Laboratorio

Introducción al lenguaje SPARQL

Ulises Cortés
Ignasi Gómez-Sebastià
Sergio Álvarez

SID2018

Introducción

- **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- Lenguaje de consultas de-facto para la web semántica
- Funciona bajo RDF
  - Consultas basadas en tripletas
- Permite consultas en datos estructurados y semi-estructurados
- Permite consultas sobre las estructuras de datos de forma natural

Introducción

- Permite *joins* sobre bases de conocimiento no homogéneas
- Dispone de algunas extensiones
  - GeoSPARQL
  - SPARUL
    - SPARQL con DML
      - INSERT
      - UPDATE
- Versión 1.0 de 2008
  - Versión 1.1 de 2013

Estructura de una consulta

- **Prefijos: Permiten abreviar URIs que de otra forma serían muy largas (opcionales)**
- **Definición de modelos: Sobre que modelos RDF hacemos la consulta**
- **Resultado: Información que estamos obteniendo**
- **Patrón de consulta: Cómo obtenemos dicha información**
- **Modificadores de consulta: División, ordenado, etc**

```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...
# result
                    clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

Ejemplo de consulta I

```
select distinct ?Concept where {[] a ?Concept} LIMIT 100

PREFIX dbpedia: <http://dbpedia.org/property/birthName>
PREFIX actor: <http://dbpedia.org/ontology/Actor>
SELECT *
WHERE {
    ?actor a <http://dbpedia.org/ontology/Actor>.
?actor <http://dbpedia.org/property/birthName> ?nombre

} LIMIT 50
```

- ?Variable
- . Al final de cada línea en el WHERE
- WHERE va entre corchetes, no paréntesis
- Uso de tripletas
  - Notad como saltamos entre nodos

Ejemplo de una consulta

- https://sparql.uniprot.org/
- **Obtener los acrónimos y nombres de 100 enfermedades**

Ejemplo de una consulta

- https://sparql.uniprot.org/
- **Obtener los acrónimos y nombres de todas las enfermedades**

```
SELECT DISTINCT
?Concept
WHERE
{[] a ?Concept}
Limit 100
```

Ejemplo de una consulta

- https://sparql.uniprot.org/
- **Obtener los acrónimos y nombres de todas las enfermedades**

```
SELECT DISTINCT
?Concept
WHERE
{[] a ?Concept}
Limit 100
```

```
PREFIX up:<http://purl.uniprot.org/core/>
SELECT ?Disease
WHERE
{?Disease a up:Disease.}
Limit 100
```

Ejemplo de una consulta

- https://sparql.uniprot.org/
- **Obtener los acrónimos y nombres de todas las enfermedades**

```
PREFIX up:<http://purl.uniprot.org/core/>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
SELECT ?Disease ?Acronym ?Name
WHERE
{
?Disease a up:Disease.
?Disease skos:prefLabel ?Name.
?Disease up:mnemonic ?Acronym.
}
Limit 100
```

Ejemplo de consulta II

```
PREFIX actor: <http://dbpedia.org/ontology/Actor>
SELECT ?nombreActor, ?nombrePeli
WHERE {
    ?actor a <http://dbpedia.org/ontology/Actor>.
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://dbpedia.org/property/name> ?nombreActor.
?peli <http://xmlns.com/foaf/0.1/name> ?nombrePeli.
} LIMIT 50
```

```
PREFIX actor: <http://dbpedia.org/ontology/Actor>
SELECT ?nombreActor, ?nombrePeli
WHERE {
    ?actor a <http://dbpedia.org/ontology/Actor>.
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://dbpedia.org/property/name> ?nombreActor.
?peli <http://xmlns.com/foaf/0.1/name> ?nombrePeli.
?actor <http://dbpedia.org/property/name> "Bruce Lee"@en.
} LIMIT 50
```

```
PREFIX actor: <http://dbpedia.org/ontology/Actor>
SELECT ?nombreActor, ?nombrePeli
WHERE {
    ?actor a <http://dbpedia.org/ontology/Actor>.
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://dbpedia.org/property/name> ?nombreActor.
?peli <http://xmlns.com/foaf/0.1/name> ?nombrePeli.
FILTER regex(?nombreActor, "^Bruce")
} LIMIT 50
```

{ia,salvarez,igomez}@cs.upc.edu

Modificadores útiles

- Limit
- Ordery by
  - Order by ?X desc ?Y
- Offset
- Distinct
- A (rdf:type)
- Construct (Grafo RDF como resultado)
- Ask (boolean como resultado)
- Optional (Bases heterogeneas)
- SameTerm (?X, ?Y)

Modificadores útiles

- Limit
- Ordery by
  - Order by ?X desc ?Y

```
PREFIX up:<http://purl.uniprot.org/core/>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
SELECT ?Disease ?Acronym ?Name
WHERE
{
?Disease a up:Disease.
?Disease skos:prefLabel ?Name.
?Disease up:mnemonic ?Acronym.
}
ORDER BY ?Acronym
Limit 100
```

Modificadores útiles

- Limit
- Ordery by
  - Order by ?X desc ?Y
- Offset
- Distinct
  - Reduced
- A (rdf:type)
- Construct (Grafo RDF como resultado)
  - Describe (Descripción del grafo como resultado)

Modificadores útiles

- Limit
- Ordery by
  - Order by ?X desc ?Y
- Offset
- Distinct
  - Reduced
- A (rdf:type)
- Construct (Grafo RDF como resultado)
  - Describe (Descripción del grafo como resultado)

```
DESCRIBE <http://purl.uniprot.org/core/>
```

Modificadores útiles

- Ask (boolean como resultado)
- Optional (Bases heterogeneas)
- SameTerm (?X, ?Y)
- Comentarios
  - #Esto es un comentario

Modificadores útiles II

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

- BIND
  - Variables
    - SELECT ?nombre ?precioFinal { ?x precio ?precio . ?x descuento ?descuento BIND (?precio*(1-?descuento) AS ?precioFinal) ?x nombre ?nombre . }

Ejercicios

- **https://sparql.uniprot.org/sparql/**
- Obtener las enfermedades cuyo acrónimo sea *3KTD*

Ejercicios

- **https://sparql.uniprot.org/sparql/**
- Obtener las enfermedades cuyo acrónimo sea *3KTD*

```
PREFIX up:<http://purl.uniprot.org/core/>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
SELECT ?Disease ?Acronym ?Name
WHERE
{
?Disease a up:Disease.
?Disease skos:prefLabel ?Name.
?Disease up:mnemonic ?Acronym.
FILTER regex(?Acronym, "3KTD", "i")
}
Limit 100
```

DML

- **INSERT**

INSERT { <http://example/libro> titulo “Snowcrash" ;
Autor “Neal Stephenson”; Precio “Un huevo”} WHERE{}

- **DELETE**

DELETE{ ?x}WHERE{?x :titulo “SnowCrash”}

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

WITH <http://example/addresses>
DELETE { ?person foaf:givenName 'Bill' }
INSERT { ?person foaf:givenName 'William' }
WHERE
{ ?person foaf:givenName 'Bill'
}

DML II

- **UPDATE**

INSERT DATA { <http://example/libro> PrecioReal “Un huevo de rey”}

DELETE DATA { <http://example/libro> PrecioReal “Un huevo de rey”}

- **OTRAS**
  - **Load**
  - **Copy**
  - **Move**
  - **Clear**

Ejercicios

- Probar las consultas en la DBPedia
  - http://dbpedia.org/sparql
- Buscar películas de “Quentin Tarantino”
- Buscar nombres de actores porno y películas en las que han participado
- Añadir el director a la búsqueda anterior
- Buscar actores que han ganado un goya al mejor actor

- No olvidéis limitar a 50 la búsqueda!

Ejercicios

- Probar las consultas en la DBPedia
  - http://dbpedia.org/sparql
- Buscar películas de “Quentin Tarantino”

```
SELECT DISTINCT ?director, ?nombreDirector, ?peli, ?nombrePeli
WHERE {
?director a <http://dbpedia.org/ontology/Person> .
?peli <http://dbpedia.org/ontology/director> ?director .
?peli <http://www.w3.org/2000/01/rdf-schema#label> ?nombrePeli .
?director <http://www.w3.org/2000/01/rdf-schema#label> ?nombreDirector .
FILTER regex(?nombreDirector, "Tarantino")
FILTER regex(?nombreDirector, "Quentin")
FILTER(LANG(?nombrePeli ) = "" || LANGMATCHES(LANG(?nombrePeli ), "en"))
}
LIMIT 50
```

Ejercicios

- **Probar las consultas en la DBPedia**
  - http://dbpedia.org/sparql
- **Buscar nombres de actores porno y películas en las que han participado**

```
SELECT ?nombreActor, ?actor, ?nombrePeli, ?peli
WHERE {
?actor a <http://dbpedia.org/ontology/AdultActor>.
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://www.w3.org/2000/01/rdf-schema#label>?nombreActor.
?peli <http://www.w3.org/2000/01/rdf-schema#label> ?nombrePeli.
FILTER(LANG(?nombreActor ) = "" || LANGMATCHES(LANG(?nombreActor ), "en"))
FILTER(LANG(?nombrePeli ) = "" || LANGMATCHES(LANG(?nombrePeli ), "en"))
}
LIMIT 50
```

Ejercicios

- **Probar las consultas en la DBPedia**
  - http://dbpedia.org/sparql
- **Añadir el director a la búsqueda anterior**

```
SELECT ?nombreActor, ?actor, ?nombrePeli, ?peli, ?director
WHERE {
?actor a <http://dbpedia.org/ontology/AdultActor>.
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://www.w3.org/2000/01/rdf-schema#label>?nombreActor.
?peli <http://www.w3.org/2000/01/rdf-schema#label> ?nombrePeli.
?peli <http://dbpedia.org/property/director> ?director.
FILTER(LANG(?nombreActor ) = "" || LANGMATCHES(LANG(?nombreActor ), "en"))
FILTER(LANG(?nombrePeli ) = "" || LANGMATCHES(LANG(?nombrePeli ), "en"))
}
LIMIT 50
```

Ejercicios

- **Probar las consultas en la DBPedia**
  - http://dbpedia.org/sparql
- **Buscar actores que han ganado un goya al mejor actor**

```
SELECT ?nombreActor, ?actor, ?nombrePeli, ?peli, ?premio
WHERE {
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://www.w3.org/2000/01/rdf-schema#label>?nombreActor.
?peli <http://www.w3.org/2000/01/rdf-schema#label> ?nombrePeli.
?premio <http://dbpedia.org/property/bestActor> ?actor.
?premio <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Goya_Awards>.
FILTER regex(?nombreActor, "Bardem")
FILTER(LANG(?nombreActor ) = "" || LANGMATCHES(LANG(?nombreActor ), "en"))
FILTER(LANG(?nombrePeli ) = "" || LANGMATCHES(LANG(?nombrePeli ), "en"))
}
LIMIT 50

SELECT DISTINCT ?nombreActor
WHERE {
?peli <http://dbpedia.org/ontology/starring> ?actor.
?actor <http://www.w3.org/2000/01/rdf-schema#label>?nombreActor.
?peli <http://www.w3.org/2000/01/rdf-schema#label> ?nombrePeli.
?premio <http://dbpedia.org/property/bestActor> ?actor.
?premio <http://purl.org/dc/terms/subject> <http://dbpedia.org/resource/Category:Goya_Awards>.
FILTER(LANG(?nombreActor ) = "" || LANGMATCHES(LANG(?nombreActor ), "en"))
FILTER(LANG(?nombrePeli ) = "" || LANGMATCHES(LANG(?nombrePeli ), "en"))
}
LIMIT 50
```

Referencias

- https://www.w3.org/TR/2013/REC-sparql11-query-20130321/
- https://www.w3.org/TR/2013/REC-sparql11-update-20130321/