ISIT315
Week 6

SPARQL

Reference: https://www.w3.org/TR/rdf-sparql-query/

Allemang, D. & Hendler, J. Semantic Web for the working ontologist: effective modelling in RDFS and OWL (second edition) – available as ebook in Library – Chapter 5

SPARQL Basic Queries

- SPARQL is based on matching graph patterns
- Most forms of SPARQL query contain a set of triple patterns called a *basic graph pattern*
    - Each of the subject, predicate and object may be a variable

Example 1

- Given the following triple

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

- Question
  - What is the title of book1?

Query

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

Query Result

```
"SPARQL Tutorial"
```

Example 2 - Triple

```
@prefix :
  <http://aabs.purl.org/music#> .
:andrew :playsInstrument :guitar .
```

Query: which instrument does Andrew play?

```
PREFIX : <http://aabs.purl.org/music#>
SELECT ?instrument
WHERE {
      :andrew :playsInstrument ?instrument .
      }
```

Result

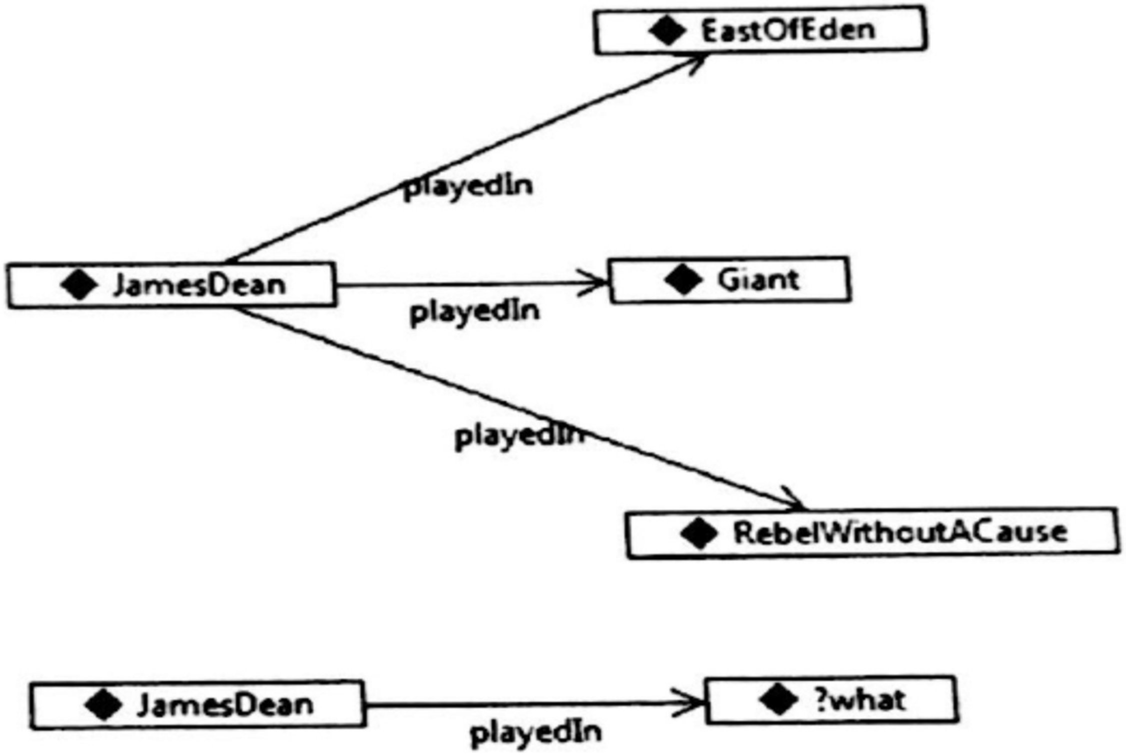
```
:guitar
```

SPARQL SELECT

- has two parts
  - a set of question words and a question pattern
  - WHERE indicate selection pattern
- example

```
SELECT ?instrument WHERE {:andrew
  :playsInstrument ?instrument .}
SELECT ?who WHERE {?who
  :playsInstrument :guitar .}
SELECT ?what WHERE {:andrew :?what
  :guitar .}
```

Consider the following RDF graph

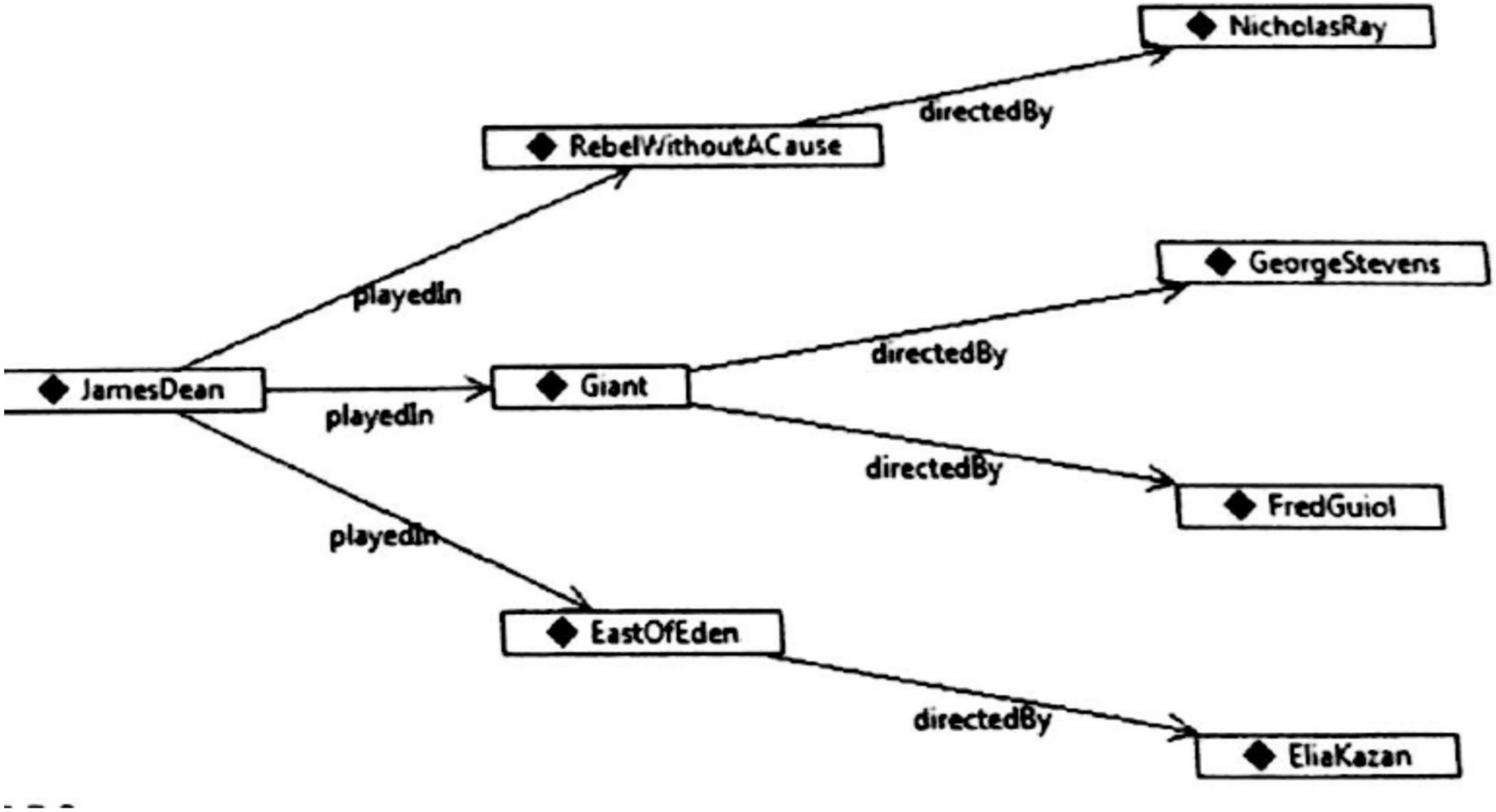


Question

- Write a SPARQL query to find out the movies that JamesDean played in.

```
SELECT ?what
WHERE {:JamesDean :playrdIn
?what .}
```

Consider the following RDF graph



Question

- Write a SPARQL query to find out the directors that direct the movies that JamesDean played in.

```
SELECT ?who
 WHERE {:JamesDean :playrdIn ?what .
        ?what :directedBy ?who .}
```

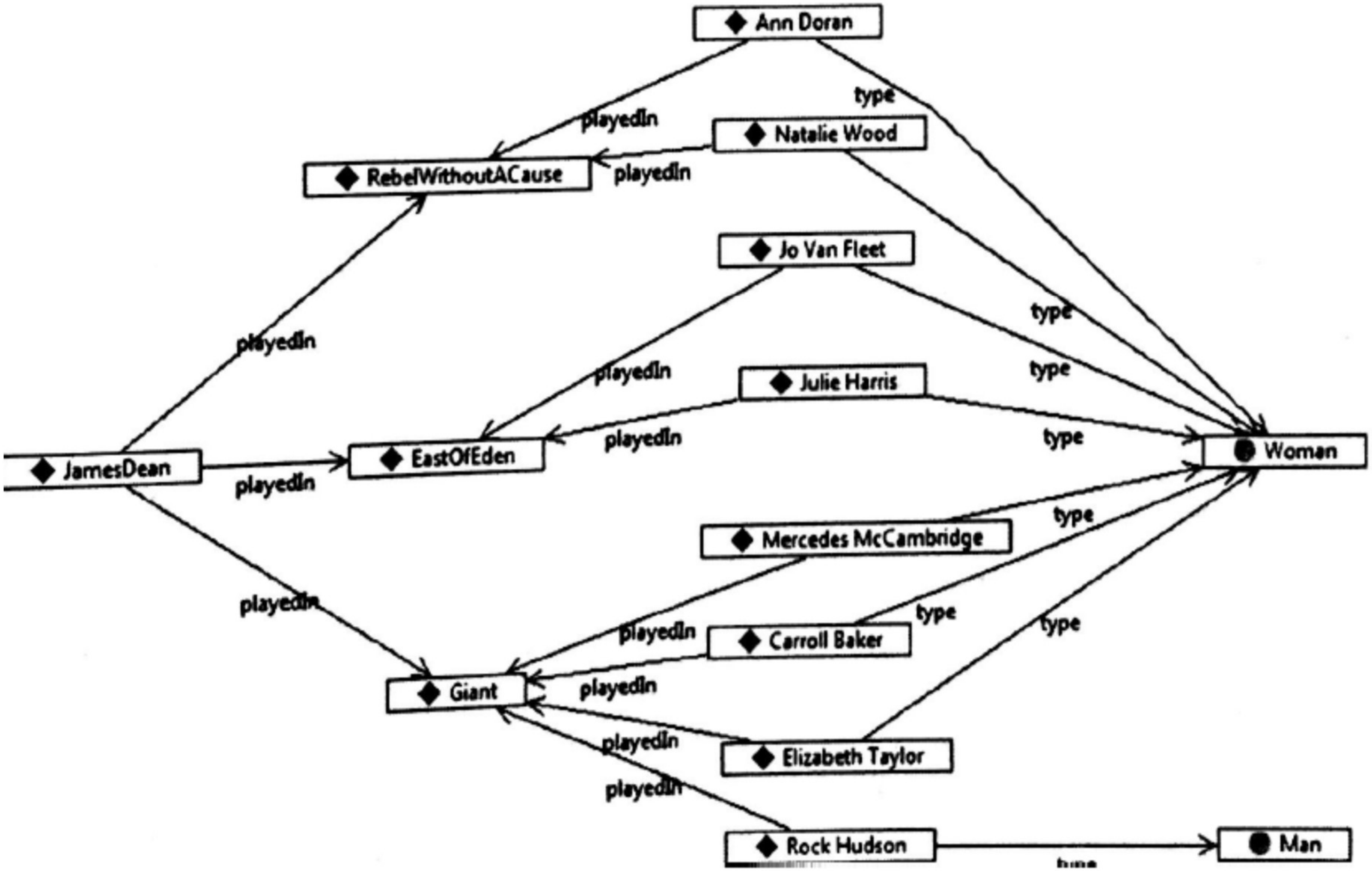
Result

```
?who
:GeorgeStevens
:EliaKazan
:NicholasRay
:FredGuiol
```

Additional question

- Modify SPARQL query in previous slide to find the movies JamesDean playedIn and the director of that movies.

Consider the following RDF graph



Query

- Write a SPARQL query to find the actresses who playedIn the same movies as JamesDean.

```
SELECT ?actress
 WHERE {:JamesDean :playrdIn ?movie .
        ?actress :playedIn ?movie .
        ?actress rdf:type :Woman .}
```

Query for properties

- Example: What do you know about JamesDean?

```
SELECT ?property ?value
WHERE ( :JamesDean ?property ?value)
```

SELECT DISTINCT

- You can use SELECT DISTINCT to filter out duplicate results

```
SELECT DISTINCT ?property ?value
WHERE ( :JamesDean ?property ?value)
```

Multiple matches – Example 1

Given the following:

```
@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 }
```

Result

name	**mailbox**
`“Johnny Lee Outlaw”`	`<mailto:jlow@example.com>`
`“Peter Goodguy”`	`<mailto:peter@example.com>`

Question:
Why is mailto:carol@example.org not output in result?

Consider another example

```
@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" .
```

If this query is executed, what is the result?

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

Result

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

Question: Why is the result for Bob is displayed in this case?
Hint: What does OPTIONAL mean in the query?

Matching RDF Literals - Example

```
@prefix dt: <http://example.org/datatype#> .
@prefix ns: <http://example.org/ns#> .
@prefix : <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:x ns:p "cat"@en .
:y ns:p "42"^^xsd:integer .
:z ns:p "abc"^^dt:specialDatatype .
```

Query 1

```
SELECT ?v
WHERE { ?v ?p “cat”}
```

What is the result?

Result

- No output

WHY?

Query 2

```
SELECT ?v
WHERE { ?v ?p “cat”@en}
```

What is the result?

Result

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

Query 3

```
SELECT ?v
WHERE { ?v ?p 42}
```

What is the result?

Result

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

Query 4

```
SELECT ?v
WHERE { ?v ?p
  "abc"^^<http://example.org/data
  type#specialDatatype> }
```

What is the result?

Result

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

Blank node labels

```
@prefix foaf:
  <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:b foaf:name "Bob" .
```

SPARQL – Filter

- restrict solutions to whose for which the filter expression evaluates to be TRUE

Example

```
@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 1

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

What is the result?

Result

Title
“SPARQL Tutorial”

Query 2

```
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 .
      }
```

What is the result?

Result

Title	price
“The Semantic Web”	23

CONSTRUCT

- The SELECT query form returns variable bindings.
- The **CONSTRUCT** query form returns an RDF graph.

Given the following 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" .
```