Lenguaje de consulta SPARQL

PID_00271448

Blas Torregrosa García

Tiempo mínimo de dedicación recomendado: 1 hora

Universitat Oberta de Catalunya

Blas Torregrosa García

Ingeniero en Informática y máster universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC) por la Universitat Oberta de Catalunya (UOC). Especializado en ciberseguridad. Profesor colaborador en el máster de Ciencia de Datos de la UOC y profesor asociado en la Universidad de Valladolid (UVA).

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Ferran Prados Carrasco (2020)

Primera edición: febrero 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción

El lenguaje de consulta semántico SPARQL[1] permite la recuperación de datos mediante programación. Al igual que las bases de datos relacionales tienen su propio lenguaje de consulta (SQL o *Structured Query Language*), las tecnologías web semánticas también tienen su propio lenguaje de consulta, que es el mecanismo que permite enviar consultas y obtener resultados.

[(1)]Recursivamente en inglés, *SPARQL Protocol and RDF Query Language* pronunciado /sparkel/.

El lenguaje se basa en dos especificaciones: la de 2008 (**SPARQL 1.0**) y la de 2013 (**SPARQL 1.1**). Desde la versión 1.1 el lenguaje se puede, no solo realizar consultas sobre los datos, sino también modificar e insertar datos RDF. En este módulo solo trataremos la consulta de datos con SPARQL.

Cómo utilizar SPARQL

En este módulo se muestra cómo utilizar SPARQL como lenguaje de consulta, similar a SQL, para los modelos de datos del ecosistema de RDF.

Las características de SPARQL son, entre otras:

- **Extracción de datos** como grafos RDF, URI, literales (tipados o no tipados), con funciones de agregación, subconsultas, etc.
- **Exploración de datos** mediante consultas para relaciones desconocidas.
- **Transformación de datos RDF** de un vocabulario a otro.
- **Construcción de nuevos grafos RDF** basados en grafos de consultas.
- **Actualizaciones de grafos RDF** como lenguaje de manipulación de datos (*Data Manipulation Language*, DML) completo.
- **Vinculación lógica** para RDF, RDFS y OWL.
- **Consultas federadas** distribuidas en diferentes puntos de acceso (*Endpoint*) SPARQL.

1. Sintaxis básica

Las consultas SPARQL pueden tener diferentes formas. La más frecuente es la consulta *Select* que obtiene información basada en restricciones sobre los datos.

Cada consulta SELECT de SPARQL SELECT se organiza de la siguiente manera:

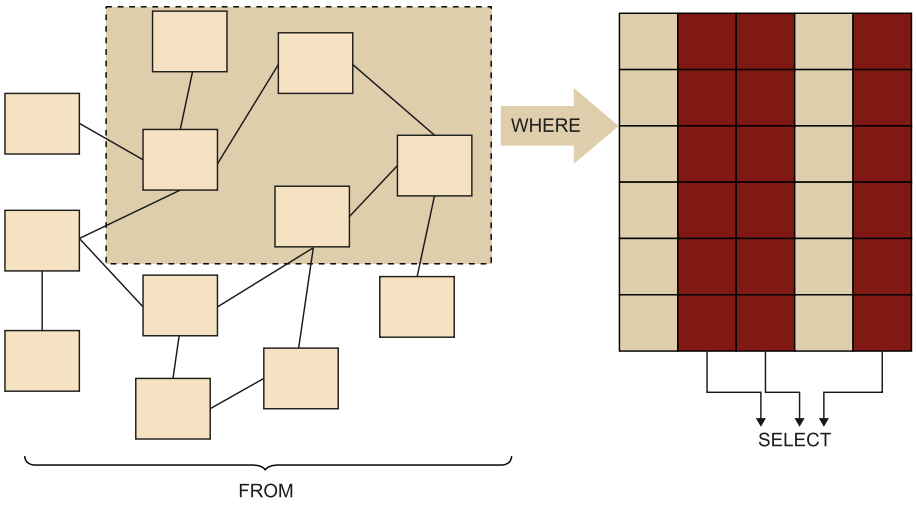
- **PREFIX** (prefijos de los espacios de nombres).
- **SELECT** (define lo que se desea recuperar).
- **FROM** (especifica el conjunto de datos del que se extraen los datos).
- **WHERE** (criterios de la restricción de los datos. Es una descripción en forma de tripletas de consulta).
- **ORDER BY** (ordenación del resultado).
- **LIMIT** (modificación del resultado).

Figura 1. Consulta SPARQL que pregunta cuándo se descubrió Plutón

```
BASE  <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>


SELECT  ?x
WHERE {
  <Pluto> dbo:discovered ?x.
}
```

Figura 2. Consultas SPARQL



1.1. Prólogo

El prólogo permite definir los espacios de nombres, es decir, las abreviaturas de los vocabularios que vamos a utilizar en la consulta. Mediante prefijos se pueden añadir tantos espacios de nombres como sea necesario. Para hacerlo deberemos utilizar la palabra reservada «PREFIX».

Por ejemplo, podríamos definir prefijos para los vocabularios de dbpedia y de foaf de la siguiente forma:

Figura 3. Declaración de prefijos

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
```

Una vez definidos, podemos referirnos a las propiedades de los vocabularios de la siguiente forma:

Figura 4. Uso de vocabularios declarados

```
dbpedia:Solar_System
foaf:name
```

El término «BASE» se puede utilizar solamente una vez en una consulta SPARQL y permite definir el vocabulario por defecto de la consulta. Por tanto se sobreentiende que todos los URI que no utilicen espacio de nombres pertenecerán al vocabulario base.

1.2. Select

Indica la operación a ejecutar (SELECT) y el resultado esperado de la consulta. Es decir, los valores que debe devolver la consulta SPARQL, y es un elemento obligatorio de una consulta.

De la misma manera que en SQL, en SPARQL se puede usar un asterisco (*) para expresar que queremos que la consulta devuelva todos los datos. También es posible indicar una lista de expresiones para determinar los datos deseados. Y al igual que en SQL se puede usar la cláusula «DISTINCT» para indicar que no se deben devolver datos duplicados.

Aunque la consulta se realiza sobre un grafo, la salida de SELECT es una tabla. CONSTRUCT permite obtener los resultados en forma de grafo RDF, en lugar de en forma tabular.

Figura 5. Consulta SPARQL que devuelve las astronautas mujeres en forma de tripletas

```
CONSTRUCT  {?x foaf:gender ?g }
WHERE {
   ?x  rdf:type   dbo:Astronaut;
       foaf:name   ?nombre ;
       foaf:gender ?g ..
   FILTER ((¿g = "female”@en) ).
}
```

Hay que indicar a CONSTRUCT una plantilla de cómo queremos obtener los resultados, y esa plantilla es un conjunto de tripletas con variables.

ASK permite ejecutar una consulta para comprobar si una determinada condición se cumple en el conjunto de datos consultados. ASK devuelve un valor booleano que indica si el patrón de consulta se satisface o no. No devuelve el resultado.

DESCRIBE devuelve un grafo RDF que describe un recurso RDF. El recurso puede expresarse mediante un URI o mediante una variable resultante de un patrón de consulta.

1.3. Cuerpo

El cuerpo de la consulta permitirá determinar qué elementos del grafo deben recuperarse y en qué formato (orden y agrupación) deberán ser entregados. Para ello el cuerpo de la consulta consta de tres partes:

1) Origen de la consulta: permite definir el conjunto de datos que se deben consultar. Hace referencia al grafo RDF (o a un fragmento del mismo) que va a ser el origen de los datos. Para indicar el conjunto de datos que vamos a utilizar se usan las palabras clave «FROM» (grafo RDF por defecto) y «FROM NAMED» o «GRAPH» (grafo RDF con nombre). Esta cláusula es opcional (los puntos de acceso SPARQL suelen tener un conjunto de datos de referencia y la cláusula FROM suele estar vacía).

2) Patrón de consulta: indica las condiciones que deben cumplir los datos del grafo para poder ser recuperados por la consulta. Permitirá seleccionar un conjunto de datos que satisfagan la estructura y los valores indicados en el patrón. El patrón estará contenido dentro de la cláusula WHERE y puede ser tan complejo como sea necesario, permitiendo conjunciones de patrones, disyunciones, partes opcionales variables y restricciones de valores.

3) Modificadores: son operaciones que se ejecutarán sobre los datos seleccionados, ya sea para cambiar su orden (la cláusula ORDER BY permite ordenar los datos de acuerdo a un conjunto de propiedades), su nivel de agregación (la cláusula GROUP BY permite agrupar los datos de acuerdo a una o más propiedades), limitar su número (la cláusula LIMIT n permite limitar los resulta-

dos devueltos a los *n* primeros) y saltarse algunos (la cláusula OFSET i permite empezar a mostrar los datos a partir del elemento *i*-ésimo devuelto por la consulta).

2. Patrones de consulta

Los patrones de consulta son los elementos clave para entender el funcionamiento de las consultas SPARQL.

> Un **patrón de consulta** es una condición que deben satisfacen los datos del grafo RDF para poder ser seleccionados por la consulta. El patrón más frecuente es el de tripletas.

Un patrón es una tripleta RDF (sujeto-predicado-objeto) en la que uno o más de sus componentes son una variable. Las variables se representan mediante un símbolo de interrogación (?) y el nombre de la variable.

Supongamos que estamos consultando DBpedia con SPARQL utilizando el siguiente patrón de tripleta en el cuerpo de una consulta SPARQL:

Figura 6. Consulta SPARQL a la DBpedia

```
SELECT  ?satelites ?nombre
WHERE {
   ?satelites  prop:satelliteOf dbpedia:Pluto .
   ?satelites rdfs:label ?nombre.
   FILTER (lang(?nombre) = "es")
}
```

La consulta devolvería todas aquellas tripletas RDF de la DBpedia que cumplan con el patrón. Una tripleta cumplirá (*matching*) con un patrón si existen valores de la tripleta que aparezcan en los datos consultados. En el caso de ejemplo, el patrón prefija el objeto (recurso Plutón) y el predicado (satelliteOf). Por tanto todas las tripletas con objeto Plutón y predicado satelliteOf satisfarían el patrón planteado.

El resultado obtenido será el siguiente:

Satélites	Nombre
http://dbpedia.org/resource/Hydra_(moon)	"Hidra (satélite)"@es
http://dbpedia.org/resource/Charon_(moon)	"Caronte (satélite)"@es
http://dbpedia.org/resource/Nix_(moon)	"Nix (satélite)"@es

Aparecerán tantos resultados de nombre como tripletas en el grafo satisfagan la condición: están los nombres de los tres satélites en todos los idiomas distintos. Por tanto se ha añadido un filtro para que solo muestre los que están en español.

Como se ha visto, en una consulta SPARQL pueden definirse patrones simples (de una sola tripleta) o conjuntos de patrones (de más de una tripleta). Un conjunto de patrones se compone de distintos patrones simples concatenados mediante un punto. En nuestro caso, tenemos dos patrones:

1) El primer patrón identificará aquellos recursos que cumplen que son satélites de Plutón.

2) El segundo patrón identificará los nombres definidos en la DBpedia para los nombres de los satélites en los diferentes idiomas (filtrando solo los que estén en español).

Podríamos complicar el patrón añadiendo más patrones de tripletas. Las tripletas separadas con el punto (.) se comportan como el operador `AND (Y)`, es decir, se tienen que cumplir todas (forma conjuntiva).

2.1. Usando literales

Hasta ahora hemos visto cómo realizar consultas SPARQL filtrando los datos en función del esquema que siguen (en qué propiedades participa cada recurso) y de sus URI (con qué recursos está relacionado). Resulta interesante también poder filtrar datos en función de literales. Por ejemplo, identificar los recursos que contengan cierta cadena o que un valor numérico supere una cantidad. Para poder realizar estos filtros es necesario conocer la representación de los distintos tipos de datos en SPARQL.

La sintaxis general para literales es una cadena de caracteres (entre comillas dobles " o simples ') y, opcionalmente, el nombre de su tipo precedido de ^^. Por ejemplo:

- `"1"^^xsd:integer` para indicar que 1 es un entero.
- `"1"^^xsd:string` para indicar que es una cadena de caracteres con el 1.
- `"3.1415"^^xsd:decimal` para indicar el número 3.1415.
- `"1.0e3"^^xsd:double` para indicar que es un número real.
- `"true"^^xsd:boolean` para indicar el valor *true* de tipo booleano.
- `"1930-02-18"^^xsd:date` para indicar que es una fecha.

Para añadir patrones que utilicen comparaciones con posibles valores literales de las tripletas podemos utilizar la cláusula FILTER. Esta función nos permite filtrar solo aquellas tripletas que satisfagan una determinada condición. La condición se indica mediante un conjunto de expresiones booleanas, de la misma forma que en SQL.

La cláusula FILTER funciona como un patrón de tripleta más. Por lo tanto se puede utilizar la cláusula FILTER repetidas veces en una misma consulta. FILTER no puede asignar ni crear nuevos valores.

Las expresiones de la cláusula FILTER pueden utilizar distintos operadores. Los más comunes son:

- Los operadores booleanos (! para representar `NOT` («no» lógico), && para representar un `AND` («y» lógico) y || para representar un `OR` («o» lógico).
- Los operadores de comparación (=, !=, >=, <, <=, > y >=).
- Los operadores matemáticos (+, -, *, /, %).

Las cadenas de caracteres en RDF pueden tener asociado un idioma. Eso permite definir una misma propiedad en distintos idiomas; por ejemplo, el nombre del planeta enano Plutón es **Pluto** en inglés o *Plutó* en catalán. Para indicar esto, en RDF se añade un sufijo a la cadena de caracteres que contiene una @ y el código del idioma. Así, si consultamos el nombre de Plutón en la DBpedia, tendríamos «"Plutón"@es», «"Pluto"@en» o «冥王星@ja» entre otros valores. Los tres valores representan el nombre del planeta enano en castellano, inglés y japonés respectivamente.

Para filtrar por cadenas de caracteres se puede utilizar la cláusula FILTER en combinación con la función regex. La función regex permite definir la expresión regular que los valores de la variable deberán satisfacer (de una manera muy parecida al LIKE en SQL).

Función regex

Más información sobre el funcionamiento de regex en www.w3.org/TR/xpath-functions/#regex-syntax.

La función regex permite utilizar caracteres comodín, como por ejemplo el ^ para indicar el inicio de una cadena de caracteres, el $ para indicar el final y el * para indicar una cadena de cero, uno o más caracteres.

Además de la función regex existen otras funciones útiles para filtrar los valores:

Filtrar valores

La lista completa de las funciones útiles para filtrar valores se puede encontrar en: www.w3.org/TR/sparql11-query/#SparqlOps.

- `datatype`: que devuelve el tipo de datos de un elemento,
- `str`: que convierte a texto un literal,
- `isUri`, que indica si un recurso es un URI,
- `lang`, que indica el idioma asociado a una cadena de texto,
- `bound`, que indica si el literal tiene asignado un valor.

2.2. Patrones opcionales

Las consultas vistas hasta ahora exigen que se satisfagan todos los patrones para obtener los resultados. En algunos casos esto puede ser demasiado restrictivo. Para resolver este problema están los **patrones opcionales** que se definen mediante una cláusula OPTIONAL.

Tanto las tripletas que satisfagan el patrón opcional como las que no lo hagan se seleccionarán en la consulta. Por tanto, los resultados de la búsqueda serán aquellos en los que se cumple el patrón opcional, pero también los datos en los que no se cumpla. Las variables ligadas al patrón opcional no tendrán valor para las tripletas en las que el patrón no se ha cumplido.

Figura 7. Consulta SPARQL sobre los miembros de la misión Apollo 11

```
SELECT  ?nombre ?fn ?fd
WHERE {
   dbp:Apollo_11 prop:crewMembers ?x .

  ?x rdf:type dbo:Astronaut ;
     foaf:name ?nombre ;
     dbo:birthDate ?fn .

   OPTIONAL { ?x dbo:deathDate ?fd }
}
```

A continuación podemos ver el resultado de la anterior consulta, que ahora devuelve los tres miembros de la tripulación. Para algunos la fecha de defunción está vacía, ya que, en su caso, el patrón de tripleta con el que se calculaba no se satisfizo.

Figura 8. Resultado de la consulta

Nombre	Fecha Nacimiento	Fecha Defunción
"Neil Armstrong"@en	1930-08-05	2012-08-25
"Buzz Aldrin"@en	1930-01-20	
"Michael Collins"@en	1930-10-31	

2.3. Patrones alternativos

Hasta ahora hemos visto cómo utilizar combinaciones de patrones de tripletas de forma que todos los patrones se cumplan a la vez. En algunas ocasiones será necesario utilizar patrones de forma que se garantice que se satisfaga un patrón u otro, como en una `OR` lógica.

Este tipo de patrones se denominan *patrones alternativos*. Los **patrones alternativos** se expresan enmarcando los dos patrones disjuntos entre los símbolos { } y uniéndolos mediante la cláusula UNION.

Figura 9. Consulta SPARQL con la unión de las tripulaciones de las misiones Apolo 11 y Apolo 13

```
SELECT  ?nombre ?fn ?fd
WHERE {
  {
      dbp:Apollo_11 prop:crewMembers ?x .
      ?x rdf:type dbo:Astronaut ;
         foaf:name ?nombre ;
         dbo:birthDate ?fn .
      OPTIONAL { ?x dbo:deathDate ?fd }
   }
  UNION
   {
      dbp:Apollo_13 prop:crewList ?x .
      ?x rdf:type dbo:Astronaut ;
         foaf:name ?nombre ;
         dbo:birthDate ?fn .
      OPTIONAL { ?x dbo:deathDate ?fd }
   }
}
```

Para integrar en una consulta ambos conjuntos de patrones de forma alternativa, se engloba cada patrón entre las marcas { } y se intercala la cláusula UNION en medio. El resultado de la consulta anterior SPARQL es:

Figura 10. Resultado de la consulta

Nombre	Fecha Nacimiento	Fecha Defunción
"Neil Armstrong"@en	1930-08-05	2012-08-25
"Buzz Aldrin"@en	1930-01-20	
"Michael Collins"@en	1930-10-31	
"Fred Haise"@en	1933-11-14	
"James Lovell"@en	1928-03-25	
"Jack Swigert"@en	1931-08-30	1982-12-27

3. Puntos de acceso SPARQL

Para poder ejecutar consultas SPARQL necesitamos un punto de acceso.[2] Estos puntos de acceso serían el equivalente a la consola de un sistema gestor de bases de datos. Existen distintos puntos de acceso disponibles desde Internet que nos permiten consultar conjuntos de datos RDF localizados.

(2) *Endpoint*, en inglés.

Desde la versión SPARQL 1.1 los puntos de acceso permiten obtener los resultados de una consulta en distintos formatos (XML, JSON, CSV, etc.), o crear un nuevo grafo RDF como resultado de una consulta. En SPARQL también se pueden consultar datos RDF de más de un conjunto de datos mediante lo que se denominan **consultas federadas** (*Federated Query*).

Algunos de los puntos de acceso más populares son los siguientes:

Puntos de acceso

La lista de los puntos de acceso más relevantes se puede encontrar en www.w3.org/wiki/SparqlEndpoints.

1) Para consultar datos de carácter general:

- DBpedia permite acceder a sus datos.
- Wikidata permite acceder a los datos de www.wikidata.org.

2) Para consultar datos geográficos:

- Geonames permite acceder a los datos geográficos disponibles en Geonames.
- LinkedGeoData permite acceder a los datos geograficos de la web de OpenStreetMaps.

Bibliografía

DuCharme, R. (2013). *Learning SPARQL* (2.ª ed.). O'Reilly Media.

Guarino, N.; Oberle, D.; Staab, S. (2009). *What Is an Ontology?* [en línea]. [Fecha consulta: enero 2020]. Disponible en: <https://iaoa.org/isc2012/docs/Guarino2009_What_is_an_Ontology.pdf>

Kumar, A. (2018). *Architecting Data-Intensive Applications*. Packt Pub.

Noy, N. F.; McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report (Informe SMI-2001-0880).

Powers, S. (2003). *Practical RDF*. O'Reilly Media.