INFOH509 XML & WEB TECHNOLOGIES

LECTURE 9: SPARQL

Stijn Vansummeren
February 14, 2017
WHAT HAVE WE GAINED?

Current – no structure

In modern molecular biology, the genome is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of virus, in RNA.

The genome includes both the genes and the non-coding sequences of the DNA. The term was adapted in 1920 by Hans Winkler, Professor of Botany at the University of Hamburg, Germany. The Oxford English Dictionary suggests the name to be a portmanteau of the words gene and chromosome. A few related -ome words already existed, such as biome and rhizome, forming a vocabulary into which genome fits systematically.

Future – structured by RDF
(subject, predicate, object)

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>b:genome</td>
<td>b:field</td>
<td>b:molecular-bio</td>
</tr>
<tr>
<td>b:DNA</td>
<td>b:encode</td>
<td>b:genes</td>
</tr>
<tr>
<td>b:DNA</td>
<td>b:encode</td>
<td>b:non-coding-seq</td>
</tr>
<tr>
<td>b:genome</td>
<td>b:include</td>
<td>b:non-coding-seq</td>
</tr>
<tr>
<td>b:genome</td>
<td>b:include</td>
<td>b:gene</td>
</tr>
<tr>
<td>b:genome</td>
<td>b:related-to</td>
<td>b:rhizome</td>
</tr>
</tbody>
</table>

- RDF is meant to assert knowledge (statements) about entities (resources)
- By convention is clear what the subject, predicate, and object are
- With extra knowledge more inferences can be made (e.g.: John Doe is a person)
WHAT HAVE WE GAINED?

Current – no structure

Future – structured by RDF
(subject, predicate, object)

• RDF is meant to assert knowledge (statements) about entities (resources)
• By convention is clear what the subject, predicate, and object are
• With extra knowledge more inferences can be made (e.g.: John Doe is a person)
Envisions the Web as a single HUGE database consisting of RDF data
Query: What proteins are absent in diabetes patients?
Query: What proteins are absent in diabetes patients?
• **Query:** What proteins are absent in diabetes patients?

• **Required:** A means to query RDF
To support structured queries like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language

- SPARQL is essentially the SQL for RDF
To support structured queries like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language

- **SPARQL** is essentially the SQL for RDF

**Example**

```
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {

}
```
To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language

- SPARQL is essentially the SQL for RDF

**Example**

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```

To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language

- SPARQL is essentially the SQL for RDF

**Example**

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```
To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language

**SPARQL** is essentially the SQL for RDF

---

**Example**

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```

subject: ?disease

predicate: bio:scientific_name

object: "diabetes mellitus"
To support structured queries like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language

**SPARQL** is essentially the SQL for RDF

**Example**

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```
To support structured queries like “What proteins are absent in diabetes patients?”, the W3C has proposed SPARQL: Simple Protocol and RDF Query Language.

SPARQL is essentially the SQL for RDF.

Example

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```

subject  predicate  object
To support structured queries like “What proteins are absent in diabetes patients?”, the W3C has proposed SPARQL: Simple Protocol and RDF Query Language.

SPARQL is essentially the SQL for RDF.

Example

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```

subject predicate object
To support **structured queries** like “What proteins are absent in diabetes patients?”, the W3C has proposed **SPARQL**: Simple Protocol and RDF Query Language.

SPARQL is essentially the SQL for RDF.

**Example**

```sparql
PREFIX bio: <http://science.org/biology/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?prot_name
WHERE {
  ?disease bio:scientific_name "diabetes mellitus".
}
```

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>?disease</td>
<td>bio:scientific_name</td>
<td>&quot;diabetes mellitus&quot;.</td>
</tr>
</tbody>
</table>
• PREFIX directives can be used to abbreviate URIs
• The basic syntax is a SELECT - WHERE clause
• A FROM clause is optional
• The where clause consists of graph patterns: RDF triples with variables
• Variables are denoted by ?var with var a variable name

Example: directors who have acted in their own movies

PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
WHERE {
  ?movie mov:has-actor ?director .
}
Example Input:

```
mov:woody-allen
1-12-1935
mov:name
Woodie Allen

mov:directed-by
mov:has-actor
mov:title
mov:name
mov:birthdate
mov:directed-by
mov:has-actor
mov:title
mov:name
mov:birthdate

mov:clint-eastwood
Clint Eastwood

mov:the-rookie
The Rookie
```
SPARQL

- **PREFIX** directives can be used to abbreviate URIs
- The basic syntax is a **SELECT** - **WHERE** clause
- A **FROM** clause is optional
- The where clause consists of **graph patterns**: RDF triples with variables
- Variables are denoted by **?var** with **var** a variable name

Example: directors who have acted in their own movies

```sparql
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
WHERE {
  ?movie mov:has-actor ?director .
}
```
SPARQL

- **PREFIX** directives can be used to abbreviate URIs
- The basic syntax is a **SELECT** - **WHERE** clause
- A **FROM** clause is optional
- The where clause consists of **graph patterns**: RDF triples with variables
- Variables are denoted by `?var` with `var` a variable name

**Example: directors who have acted in their own movies**

PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
WHERE {
  ?movie mov:has-actor ?director .
  ?movie mov:has-actor ?director .
}

<table>
<thead>
<tr>
<th>?director</th>
<th>?movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov:woody-allen</td>
<td>mov:scoop</td>
</tr>
<tr>
<td>mov:clint-eastwood</td>
<td>mov:the-rookie</td>
</tr>
</tbody>
</table>
Prefix directives can be used to abbreviate URIs

The basic syntax is a **SELECT** - **WHERE** clause

A **FROM** clause is optional

The where clause consists of **graph patterns**: RDF triples with variables

Variables are denoted by `?var` with `var` a variable name

Example: directors who have acted in their own movies

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie
FROM <http://example.org/mdb.ttl>
WHERE {
  ?movie mov:has-actor ?director .
}
```

<table>
<thead>
<tr>
<th>?director</th>
<th>?movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov:woody-allen</td>
<td>mov:scoop</td>
</tr>
<tr>
<td>mov:clint-eastwood</td>
<td>mov:the-rookie</td>
</tr>
</tbody>
</table>
• **OPTIONAL** clauses allow you to use information in the RDF graph if it is present, but does not eliminate solutions if it is missing.

**Example: director must have a birthdate**

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie ?bd
WHERE {
    ?movie mov:has-actor ?director .
}
```

<table>
<thead>
<tr>
<th>?director</th>
<th>?movie</th>
<th>?bd</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov:woody-allen</td>
<td>mov:scoop</td>
<td>&quot;1-12-1935&quot;</td>
</tr>
</tbody>
</table>
**OPTIONAL** clauses allow you to use information in the RDF graph if it is present, but does not eliminate solutions if it is missing.

**Example: return birthdate if available**

```sparql
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director ?movie ?bd
WHERE {
    ?movie mov:has-actor ?director .
    OPTIONAL {?director mov:birthdate ?bd .}
}
```

<table>
<thead>
<tr>
<th>director</th>
<th>movie</th>
<th>bd</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov:woody-allen</td>
<td>mov:scoop</td>
<td>&quot;1-12-1935&quot;</td>
</tr>
<tr>
<td>mov:clint-eastwood</td>
<td>mov:the-rookie</td>
<td></td>
</tr>
</tbody>
</table>
FILTER clauses allow you to specify additional constraints on candidate solutions.

These constraints can use a small set of operators from XPath 2.0:

- The operator `bound` can be used to test if a variable is bound or not.
- The operator `regex` can be used to test if a variable matches a regular expression.
- The operators `<`, `>`, `<=`, `>=`, `=`, `!=` can be used to compare.

**Example: directors that do not have a birthdate**

```sparql
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
    OPTIONAL {?director mov:birthdate ?bd .}
    FILTER (!bound(?bd))
}
```

The result for `?director` includes `mov:clint-eastwood`.
SPARQL: FILTER

- FILTER clauses allow you to specify additional constraints on candidate solutions.
- These constraints can use a small set of operators from XPath 2.0.
- The operator `bound` can be used to test if a variable is bound or not.
- The operator `regex` can be used to test if a variable matches a regular expression.
- The operators `<`, `>`, `<=`, `>=`, `=`, `!=` can be used to compare.

Example: directors who’s name contains ”allen” as a substring

PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
    ?director mov:name ?name .
    FILTER regex(?name, "allen", "i")
}

?director

mov:woody-allen
FILTER clauses allow you to specify additional constraints on candidate solutions. These constraints can use a small set of operators from XPath 2.0:

- The operator `bound` can be used to test if a variable is bound or not.
- The operator `regex` can be used to test if a variable matches a regular expression.
- The operators `<`, `>`, `<=`, `>=`, `=`, `!=` can be used to compare.

Example: movies that have rating at least 3

```sparql
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?movie
WHERE {
  ?movie mov:rating ?x .
  FILTER (?x >= 3)
}
```
SPARQL: FILTER NOT EXISTS

- As of SPARQL 1.1 there is an easier way to find data that does not meet certain data using **FILTER NOT EXISTS**.
- **FILTER NOT EXISTS** takes as argument another graph pattern that should not be found in the data.

**Example: directors that do not have a birthdate**

```sparql
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
  FILTER NOT EXISTS { ?director mov:birthdate ?bd . }
}
```

?director mov:clint-eastwood
Using the **UNION** keyword, we can let patterns be evaluated independently.

Example: Directors who’s name contains ”allen” as a substring or do not have a birthdate.

```sparql
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?director
WHERE {
  {
    ?director mov:name ?name .
    FILTER regex(?name, "allen", "i")
  }
  UNION
  {
    OPTIONAL {?director mov:birthdate ?bd .}
    FILTER (!bound(?bd))
  }
}
SPARQL: CONSTRUCT

- **SELECT** queries returns tables listing variable bindings
- **CONSTRUCT** queries construct a new RDF graph

Example: assign all courses to John Doe

```
PREFIX terms: <http://ulb.be/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT
{
   <http://ulb.ac.be/staff/jdooe> terms:teaches ?x
}
WHERE { ?x rdf:type terms:course }
```
SPARQL: CONSTRUCT

- SELECT queries return tables listing variable bindings
- CONSTRUCT queries construct a new RDF graph

Example: assign all courses to John Doe and give them 5 credits

PREFIX terms: <http://ulb.be/terms>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT
{
  ?x terms:credits "5"
}
WHERE { ?x rdf:type terms:course }
### SPARQL: ASK

- **SELECT** queries returns tables listing variable bindings
- **CONSTRUCT** queries construct a new RDF graph
- **ASK** queries check whether the graph pattern occurs in the RDF graph (returns a boolean)

**Example: is there some resource with both a birthdate and a name?**

```
PREFIX mov: <http://movies-in-rdf.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
ASK
{
  ?x mov:name ?n .
  ?x mov:birthdate ?d
}
```
The **FROM** clause states that the query should be answered on a particular dataset (argument to the FROM clause)

The dataset is downloaded (if necessary) and then the query answer is computed.

**Example: query Tim-Berners-Lee’s FOAF file?**

```sparql
PREFIX dc: <http://purl.org/dc/elements/1.1>
SELECT ?title
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
}
```
The **SERVICE** keyword ships (a part of) a query to a remote SPARQL endpoint, and retrieves the results.

*(New in SPARQL 1.1.)*

Example: Ship a subquery to DBpedia?

```
SELECT ?p ?o
WHERE {
  SERVICE <http://dbpedia.org/sparql>
  { SELECT ?p ?o
    WHERE { <http://dbpedia.org/resource/Brussels> ?p ?o. } }
}
```
The **ORDER BY** modifier allows to sort the results (by default ascendingly, but one can specify descending also).

The **LIMIT** $N$ modifies retrieves only the first $N$ results.

Example: sort results by price (ascending) and, for items with the same price, by name (descending)

```sparql
PREFIX terms: <http://ulb.be/terms>
SELECT ?name ?price
WHERE {
  ?prod rdf:type terms:product .
  ?prod terms:name ?name .
}
ORDER BY ?price DESC(?name)
```
The ORDER BY modifier allows to sort the results (by default ascendingly, but one can specify descending also).

The LIMIT N modifies retrieves only the first N results.

Example: retrieve the product with the highest price

```sparql
PREFIX terms: <http://ulb.be/terms>
SELECT ?name ?price
WHERE {
    ?prod rdf:type terms:product .
    ?prod terms:name ?name .
}
ORDER BY ?price
LIMIT 1
```
SPARQL 1.1 allows aggregate operators \((MAX, MIN, AVG, \ldots)\) in the SELECT clause to compute aggregate values.

Example: compute the average price

```sparql
PREFIX terms: <http://ulb.be/terms>
SELECT AVG(?price) as ?avgPrice
WHERE {
  ?prod rdf:type terms:product .
  ?prod terms:name ?name .
}
```
REFERENCES