Lecture 10

SPARQL (Advanced)

The contents are taken from http://www.w3.org/TR/rdf-sparql-query/

The slides are prepared by Dr. Davoud Mougouei

SPARQL in 11 minutes https://www.youtube.com/watch?v=FvGndkpa4K0&ab_channel=bobdc

Functions on Strings

- Certain functions (e.g. REGEX, STRLEN, CONTAINS) take a string literal as an argument and accept a simple literal, a plain literal with language tag, or a literal with datatype xsd:string.
- They then act on the lexcial form of the literal. The term string literal is used in the function descriptions for this.
- Use of any other RDF term will cause a call to the function to raise an error.

Functions on Strings

langMatches

Returns true if language-tag (first argument) matches language-range (second argument). A language-range of "*" matches any non-empty language-tag string.

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
_:a dc:title "That Seventies Show"@en .
_:a dc:title "Cette Série des Années Soixante-dix"@fr .
_:a dc:title "Cette Série des Années Septante"@fr-BE .
_:b dc:title "Il Buono, il Bruto, il Cattivo" .
```

Query :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title WHERE { ?x dc:title "That Seventies Show"@en ; dc:title ?title .
FILTER langMatches( lang(?title), "FR" ) }
```

Result:

???

What does this query return?


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

Subqueries

- Subqueries are a way to embed SPARQL queries within other queries.
- To achieve results which cannot otherwise be achieved, such as limiting the number of results from some sub-expression within the query.
- Due to the bottom-up nature of SPARQL query evaluation, the subqueries are evaluated logically first, and the results are projected up to the outer query.
- Note that only variables projected out of the subquery will be visible, or in scope, to the outer query.

Subqueries

Data:

```
@prefix : <http://people.example/> .
:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```


Query :


```
PREFIX : <http://people.example/>
SELECT ?y ?minName WHERE {
   :alice :knows ?y . {
      SELECT ?y (MIN(?name) AS ?minName) WHERE { ?y :name ?name . } GROUP BY ?y
   }
}
```


Result:

???

Subqueries

This result is achieved by first evaluating the inner query:

```
SELECT ?y (MIN(?name) AS ?minName)
WHERE {
  ?y :name ?name .
} GROUP BY ?y
```


This produces the following solution sequence:

y	minName
:alice	"A. Foo"
:bob	"B. Bar"
:carol	"C. Baz"

Which is joined with the results of the outer query:

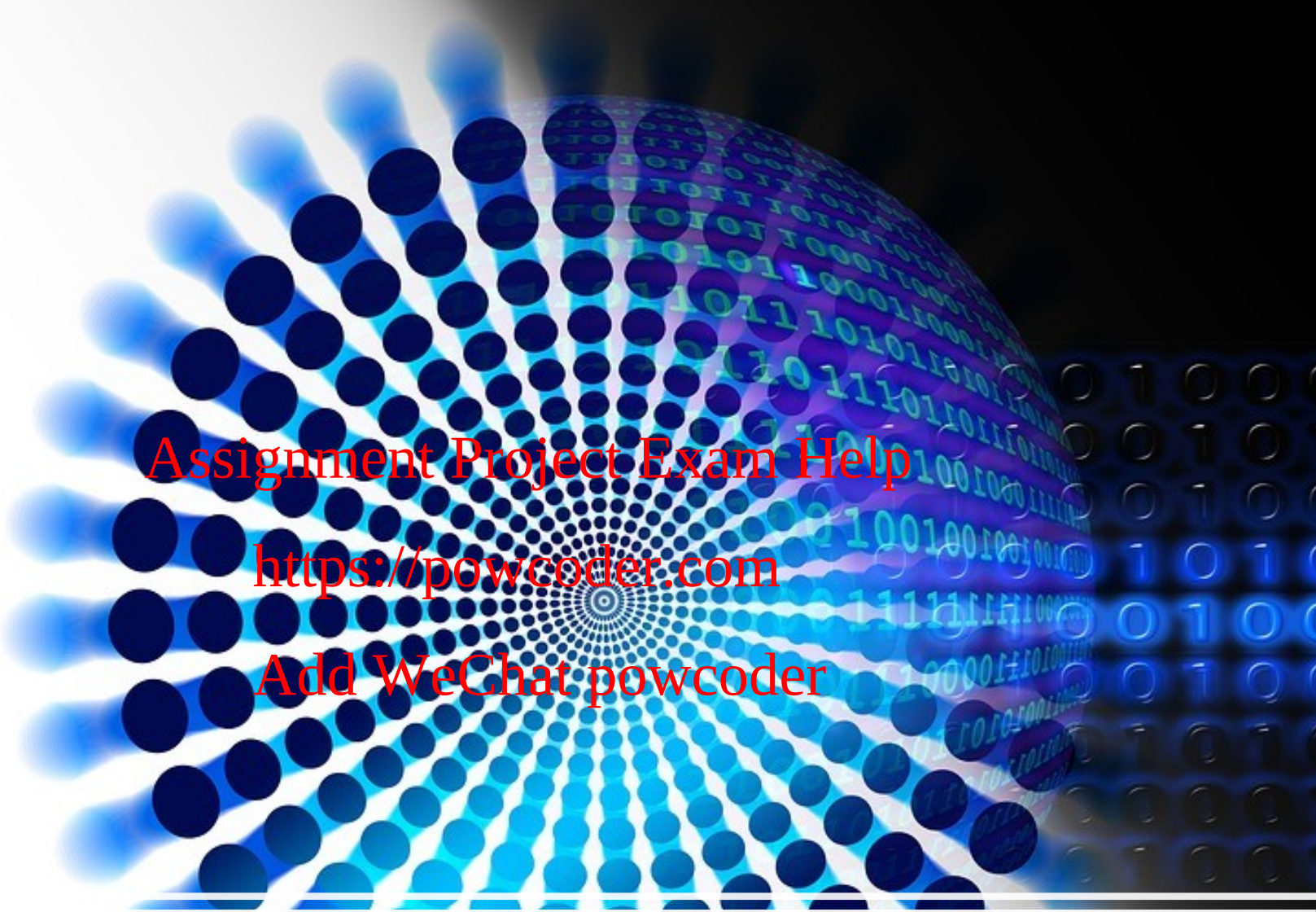
y
:bob
:carol

Modify the previous query to return the “length of the shortest name” rather than “smallest name”.

Modify the previous query to return the “shortest name” rather than the “smallest name”.

Complex Queries with SPARQL

- https://www.youtube.com/watch?v=A2kkR1-qn5k&t=410s&ab_channel=OpenHPITutorials



RDF Datasets

RDF Datasets

The definition of RDF Dataset does not restrict the relationships of named and default graphs.

Information can be repeated in different graphs; relationships between graphs can be exposed.

Two useful arrangements:

- to have information in the default graph that includes provenance information about the named graphs
- to include the information in the named graphs in the default graph as well.

RDF Datasets

Example 1

The default graph contains the names of the publishers of two named graphs. The triples in the named graphs are not visible in the default graph in this example.


```
# 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> .
```

RDF Datasets

Example 2

- RDF data can be combined by the RDF merge of graphs.

- One possible arrangement of graphs in an RDF Dataset is to have the default graph be the RDF merge of some or all of the information in the named graphs.

- In this example, the RDF dataset includes an RDF merge of the named graphs in the default graph, re-labeling blank nodes to keep them distinct.


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

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

_:y foaf:name "Alice" .
_:y foaf:mbox <mailto:alice@work.example.org> .
```

RDF Datasets

Specifying RDF Graphs

- A SPARQL query may specify the dataset to be used for matching by using the FROM clause and the FROM NAMED clause to describe the RDF dataset.
- If a query provides such a dataset description, then it is used in place of any dataset that the query service would use if no dataset description is provided in a query.
- The FROM and FROM NAMED keywords allow a query to specify an RDF dataset by reference; they indicate that the dataset should include graphs that are obtained from representations of the resources identified by the given IRIs.

RDF Datasets

Specifying RDF Graphs

- The dataset resulting from a number of FROM and FROM NAMED clauses is:
  - a default graph consisting of the RDF merge of the graphs referred to in the FROM clauses, and
  - a set of (IRI, graph) pairs, one from each FROM NAMED clause.
- If there is no FROM clause, but there is one or more FROM NAMED clauses, then the dataset includes an empty graph for the default graph.
- The RDF dataset may also be specified in a SPARQL protocol request, in which case the protocol description overrides any description in the query itself.
- A query service may refuse a query request if the dataset description is not acceptable to the service.

RDF Datasets

SELECT with service-supplied RDF Dataset

Query:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?book ?who WHERE { ?book dc:creator ?who }
```


Request (SPARQL query service, http://www.example/sparql/):

```
GET /sparql/?query=PREFIX%20dc%3A%20%3Chttp%3A%2F%2Fpurl.org%2Fdc
%2Felements%2F1.1%2F%3E%20%0ASELECT%20%3Fbook%20%3Fwho
%20%0AWHERE%20%7B%20%3Fbook%20dc%3Acreator%20%3Fwho%20%7D%0A
HTTP/1.1 Host: www.example User-agent: my-sparql-client/0.
```


Result:

???

RDF Datasets

SELECT with service-supplied RDF Dataset

```
HTTP/1.1 200 OK
Date: Fri, 06 May 2005 20:55:12 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4 DAV/1.0.3
Connection: close
Content-Type: application/sparql-results+xml

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

 <head>
   <variable name="book"/>
   <variable name="who"/>
 </head>
 <results>
   <result>
     <binding name="book"><uri>http://www.example/book/book5</uri></binding>
     <binding name="who"><bnode>r29392923r2922</bnode></binding>
   </result>
...
</sparql>
```

RDF Datasets

Specifying the Default Graph

- Each FROM clause contains an IRI that indicates a graph to be used to form the default graph. This does not put the graph in as a named graph.


```
# Default graph (located 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"

The RDF Dataset contains a single default graph and no named graphs.

RDF Datasets

Specifying Named Graphs

- A query can supply IRIs for the named graphs in the RDF Dataset using the FROM NAMED clause. Each IRI is used to provide one named graph in the RDF Dataset.

- Using the same IRI in two or more FROM NAMED clauses results in one named graph with that IRI appearing in the dataset.


```
# 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> .
```



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

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

```
...
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
...
```

RDF Datasets

Combining FROM and FROM NAMED

```
# Default graph (located 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 }
}
```

RDF Datasets

Querying the Dataset

- When querying a collection of graphs, the GRAPH keyword is used to match patterns against named graphs.
- GRAPH can provide an IRI to select one graph or use a variable which will range over the IRI of all the named graphs in the query's RDF dataset.
- The use of GRAPH changes the active graph for matching graph patterns within that part of the query.
- Outside the use of GRAPH, matching is done using the default graph.

RDF Datasets

Querying the Dataset

```
# Named graph: http://example.org/foaf/aliceFoaf
@prefix  foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .

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

_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
_:b  foaf:nick     "Bobby" .
_:b  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .

<http://example.org/foaf/bobFoaf>
     rdf:type      foaf:PersonalProfileDocument .
```



```
# Named graph: http://example.org/foaf/bobFoaf
@prefix  foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .

_:z  foaf:mbox     <mailto:bob@work.example> .
_:z  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .
_:z  foaf:nick     "Robert" .

<http://example.org/foaf/bobFoaf>
     rdf:type      foaf:PersonalProfileDocument .
```

RDF Datasets

Querying the Dataset: Accessing Graph names

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

SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
  {
    GRAPH ?src
    { ?x foaf:mbox <mailto:bob@work.example> .
      ?x foaf:nick ?bobNick
    }
  }
```


src	bobNick
<http://example.org/foaf/aliceFoaf>	"Bobby"
<http://example.org/foaf/bobFoaf>	"Robert"

Modify the query to return triples where foaf:nick may or may not exit.

RDF Datasets

Querying the Dataset: Restricting by Graph IRI

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

SELECT ?nick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
  {
     GRAPH data:bobFoaf {
         ?x foaf:mbox <mailto:bob@work.example> .
         ?x foaf:nick ?nick }
  }
```


which yields a single solution:

nick
"Robert"

RDF Datasets

Querying the Dataset: Restricting Possible Graph IRIs

- A variable used in the GRAPH clause may also be used in another GRAPH clause or in a graph pattern matched against the default graph in the dataset.

- The query below uses the graph with IRI http://example.org/foaf/aliceFoaf to find the profile document for Bob; it then matches another pattern against that graph.

- The pattern in the second GRAPH clause finds the blank node (variable w) for the person with the same mail box (given by variable mbox) as found in the first GRAPH clause (variable whom), because the blank node used to match for variable whom from Alice's FOAF file is not the same as the blank node in the profile document (they are in different graphs).

RDF Datasets

Querying the Dataset: Restricting Possible Graph IRIs

```
PREFIX  data:  <http://example.org/foaf/>
PREFIX  foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX  rdfs:  <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?mbox ?nick ?ppd
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
{
  GRAPH data:aliceFoaf
  {
    ?alice foaf:mbox <mailto:alice@work.example> ;
           foaf:knows ?whom .
    ?whom  foaf:mbox ?mbox ;
           rdfs:seeAlso ?ppd .
    ?ppd  a foaf:PersonalProfileDocument .
  } .
  GRAPH ?ppd
  {
      ?w foaf:mbox ?mbox ;
         foaf:nick ?nick
  }
}
```


mbox	nick	ppd
<mailto:bob@work.example>	"Robert"	<http://example.org/foaf/bobFoaf>

RDF Datasets

Querying the Dataset: Restricting Possible Graph IRIs

- Any triple in Alice's FOAF file giving Bob's nick is not used to provide a nick for Bob because the pattern involving variable nick is restricted by ppd to a particular Personal Profile Document

RDF Datasets

Querying the Dataset: Named and Default Graphs

- Query patterns can involve both the default graph and the named graphs.

- In this example, an aggregator has read in a Web resource on two different occasions. Each time a graph is read into the aggregator, it is given an IRI by the local system. The graphs are nearly the same but the email address for "Bob" has changed.

- In this example, the default graph is being used to record the provenance information and the RDF data actually read is kept in two separate graphs, each of which is given a different IRI by the system. The RDF dataset consists of two named graphs and the information about them.

RDF Datasets

Querying the Dataset: Named and Default Graphs

```
# 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> .
```

RDF Datasets

Querying the Dataset: Named and Default Graphs

This query 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 . }
  }
```


The results show that the email address for "Bob" has changed.

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

Security Considerations

- SPARQL queries using FROM, FROM NAMED, or GRAPH may cause the specified URI to be dereferenced. This may cause additional use of network, disk or CPU resources along with associated secondary issues such as denial of service.

- In addition, the contents of file: URIs can in some cases be accessed, processed and returned as results, providing unintended access to local resources.

Security Considerations

- SPARQL requests may cause additional requests to be issued from the SPARQL endpoint, such as FROM NAMED. The endpoint is potentially within an organizations firewall or DMZ, and so such queries may be a source of indirection attacks.
- The SPARQL language permits extensions, which will have their own security implications.

Security Considerations

- Multiple IRIs may have the same appearance. Characters in different scripts may look similar (a Cyrillic "o" may appear similar to a Latin "o"). A character followed by combining characters may have the same visual representation as another character (LATIN SMALL LETTER E followed by COMBINING ACUTE ACCENT has the same visual representation as LATIN SMALL LETTER E WITH ACUTE). Users of SPARQL must take care to construct queries with IRIs that match the IRIs in the data.

Querying Wikidata with SPARQL for Absolute Beginners

- https://www.youtube.com/watch?v=kJph4q0Im98&ab_channel=WikimediaFoundation