SPARQL

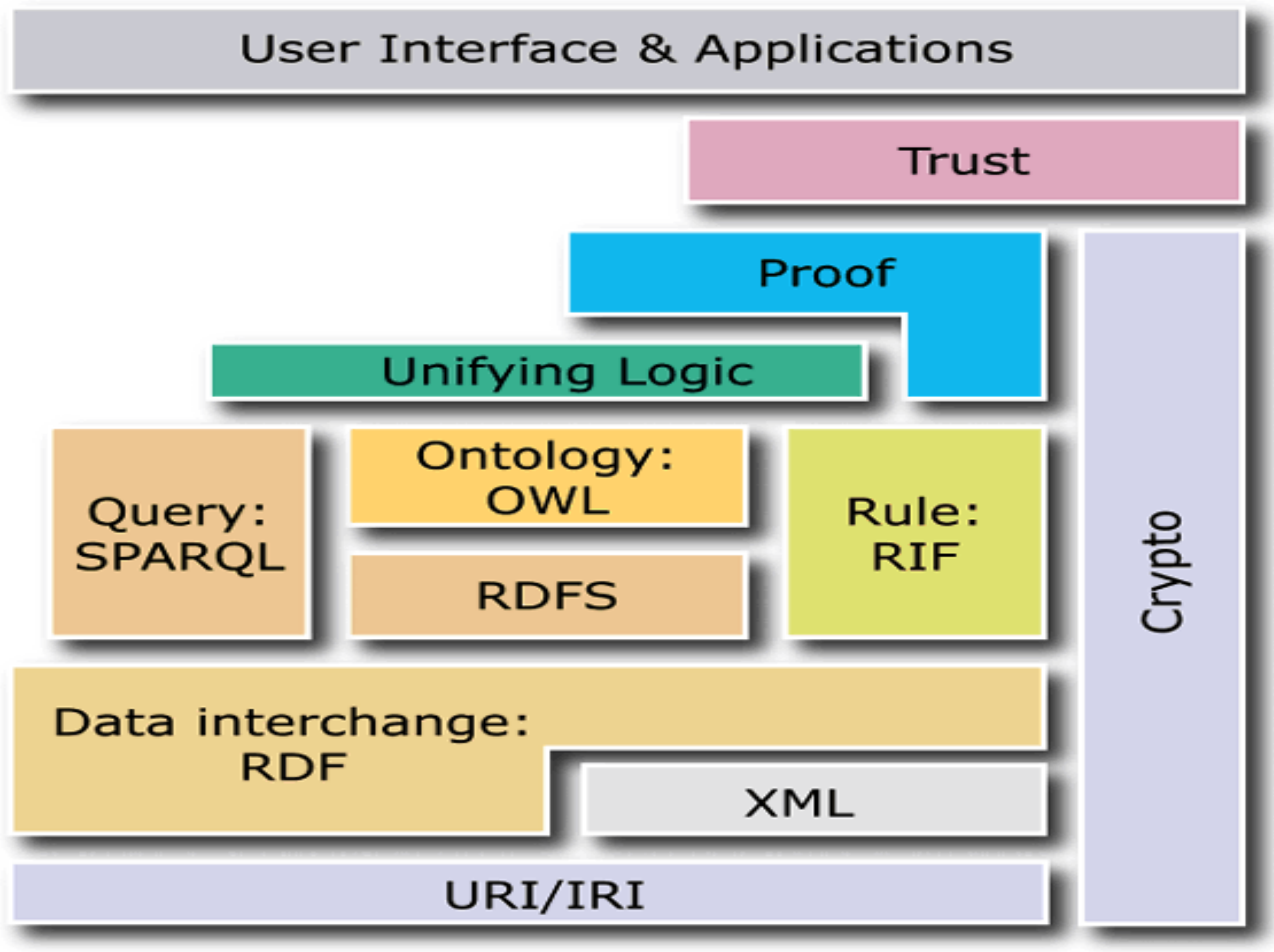
Hala Skaf-Molli
Associate Professor
Nantes University
Hala.Skaf@univ-nantes.fr
http://pagesperso.lina.univ-nantes.fr/~skaf-h

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

Semantic Web Cake



SPARQL

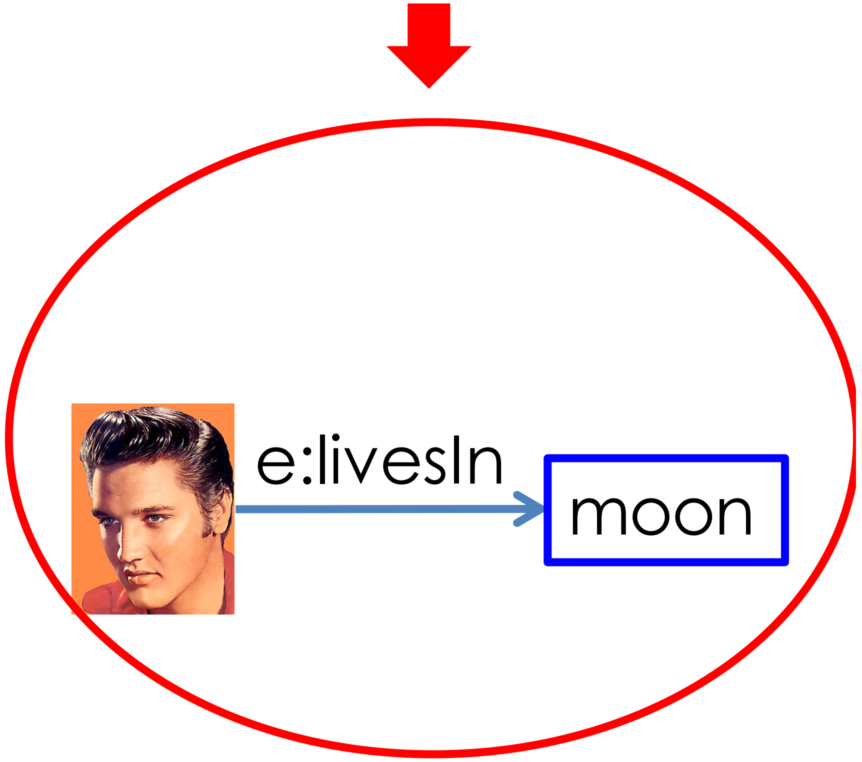
- SAPRQL is the query language for the Semantic Web
- RDF query language + access protocol
- SPARQL Protocol for RDF
  - Transmission of SPARQL queries and the results
  - **SPARQL endpoint**: Web service that implements the protocol

SPARQL

```
PREFIX e: <http://elvis.org/>

SELECT ?loc
WHERE {
   e:elvis   e:livesIn   ?loc
}
```

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

Elvis, where are you?

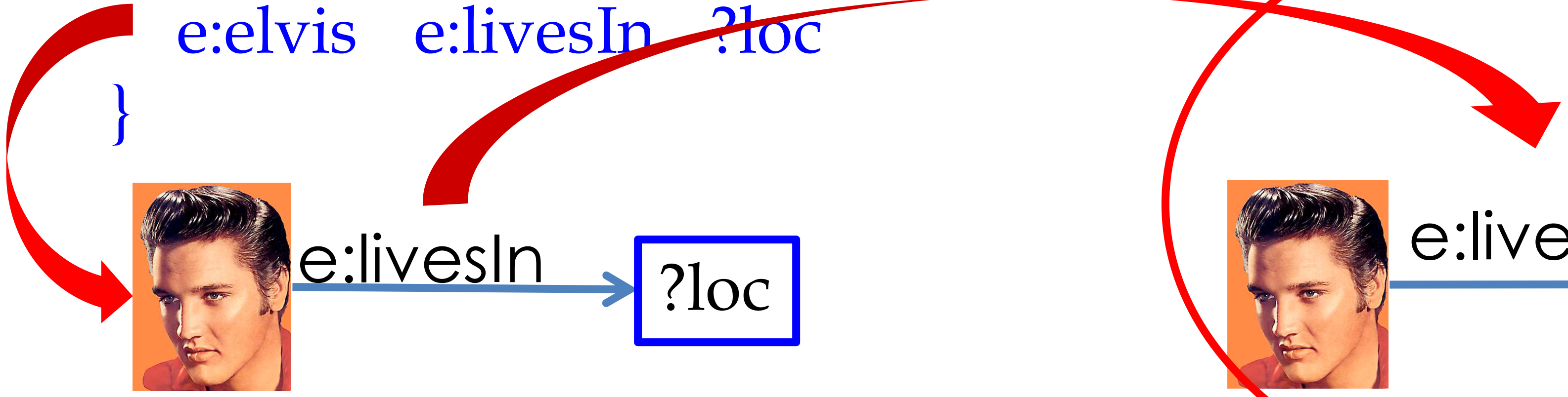


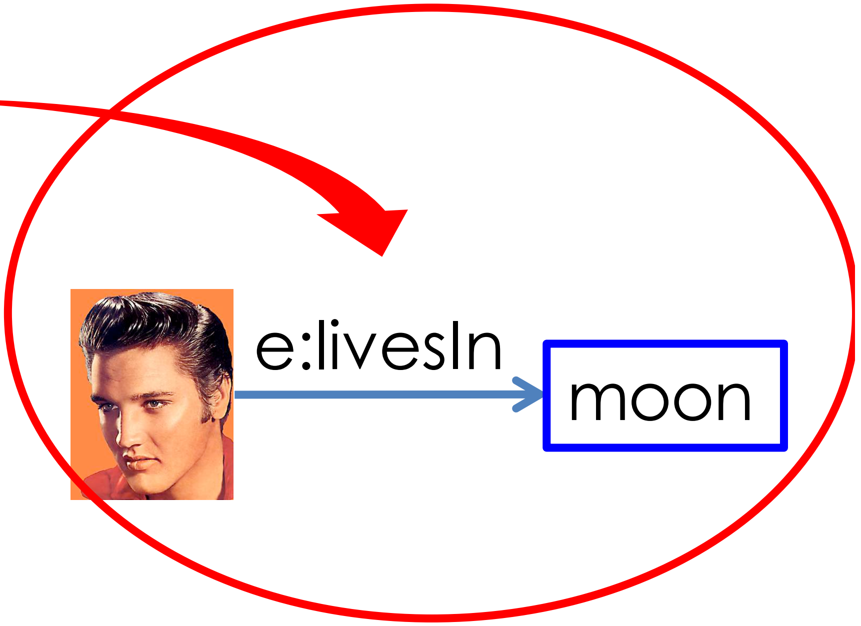
SPARQL Matching

SPARQL is based on matching graph patterns

```
PREFIX e: <http://elvis.org/>

SELECT ?loc
WHERE {
  e:elvis   e:livesIn   ?loc
}
```

Elvis, where are you?





```
?loc
-----------
 e:moon
```

SPARQL Matching

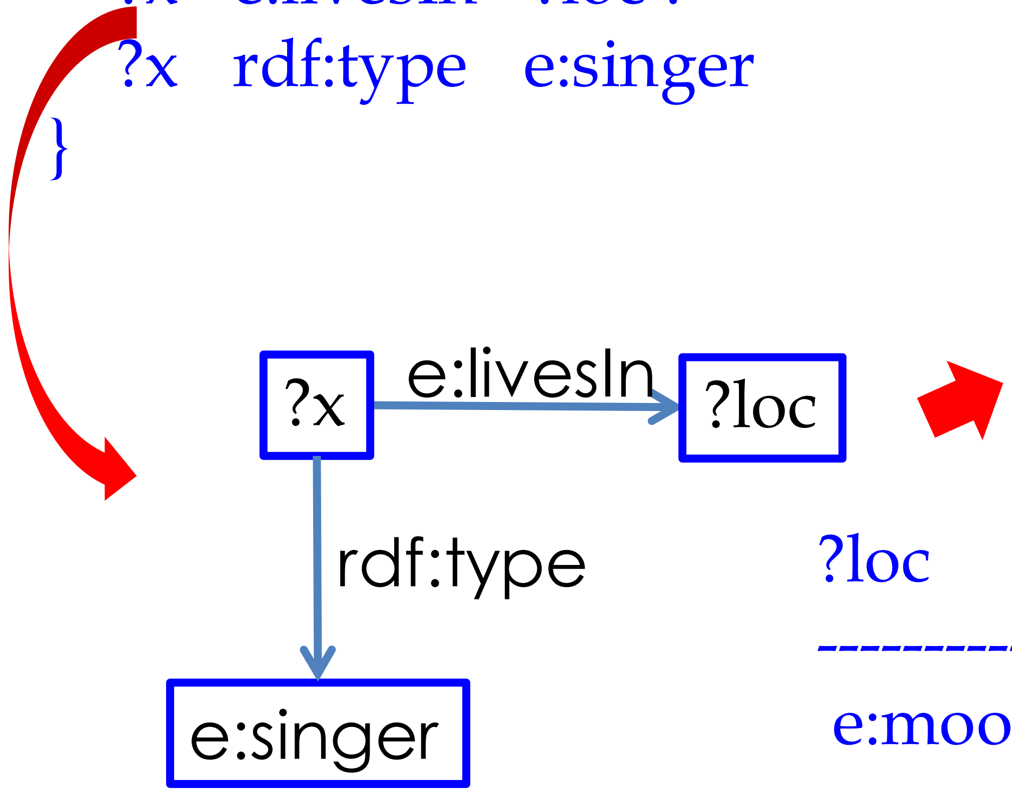
- A triple pattern is *matched* against the RDF data
- Each way a pattern can be matched yields a solution
- *Matches the graph*
  - find a set of bindings such that the substitution of variables for values creates a triple that is in the set of triples making up the graph.

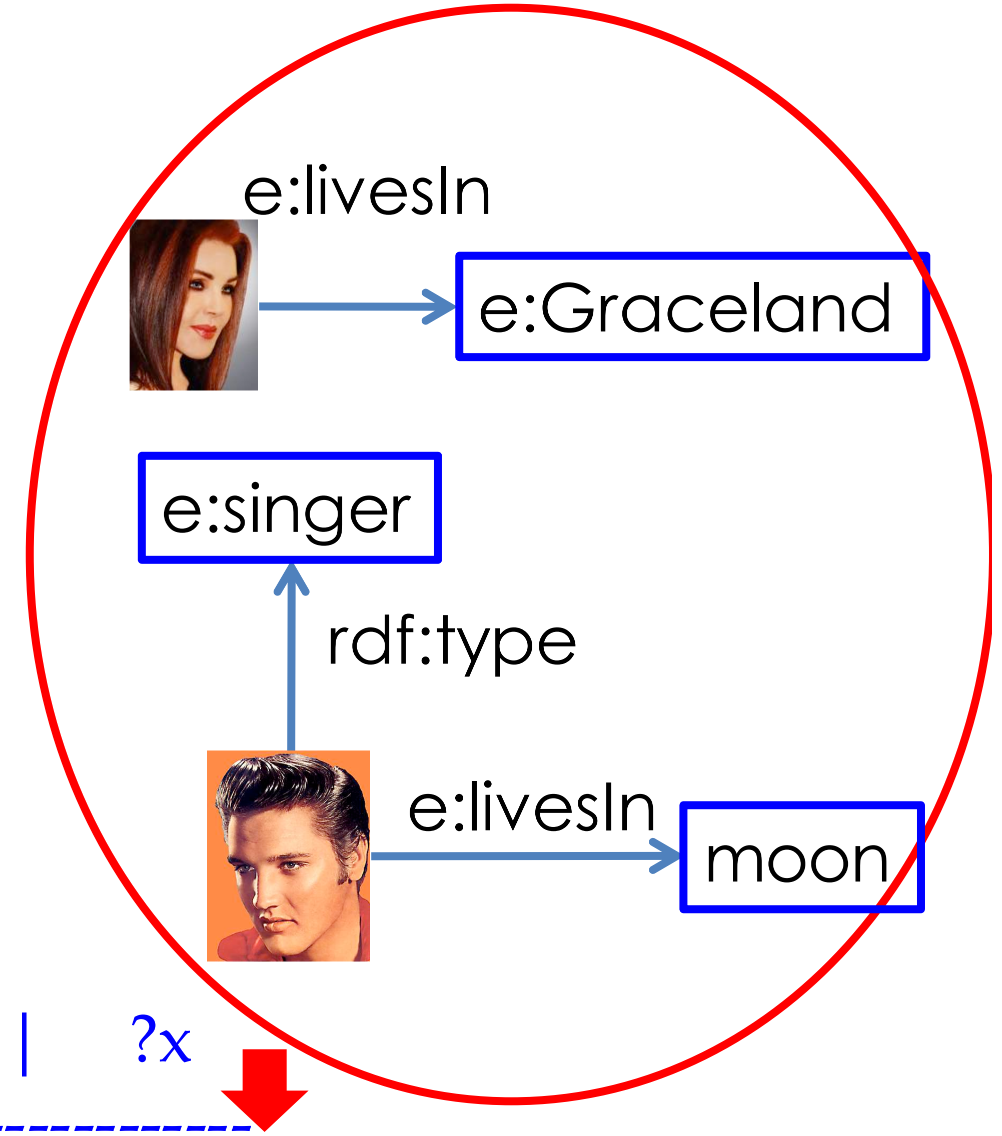
SPARQL Patterns

- **Triple pattern :** like an RDF triple, but with the possibility of a variable instead of an RDF term in the subject, predicate, or object positions
  - e:elvis e:livesIn **?loc**
- **Basic Graph Pattern (BGP)** is a set of triple patterns
- **Group Graph Pattern** set of BGP delimited by { }
- **Optional Graph Pattern**
- **Alternative Graph Pattern**
- **Patterns on Named Graphs**

Basic Graph Patterns

```
PREFIX e: <http://elvis.org/>
PREFIX rdf: <http://w3c.org/...>

SELECT ?loc, ?x
WHERE {
   ?x   e:livesIn   ?loc .
   ?x   rdf:type   e:singer
}
```

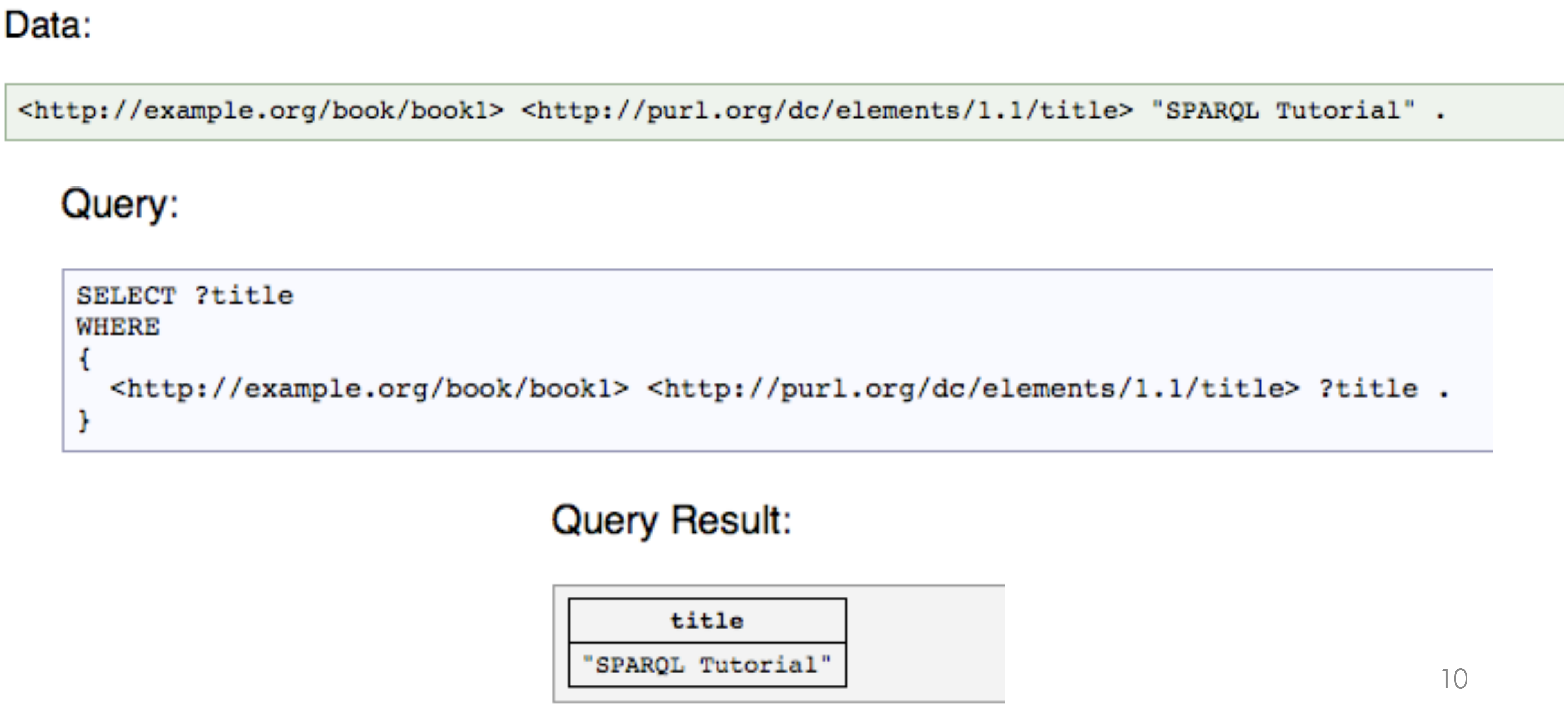




```
?loc      |    ?x
----------------------
 e:moon  |  e:Elvis
```

BGP Example 1: Simple Matching

- Query: Find the title of a book.



Data:

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

Query:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Query Result:

title
"SPARQL Tutorial"

BGP Example 2: Multiple Matches

Q: Retrieve name and mailbox ..

Data:

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

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Query:

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

Query Result:

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

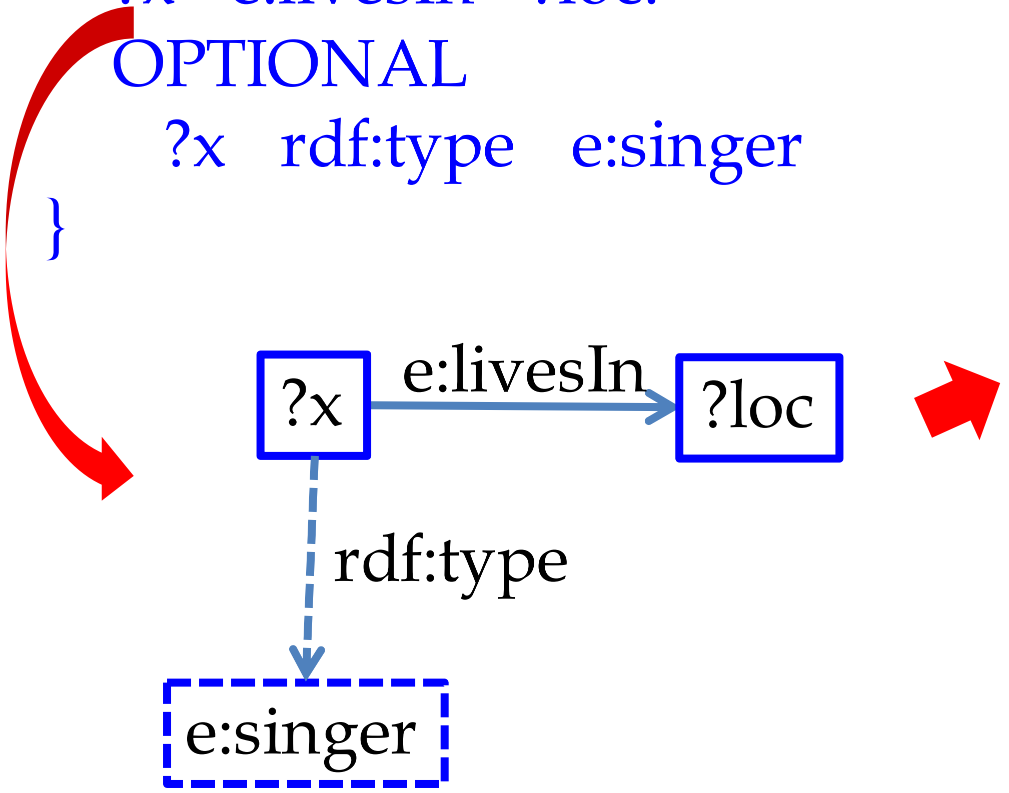
Simplified Syntaxes

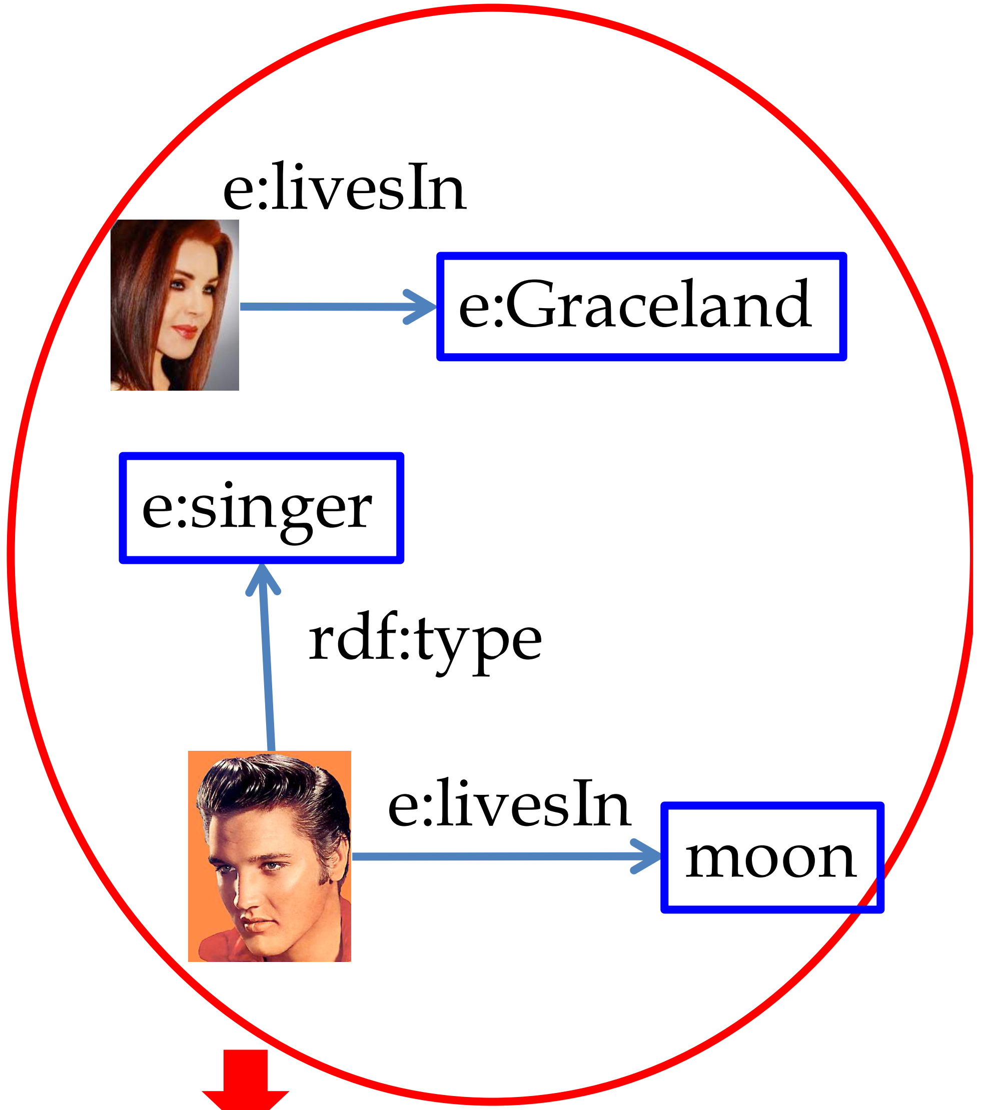
```
SELECT ?name ?mbox
   WHERE {
  ?x foaf:name ?name .
   ?x foaf:mbox ?mbox
}
```

```
SELECT ?name ?mbox
   WHERE {
  ?x foaf:name ?name ;
       foaf:mbox ?mbox.
}
```

Optional Graph Pattern (1)

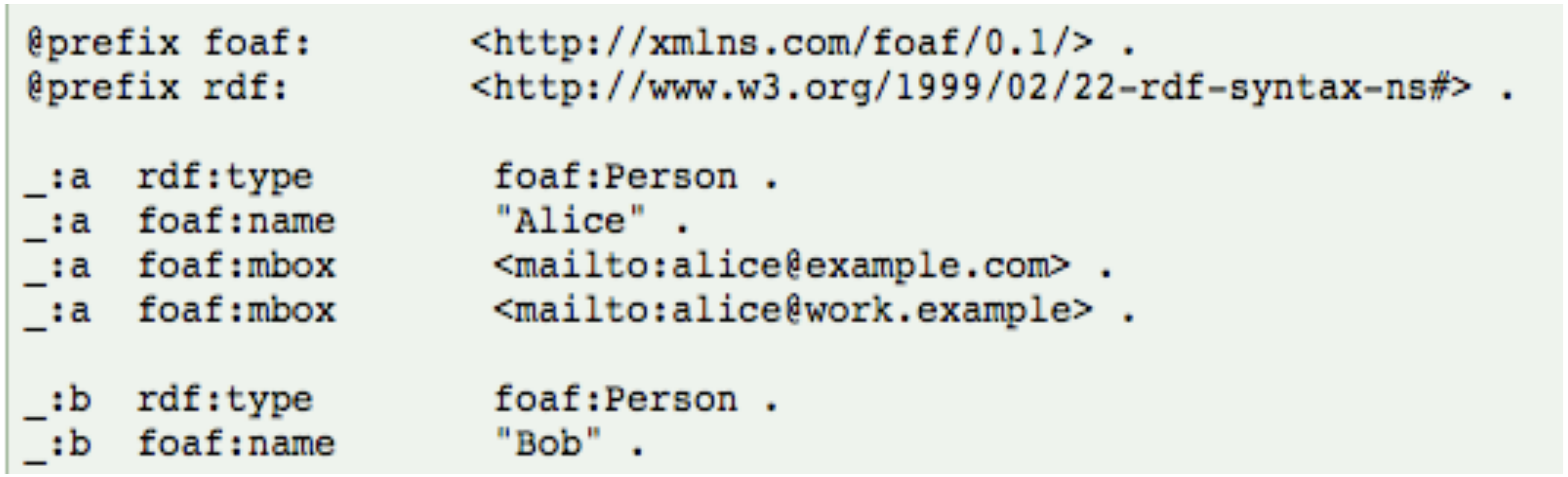
```
PREFIX e: <http://elvis.org/>
PREFIX rdf: <http://w3c.org/...>

SELECT  ?x
WHERE {
   ?x   e:livesIn   ?loc.
   OPTIONAL
      ?x   rdf:type   e:singer
}
```



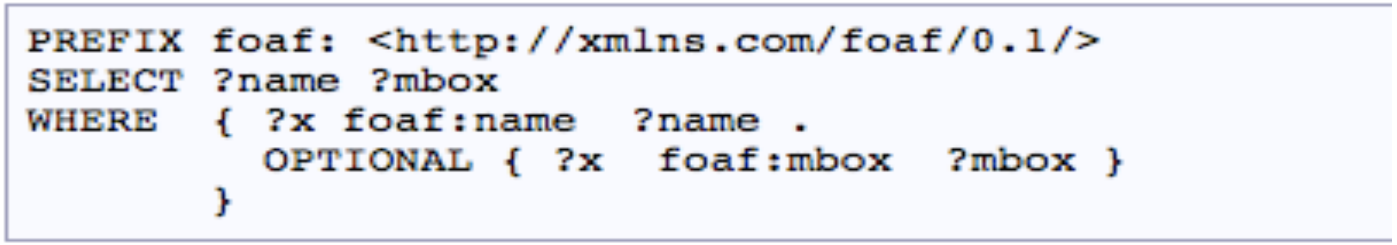


?x = e:Priscilla
?x = e:Elvis

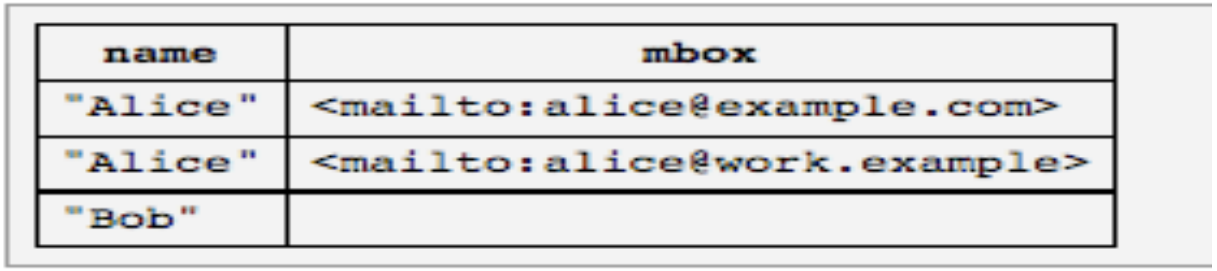
Optional Graph Pattern (2)



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

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

_:b  rdf:type        foaf:Person .
_:b  foaf:name       "Bob" .
```



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



name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example>
"Bob"	

Exercise (1)

Retrieve the names of friends and their nickname, if it exist..

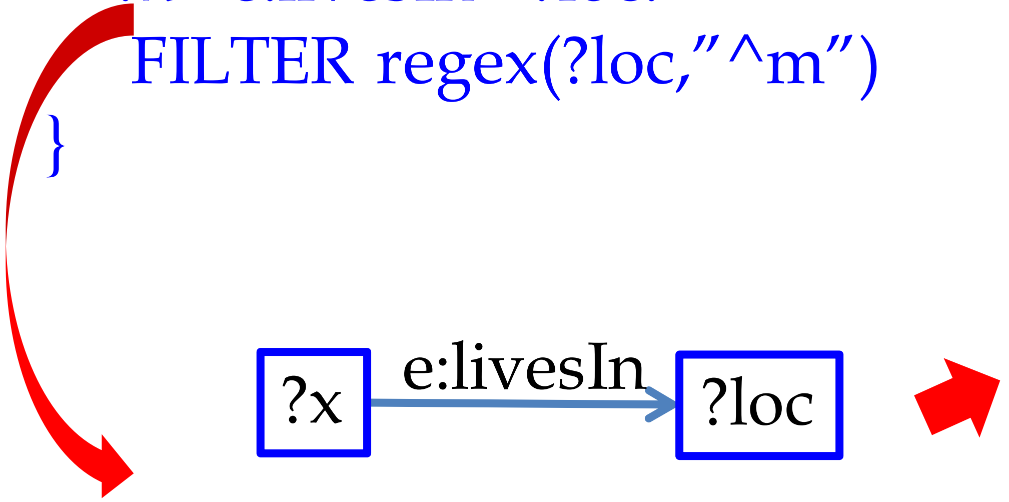
```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .

_:a     foaf:name    "Alice" .
_:a     foaf:knows   _:b .
_:a     foaf:knows   _:c .

_:b     foaf:name    "Bob" .

_:c     foaf:name    "Clare" .
_:c     foaf:nick    "CT" .
```

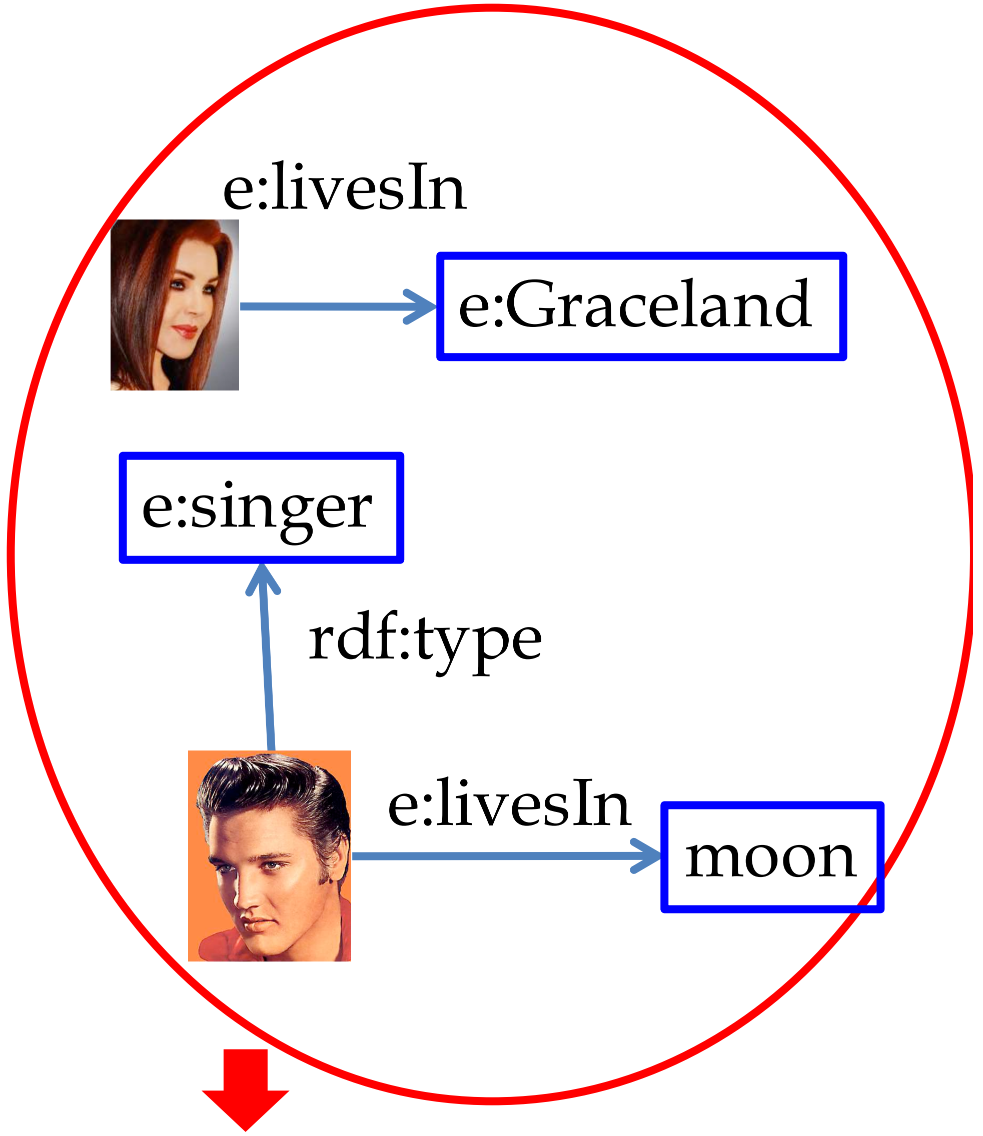
```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
  { ?x foaf:knows ?y ;
       foaf:name ?nameX .
    ?y foaf:name ?nameY .
    OPTIONAL { ?y foaf:nick ?nickY }
  }
```

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

SPARQL Filters (1)

```
PREFIX e: <http://elvis.org/>
PREFIX rdf: <http://w3c.org/...>

SELECT ?loc ?x
WHERE {
   ?x   e:livesIn   ?loc.
   FILTER regex(?loc,"^m")
}
```





?x = e:Elvis

Results can be filtered through external boolean functions.

SPARQL Filters (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")
        }
```

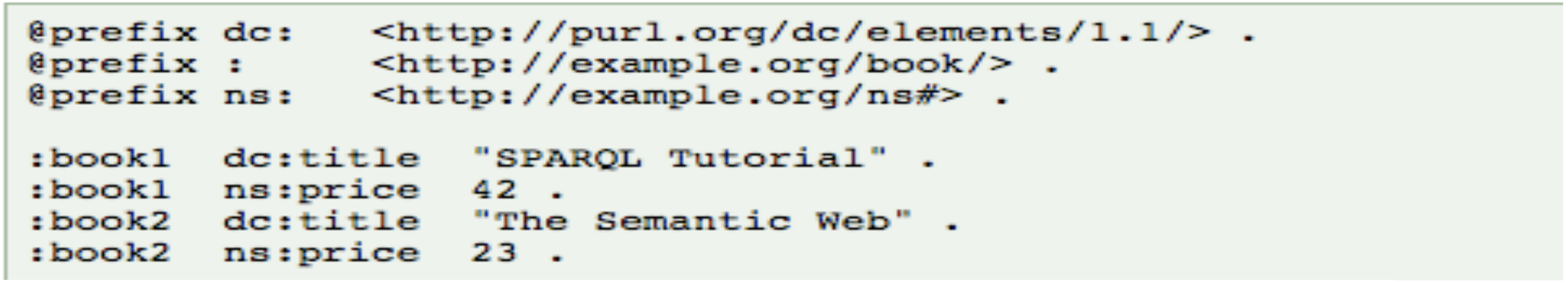
Query Result:

title
"SPARQL Tutorial"

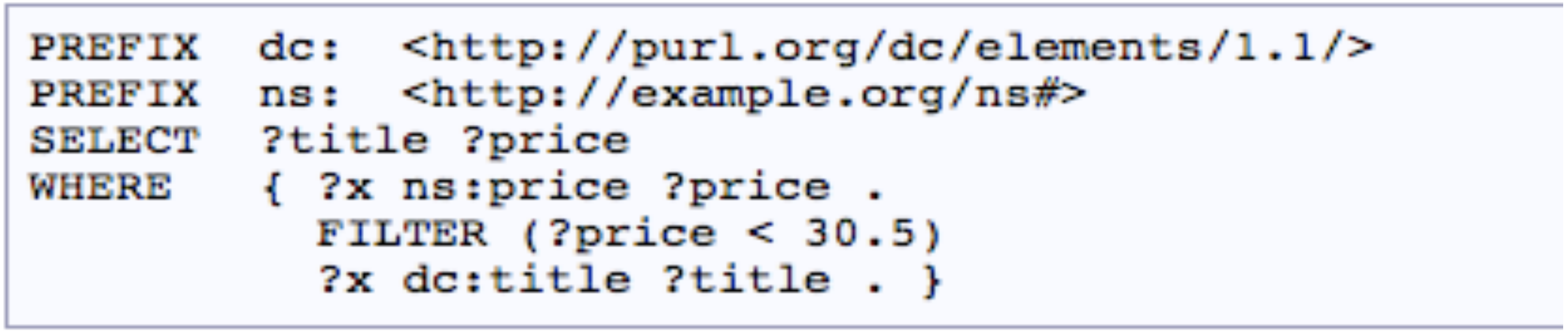
Scope of Filters (3)

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

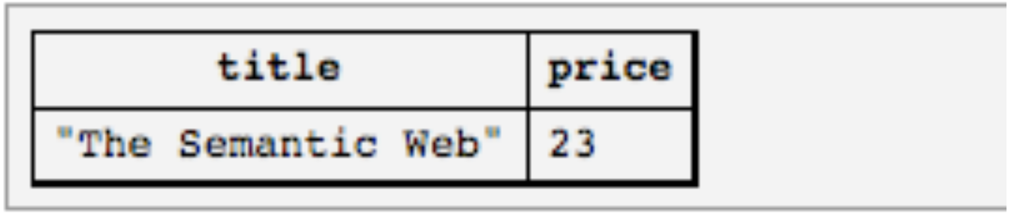
Data:



```
@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 .
```

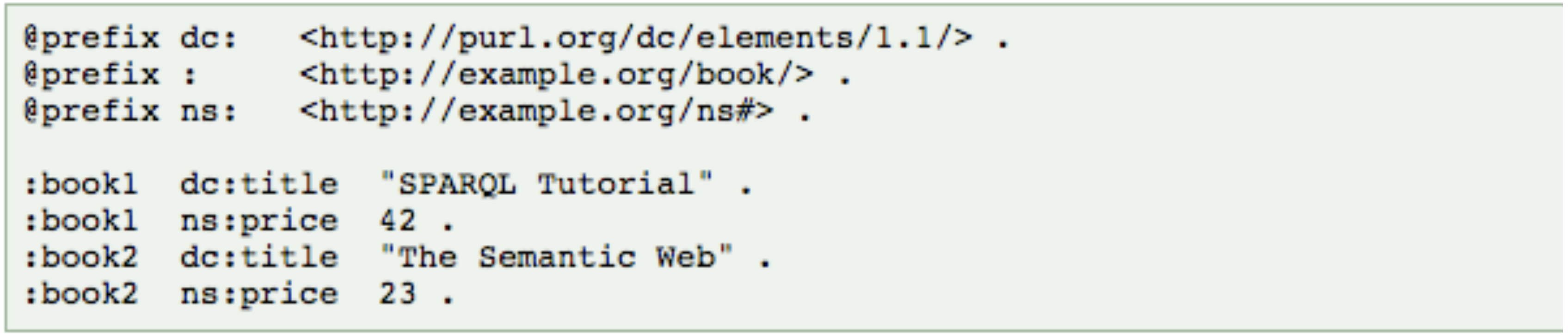


```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

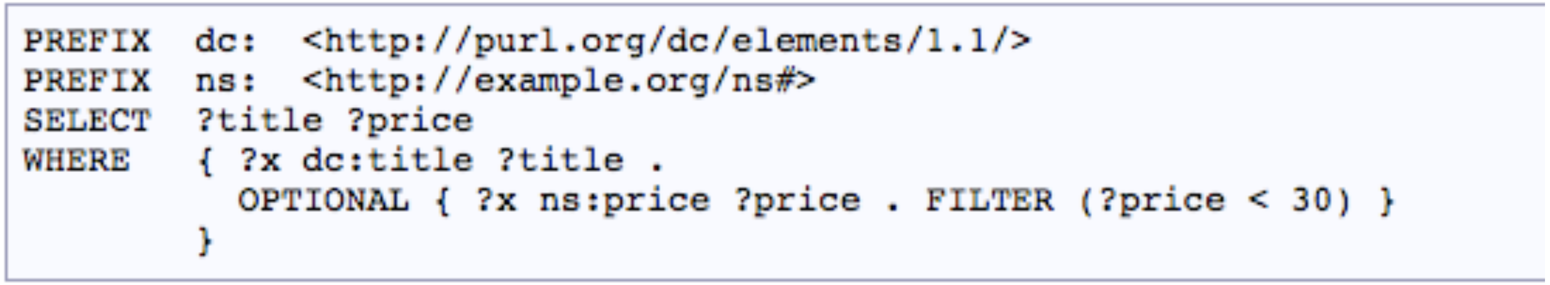


title	price
"The Semantic Web"	23

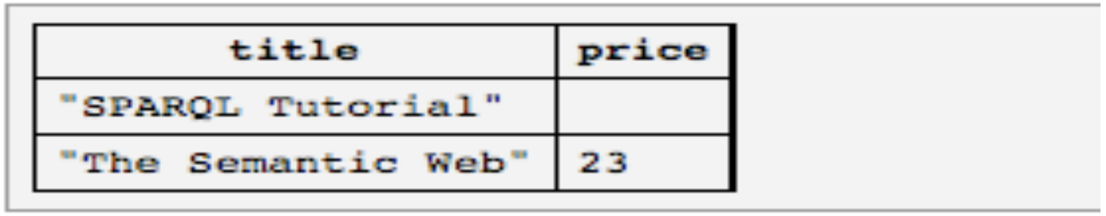
SPARQL Filters and Optional



```
@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

Filtering Using Graph Patterns Negation: not exists

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

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

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 Pattern

- 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
```

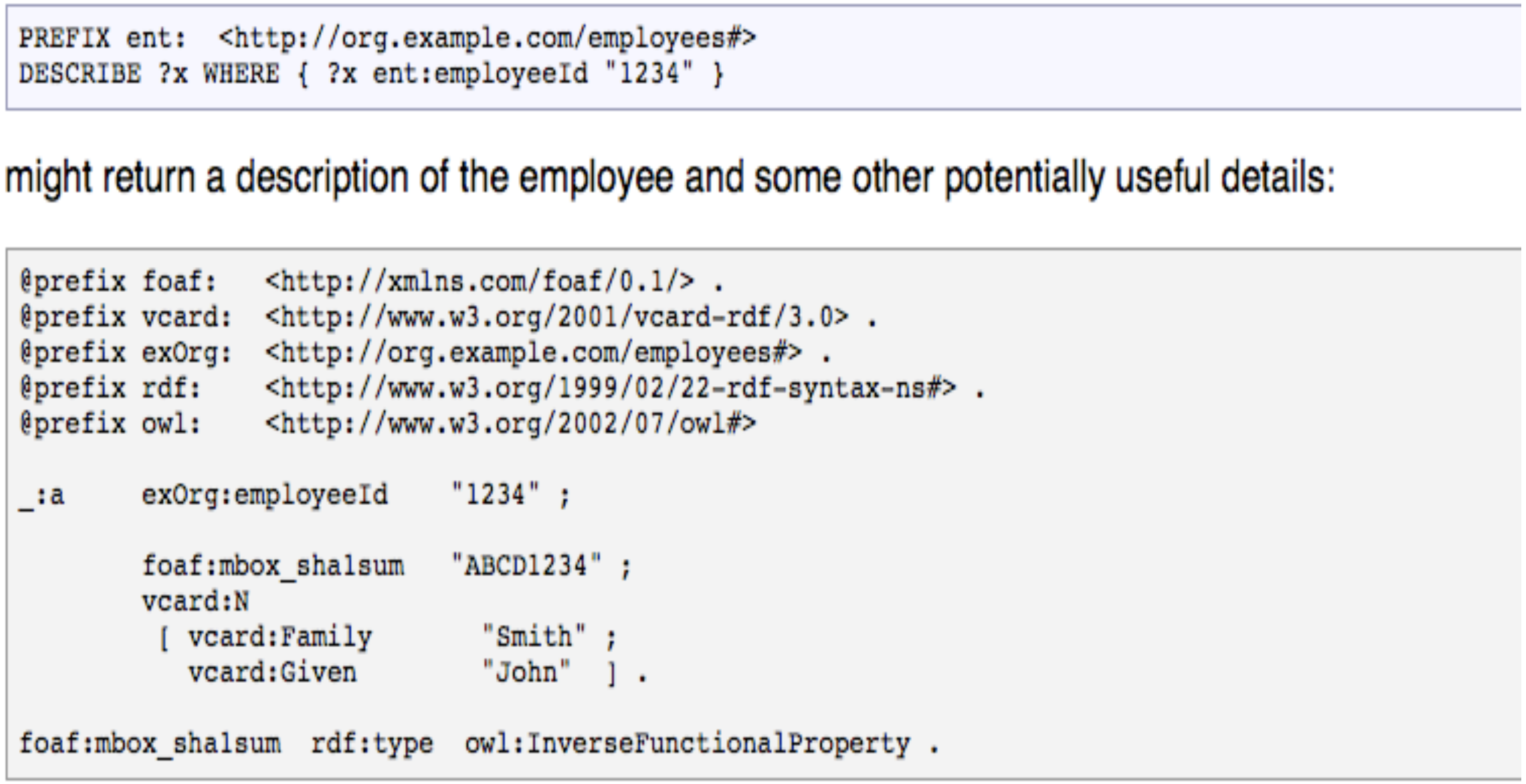
Does Alice has a mailbox ?

```
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_sha1sum   "ABCD1234" ;
        vcard:N
         [ vcard:Family       "Smith" ;
           vcard:Given        "John"  ] .

foaf:mbox_sha1sum  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

- **SELECT** specifies the projection: the number and order of retrieved data
- **FROM** is used to specify the source **(RDF data set)** being queried (optional)
- **WHERE** imposes constraints on possible solutions in the form of graph pattern templates and boolean constraints

Specifying RDF Datasets

```
# Default graph (stored 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 }
```

name
"Alice"

Example of Dataset

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .

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

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

_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@oldcorp.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

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

<http://example.org/bob>     dc:publisher  "Bob Hacker" .
<http://example.org/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@oldcorp.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> .
```

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

SELECT ?who ?g ?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 ?g { ?x foaf:mbox ?mbox }
}
```

DataSet:

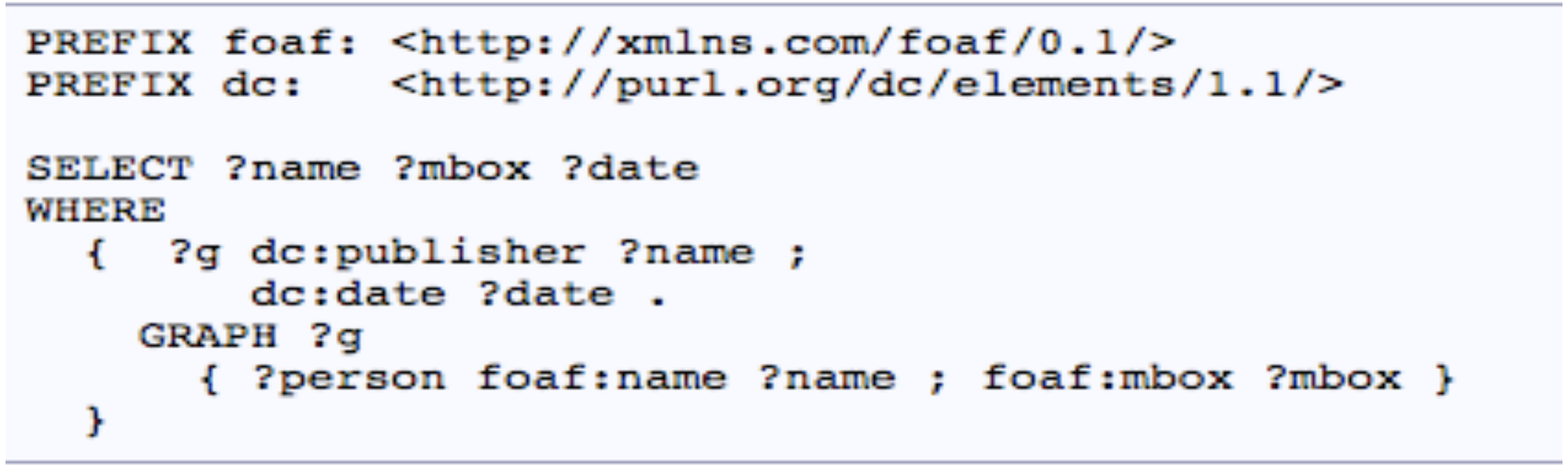
```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix g:  <tag:example.org,2005-06-06:> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

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> .
```

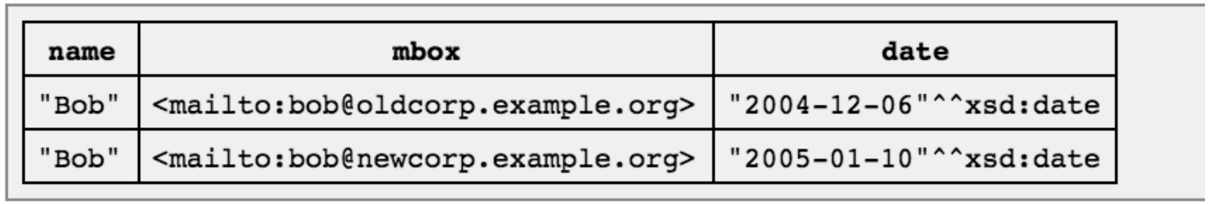
```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph2
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

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

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

Question

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



```
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 }
  }
```

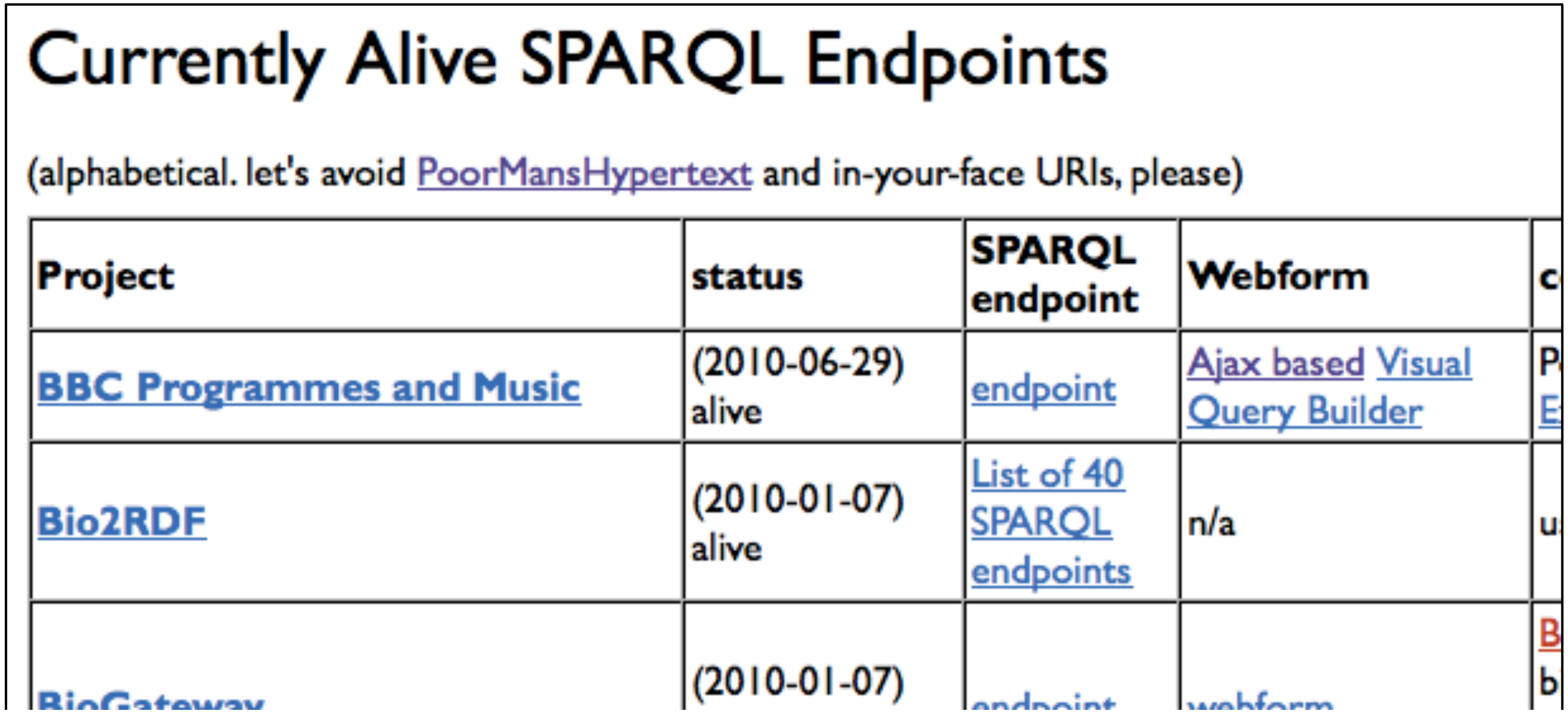


name	mbox	date
"Bob"	<mailto:bob@oldcorp.example.org>	"2004-12-06"^^xsd:date
"Bob"	<mailto:bob@newcorp.example.org>	"2005-01-10"^^xsd:date

SPARQL 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



Currently Alive SPARQL Endpoints

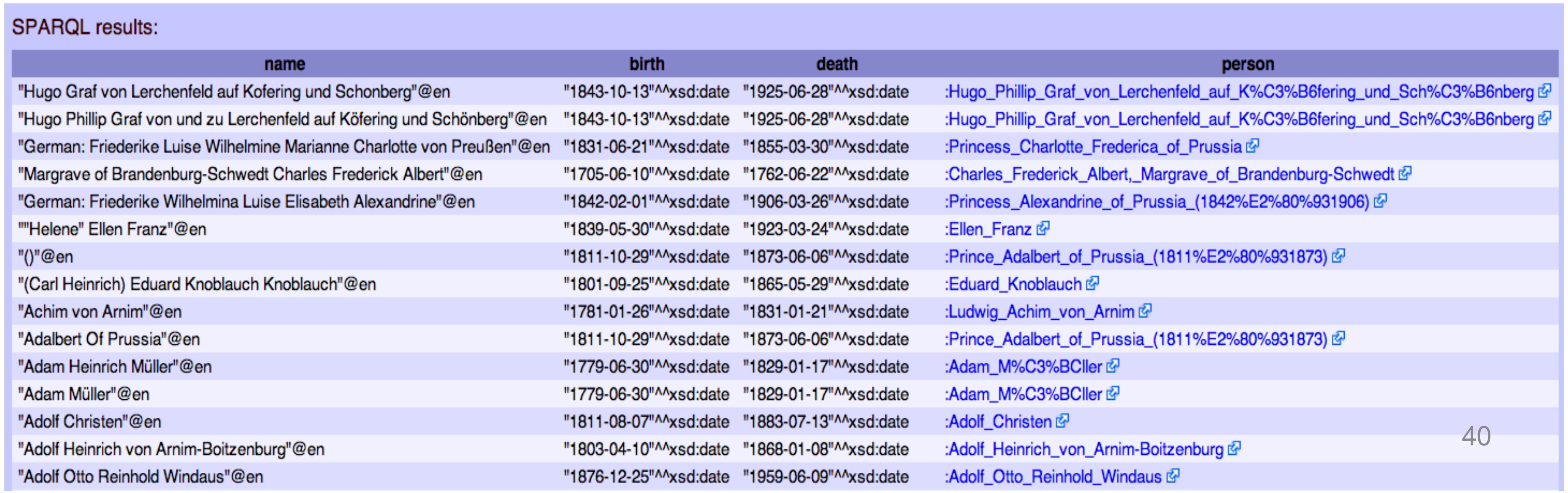
(alphabetical. let's avoid PoorMansHypertext and in-your-face URIs, please)

Project	status	SPARQL endpoint	Webform	c
BBC Programmes and Music	(2010-06-29) alive	endpoint	Ajax based Visual Query Builder	P E
Bio2RDF	(2010-01-07) alive	List of 40 SPARQL endpoints	n/a	u
BioGateway	(2010-01-07)	endpoint	webform	B b

SPARQL Example

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

```
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name ?birth ?death ?person WHERE {
     ?person dbo:birthPlace :Berlin .
     ?person dbo:birthDate ?birth .
     ?person foaf:name ?name .
     ?person dbo:deathDate ?death .
     FILTER (?birth < "1900-01-01"^^xsd:date) .
}
```



SPARQL results:

name	birth	death	person
"Hugo Graf von Lerchenfeld auf Kofering und Schonberg"@en	"1843-10-13"^^xsd:date	"1925-06-28"^^xsd:date	:Hugo_Phillip_Graf_von_Lerchenfeld_auf_K%C3%B6fering_und_Sch%C3%B6nberg
"Hugo Phillip Graf von und zu Lerchenfeld auf Köfering und Schönberg"@en	"1843-10-13"^^xsd:date	"1925-06-28"^^xsd:date	:Hugo_Phillip_Graf_von_Lerchenfeld_auf_K%C3%B6fering_und_Sch%C3%B6nberg
"German: Friederike Luise Wilhelmine Marianne Charlotte von Preußen"@en	"1831-06-21"^^xsd:date	"1855-03-30"^^xsd:date	:Princess_Charlotte_Frederica_of_Prussia
"Margrave of Brandenburg-Schwedt Charles Frederick Albert"@en	"1705-06-10"^^xsd:date	"1762-06-22"^^xsd:date	:Charles_Frederick_Albert,_Margrave_of_Brandenburg-Schwedt
"German: Friederike Wilhelmina Luise Elisabeth Alexandrine"@en	"1842-02-01"^^xsd:date	"1906-03-26"^^xsd:date	:Princess_Alexandrine_of_Prussia_(1842%E2%80%931906)
""Helene" Ellen Franz"@en	"1839-05-30"^^xsd:date	"1923-03-24"^^xsd:date	:Ellen_Franz
"()"@en	"1811-10-29"^^xsd:date	"1873-06-06"^^xsd:date	:Prince_Adalbert_of_Prussia_(1811%E2%80%931873)
"(Carl Heinrich) Eduard Knoblauch Knoblauch"@en	"1801-09-25"^^xsd:date	"1865-05-29"^^xsd:date	:Eduard_Knoblauch
"Achim von Arnim"@en	"1781-01-26"^^xsd:date	"1831-01-21"^^xsd:date	:Ludwig_Achim_von_Arnim
"Adalbert Of Prussia"@en	"1811-10-29"^^xsd:date	"1873-06-06"^^xsd:date	:Prince_Adalbert_of_Prussia_(1811%E2%80%931873)
"Adam Heinrich Müller"@en	"1779-06-30"^^xsd:date	"1829-01-17"^^xsd:date	:Adam_M%C3%BCller
"Adam Müller"@en	"1779-06-30"^^xsd:date	"1829-01-17"^^xsd:date	:Adam_M%C3%BCller
"Adolf Christen"@en	"1811-08-07"^^xsd:date	"1883-07-13"^^xsd:date	:Adolf_Christen
"Adolf Heinrich von Arnim-Boitzenburg"@en	"1803-04-10"^^xsd:date	"1868-01-08"^^xsd:date	:Adolf_Heinrich_von_Arnim-Boitzenburg
"Adolf Otto Reinhold Windaus"@en	"1876-12-25"^^xsd:date	"1959-06-09"^^xsd:date	:Adolf_Otto_Reinhold_Windaus

Summary

- SPARQL is a protocol and query language for RDF data model..
- It designed for open, decentralized Web
- **Select** and **Construct** forms are suitable for querying known endpoint with known vocabularies
- **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