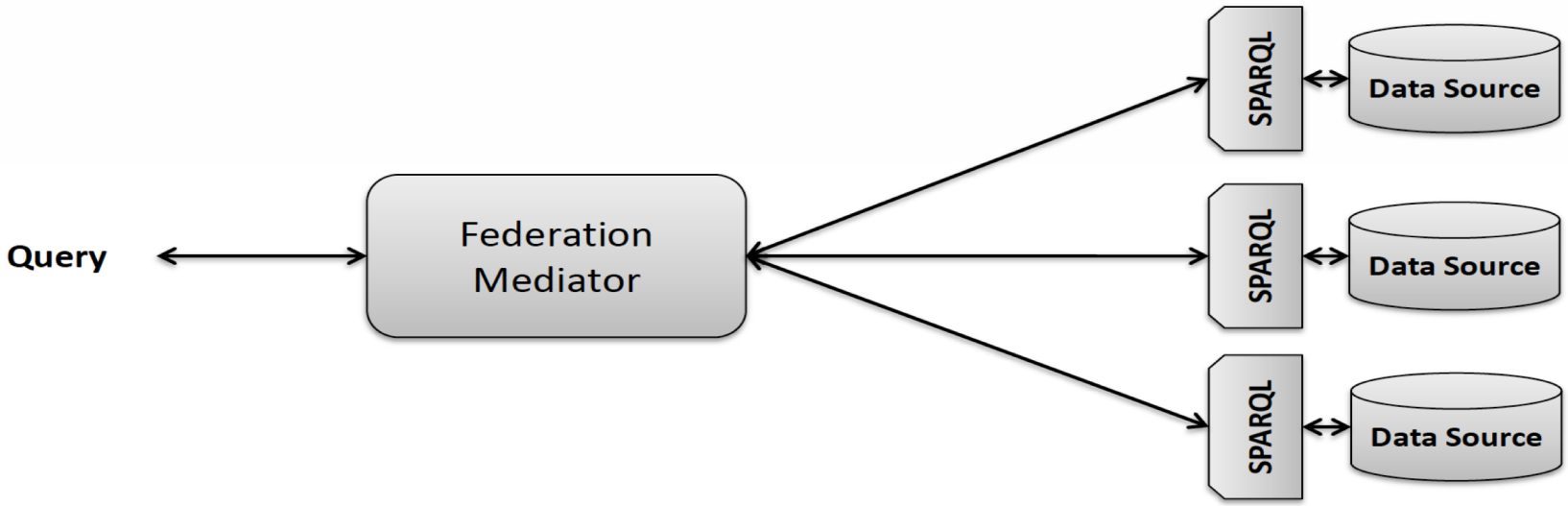
```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/myfoaf.rdf>
WHERE
{
  <http://example.org/myfoaf/I>
foaf:knows ?person .
  SERVICE
<http://people.example.org/sparql> {
    ?person foaf:name ?name . }
}
```

```
name
--------
Alice
```

Federated Query Processing

Federation mediator at the server

- ➔ Virtual integration of (remote) data sources
- ➔ Communication via SPARQL protocol



Source: https://www.semanticscholar.org/paper/FedX

Federated Query Processing

Example Query

Find US presidents and associated news articles

```
SELECT ?President ?Party ?TopicPage WHERE {
   ?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .
   ?nytPresident owl:sameAs ?President .
   ?President dbpedia:party ?Party .
   ?nytPresident nytimes:topicPage ?TopicPage .
}
```

Summary

- SPARQL is a protocol and query language for RDF data model..
- It designed for open, decentralized Web
- **Select** is suitable for querying known endpoint with known vocabularies
- **Other Forms:**
  - **Describe** is suitable for known IRI and unknown vocabularies, the results is a RDF graph describing the requested resource
  - **Ask** discover which SPARQL endpoint could answer the query
  - **Construct** to build a graph as a result