SPARQL
(RDF Query Language)

Mustafa Jarrar

Birzeit University

Watch this lecture and download the slides

Mustafa Jarrar: Online Courses – Watch and Download Slides

Reference:
Mustafa Jarrar: Lecture Notes on Conceptual Analyses
University of Birzeit, Palestine, 2015

Online Course

Watch Do ad Slides

Mustafa Jarrar
Birzeit University, Palestine
mjarrar@birzeit.edu

Online Courses : http://www.jarrar.info/courses/

Lecture Outline

Part I: SPARQL Basics

Part 2: SPARQL Practical Session

Keywords: SPARQL**,** RDF, RDF Stores, RDF Query Language, Graph Databases, Querying Graph, Semantic Web, Data Web,

SPARQL

As we have learned, RDF is a graph-shaped data model.

Until now, we have queried RDF stored in relational databases using standard SQL.

What about a standard query language that is dedicated for querying RDF graphs?

- Offering a more intuitive method for querying graph-shaped data (using graph patterns).
- Offering a way for the queries and their respective results to be transported between applications / services.
- Allowing querying information from multiple Web sites (mashups).
- Allowing querying information from multiple enterprise databases.

SPARQL

SPARQL (pronounced: Sparkle). The name is a recursive acronym for:

“**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage”

The “Protocol” part of SPARQL’s name refers to the rules for how a client program and a SPARQL processing server exchange SPARQL queries and results (here, we focus on the query language).

Official W3C Recommendation: January 2008, SPARQL 1.0 and SPARQL 1.1 in March, 2013

SPARQL: Jumping right in

A SPARQL query typically says “I want these pieces of information from the subset of the data that meets these conditions.”

Q1: What is the name of director D3?

```
SELECT ?directorName
WHERE {:D3  :Name  ?directorName}
```

S	P	O
…	…	…
D2	Name	Mel Gibson
D2	actedIn	M3
D3	Name	Nadine Labaki
D3	Country	C2
D3	hasWonPrizeIn	P3
D3	actedIn	M4
…	…	…

Variables

The Variable:

- It tells the query engine that triples with any value at all in that position are OK to match this triple pattern.
- The values are stored in the variable so that we can use them elsewhere in the query.

Q2: What is the name of the director of the movie M1?

```
SELECT ?directorName
WHERE
 {
    :M1 :directedBy ?director .
    ?director :name ?directorName
  }
```

Answer: Michael Moore

S	P	O
M1	year	2007
M1	Name	Sicko
M1	directedBy	D1
…	…	…
M4	Name	Caramel
D1	Name	Michael Moore
D1	hasWonPrizeIn	P1
D1	Country	C1
…	…	…

Example

Q3: List all the movies who have directors from the USA and their directors.

```
Select ?movie ?director
Where {?movie :directedBy ?director.
    ?director :country ?country.
    ?country :name ‘USA’}
```

Answer: M1 D1; M2 D1; M3 D2

S	P	O
M1	year	2007
M1	Name	Sicko
M1	directedBy	D1
M2	directedBy	D1
M2	Year	2009
M2	Name	Capitalism
M3	Year	1995
M3	directedBy	D2
M3	Name	Brave Heart
…	…	…
D1	Name	Michael Moore
D1	hasWonPrizeIn	P1
D1	Country	C1
D2	Counrty	C1
D2	hasWonPrizeIn	P2
D2	Name	Mel Gibson
D2	actedIn	M3
…	…	…
C1	Name	USA
C1	Capital	Washington DC
C2	Name	Lebanon
C2	Capital	Beirut
…	…	…

How to Query RDF data stored in one table?

Q4: List all the names of the directors from Lebanon who have won prizes and the prizes they have won.

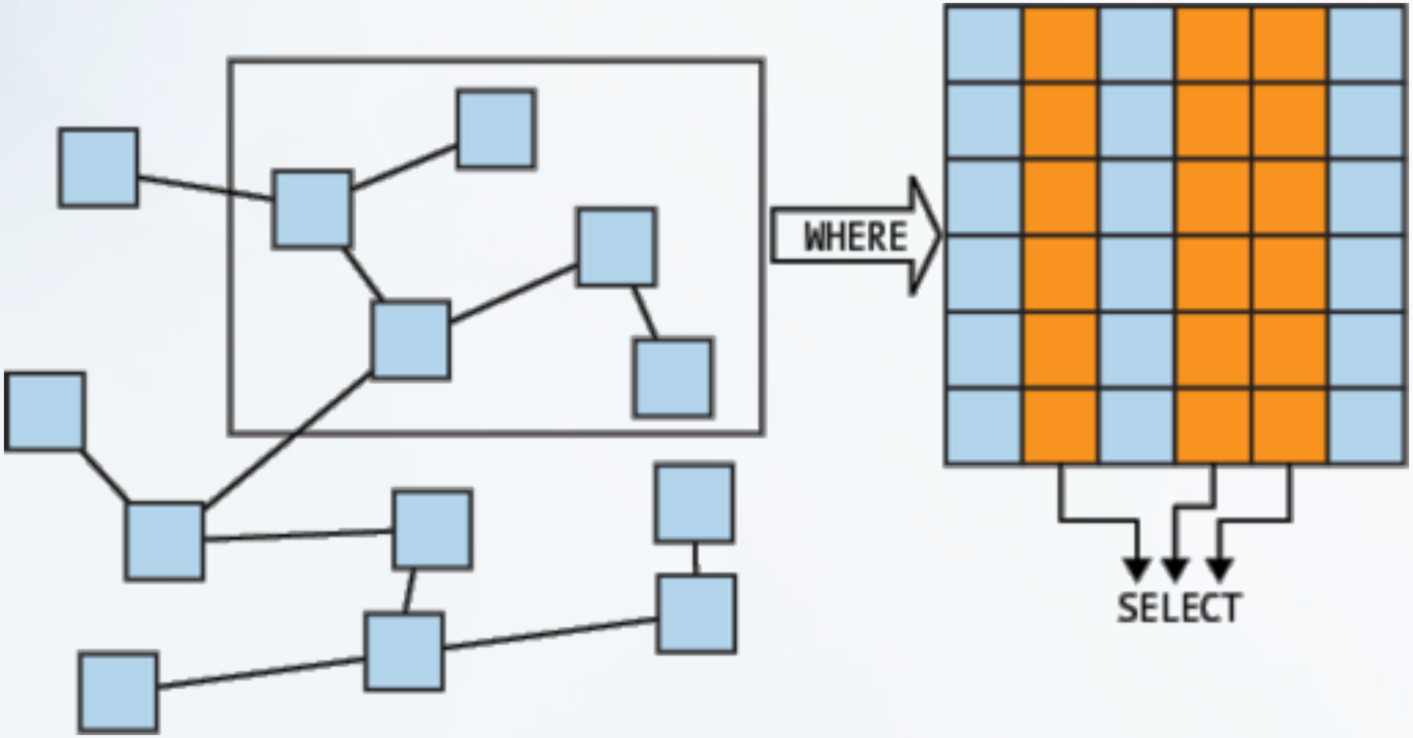
```
Select ?directorName ?prize
Where { ?director :name ?directorName.
          ?director :country ?c.
          ?c :name ‘Lebanon’.
          ?director :hasWonPrizeIn ?prize
        }
```

S	P	O
D1	Name	Michael Moore
D1	hasWonPrizeIn	P1
D1	Country	C1
D2	Counrty	C1
D2	hasWonPrizeIn	P2
D2	Name	Mel Gibson
D2	actedIn	M3
D3	Name	Nadine Labaki
D3	Country	C2
D3	hasWonPrizeIn	P3
D3	actedIn	M4
…	…	…
C1	Name	USA
C1	Capital	Washington DC
C2	Name	Lebanon
C2	Capital	Beirut
…	…	…

Answer: ‘Nadine Labaki’ , P3

A SPARQL query

Source: "*Learning SPARQL* by Bob DuCharme (O'Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5."



- WHERE specifies data to pull out
- SELECT picks which data to display

RDF and SPARQL in accurate syntax

Recall that RDF triple’s Subject and Predicate must always be URIs. RDF’s object can either be a URI or a literal.

RDF can be written in many ways such as RDF/XML, Notation 3, and Turtle. Consider our RDF graph written in Turtle format:

```
@prefix ab: <http://example.com/ns/movies#> .
@prefix da: <http://example.com/ns/data#> .
...
da:M1 ab:year “2007”.
da:M1 ab:name “Sicko”.
da:M1 ab:directedBy da:D1.
da:D1 ab:name “Michael Moore”. ...
```

- Consider Q2 again:

```
PREFIX ab: <http://example.com/ns/movies#>
PREFIX da: <http://example.com/ns/data#>

SELECT ?directorName
WHERE
{ da:M1 ab:directedBy ?director .
  ?director ab:name ?directorName }
```

Namespaces where the vocabulary used is defined (usually an ontology)

Prefixes are used to make the query more compact

Consider the use of URIs in the subject and predicates, and the use of strings in non-URI objects

Graph Patterns

Basic and Group Graph Patterns

So far, we have seen two graph patterns:

– **Basic Graph Pattern:** A triple pattern.

– **Group Pattern:** A set of graph patterns which must all match.

- **Triple Pattern** – similar to an RDF Triple (subject, predicate, object), but may include variables to add flexibility in how they match against the data.

```
da:M1 ab:directedBy ?director
```

- Matching a triple pattern to a graph: bindings between variables and RDF Terms.

Matching of Basic Graph Patterns

– A **Pattern Solution** of Graph Pattern GP on graph G is any substitution S such that S(GP) is a subgraph of G.

Graph Patterns

Basic and Group Graph Patterns

Basic Graph Pattern

```
SELECT ?directorName
WHERE {da:D3  ab:name  ?directorName}
```

```
SELECT ?directorName
WHERE
  {
    da:M1 ab:directedBy ?director .
    ?director ab:name ?directorName
  }
```

Group Graph Pattern

Graph Patterns

Value Constraint

Source: http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/

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/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
        FILTER ?price < 30 .
        ?x dc:title ?title . }
```

Query Results

title	price
"The Semantic Web"	23

Graph Patterns

Optional Graph Patterns

Source: http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/

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 .
```

Returns titles and their prices if it is less than 30

Query

```
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 }}
```

Query Result

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

Graph Patterns

Alternative Graph Pattern (UNION)

Source: http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050721/

Data

```
@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" .
_:b dc11:title "SPARQL Protocol Tutorial" .
_:c dc10:title "SPARQL" .
_:c dc11:title "SPARQL (updated)" .
```

Query

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?x ?y
WHERE { { ?book dc10:title ?x } UNION
            { ?book dc11:title ?y } }
```

Query Result

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

Sorting, Aggregating, Finding the Biggest, …

Source: “Learning SPARQL by Bob DuCharme (O’Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5.”

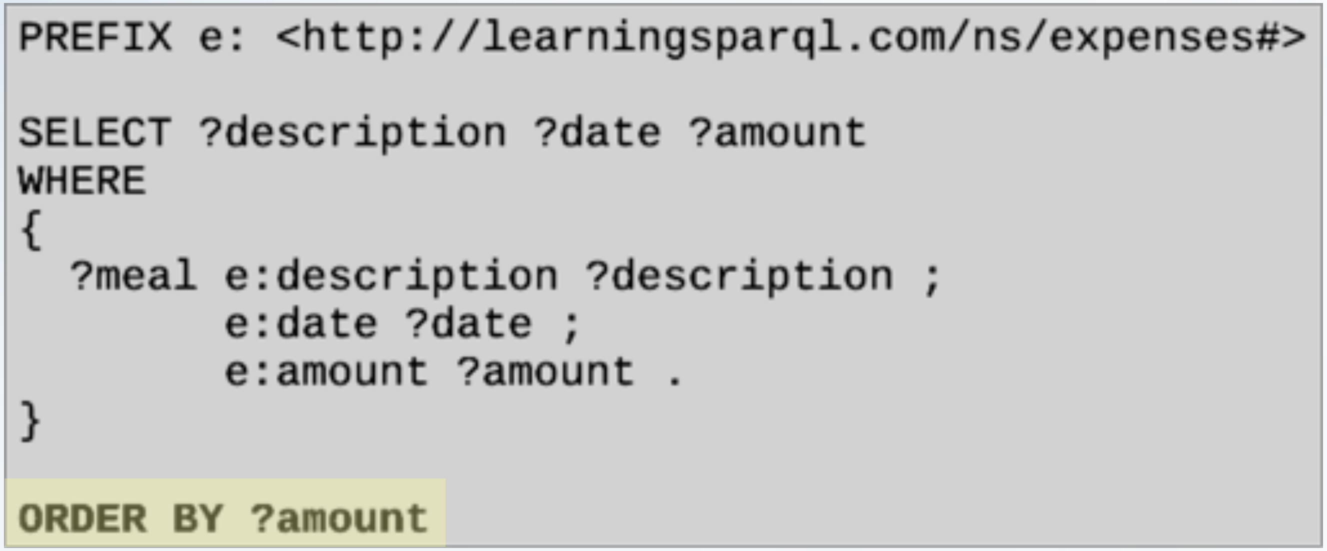
Consider the following example about restaurant expenses:

```
@prefix e: <http://learningsparql.com/ns/expenses#> .
@prefix d: <http://learningsparql.com/ns/data#> .

d:m40392 e:description "breakfast" ;
         e:date "2011-10-14T08:53" ;
         e:amount 6.53 .
d:m40393 e:description "lunch" ;
         e:date "2011-10-14T13:19" ;
         e:amount 11.13 .
d:m40394 e:description "dinner" ;
         e:date "2011-10-14T19:04" ;
         e:amount 28.30 .
d:m40395 e:description "breakfast" ;
         e:date "2011-10-15T08:32" ;
         e:amount 4.32 .
d:m40396 e:description "lunch" ;
         e:date "2011-10-15T12:55" ;
         e:amount 9.45 .
d:m40397 e:description "dinner" ;
         e:date "2011-10-15T18:54" ;
         e:amount 31.45 .
d:m40398 e:description "breakfast" ;
         e:date "2011-10-16T09:05" ;
         e:amount 6.65 .
d:m40399 e:description "lunch" ;
         e:date "2011-10-16T13:24" ;
         e:amount 10.00 .
d:m40400 e:description "dinner" ;
         e:date "2011-10-16T19:44" ;
         e:amount 25.05 .
```

Sorting Data

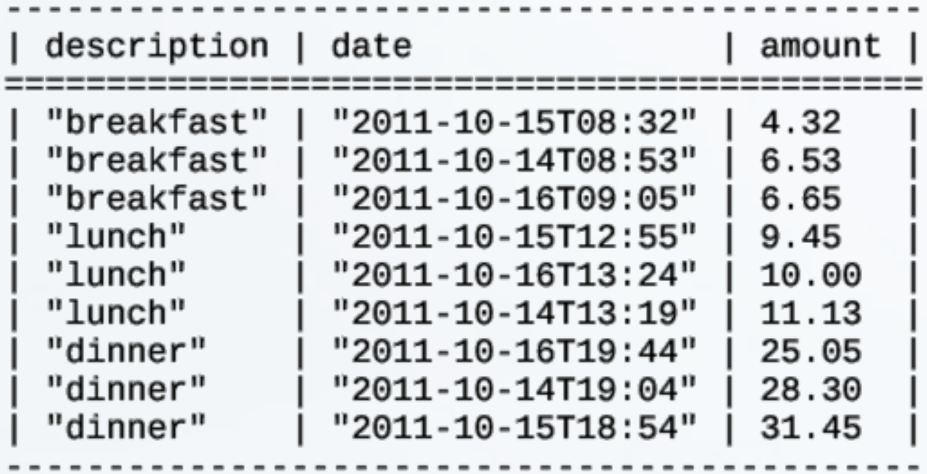
Source: "*Learning SPARQL* by Bob DuCharme (O'Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5."

Sort in ascending order:



```
PREFIX e: <http://learningsparql.com/ns/expenses#>

SELECT ?description ?date ?amount
WHERE
{
  ?meal e:description ?description ;
        e:date ?date ;
        e:amount ?amount .
}

ORDER BY ?amount
```

Result Set:

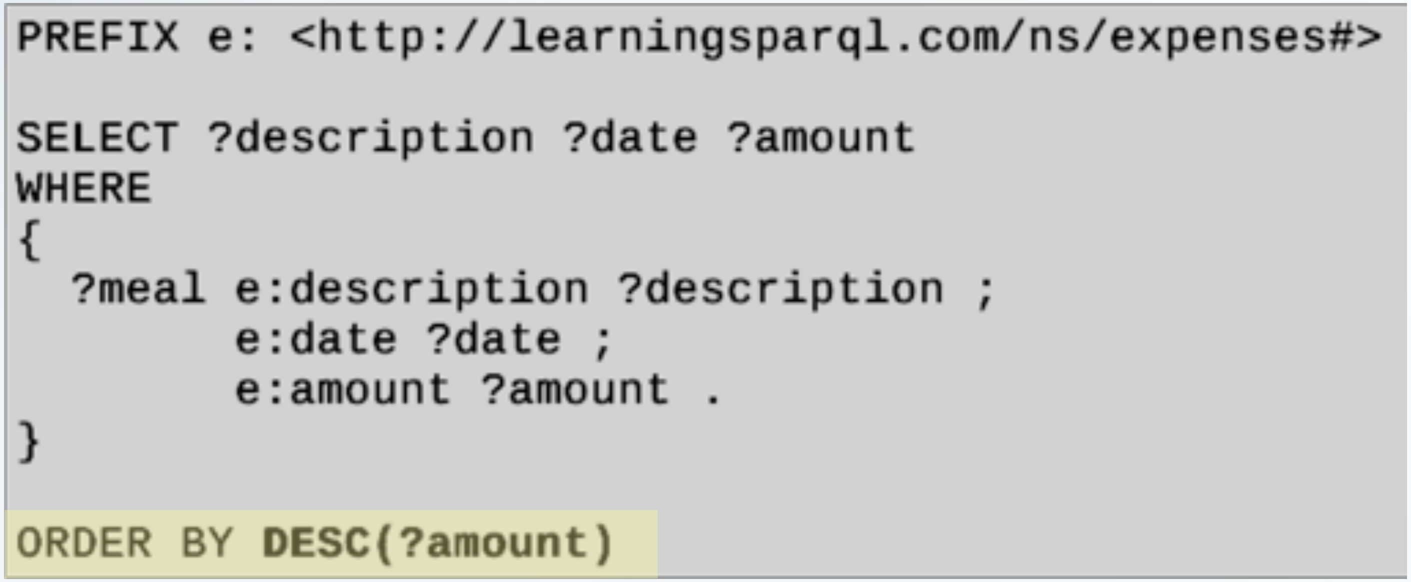


```
--------------------------------------------------
| description | date                  | amount |
==================================================
| "breakfast" | "2011-10-15T08:32" | 4.32   |
| "breakfast" | "2011-10-14T08:53" | 6.53   |
| "breakfast" | "2011-10-16T09:05" | 6.65   |
| "lunch"     | "2011-10-15T12:55" | 9.45   |
| "lunch"     | "2011-10-16T13:24" | 10.00  |
| "lunch"     | "2011-10-14T13:19" | 11.13  |
| "dinner"    | "2011-10-16T19:44" | 25.05  |
| "dinner"    | "2011-10-14T19:04" | 28.30  |
| "dinner"    | "2011-10-15T18:54" | 31.45  |
--------------------------------------------------
```

Sorting Data

Source: “*Learning SPARQL* by Bob DuCharme (O’Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5.”

How to sort in descending order?

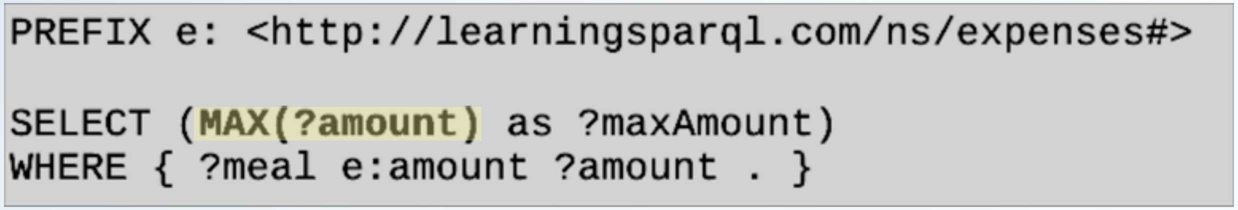


```
PREFIX e: <http://learningsparql.com/ns/expenses#>

SELECT ?description ?date ?amount
WHERE
{
  ?meal e:description ?description ;
        e:date ?date ;
        e:amount ?amount .
}

ORDER BY DESC(?amount)
```

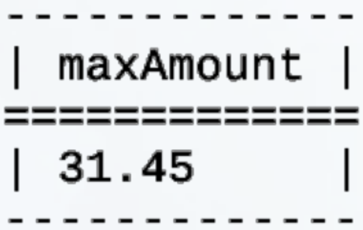
MAX and AVG

Source: “*Learning SPARQL* by Bob DuCharme (O’Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5.”

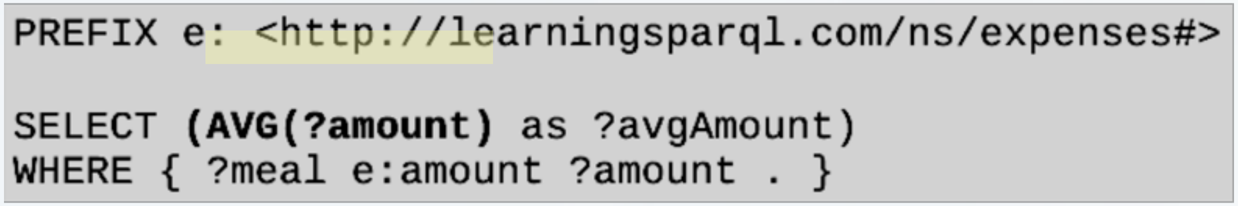
NOTE: MAX() and the remaining functions described here are new in SPARQL 1.1.



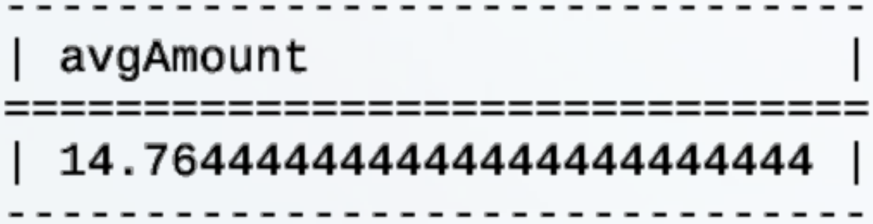
```
PREFIX e: <http://learningsparql.com/ns/expenses#>

SELECT (MAX(?amount) as ?maxAmount)
WHERE { ?meal e:amount ?amount . }
```



```
-------------
| maxAmount |
=============
| 31.45     |
-------------
```



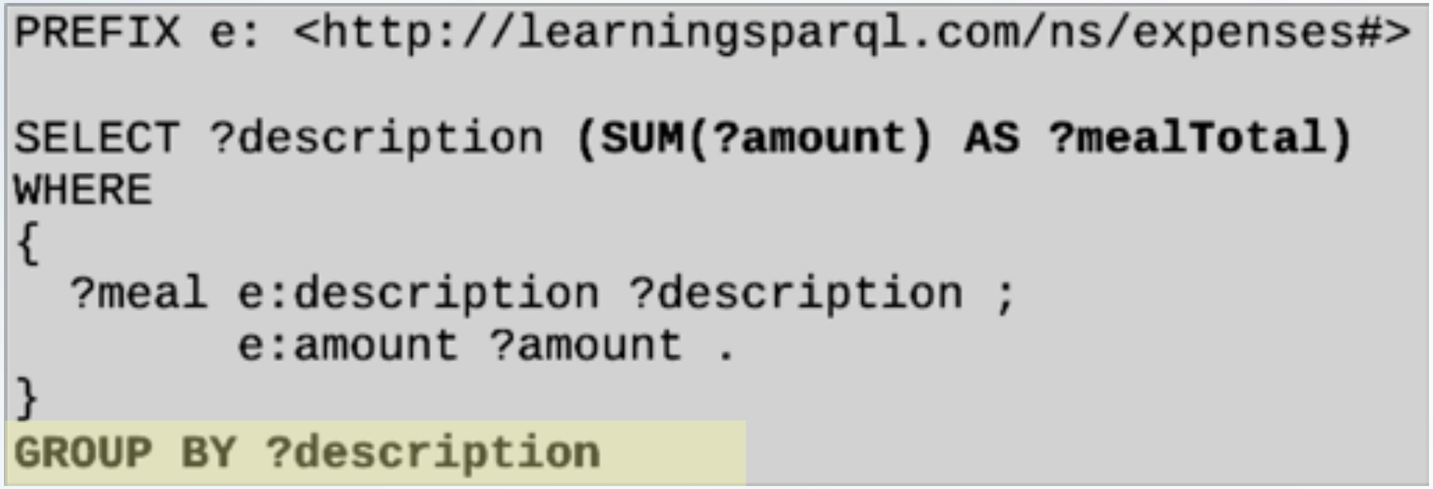
```
PREFIX e: <http://learningsparql.com/ns/expenses#>

SELECT (AVG(?amount) as ?avgAmount)
WHERE { ?meal e:amount ?amount . }
```



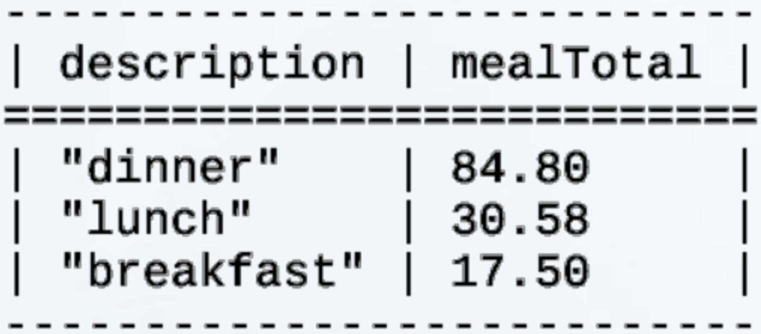
```
------------------------------
| avgAmount                  |
==============================
| 14.764444444444444444444   |
------------------------------
```

Group Data

Source: "*Learning SPARQL* by Bob DuCharme (O'Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5."



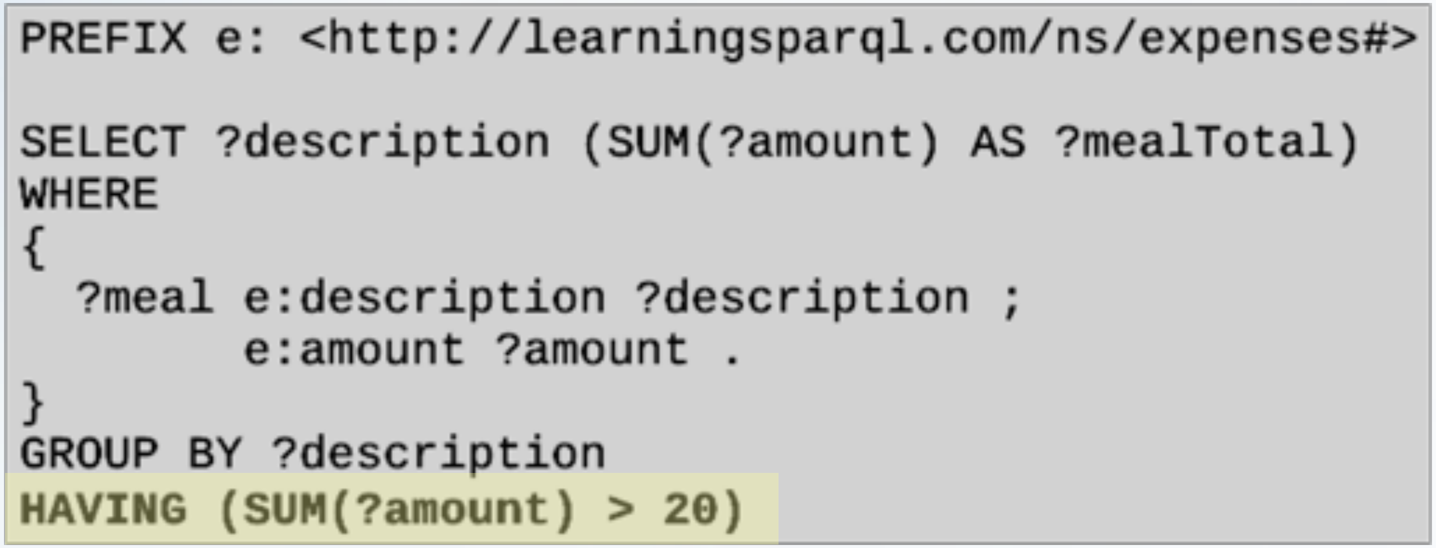
```
PREFIX e: <http://learningsparql.com/ns/expenses#>

SELECT ?description (SUM(?amount) AS ?mealTotal)
WHERE
{
  ?meal e:description ?description ;
        e:amount ?amount .
}
GROUP BY ?description
```



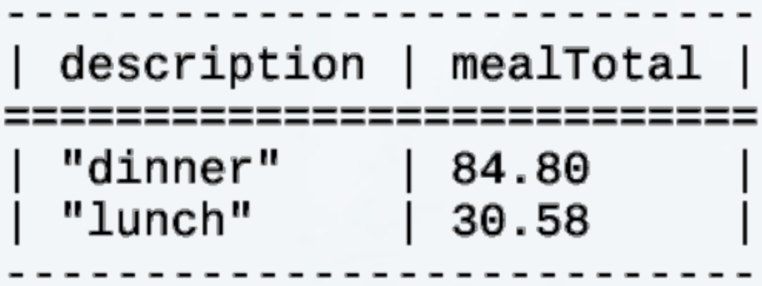
```
---------------------------
| description | mealTotal |
===========================
| "dinner"    | 84.80     |
| "lunch"     | 30.58     |
| "breakfast" | 17.50     |
---------------------------
```

Having Function

Source: "*Learning SPARQL* by Bob DuCharme (O'Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5."



```
PREFIX e: <http://learningsparql.com/ns/expenses#>

SELECT ?description (SUM(?amount) AS ?mealTotal)
WHERE
{
  ?meal e:description ?description ;
        e:amount ?amount .
}
GROUP BY ?description
HAVING (SUM(?amount) > 20)
```



```
---------------------------
| description | mealTotal |
===========================
| "dinner"    | 84.80     |
| "lunch"     | 30.58     |
---------------------------
```

Other SPARQL Query Forms

- SELECT
  - The SELECT form of results returns variables and their bindings directly.
- CONSTRUCT
  - The CONSTRUCT query form returns a single RDF graph specified by a graph template.
- DESCRIBE
  - The DESCRIBE form returns a single result RDF graph containing RDF data about resources.
- ASK
  - Applications can use the ASK form to test whether or not a query pattern has a solution.

Part 2

SPARQL
Practical Session

Practical Session

This practical session is divided into two parts:

(1) Querying DBPedia (a huge RDF dataset built from Wikipedia Infoboxes and data), using the online SPARQL endpoint.

(2) Querying the same graph of Practical Session I, but this time using SPARQL.

PART 1:

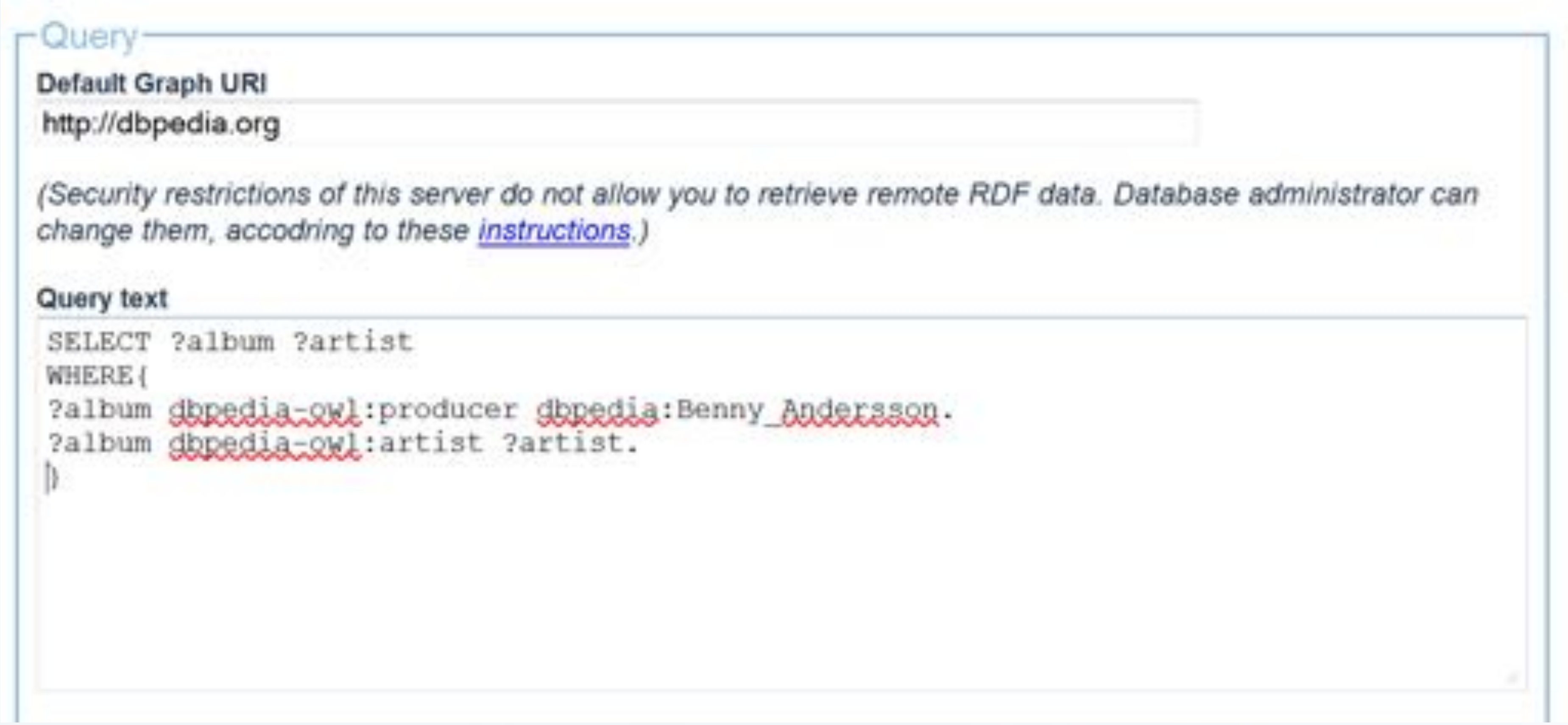
Each student should do the following:

(i) Execute the following three sample queries using the online Virtuoso SPARQL Query Editor: http://dbpedia.org/sparql.

(ii) Construct additional 3 *meaningful complex* queries on DBPedia or any other dataset using an online SPARQL endpoint.

PART1: Query 1

Find all the albums that have the producer Benny Anderson with their artists



Default Graph URI

http://dbpedia.org

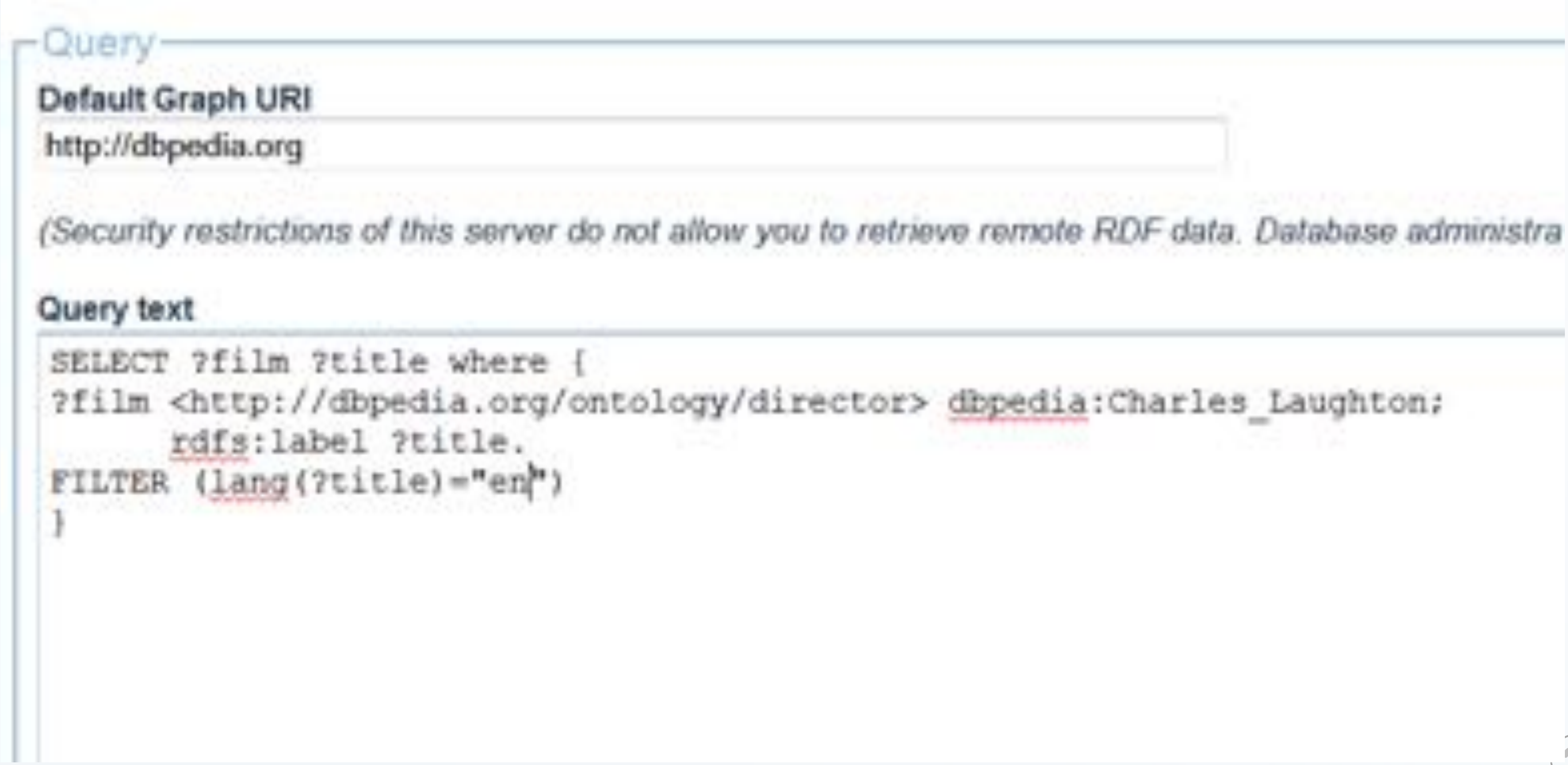
(Security restrictions of this server do not allow you to retrieve remote RDF data. Database administrator can change them, accodring to these instructions.)

Query text

```
SELECT ?album ?artist
WHERE{
?album dbpedia-owl:producer dbpedia:Benny_Andersson.
?album dbpedia-owl:artist ?artist.
}
```

PART1: Query 2

Find all English films whose director is Charles Laughton



Query

Default Graph URI

http://dbpedia.org

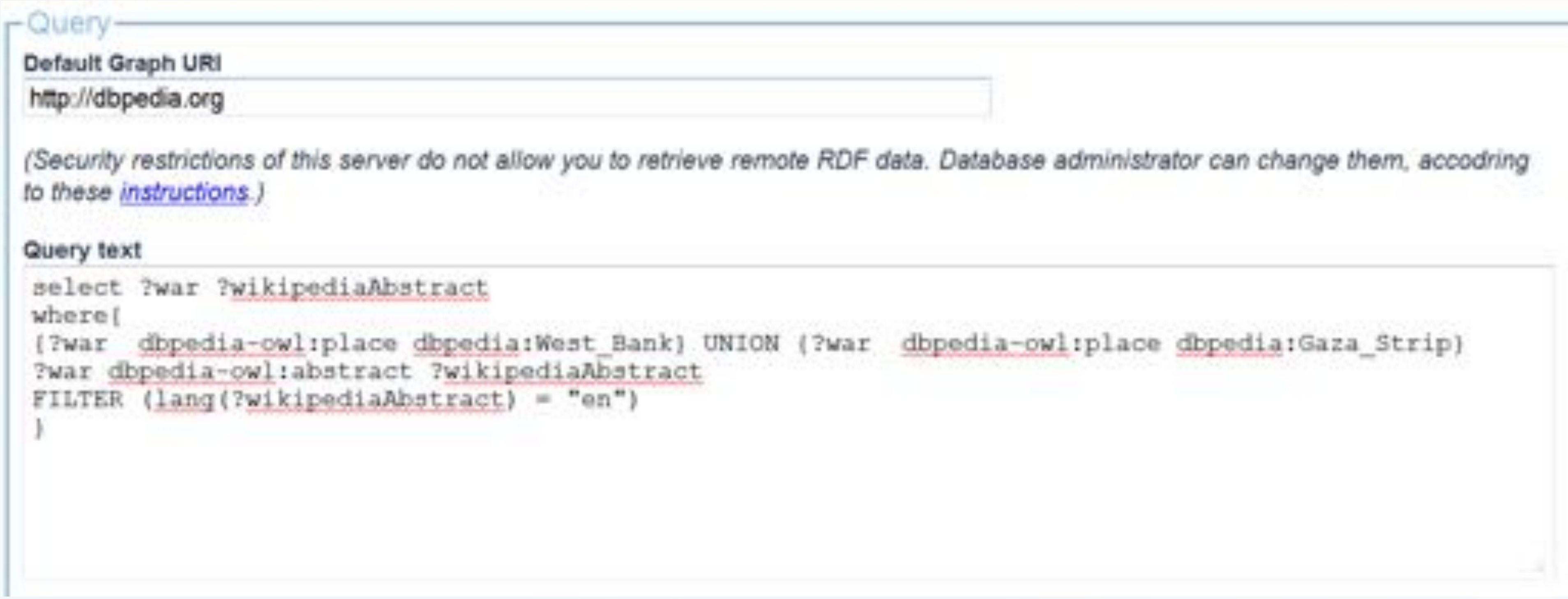
(Security restrictions of this server do not allow you to retrieve remote RDF data. Database administra

Query text

```
SELECT ?film ?title where {
?film <http://dbpedia.org/ontology/director> dbpedia:Charles_Laughton;
      rdfs:label ?title.
FILTER (lang(?title)="en")
}
```

PART1: Query 3

Find all wars that happened in the West Bank or Gaza Strip with their abstracts in English.



Default Graph URI

http://dbpedia.org

(Security restrictions of this server do not allow you to retrieve remote RDF data. Database administrator can change them, accodring to these instructions.)

Query text

```
select ?war ?wikipediaAbstract
where{
{?war  dbpedia-owl:place dbpedia:West_Bank} UNION {?war  dbpedia-owl:place dbpedia:Gaza_Strip}
?war dbpedia-owl:abstract ?wikipediaAbstract
FILTER (lang(?wikipediaAbstract) = "en")
}
```

PART2: Querying the Books Graph

Given the RDF graph of Practical Session I (also included in the next slide), do the following:

(1) Write the data graph using any suitable RDF syntax (XML, N3, or Turtle).

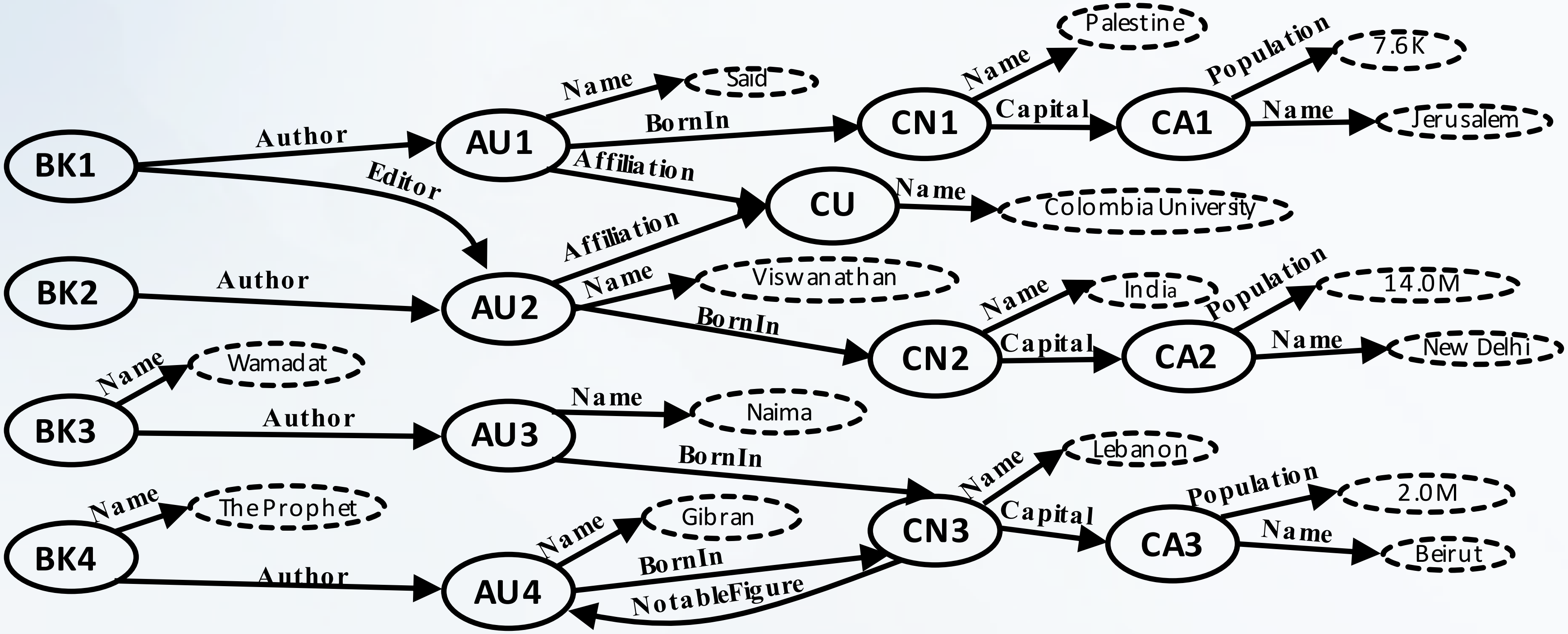
(2) Open http://sparql.us (Pronounced Sparklous)

(3) Upload your RDF file using the sparql.us tool.

(4) Write the following queries in SPARQL and execute them over the loaded file:

- List all the authors born in a country which has the name Palestine.
- List the names of all authors with the name of their affiliation who are born in a country whose capital's population is14M. Note that the author must have an affiliation.
- List the names of all books whose authors are born in Lebanon along with the name of the author.

PART2: Querying the Books Graph



This data graph is about books. It talks about four books (BK1-BK4). Information recorded about a book includes data such as; its author, affiliation, country of birth including its capital and the population of its capital.

Practical Session - Instructions

- Each student should work alone.
- In part 2 of this practical session, the student is *strongly recommended* to write two additional queries, execute them on the data graph, and hand them along with the required queries.
- In part 2 of this practical session, the student is encouraged to compare the results of the queries with those from Practical Session I.
- Each student must expect to present and discuss his/her queries at class and compare them with the work of other students.
- The final delivery should include: (i) The 6 queries constructed in Part 1 with the links of their results. (ii) A link to the RDF file. (iii) The queries executed over the RDF file in sparq.us along with snapshots of their results. These must be handed in a report form in PDF Format.

References

- http://www.w3.org

- Anton Deik, Bilal Faraj, Ala Hawash, Mustafa Jarrar: Towards Query Optimization for the Data Web - Two Disk-Based algorithms: Trace Equivalence and Bisimilarity.

- Learning SPARQL by Bob DuCharme (O'Reilly). Copyright 2011 Bob DuCharme, 978-1-449-30659-5.

SPARQL Project

Goal

This project aims to train students how to use Graph queries using both: 1) an SPARQL endpoint, and using 2) Oracle Semantic Technology.

Data will be used from pervious projects (marksheets)

Oracle Semantic Technology Project

1. Each student alone should do the following:
2. Convert his/her two RDF Mark sheets into an RDF1(S,P,O) table,
3. Convert two RDF Mark sheets (from another student) into an RDF2(S,P,O) table.
4. Create a table called SamaAs(URI1,UR2) and populate it with the same entities in RDF1 and RDF2.

Practice Oracle Semantic Technology:

1. Create an RDF(S,P,O) table and populate it with RDF1 and RDF2, taking into account linked entities in the SameAs table.
2. Load this RDF table into an Oracle Semantic Technology table.
3. Write three different queries using Oracle Smatch table function: 1) a simple start query, a start query with a path with two edges length, a start query with a path with four edges length.

Practice SPARQL:

1. Load the graph in the RDF table (above) into the Query Editor: http://sparql.us/ .
2. Execute the same queries above using SPARQL.

- Each student will deliver a report that contains the following:
- Snapshot/screenshot of RDF1, RDF2, RDF, and SameAs tables.
- A screenshot of each query and its results (on both sparql.us and Oracle), and description about what this query mean.

- Each student will be asked to demonstrate all queries in his/her (own laptop), and will be asked to execute additional queries.

References

1. Mustafa Jarrar, Anton Deik: **The Graph Signature: A Scalable Query Optimization Index for RDF Graph Databases Using Bisimulation and Trace Equivalence Summarization.** International Journal on Semantic Web and Information Systems, 11(2), 36-65, DOI: 10.4018/IJSWIS.2015040102. April-June 2015

2. Anton Deik, Bilal Faraj, Ala Hawash, Mustafa Jarrar: **Towards Query Optimization for the Data Web - Two Disk-Based algorithms: Trace Equivalence and Bisimilarity**. Proceedings of the 3rd Palestinian International Conference on Computer and Information Technology (PICCIT 2010). Hebron, Palestine. March 2010.

3. Mustafa Jarrar and Marios D. Dikaiakos: A Query Formulation Language for the Data Web. IEEE Transactions on Knowledge and Data Engineering. IEEE Computer Society.

4. Mustafa Jarrar, Marios D. Dikaiakos: **Querying the Data Web: the MashQL Approach**. IEEE Internet Computing. Volume 14, No. 3. Pages (58-670). IEEE Computer Society, ISSN 1089-7801. May 2010.Mustafa Jarrar and Marios D. Dikaiakos: **A Data Mashup Language for the Data Web** . Proceedings of LDOW, WWW'09. ACM. ISSN 1613-0073. (2009).

5. Mustafa Jarrar and Marios D. Dikaiakos: **MashQL: a query-by-diagram topping SPARQL -Towards Semantic Data Mashups**. Proceedings of ONISW'08, part of the ACM CiKM conference. ACM. pages (89-96) ISBN 9781605582559.(2008).