SPARQL: SPARQL Protocol And Query Language

Lenguajes y Tecnologías de la Web Semántica

Concepción de Sistemas de Información

2021

FACULTAD DE INGENIERIA

Contenidos

- Ideas Básicas

Contenidos

- Ideas Básicas

Qué es SPARQL?

- SPARQL Protocol And RDF Query Language.
- Simple Protocol And RDF Query Language ([Della Valle y Ceri, 2011])
- Tiene dos partes:
  - Un lenguaje de consulta sobre RDF.
  - Un protocolo que describe cómo hacer las consultas y cómo recuperar los resultados.
- Hay 2 versiones:
  - 1.0 : Recomendación desde enero 2008.
  - 1.1 : Recomendación desde marzo 2013 .

Triplestore

- Es una base de datos que utiliza RDF como modelo de datos nativo.
- Almacena grafos.
- Su interfase básica es SPARQL (Protocolo y Lenguaje).

RDF: Definiciones (repaso)

Terna RDF

Es una terna $< S, P, O >$ en donde:

- S es una URI o un nodo blanco.
- P es una URI
- O es una URI, un nodo blanco o un literal.

Grafo RDF

Conjunto de ternas

Dataset

Es un conjunto de grafos en donde hay un **grafo por defecto y un conjunto de Named Graphs**.

RDF: Definiciones

Named Graph

Es un grafo RDF con una URI asociada.

- Se considera que está formado por $QUADS$ - cuaternas formadas por la URI del grafo y una terna.

Anatomía de una Consulta

Abreviaturas

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
...
select  ?s ?p ?o
from    <http://dbpedia.org>
where   {
        ?s rdfs:label ``Raiders of the Lost Ark''@en.
         ?s ?p ?o
        }
order by ?s
limit 100
```

Resultado

Origen

Patrón de Grafo

Modificadores

La Consulta

- Abreviaturas
  - Pueden ser namespaces.
  - Pueden ser cualquier dirección que convenga abreviar.
- Resultados
  - Variables de las que se debe devolver el valor.
  - Deben aparecer en el patrón.
- Origen
  - Debe ser un Dataset RDF
- Modificadores
  - Típicos de SQL
  - Limit, Order by, Limit, Group by, etc.

Graph Patterns

- SPARQL se basa en pattern matching contra los grafos.
  - Grafo para ejemplos:
  - http://dbpedia.org
- Los patrones se escriben en Turtle con algunos agregados para construir patrones complejos.
- BGP: Basic Graph Pattern:
  - La forma más simple de patrones.
  - Describen un grafo, usando URIS, literales o variables en donde corresponda.
  - El patrón más simple que coindice con cualquier terna es

```
?s ?p ?o.
```

Ejemplos de BGP

?s skos:broader dbc:Tourist_attractions_in_London.

Devuelve los recursos que
son más específicos que
"Tourist attractions in London"

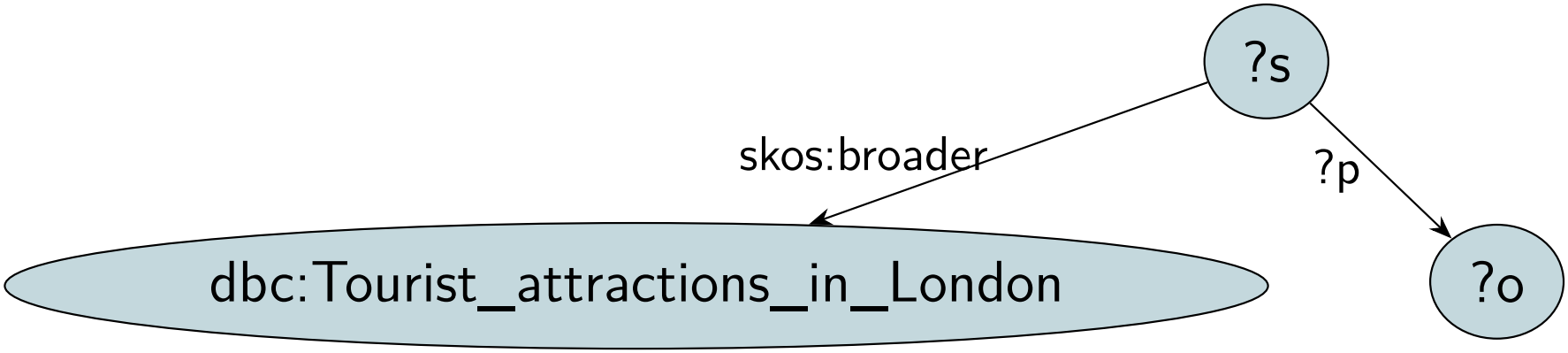
?s ?p dbc:Tourist_attractions_in_London.

Devuelve los recursos y las
propiedades que tienen esos recursos y apuntan a
"Tourist attractions in London"

Ejemplos de BGP

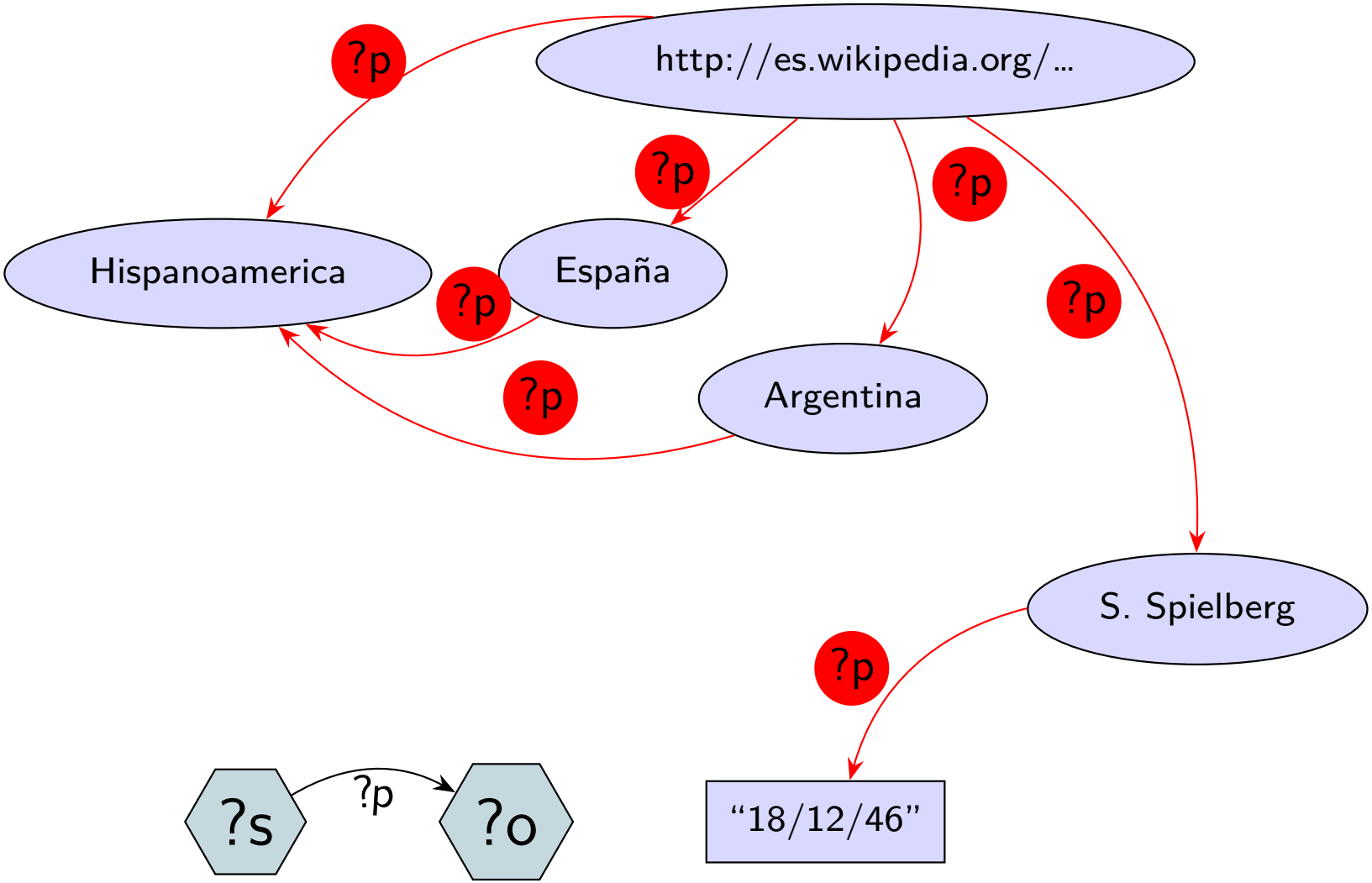
- Qué devuelve este patrón?

```
?s skos:broader dbc:Tourist_attractions_in_London.
?s ?p ?o.
```



Los objetos más específicos que "Tourist attractions in London" con todos los objetos (o valores) con que se relaciona y las propiedades correspondientes

Cómo se Computan los BGP? (intuición)



Cómo se Computan los BGP

- Un BGP devuelve un **conjunto de soluciones**.
- Cada solución es una **función parcial** que asocia una variable con un término rdf del grafo en cuestión.
- Luego se combinan o filtran esas funciones de acuerdo los nombres de las variables y la semántica de los operadores.

Cómo se Computan los BGP (ejemplo)

- Observar la siguiente consulta:

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core\#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
SELECT ?s ?o
WHERE {
   ?s rdfs:label ``Tourist attractions in London''@en
   ?s rdf:type skos:Concept.
   ?s skos:broader ?o .
}
```

- El BGP es:

```
?s rdfs:label ``Tourist attractions in London''@en.
?s rdf:type skos:Concept.
?s skos:broader ?o .
```

Cómo se Computan los BGP

- El resultado es un conjunto de funciones:

$$\mu_1[?s] = dbc : Tourist\_attractions\_in\_London$$
$$\mu_1[?o] = dbc : London$$
$$\mu_2[?s] = dbc : Tourist\_attractions\_in\_London$$
$$\mu_2[?o] = dbc : Tourist\_attractions\_in\_England\_by\_city$$
$$\mu_3[?s] = dbc : Tourist\_attractions\_in\_London$$
$$\mu_3[?o] = dbc : Tourism\_in\_London$$

- El dominio de cada función son los identificadores de variables y el rango, los valores.

Usando Literales

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?s
WHERE {
  ?s rdfs:label ``St Paul's Cathedral''.
}
```

Sin Idioma

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?s
WHERE {
   ?s rdfs:label ``St Paul's Cathedral''@en
}
```

Con Idioma

Usando Literales

```
SELECT ?s
WHERE  {
  ?s dbo:location dbr:London.
  ?s dbo:foundingYear "1968""^^xsd:gYear.
}
```

Con Tipo

```
SELECT ?s
WHERE {
  ?s dbo:location dbr:London.
  ?s dbo:foundingYear  1968.
}
```

Sin Tipo

Filtros

```
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT ?s
WHERE {
  ?s dbo:location dbr:London.
  ?s dbo:foundingYear ?c
      FILTER(?c>1950)
}
```

- El FILTER deja sólo las soluciones que cumplen con la condición.
- Hay muchas funciones para utilizar en un FILTER.
- http://www.w3.org/TR/sparql11-query/#OperatorMapping

Graph

- Es posible controlar sobre qué *named graph* se consulta:

```
select *
where { graph <http://dbpedia.org/resource/Uruguay> {
dbpedia:Uruguay ?p ?n }
}
```

- Trae datos sólo del *named graph* indicado indicado de los datasets indicados.
- Es posible iterar sobre los grafos de los datasets:

```
select distinct ?g
where { graph ?g { [] [] [] }
}
```

- Devuelve las uri's de los grafos del dataset que tienen algún dato.

Graph

- También se puede limitar sobre qué grafos del dataset se itera:

```
select distinct ?g ?p
from named <http://sws.geonames.org/Uruguay>
from named <http://www.bbc.co.uk/things/Uruguay>
where { graph ?g {
          [] ?p [] }
}
```

Optional

- Si hay soluciones que satizfazgan el patrón, entonces devuelve los valores.
- Si no existen estas soluciones, entonces el patrón devuelve verdadero.

```
select ?s  ?n  ?d
where
{?s dbo:birthPlace dbr:Uruguay.
?s a dbo:Person.
 ?s dbo:birthDate ?n.
optional { ?s dbo:deathDate ?d } }
```

Not Exists

- Se filtran las soluciones que cumplan con el patrón del *not exists*.

```
select ?s  ?n ?d
where
{?s dbo:birthPlace dbr:Uruguay.
?s a dbo:Person.
 ?s dbo:birthDate ?n.
 optional { ?s dbo:deathDate ?d } }
filter not exists { ?s dbo:deathDate [] } }
```

Minus

- Con `MINUS` se restan los conjuntos de soluciones de los patrones.

```
select ?s  ?n ?d
where {
{?s dbo:birthPlace dbr:Uruguay.
?s a dbo:Person.
 ?s dbo:birthDate ?n.
 minus
 { ?s dbo:deathDate [] } }
```

Path Expressions

- Es posible expresar caminos en el grafo usando expresiones regulares.

```
select *
where {
  dbr:Elizabeth_II
        dbo:birthPlace/dbo:country
              ?o.
}
```

```
select *
where {
  dbr:Elizabeth_II
        dbo:predecesor+
              ?o.
}
```

Consultas Federadas

- En la siguiente consulta, uno de los patrones es resuelto directamente por el endpoint de DBPEDIA.

```
SELECT ?s  ?o
FROM <http://dbpedia.org/iglesiasLondon>
WHERE {
?s rdfs:label [].
SERVICE <http://dbpedia.org/sparql> {
  ?o dcterms:subject ?s .
}
}
LIMIT 30
```

Lectura Recomendada

- http://www.cambridgesemantics.com/semantic-university/sparql-by-example
- http://www.w3.org/TR/sparql11-query/

Bibliografía

- E. D. Valle and S. Ceri, Querying the Semantic Web: SPARQL, in *Handbook of Semantic Web Technologies*, J. Domingue, D. Fensel, and J. A. Hendler, Eds. Springer Berlin Heidelberg, 2011, pp. 299–363.
- A. Kiryakov and M. Damova, Storing the Semantic Web: Repositories, in *Handbook of Semantic Web Technologies*, J. Domingue, D. Fensel, and J. A. Hendler, Eds. Springer Berlin Heidelberg, 2011, pp. 231–297.
- S. H. Garlik, A. Seaborne, and E. Prud'hommeaux, SPARQL 1.1 Query Language, Mar. 2013.
- Hitzler, P., Krötzsch, M., , Rudolph, S. (2009). *Foundations of Semantic Web Technologies*. Chapman , Hall/CRC.