UVa

Introducción a la Web Semántica

Web Semántica
Universidad de Valladolid
Curso 2016-2017

M. Mercedes Martínez
Dep. Informática (U. Valladolid, España)

SPARQL Query Language for RDF

Qué es

- Permite consultar y manipular grafos RDF. En la web o en un almacén RDF.

- Estabilizado en 2008, como un conjunto de tres especificaciones:
  - Lenguaje de consulta, *SPARQL Query Language for RDF*; es el núcleo de SPARQL
  - Formato XML para los resultados (de las consultas SPARQL), *SPARQL Query Results XML Format*
    - Fácil de procesar con herramientas XML como XSLT
  - Protocolo de acceso a datos para consultar de modo remoto bases de datos RDF, *SPARQL Protocol for RDF*

- Actualizado en marzo de 2013, como SPARQL 1.1.

SPARQL 1.1

- Estabilizado como recomendación del W3C con fecha 21 de marzo de 2013.

- Once especificaciones:
  1. SPARQL 1.1 Overview
  2. SPARQL 1.1 Query Language.
  3. SPARQL 1.1 Update
  4. SPARQL 1.1 Service Description
  5. SPARQL 1.1 Federated Query
  6. SPARQL 1.1 Query Results JSON Format
  7. SPARQL 1.1 Query Results CSV and TSV Formats
  8. SPARQL Query Results XML Format (Second Edition)
  9. SPARQL 1.1 Entailment Regimes
  10. SPARQL 1.1 Protocol
  11. SPARQL 1.1 Graph Store HTTP Protocol

SPARQL 1.1

- Estabilizado como recomendación del W3C con fecha 21 de marzo de 2013.

- Once especificaciones:
  1. SPARQL 1.1 Overview
  2. **SPARQL 1.1 Query Language.**
  3. **SPARQL 1.1 Update**
  4. SPARQL 1.1 Service Description
  5. SPARQL 1.1 Federated Query
  6. SPARQL 1.1 Query Results JSON Format
  7. SPARQL 1.1 Query Results CSV and TSV Formats
  8. **SPARQL Query Results XML Format (Second Edition)**
  9. SPARQL 1.1 Entailment Regimes
  10. SPARQL 1.1 Protocol
  11. SPARQL 1.1 Graph Store HTTP Protocol

Características

- Lenguaje de consulta para RDF
  - A diferencia de XQuery, no depende de la sintaxis del documento XML/RDF ⇒ no varía la consulta si la sintaxis XML del documento varía
- Basado en características de SQL
  - Se basa en un SELECT
- Permite expresar consultas sobre múltiples fuentes de datos, tanto si son datos RDF como si son accesibles bajo la forma de RDF a través de un middleware
- Los resultados de una consulta SPARQL pueden ser conjuntos de resultados XML (*result set*) o grafos RDF

Infraestructura necesaria

- Almacén RDF (*Triple store*)
  - Equivalente a una base de datos para RDF: 4store, Virtuoso, Sesame, …
- Se consulta usando el protocolo SPARQL
- El almacén proporciona un “punto SPARQL” (*SPARQL endpoint*)
  - Se pueden enviar consultas usando HTTP (en la URL del navegador)
  - Se puede usar un cliente diseñado específicamente para ello

Sintaxis SPARQL para las tripletas RDF

UVa

- Se basa en Turtle
- Se indica el final de cada tripleta con un punto

  <sujeto> <predicado> <objeto> .
- Las URI se escriben entre corchetes angulares

  <http://example.org/book/book1>
- Los literales es escriben entre comillas (simples o dobles)
- Las propiedades se suelen especificar usando una sintaxis de tipo ‘qname’ (prefijos) para hacerlas más legibles, aunque también pueden usarse las URI correspondientes

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

Variables en las consultas SPARQL

UVa

- Los patrones de tripletas SPARQL pueden incluir variables

  ?element table:name ?name .

- Se establece una ligadura entre cada variable y cada tripleta de datos que se ajusta al patrón
  - Se consideran todas las posibilidades

- Ejemplo: Consulta que retorna todas las tripletas de un grafo RDF

  ?subject ?predicate ?object

Estructura de una consulta SPARQL

UVa

```
PREFIX
SELECT
FROM
WHERE {  }
```

- PREFIX: Equivalente a declarar un espacio de nombres XML. Asocia un alias (label) con una URI
- SELECT: Inicia la consulta. Equivalente a la claúsula SELECT de SQL: se usa para definir los ítems de datos que devolverá la consulta
- FROM: Identifica los datos sobre los que se realiza la consulta. Puede haber múltiples claúsulas FROM en la misma consulta. Si no aparece, se consulta sobre el grafo activo
- WHERE: Especifica un patrón de tripletas contra el cual se constrastarán los datos consultados. El uso de la claúsula WHERE es opcional.

Estructura de una consulta SPARQL

UVa

```
PREFIX
SELECT
FROM
WHERE {  }
```

```
BASE
  PREFIX
  SELECT
  FROM
  WHERE {  }
```

- PREFIX: Equivalente a declarar un espacio de nombres XML. Asocia un alias (label) con una URI
- SELECT: Inicia la consulta. Equivalente a la claúsula SELECT de SQL: se usa para definir los ítems de datos que devolverá la consulta
- FROM: Identifica los datos sobre los que se realiza la consulta. Puede haber múltiples claúsulas FROM en la misma consulta. Si no aparece, se consulta sobre el grafo activo
- WHERE: Especifica un patrón de tripletas contra el cual se constrastarán los datos consultados. El uso de la claúsula WHERE es opcional.

Variante para simplificar el uso de las URIs

Estructura de los resultados

- El resultado de una consulta SPARQL es una tabla donde
  - Cada fila es un resultado
  - Cada columna se corresponde con una de las variables del SELECT

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

La claúsula WHERE

- Se pueden especificar varios patrones de tripletas en una claúsula WHERE, dando lugar a un grafo RDF

- No se puede usar en el SELECT una variable que no se usa en el patrón de grafo de la claúsula WHERE

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

Una consulta sencilla

- Grafo RDF

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

- Consulta SPARQL

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

- Resultado

title
"SPARQL Tutorial"

Consultas con múltiples resultados (matching)

- Las consultas pueden retornar múltiples resultados
  - Múltiples grafos se ajustan encajan con el patrón establecido

Datos

```
@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> .
```

Consulta

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

Resultado

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Patrones opcionales en el WHERE (I)

UVa

- Las consultas pueden retornar múltiples resultados
  - Múltiples grafos se ajustan encajan con el patrón establecido

Datos

```
@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> .
```

¿Por qué no está el nodo '_:c' entre los resultados?

¿Cómo conseguir que '_:c' esté entre los resultados?

Consulta

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

Resultado

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Patrones opcionales en el WHERE (II)

UVa

- Sirven para especificar la aparición opcional de ese patrón
  - Tienen en cuenta una característica de RDF (y XML): la aparición de elementos es opcional

- Se incluye la palabra clave OPTIONAL delante del patrón cuya aparición es opcional

patrón OPTIONAL {patrón}

Patrones opcionales en el WHERE (III)

- Se usa la palabra clave OPTIONAL

Datos

```
@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> .
```

Consulta

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

Resultado

Name	mbox
“Johnny Lee Outlaw”	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>
	<mailto:carol@example.org>

Patrones opcionales: Ejemplo

- Ejemplo de consulta con patrón opcional de la Recomendación SPARQL

Datos

```
@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" .
```

Consulta

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

Resultado

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

Patrones alternativos en la claúsula WHERE

- Unión de patrones, similar a SQL, expresada con la palabra clave UNION

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title  ?title } }
```

Patrones alternativos: Ejemplo

- Ejemplo de consulta con patrones alternativos de la Recomendación SPARQL

Datos

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .

_:a  dc10:title     "SPARQL Query Language Tutorial" .
_:a  dc10:creator   "Alice" .

_:b  dc11:title     "SPARQL Protocol Tutorial" .
_:b  dc11:creator   "Bob" .

_:c  dc10:title     "SPARQL" .
_:c  dc11:title     "SPARQL (updated)" .
```

Consulta

```
PREFIX dc10:
<http://purl.org/dc/elements/1.0/>
PREFIX dc11:
<http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION
{ ?book dc11:title  ?title } }
```

Resultado

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Literales: indicación del idioma (I)

- Ejemplo donde usamos un literal, indicando el idioma en que lo expresamos

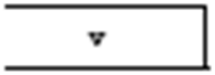
Datos

```
@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 .
```

Consulta

```
SELECT ?v WHERE { ?v ?p "cat" }
```

Resultado



Literales: indicación del idioma (II)

- Ejemplo donde usamos un literal, indicando el idioma en que lo expresamos

Datos

```
@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 .
```

Consulta

```
SELECT ?v WHERE { ?v ?p "cat"@en }
```

Resultado

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

Literales: números enteros

- Los enteros se corresponden con el tipo de datos *xsd:integer*

Datos

```
@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 .
```

Consulta

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

Resultado

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

Filtros basados en cadenas: FILTER + regex()

- FILTER permite filtrar. Por ejemplo, seleccionar cadenas en base a patrones (uso combinado con regex()
  - Una función muy utilizada con FILTER es **regex()**

Datos

```
@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 .
```

Consulta

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

Resultado

title
"SPARQL Tutorial"

Selección de enteros: FILTER + expresiones aritméticas

UVa

- FILTER permite seleccionar resultados en base a expresiones aritméticas

Datos

```
@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 .
```

Consulta

```
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 . }
```

Resultado

title	price
"The Semantic Web"	23

Uso de FILTER en patrones opcionales

UVa

Datos

```
@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 .
```

Consulta

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x dc:title ?title .
        OPTIONAL { ?x ns:price ?price .
                   FILTER (?price < 30) }
}
```

Resultado

title	price
"SPARQL Tutorial"	
"The Semantic Web"	23

Especificar los conjuntos RDF: FROM

- Fijar el grafo por defecto para la consulta

Datos

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

Consulta

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

Resultado

name
"Alice"

Especificar los conjuntos RDF: FROM NAMED

UVa

- Indicar los grafos usando su IRI

Datos

```
# Default graph (located at
http://example.org/foaf/aliceFoaf)
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
```

Consulta

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM  NAMED <http://example.org/foaf/aliceFoaf>
WHERE { ?x foaf:name ?name }
```

Resultado

name
"Alice"

Especificar los conjuntos RDF: FROM NAMED + GRAPH

UVa

- Indicar los grafos usando su IRI, y saber en qué grafo se encuentran los patrones

Consulta

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE  {
            GRAPH ?src
             { ?x foaf:mbox <mailto:bob@work.example> .
               ?x foaf:nick ?bobNick
             } }
```

Resultado

src	bobNick
<http://example.org/foaf/aliceFoaf>	"Bobby"
<http://example.org/foaf/bobFoaf>	"Robert"

Especificar los conjuntos RDF: FROM NAMED + GRAPH

UVa

- Indicar los grafos usando su IRI, y saber en qué grafo se encuentran los patrones

Datos (grafo 1)

```
# Named graph: http://example.org/foaf/aliceFoaf
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
_:a foaf:knows

_:b . _:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@work.example> .
_:b foaf:nick "Bobby" .
_:b rdfs:seeAlso <http://example.org/foaf/bobFoaf> .

<http://example.org/foaf/bobFoaf>
                rdf:type foaf:PersonalProfileDocument .
```

Datos (grafo 2)

```
# Named graph: http://example.org/foaf/bobFoaf
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
_:z foaf:mbox <mailto:bob@work.example> .
 _:z rdfs:seeAlso <http://example.org/foaf/bobFoaf> .
_:z foaf:nick "Robert" .

<http://example.org/foaf/bobFoaf> rdf:type
                foaf:PersonalProfileDocument .
```

Más sobre el uso de *Named Graphs*

UVa

Consulta	Acción
SELECT * FROM NAMED <http//a> FROM NAMED <http://b> {GRAPH <http://a> {?s ?p ?o} }	Los resultados proceden únicamente del grafo A, porque el uso de GRAPH restringe la aplicación del patrón al grafo A y el grafo A se especifica en la claúsula FROM NAMED
SELECT * FROM NAMED <http//a> {GRAPH <http://b> {?s ?p ?o} }	El resultado es el conjunto vacío en todos los casos. Explicación: GRAPH restringe la aplicación del patrón al grafo B, pero la claúsula FROM NAMED ha especificado que el grafo que se usará en la consulta es el grafo A (esto es, no se usará el grafo B)
SELECT ?s ?p ?o FROM NAMED <http//a> {GRAPH ?g {?s ?p ?o} } SELECT ?s ?p ?o {GRAPH <http://a> {?s ?p ?o} }	Dos consultas equivalentes (devuelven el mismo resultado)

Property Paths

- Permiten describir patrones de modo más breve, y trabajar con caminos (en los grafos) de longitud variable

- Más sobre la sintaxis: https://www.w3.org/TR/sparql11-query/#propertypaths

Property Paths: Ejemplos

- Mostrar los nombres de las personas relacionadas a través de dos saltos de *foaf:knows*

Consulta sin paths

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows ?a1 .
  ?a1 foaf:knows ?a2 .
  ?a2 foaf:name ?name .
}
```

Consulta con paths

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:knows/foaf:name ?name .
}
```

- Mostrar los nombres de las personas relacionadas a través de *foaf:knows* con Alice

Consulta con paths

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name .
}
```

Resultados ordenados

- Se usa la claúsula ORDER BY, que siempre debe ir después de la claúsula WHERE
- Similar al ORDER BY de SQL

```
PREFIX     :    <http://example.org/ns#>
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>

SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY DESC(?emp)
```

Limitar el tamaño del resultado

- Se usa LIMIT
  - Los valores deben ser positivos

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name }
LIMIT 20
```

Construcción de grafos: CONSTRUCT

- CONSTRUCT retorna un grafo RDF

Datos

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

Consulta

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

Resultado

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

Construcción de grafos: CONSTRUCT

- CONSTRUCT retorna un grafo RDF

Datos

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

Consulta

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

Resultado (expresado con RDF/XML)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
<rdf:Description><foaf:name>Alice</foaf:name></rdf:Description>
<rdf:Description><foaf:name>Bob</foaf:name></rdf:Description>
</rdf:RDF>
```

Consulta sobre la existencia de un grafo: ASK

UVa

- La claúsula ASK sirve para preguntar si existe un grafo que cumpla las condiciones de la consulta
  - La respuesta es verdadero (*true*) o falso (*false*)

Datos

```
@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> .
```

Consulta

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

Resultado

```
true
```

Consulta sobre la existencia de un grafo: ASK

UVa

- La claúsula ASK sirve para preguntar si existe un grafo que cumpla las condiciones de la consulta
  - La respuesta es verdadero (*true*) o falso (*false*)

Datos

```
@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> .
```

Consulta

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

Resultado

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
   <head></head>
   <results><boolean>true</boolean></results>
</sparql>
```

Formato XML de los resultados (I)

UVa

- Existe un espacio de nombres específico:
  http://www.w3.org/2005/sparql-results#
- El elemento raíz es *sparql*. Contiene un elemento *head* y un elemento *results*
- Dentro del elemento *head* se declaran todas las variables que se devolverán en el resultado
- Dentro del elemento *results* se listan los resultados: un elemento *result* por cada fila del conjunto resultado
- Dentro de cada elemento *result* hay una ligadura (*binding*) para cada variable, que es una URI o un literales resultado. Si no se ha podido asociar una variable con nada (OPTIONAL) aparece marcada como *unbound*

Formato XML de los resultados (II)

UVa

- Patrón de un documento XML con los resultados de una consulta SPARQL

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 ...
</sparql>
```

Formato XML de los resultados: elemento HEAD

UVa

- Contiene la secuencia de variables que aparecen en el resultado, en el mismo orden en que se expresaron en el SELECT de la consulta SPARQL

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">

  <head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="mbox"/>
    <variable name="blurb"/>
  </head>
...
</sparql>
```

Formato XML de los resultados: elementos RESULT

UVa

- Cada resultado se describe mediante un elemento de tipo RESULT

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  ...  head ...

  <results>
    <result>...
    </result>
    <result>...
    </result>
    ...
  </results>

</sparql>
```

Formato XML de los resultados: elementos BINDING

UVa

- Cada ligadura entre una variable y un valor se especifica con un elemento de tipo BINDING

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="x"/>
    <variable name="hpage"/>
  </head>
  <results>
    <result>
      <binding name="x"> ... </binding>
      <binding name="hpage"> ... </binding>
    </result>
...
  </results>
</sparql>
```

SPARQL Update

- Permite actualizar grafos RDF

- Dos posibilidades de uso:
  - Mediante OPERACIONES que modifican el grafo almacenado en un almacén RDF
  - Mediante PETICIONES (request) que se envían a un almacén RDF
    - Cada petición es una secuencia de operaciones.
    - Ej.: Cuando se usa *SPARQL 1.1 Protocol for RDF* una petición es un HTTP POST.

SPARQL 1.1 Update

- OPERACIONES:
  - INSERT DATA
    - Añade tripletas a un grafo.
    - Las tripletas están incluidas en cada *request* enviada al almacén.
    - Crea el grafo de destino en caso de que no exista previamente.
  - DELETE DATA
    - Borra tripletas de un grafo.
    - Las tripletas están incluidas en cada *request* enviada al almacén.

SPARQL 1.1 Update

- INSERT / DELETE
  - Grupos de tripletas que se insertan o eliminan en el grafo.
  - Se pueden usar variables (a diferencia de INSERT DATA / DELETE DATA)

- LOAD
  - Carga el contenido de un documento que contiene un grafo RDF en el almacén RDF

- CLEAR
  - Elimina todas las tripletas de un grafo.

SPARQL 1.1 Update

- Operaciones sobre grafos
  - CREATE
    - Crea un grafo en el almacén RDF
  - DROP
    - Elimina el grafo indicado del almacén
  - COPY
    - Se copias todas las tripletas de un grafo en el grafo de destino, eliminando lo que hubiese antes
  - MOVE
    - Se mueven todas las tripletas del grafo origen hacia el grafo de destino (se borran del primero)
  - ADD
    - Se *agregan* las tripletas del grafo origen en el grafo destino

Inserción de tripletas (I)

- INSERT DATA
  - Se inserta en el grafo por defecto

Datos - ANTES

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .

 <http://example/book1> ns:price 42 ..
```

Datos - DESPUÉS

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .

<http://example/book1> ns:price 42 .
<http://example/book1> dc:title "A new book" .
<http://example/book1> dc:creator "A.N.Other" .
```

Inserción

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT DATA
 {
 <http://example/book1> dc:title "A new book" ;
                                dc:creator "A.N.Other" .
}
```

Inserción de tripletas (II)

- INSERT DATA
  - Se inserta en el grafo http://example/bookStore

Datos - ANTES

```
# Graph: http://example/bookStore
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example/book1> dc:title "Fundamentals of
Compiler Design" .
```

Inserción

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
INSERT DATA
{ GRAPH http://example/bookStore
      { <http://example/book1> ns:price 42 } }
```

Datos - DESPUÉS

```
# Graph: http://example/bookStore
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book1> dc:title "Fundamentals of
Compiler Design" .
<http://example/book1> ns:price 42 .
```

Borrado de tripletas (I)

- DELETE DATA
  - Se borra del grafo por defecto

Datos - ANTES

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book2> ns:price 42 .
<http://example/book2> dc:title "David Copperfield" .
<http://example/book2> dc:creator "Edmund Wells" .
```

Borrado

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
DELETE DATA
{
   <http://example/book2>
                        dc:title "David Copperfield" ;
                        dc:creator "Edmund Wells" .
}
```

Datos - DESPUÉS

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book2> ns:price 42 .
```

INSERT / DELETE (I)

- INSERT y DELETE combinados
  - Se actualiza el grafo http://example/addresses

Datos - ANTES

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://example/president25> foaf:givenName "Bill" .
<http://example/president25> foaf:familyName "McKinley" .
<http://example/president27> foaf:givenName "Bill" .
<http://example/president27> foaf:familyName "Taft" .
<http://example/president42> foaf:givenName "Bill" .
<http://example/president42> foaf:familyName "Clinton" .
```

Borrado

```
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' }
```

Datos - DESPUÉS

```
# Graph: http://example/addresses
 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://example/president25> foaf:givenName "William" .
<http://example/president25> foaf:familyName "McKinley" .
<http://example/president27> foaf:givenName "William" .
<http://example/president27> foaf:familyName "Taft" .
<http://example/president42> foaf:givenName "William" .
<http://example/president42> foaf:familyName "Clinton" .
```

INSERT / DELETE (II)

- INSERT y DELETE, con variables
  - Se actualiza el grafo http://example/addresses

Datos - ANTES

```
# Graph: http://example/addresses
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://example/william> a foaf:Person .
<http://example/william> foaf:givenName "William" .
<http://example/william> foaf:mbox <mailto:bill@example> .
<http://example/fred> a foaf:Person .
<http://example/fred> foaf:givenName "Fred" .
<http://example/fred> foaf:mbox <mailto:fred@example> .
```

Borrado

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
WITH <http://example/addresses>
DELETE { ?person ?property ?value }
WHERE { ?person ?property ?value ;
              foaf:givenName 'Fred' }
```

Datos - DESPUÉS

```
# Graph: http://example/addresses
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://example/william> a foaf:Person .
<http://example/william> foaf:givenName "William" .
<http://example/william> foaf:mbox <mailto:bill@example> .
```

Herramientas SPARQL

- Validadores SPARQL
  - SPARQL Validator: http://sparql.org/query-validator.html
- Consultas on-line a repositorios RDF utilizando SPARQL
  - DBPedia Public SPARQL Endpoint, http://dbpedia.org/sparql

Referencias

- Artículos:
  - [Dodds05] “Introducing SPARQL: Querying the Semantic Web”. Leigh Dodds. *XML.com,* 2005. Disponible en http://www.xml.com/pub/a/2005/11/16/introducing-sparql-querying-semantic-web.
- Tutoriales:
  - Tutorial sobre SPARQL asociado a Jena, http://jena.apache.org/tutorials/sparql.html
  - *SPARQL by Example*. W3C SPARQL Working Group. http://www.cambridgesemantics.com/semantic-university/sparql-by-example

DBpedia

- Datos extraidos de la Wikipedia y convertidos en RDF
- Disponible en http://dbpedia.org/
- La usaremos para las consultas SPARQL. Se pueden realizar en el *SPARQL Access Poin*t, accesible desde la web de Dbpedia