Lecture 4

SPARQL (Contd.)

- http://www.w3.org/TR/rdf-sparql-query/
- https://jena.apache.org/
- Chapter 3 of Semantic Web Primer

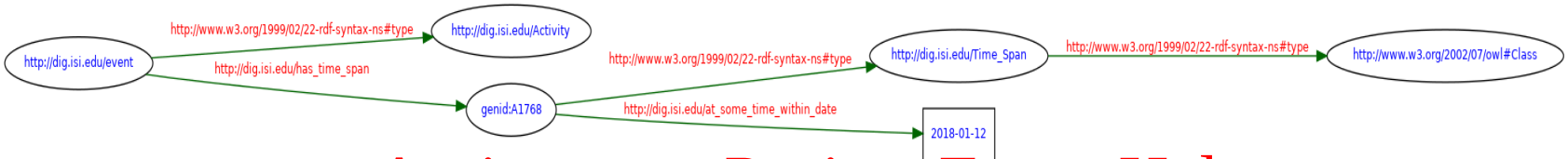
Dr. Davoud Mougouei

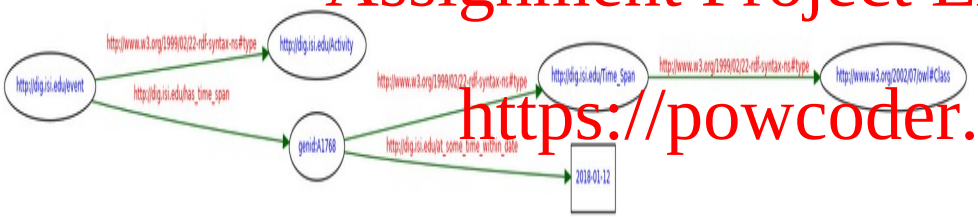
Blank Node!

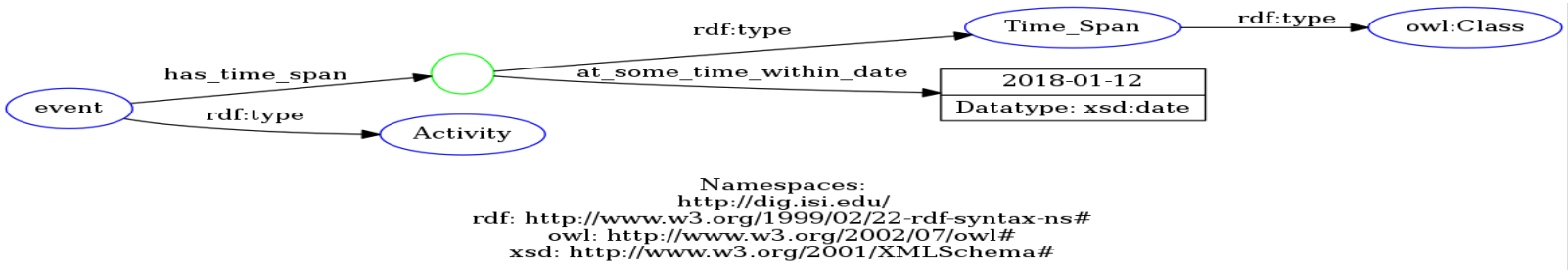
```
1.  <?xml version="1.0" encoding="utf-8" ?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:ns0="http://dig.isi.edu/"
4.           xmlns:owl="http://www.w3.org/2002/07/owl#"
5.           xmlns:dig = "http://dig.isi.edu/">

6.    <ns0:Activity rdf:about="http://dig.isi.edu/event">
7.      <ns0:has_time_span>
8.        <ns0:Time_Span>
9.          <ns0:at_some_time_within_date
            rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2018-01-12</ns0:at_some_time_wi
            thin_date>
10.       </ns0:Time_Span>
11.     </ns0:has_time_span>
12.   </ns0:Activity>
13. </rdf:RDF>
```

Blank Node!

```
1.  @prefix : <http://dig.isi.edu/> .
2.  @prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3.  @prefix owl:   <http://www.w3.org/2002/07/owl#> .
4.  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

5.  :Time_Span a owl:Class .
6.  :event a :Activity ;
7.      :has_time_span [
8.          a :Time_Span ;
9.          :at_some_time_within_date "2018-01-12"^^xsd:date
10.         ] .
```







Identify the blank node in the following XML/RDF document. Then write the TTL representation of the document.


```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ex="http://example.org/stuff/1.0/">

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar" dc:title="RDF 1.1 XML
Syntax">
        <ex:editor rdf:nodeID="abc"/>
</rdf:Description>

<rdf:Description rdf:nodeID="abc" ex:fullName="Dave Beckett">
        <ex:homePage rdf:resource="http://purl.org/net/dajobe/"/>
</rdf:Description>

</rdf:RDF>
```

Lecture Outline

1. SPARQL: Querying RDF Documents
2. Programming the semantic Web

Negation

Testing For the Presence of a Pattern

Data:

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

Result:

<http://example.org/ns#bob>

Modify the query so that the names are also included in the result set.

Negation

Testing For the Absence of a Pattern

Data:

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


Result:

```
<http://example.org/ns#bob>
```

Negation

Removing Possible Solutions

Data:

```
@prefix : <http://example.org/ns#> .
@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 foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?s WHERE { ?s ?p ?o . MINUS { ?s foaf:givenName "Bob" . } }
```

Result:

Try and see :)

Negation

NOT EXISTS vs. MINUS: sharing variables

Data:

```
@prefix : <http://example.org/ns#> .
:a :b :c .
```

Query :

```
SELECT * { ?s ?p ?o FILTER NOT EXISTS { ?x ?y ?z } }
```

Result:

???

Query :

```
SELECT * { ?s ?p ?o MINUS { ?x ?y ?z } }
```

Result:

???

Negation

NOT EXISTS vs. MINUS: sharing variables

Data:

```
@prefix : <http://example.org/ns#> .
:a :b :c .
```


Query :

```
SELECT * { ?s ?p ?o FILTER NOT EXISTS { ?x ?y ?z } }
```

Result:

No solutions: { ?x ?y ?z } matches given any ?s ?p ?o =>
NOT EXISTS { ?x ?y ?z } eliminates any solutions.

Query :

```
SELECT * { ?s ?p ?o MINUS { ?x ?y ?z } }
```

Result:

There is no shared variable between (?s ?p ?o) and (?x ?y ?z) =>
no bindings are eliminated.

Negation

NOT EXISTS vs. MINUS: fixed patterns

Data:

```
@prefix : <http://example.org/ns#> .
:a :b :c .
```


Query :

```
PREFIX : <http://example.org/ns#>
SELECT * { ?s ?p ?o FILTER NOT EXISTS { :a :b :c } }
```


Result:

???

Query :

```
SELECT * { ?s ?p ?o MINUS{ :a :b :c } }
```

Result:

???

Negation

NOT EXISTS vs. MINUS: fixed patterns

Data:

```
@prefix : <http://example.org/ns#> .
:a :b :c .
```


Query :


```
PREFIX : <http://example.org/ns#>
SELECT * { ?s ?p ?o FILTER NOT EXISTS { :a :b :c } }
```


Result:

No solutions.

Query :

```
SELECT * { ?s ?p ?o MINUS{ :a :b :c } }
```

Result:

There is no match of bindings => no solutions are eliminated.

Negation

NOT EXISTS vs. MINUS: inner filters

Data:

```
@prefix : <http://example.org/ns#> .
:a :p 1 . :a :q 1 . :a :q 2 .
:b :p 3.0 . :b :q 4.0 . :b :q 5.0 .
```

Query :


```
PREFIX : <http://example.org/ns#>
SELECT * WHERE { ?x :p ?n FILTER NOT EXISTS { ?x :q ?m . FILTER(?n = ?m) } }
```


Result:

x	n
<http://example.com/b>	3.0

Query :

```
SELECT * WHERE { ?x :p ?n MINUS { ?x :q ?m . FILTER(?n = ?m) } }
```

Result:

The FILTER inside the pattern does not have a value for ?n and it is always unbound:

x	n
<http://example.com/b>	3.0
<http://example.com/a>	1

Negation

Example (MINUS)

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:bob foaf:name "Bob" .
_:simon foaf:name "Simon" .
_:alice  foaf:name "Alice" ; foaf:knows _:simon .
 _:john  foaf:name "John" ; foaf:knows _:alice .
 _:eve   foaf:name "Eve" ; foaf:knows _:alice .
 _:dan   foaf:name "Dan" ; foaf:knows _:simon .
```

Query :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * WHERE { ?who foaf:name ?name
   MINUS {
     ?who  foaf:knows ?whom .
     ?whom foaf:name "Simon".
   }
 } LIMIT 50
```

Result:

Try and see :)

Using MINUS, Write a query that returns the names of the people except those who know someone that knows someone whose name is “Simon”.

- Using EXISTS, write a query that returns the names of the people who directly or indirectly (via someone else) know someone whose name is “Simon”

Assignment

BIND: Assigning to Variables

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 . :book1 ns:discount 0.2 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
:book2 ns:discount 0.25 .
```


Query :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
PREFIX : <http://example.org/book/>
SELECT ?title ?price { ?x ns:price ?p . ?x ns:discount ?discount BIND (?p*(1-?discount) AS ?price)
FILTER(?price < 20) ?x dc:title ?title . }
```

Result:

title	price
"The Semantic Web"	17.25

Write a query that returns the title and the price of the cheapest discounted book.

Assignment

VALUES: Providing inline data

- Data can be directly written in a graph pattern or added to a query using VALUES.
- VALUES allows multiple variables to be specified in the data block.


```
VALUES (?x ?y) {
  (:uri1 1)
  (:uri2 UNDEF)
}
```

Optionally, when there is a single variable and some values:

```
VALUES ?z { "abc" "def" }
```

which is the same as using the general form:

```
VALUES (?z) { ("abc") ("def") }
```

Assignment

VALUES: Providing inline data

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 : <http://example.org/book/>
PREFIX ns: <http://example.org/ns#>
SELECT ?book ?title ?price { VALUES ?book { :book1 :book3 }
?book dc:title ?title ; ns:price ?price . }
```

Result:

```
-------------------------------------
| book   | title             | price |
=====================================
| :book1 | "SPARQL Tutorial" | 42    |
-------------------------------------
```

Assignment

VALUES: Providing inline data

If a variable has no value for a particular solution in the VALUES clause, the keyword UNDEF is used instead of an RDF term.

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" ; ns:price 42 .
:book2 dc:title "The Semantic Web" ;
:book2 ns:price 23 .
```


Query :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/book/>
PREFIX ns: <http://example.org/ns#>
SELECT ?book ?title ?price { ?book dc:title ?title ; ns:price ?price . VALUES (?book ?title) { (UNDEF
"SPARQL Tutorial") (:book2 UNDEF) } }
```

Result:

book	title	price
:book1	"SPARQL Tutorial"	42
:book2	"The Semantic Web"	23

Assignment

VALUES: Providing inline data

The VALUES might have been specified to execute over the results of the SELECT query.

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" ; ns:price 42 .
:book2 dc:title "The Semantic Web" ;
:book2 ns:price 23 .
```


Query :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/book/>
PREFIX ns: <http://example.org/ns#>
SELECT ?book ?title ?price { ?book dc:title ?title ; ns:price ?price . } VALUES (?book ?title)
{ (UNDEF "SPARQL Tutorial") (:book2 UNDEF) }
```

Result:

book	title	price
:book1	"SPARQL Tutorial"	42
:book2	"The Semantic Web"	23

What are the results of the following queries? Compare the results.

Query 1

SELECT ?book ?title ?price {VALUES (?book ?title) { (UNDEF "SPARQL Tutorial") (:book2 UNDEF) (:book3 "SPARQL")}OPTIONAL{?book dc:title ?title} . OPTIONAL{ :book ns:price ?price} }

Query 2

SELECT ?book ?title ?price {OPTIONAL{?book dc:title ?title} . OPTIONAL{ :book ns:price ?price} VALUES (?book ?title) { (UNDEF "SPARQL Tutorial") (:book2 UNDEF) (:book3 "SPARQL")} }

Aggregates

- Aggregates apply expressions over groups of solutions, to see a result which is computed over a group of solutions, rather than a single solution.

- The maximum value that a particular variable takes, rather than each value individually.

- By default a solution set consists of a single group, containing all solutions. Grouping may be specified using the GROUP BY syntax.

- COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT, and SAMPLE.

Aggregates

GROUP BY

- In order to calculate aggregate values for a solution, the solution is first divided into one or more groups, and the aggregate value is calculated for each group.
- If aggregates are used in the query level in SELECT, HAVING or ORDER BY but the GROUP BY term is not used, then this is taken to be a single implicit group, to which all solutions belong.
- Within GROUP BY clauses the binding keyword, AS, may be used, such as GROUP BY (?x + ?y AS ?z). This is equivalent to { ... BIND (?x + ?y AS ?z) } GROUP BY ?z.
- For example, given a solution sequence S, ({?x→2, ?y→3}, {?x→2, ?y→5}, {?x→6, ?y→7}), we might wish to group the solutions according to the value of ?x, and calculate the average of the values of ?y for each group:

Query :

```
SELECT (AVG(?y) AS ?avg) WHERE { ?a :x ?x ; :y ?y . } GROUP BY ?x
```

Aggregates

Group By

Data:

```
@prefix : <http://example.org/book/>
:book1 :price 9 ; :sold 100 .
:book2 :price 5 ; :sold 106 .
:book3 :price 6 ; :sold 40 .
:book4 :price 8 ; :sold 50 .
```

Query :

```
PREFIX : <http://example.org/book/>
SELECT (AVG(?price) AS ?avg) WHERE { ?book :price ?price ; :sold ?sold . }
GROUP BY ?sold ORDER BY ?avg
```

Result:

```
-------
| avg |
=======
| 6.0 |
| 7.0 |
| 8.0 |
```

Aggregates

Having

- HAVING operates over grouped solution sets, in the same way that FILTER operates over un-grouped ones.

Data:


```
@prefix : <http://example.org/book/>
:book1 :price 9 ; :sold 100 .
:book2 :price 5 ; :sold 100 .
:book3 :price 6 ; :sold 40 .
:book4 :price 8 ; :sold 50 .
```


Query :

```
PREFIX : <http://example.org/book/>
SELECT (AVG(?price) AS ?avg) WHERE { ?book :price ?price ; :sold ?sold . } GROUP BY ?sold
HAVING (AVG(?price) > 6) ORDER BY ?avg
```

Result:

```
-------
| avg |
=======
| 7.0 |
| 8.0 |
-------
```

Can the query in the previous slide, be shortened?

Aggregates

Example

Data:

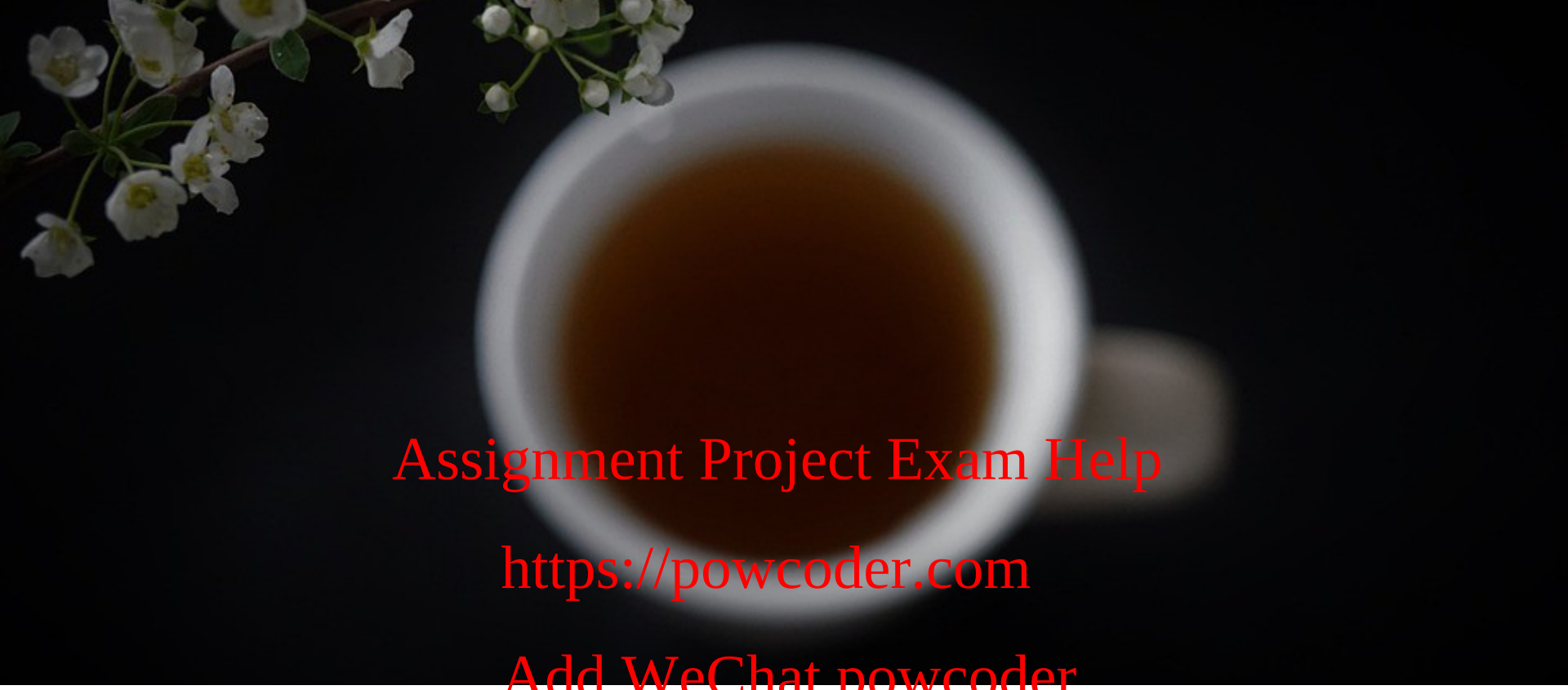
```
@prefix : <http://books.example/> .
:org1 :affiliates :auth1, :auth2 .
:auth1 :writesBook :book1, :book2 .
:book1 :price 9 .
:book2 :price 5 .
:auth2 :writesBook :book3 .
:book3 :price 7 .
:org2 :affiliates :auth3 .
:auth3 :writesBook :book4 .
:book4 :price 7 .
```


Query :

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice) WHERE { ?org :affiliates ?auth . ?auth :writesBook ?book . ?book :price ?lprice . } GROUP BY ?org HAVING (SUM(?lprice) > 10)
```

Result:

????



Take a break ☺

Solution Modifiers

ORDER BY

Data:

```
@prefix : <http://example.org/book/>
:book1 :price 9 ; :sold 100 .
:book2 :price 5 ; :sold 100 .
:book3 :price 6 ; :sold 40 .
:book4 :price 8 ; :sold 50 .
```

Query :

```
PREFIX : <http://example.org/book/>
SELECT ?book ?sold ?price WHERE { ?book :price ?price ; :sold ?sold} ORDER BY DESC(?sold) DESC(?price)
```

Result:

book	sold	price
:book1	100	9
:book2	100	5
:book4	50	8
:book3	40	6

Solution Modifiers

Projection

Data:

```
@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@work.example> .
```

Query :

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

Result:

```
----------
| name
==========
| "Bob"
| "Alice"
----------
```

Solution Modifiers

Duplicate Solutions

Data:

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


Query:

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

Result:

```
-----------
| name    |
===========
| "Alice" |
| "Alice" |
| "Alice" |
-----------
```

Solution Modifiers

Duplicate Solutions (DISTINCT)

Eliminates duplicate solutions.

Data:

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


Query :

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

Result:

```
-----------
| name    |
===========
| "Alice" |
-----------
```

Solution Modifiers

Duplicate Solutions (REDUCED)

- While the DISTINCT modifier ensures that duplicate solutions are eliminated from the solution set, REDUCED simply permits them to be eliminated.
- The cardinality of any set of variable bindings in a REDUCED solution set is at least one and not more than the cardinality of the solution set with no DISTINCT or REDUCED modifier.

Solution Modifiers

Duplicate Solutions (REDUCED)

Data:

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


Query :

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

Result:

????

Solution Modifiers

LIMIT

Data:

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


Query :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name WHERE { ?x foaf:name ?name } LIMIT 1
```

Result:

????

Solution Modifiers

OFFSET

causes the solutions generated to start after the specified number of solutions. An OFFSET of zero has no effect.

Data:


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


Query :


```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name WHERE { ?x foaf:name ?name } OFFSET 1
```

Result:

????

Solution Modifiers

Select different subsets of the query solutions

Data:

```
@prefix : <http://example.org/book/>
:book1 :price 9 ; :sold 100 .
:book2 :price 5 ; :sold 90 .
:book3 :price 6 ; :sold 60 .
:book4 :price 8 ; :sold 40 .
:book5 :price 4 ; :sold 45 .
:book6 :price 5 ; :sold 30 .
:book7 :price 3 ; :sold 20 .
:book8 :price 1 ; :sold 15 .
:book9 :price 8 ; :sold 10 .
:book10 :price 7 ; :sold 50 .
```


Query :

```
PREFIX : <http://example.org/book/>
SELECT * WHERE { ?x :sold ?sold} LIMIT 3 OFFSET 1
```

Result:

????

Solution Modifiers

Select different subsets of the query solutions

Data:

```
@prefix : <http://example.org/book/>
:book1 :price 9 ; :sold 100 .
:book2 :price 5 ; :sold 90 .
:book3 :price 6 ; :sold 60 .
:book4 :price 8 ; :sold 40 .
:book5 :price 4 ; :sold 45 .
:book6 :price 5 ; :sold 30 .
:book7 :price 3 ; :sold 20 .
:book8 :price 1 ; :sold 15 .
:book9 :price 8 ; :sold 10 .
:book10 :price 7 ; :sold 50 .
```


Query :

```
PREFIX : <http://example.org/book/>
SELECT * WHERE { ?x :sold ?sold} ORDER BY ?sold LIMIT 3 OFFSET 1
```

Result:

????

Using NOT EXISTS, write a query that returns, in descending order, the top 3 best-selling books that sold at least 30 times.

Solution Modifiers

Select different subsets of the query solutions

Data:

```
@prefix : <http://example.org/book/>
:book1 :price 9 ; :sold 100 .
:book2 :price 5 ; :sold 90 .
:book3 :price 6 ; :sold 60 .
:book4 :price 8 ; :sold 40 .
:book5 :price 4 ; :sold 45 .
:book6 :price 5 ; :sold 30 .
:book7 :price 3 ; :sold 20 .
:book8 :price 1 ; :sold 15 .
:book9 :price 8 ; :sold 10 .
:book10 :price 7 ; :sold 50 .
```


Query :

```
PREFIX : <http://example.org/book/>
SELECT ?price (AVG (?sold) AS ?S_AVG) WHERE { ?x :price ?price; :sold ?sold . FILTER (?sold>=30)}
GROUP BY ?price ORDER BY DESC (?S_AVG) LIMIT 6 OFFSET 0
```

Result:

????

Query Forms

- SPARQL has four query forms. These query forms use the solutions from pattern matching to form result sets or RDF graphs.
- SELECT Returns all, or a subset of, the variables bound in a query pattern match.
- CONSTRUCT Returns an RDF graph constructed by substituting variables in a set of triple templates.
- ASK Returns a boolean indicating whether a query pattern matches or not.
- DESCRIBE Returns an RDF graph that describes the resources found.

Query Forms (Construct)

Building RDF Graphs

- The CONSTRUCT query returns a single RDF graph specified by a graph template.
- The result is an RDF graph formed by taking each query solution in the solution sequence, substituting for the variables in the graph template, and combining the triples into a single RDF graph by set union

Query Forms (Construct)

Building RDF Graphs

Data:

```
@prefix org: <http://example.com/ns#> .
_:a org:employeeName "Alice" .
_:a org:employeeId 12345 .
_:b org:employeeName "Bob" .
_:b org:employeeId 67890 .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX org: <http://example.com/ns#>
CONSTRUCT { ?x foaf:name ?name } WHERE { ?x org:employeeName ?name }
```

Result:

```
@prefix org: <http://example.com/ns#> .
_:x foaf:name "Alice" .
_:y foaf:name "Bob" .
```

Query Forms (Construct)

Building RDF Graphs

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@example.org> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { <http://example.org/person#Alice> vcard:FN ?name}
WHERE { ?x foaf:name ?name }
```

Result:

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
<http://example.org/person#Alice> vcard:FN "Alice" .
```

Query Forms (Construct)

Templates with Blank Nodes

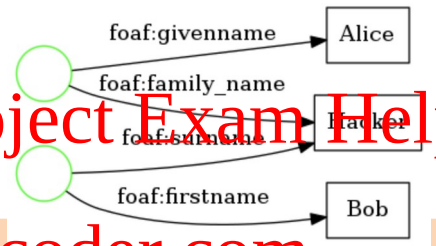
- A template can create an RDF graph containing blank nodes.
- The blank node labels are scoped to the template for each solution.
- If the same label occurs twice in a template, then there will be one blank node created for each query solution, but there will be different blank nodes for triples generated by different query solutions.

Query Forms (Construct)

Templates with Blank Nodes

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:givenname "Alice" .
_:a foaf:family_name "Hacker" .
_:b foaf:firstname "Bob" .
_:b foaf:surname "Hacker" .
```



Query:

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { ?x  vcard:N _:v . _:v vcard:givenName ?gname . _:v vcard:familyName ?fname }
WHERE
 {
   { ?x foaf:firstname ?gname } UNION  { ?x foaf:givenname   ?gname } .
   { ?x foaf:surname   ?fname } UNION  { ?x foaf:family_name ?fname } .
 }
```

Result:

???

Query Forms (Construct)

Templates with Blank Nodes

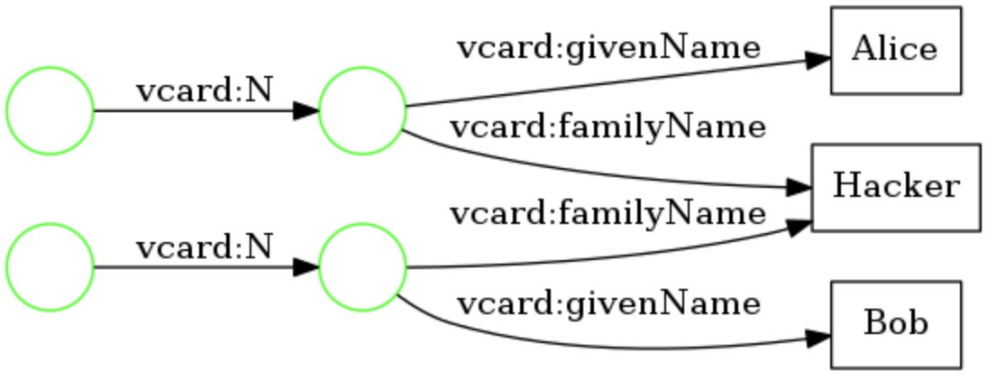
Result:

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
_:v1 vcard:N _:x . _:x vcard:givenName "Alice" . _:x vcard:familyName "Hacker" .
_:v2 vcard:N _:z . _:z vcard:givenName "Bob" . _:z vcard:familyName "Hacker" .
```

Result:


```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
[ vcard:N  [ vcard:familyName  "Hacker" ; vcard:givenName  "Bob"]] .
[ vcard:N  [ vcard:familyName  "Hacker" ; vcard:givenName  "Alice"]] .
```


Result:



Query Forms (Construct)

CONSTRUCT WHERE

Query

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

Query (short form); WHERE is essential

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

When to use the short form?

When the template and the pattern are the same and the pattern is just a basic graph pattern (no FILTERs and no complex graph patterns are used).

Write a query that constructs a graph of a data model containing all people older than 40.

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:givenname "Alice" ; foaf:family_name "Hacker" ; foaf:old 30 .
_:b foaf:firstname "Bob" ; foaf:surname "Hacker"; foaf:old 25 .
_:c foaf:givenname "Jeff" ; foaf:surname "Cool"; foaf:age 32 .
_:d foaf:firstname "David" ; foaf:surname "Goodman"; foaf:old 41 .
_:e foaf:givenname "Jenny" ; foaf:family_name "Cool"; foaf:age 50 .
_:f foaf:firstname "Sarah" ; foaf:surname "Hacker"; foaf:age 62 .
```

Query Forms (ASK)

- To 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.

Data:

```
@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> .
```

Query:

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

Result:

```
true
```

Query Forms (ASK)

Data:

```
@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> .
```


Query:

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

Result:

false

Query Forms (DESCRIBE)

- Returns a single result RDF graph containing RDF data about resources.
- This data is not prescribed by a SPARQL query, where the query client would need to know the structure of the RDF in the data source, but, instead, is determined by the SPARQL query processor.
- The query pattern is used to create a result set.
- The DESCRIBE form takes each of the resources identified in a solution, together with any resources directly named by IRI, and assembles a single RDF graph by taking a "description" which can come from any information available including the target RDF Dataset. The description is determined by the query service.

Query Forms (DESCRIBE)

Data

```
@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> .
```


Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
DESCRIBE ?x WHERE {?x foaf:mbox <mailto:bob@work.example>}
```


Result:


```
-------------------------------------------------------------------------
| s    | p                                | o                            |
=========================================================================
| _:b0 | <http://xmlns.com/foaf/0.1/name> | "Bob"                        |
| _:b0 | <http://xmlns.com/foaf/0.1/mbox> | <mailto:bob@work.example>    |
-------------------------------------------------------------------------
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

[ foaf:mbox  <mailto:bob@work.example> ;
  foaf:name  "Bob"
] .
```

Query Forms (DESCRIBE)

Data

```
@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> .
```


Query


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


Result:

```
-------------------------------------------------------------------------------
| s    | p                                    | o                                |
===============================================================================
| _:b0 | <http://xmlns.com/foaf/0.1/name>     | "Alice"                          |
| _:b0 | <http://xmlns.com/foaf/0.1/homepage> | <http://work.example.org/alice/> |
-------------------------------------------------------------------------------
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

[ foaf:homepage  <http://work.example.org/alice/> ;
  foaf:name      "Alice"
] .
```

Is DESCRIBE an essential query form? i.e., can we achieve the same results using other query forms?

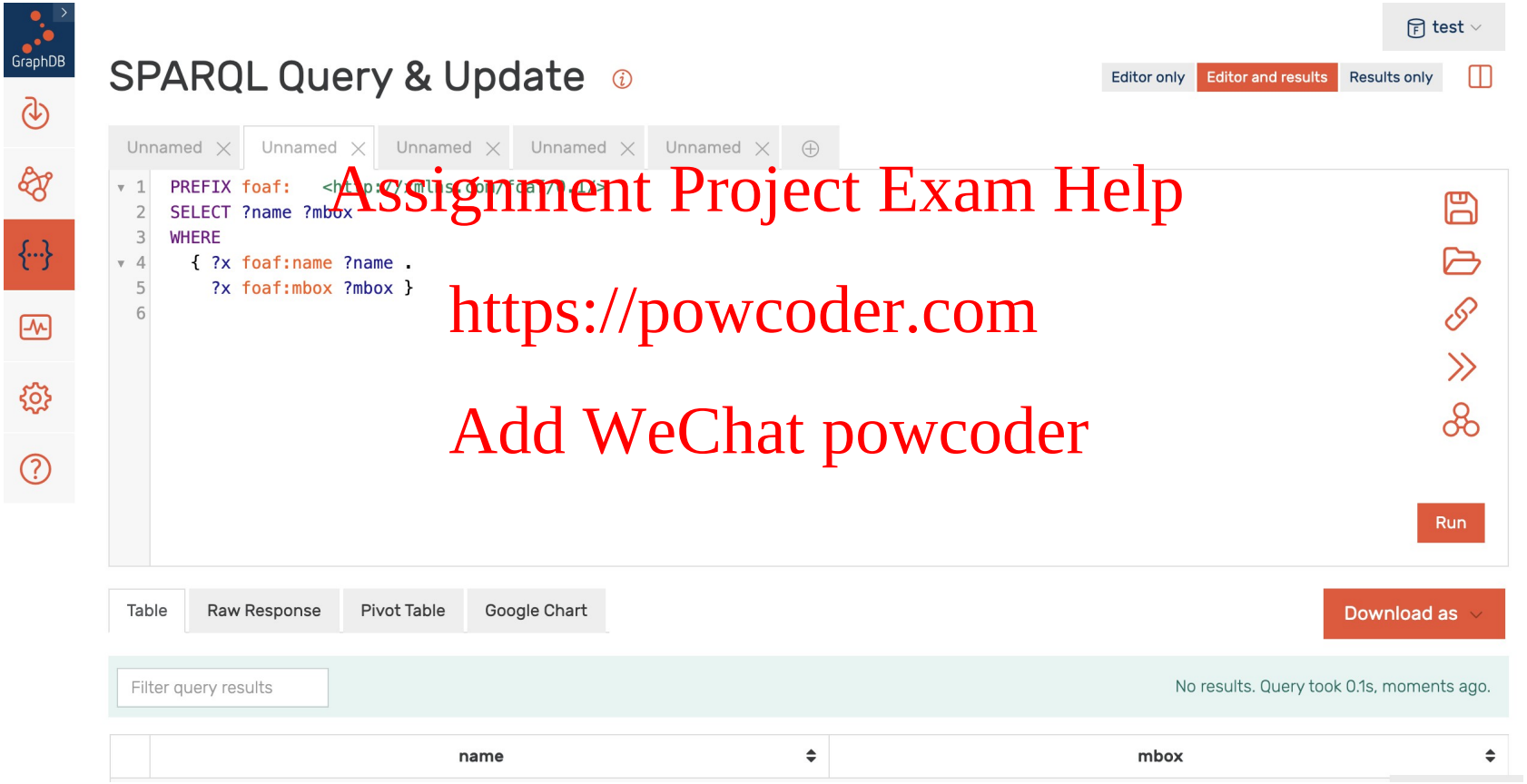
Lecture Outline

1. SPARQL: Querying RDF Documents
2. **Programming the semantic Web**

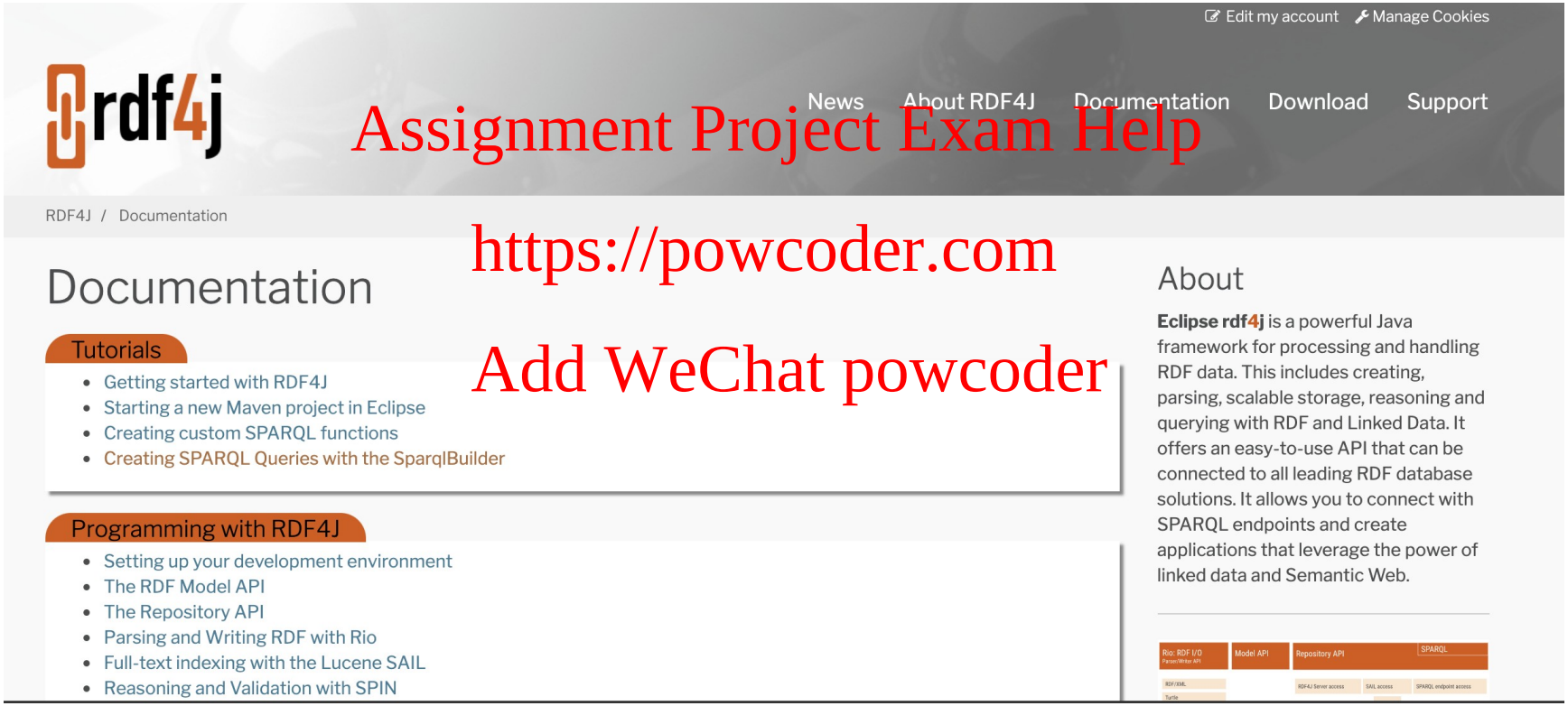
w

Programming the Semantic Web

GraphDB



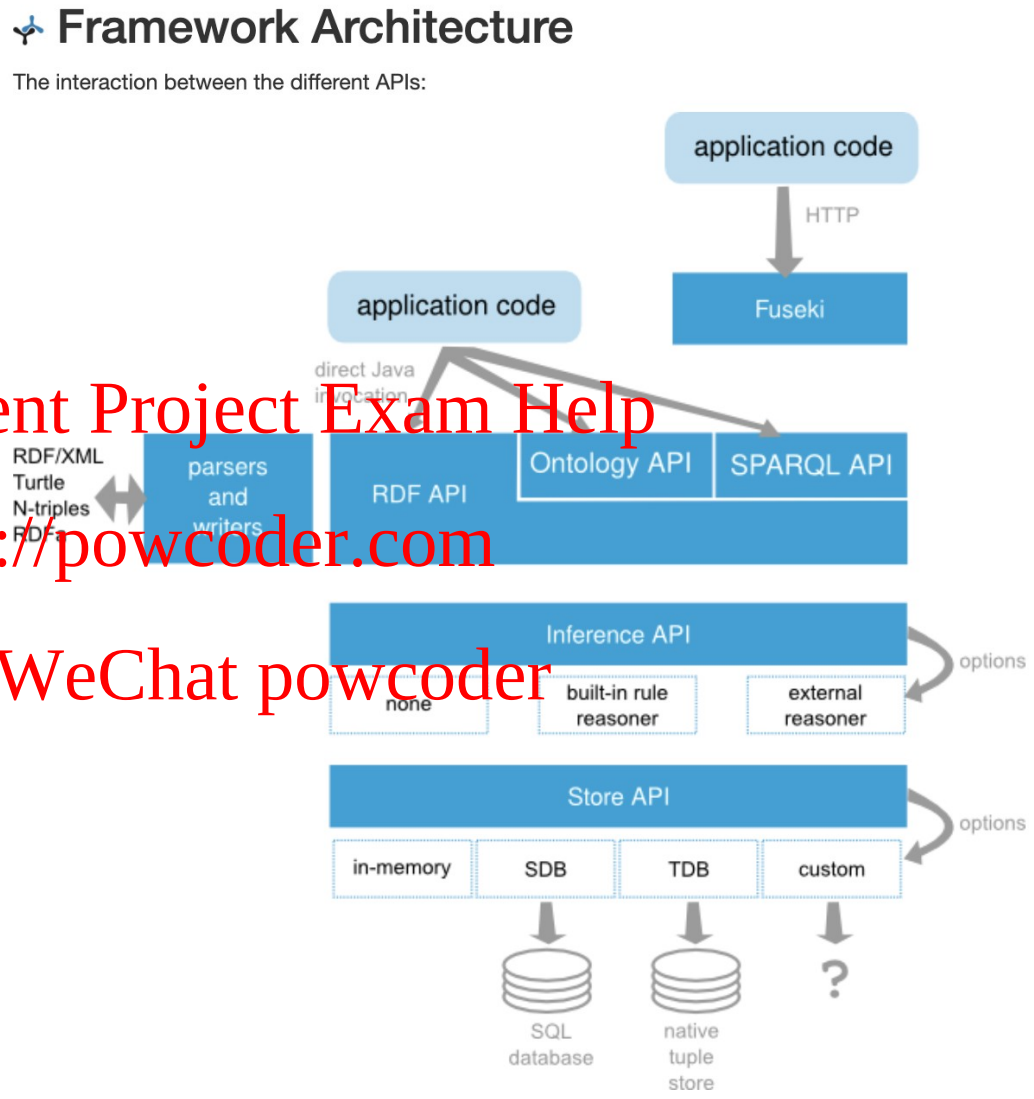
RDF4J



https://rdf4j.org/documentation/programming/

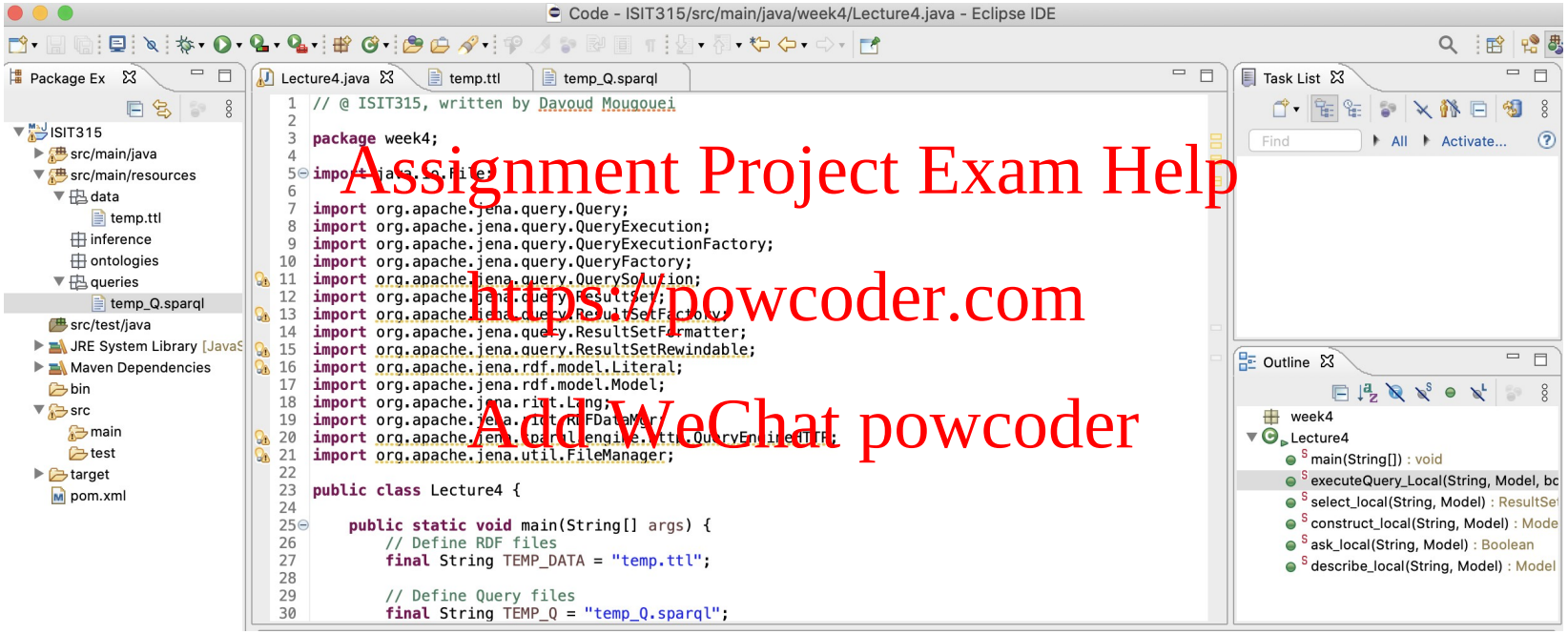
Apache Jena

Architecture



https://jena.apache.org/getting_started/index.html

Getting Started



Lab Exercise 2

- Task 1: Write and execute the SPARQL queries in Lecture 3 and Lecture 4 slides in both GraphDB and Apache Jena (download the eclipse project from Moodle). There is no need to submit your answers for this task.

- Task 2: Answer the questions in Lecture 3 and Lecture 4 slides (question slides) and submit your answers via Moodle as a single PDF file.