

Lenguajes de Recuperación

La Web representa una gran masa de contenidos heterogéneos. A pesar de su interés y utilidad, estos contenidos no podrán tener ningún uso si el usuario que los necesite no los puede encontrar. Por esto, es importante en la **recuperación y organización de información** el uso de lenguajes de recuperación.

Los lenguajes de recuperación (también query languages) son lenguajes informáticos utilizados para recuperar información de sus almacenes. Puede definirse un lenguaje de recuperación como un conjunto de órdenes, operadores y estructuras que, organizados según unas normas lógicas, permiten la consulta de fuentes y recursos de información electrónica.

Los lenguajes de recuperación pueden dividirse en dos grandes grupos, distinguiendo si su uso se enfoca a las bases de datos relacionales o a la recuperación de información. Los ejemplos son muy variados, incluyendo a **SeRQL** y **SPARQL**. Algunos otros son:

- [SQL](#), lenguaje de recuperación reconocido como estándar y ampliamente utilizado para bases de datos relacionales.
- [XQuery](#), lenguaje de recuperación para XML.
- [CODASYL](#)
- [OQL](#) (Object Query Language).

Expresiones

El resultado de la combinación de estos elementos, siguiendo las normas establecidas, es una expresión, a la que se identifica con el nombre "ecuación" o query, capaz de interrogar el contenido de la fuente de información.

Un lenguaje de recuperación especifica las normas que rigen para la formulación de estas ecuaciones, esto es, su sintaxis. Aportan la semántica necesaria para la coordinación de los diferentes elementos que componen una consulta, indicando el orden necesario entre ellos, sus posibilidades combinatorias, prioridades en la ejecución y demás informaciones.

Una ecuación se compondrá básicamente de términos y operadores. Los términos serán aquellas palabras que informarán al sistema sobre las acciones a ejecutar (como puedan ser órdenes de mostrar los documentos resultantes, ejecutar un perfil de usuario...). No todos los lenguajes utilizan el mismo vocabulario para expresar las mismas acciones. Algunos intentos para su homogeneización han surgido, como pueda ser el lenguaje CCL promovido por la Unión Europea.

Los operadores indican las relaciones que guardan entre sí los términos de una ecuación. Hay de diversos tipos, dependiendo de su semántica: booleanos, aritméticos, de comparación...

El lenguaje RDF

El lenguaje RDF (Marco de Descripción de Recursos, del inglés Resource Description Framework) es un lenguaje de propósito general para la representación de información en la web, mediante la definición de ontologías y metadatos. El lenguaje RDF se basa en XML.

El W3C ha promovido la especificación RDF/XML para su desarrollo. Desde su publicación en 1999, RDF es hoy el estándar más popular y extendido en la comunidad de la web semántica.

Estructura

Este modelo se basa en la idea de convertir las declaraciones de los recursos en expresiones con la forma sujeto-predicado-objeto. El elemento de construcción básica en RDF es el "tripleto" o sentencia, que consiste en dos nodos (sujeto y objeto) unidos por un arco (predicado), donde los nodos representan recursos, y los arcos propiedades. El objeto es el valor de la propiedad o el otro recurso con el que se establece la relación.

Por ejemplo una sentencia podría expresar el hecho de que el autor (predicado) del libro "En la cripta" (sujeto) fue el escritor H.P. Lovecraft (objeto). Encadenando estos tripletes se construyen grafos o redes semánticas para la web.

La combinación de RDF con otras herramientas como RDF Schema y OWL permite añadir significado a las páginas. Con RDF Schema (RDFS) se pueden definir jerarquías de clases de recursos, especificando las propiedades y relaciones que se admiten entre ellas.

Un ejemplo

Utilizaremos RDF para describir un CD de música:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
<cd:artist>Bob Dylan</cd:artist>
<cd:country>USA</cd:country>
<cd:company>Columbia</cd:company>
<cd:price>10.90</cd:price>
<cd:year>1985</cd:year>
</rdf:Description>
</rdf:RDF>
```

El documento está basado en XML, con su declaración en la primera línea. Tras ella, la raíz del documento RDF <rdf:RDF>. Se especifican ahí las URLs de los namespaces xmlns:rdf y xmlns:cd.

El elemento <rdf:Description> contiene la descripción del recurso identificado mediante el atributo rdf:about. Los elementos <cd:artist>, <cd:country>, <cd:company>, etc. son las propiedades del recurso.

Lenguajes de consulta

De modo similar al lenguaje SQL en las bases de datos, investigación en los últimos años trata de desarrollar lenguajes de consulta para RDF, que permita ejecutar búsquedas complejas sobre un grafo RDF mediante una sintaxis sencilla.

A falta de acuerdo sobre un lenguaje estándar comúnmente aceptado, diferentes iniciativas particulares han surgido y prosperado, como pueden ser:

- RDQL, por Hewlett Packard, posiblemente el más extendido.
- RQL, por el instituto ICS-FORTH de Grecia.
- **SeRQL**, por la empresa holandesa Administrator.
- **SPARQL**.

SeRQL

SeRQL (Sesame RDF Query Language, pronunciado como "circle") es un lenguaje de recuperación para RDF/RDFS desarrollado por Aduna como parte del software Sesame. Combina características de otros lenguajes (principalmente RQL, RDQL, N-Triples y N3) y añade otras propias.

Se asocia al programa Sesame. Sesame es un framework para RDF de código libre, con soporte para inferencias en RDF Schema y recuperación mediante queries. Originalmente desarrollado por [Aduna](#), ahora es mantenido en cooperación con NLnet Foundation, y un grupo de desarrolladores voluntarios.

El diseño de **SeRQL** tuvo el objetivo de unificar algunas de las ventajas que ofrecían otros lenguajes, como parte de un lenguaje de recuperación ligero pero potente. Algunas de las características más importantes de SeRQL para la **recuperación y organización de la información** son:

- Transformación de grafos.
- Soporte de RDF Schema.
- Soporte de los tipos de datos de XML Schema.
- Emparejado de caminos opcionales.

Sintaxis

La sintaxis de SeRQL es similar a la de RQL, añadiendo algunas modificaciones para facilitar el análisis sintáctico (parsing) del lenguaje. Como RQL, SeRQL se basa en una interpretación formal del grafo de RDF, diferenciándose de RQL en que se basa directamente en [RDF Model Theory](#).

SeRQL soporta expresiones generales de camino, restricciones booleanas y emparejado de caminos opcionales, como también dos iteradores básicos:

- **select-from-where**: Devuelve tablas con variables y valores para los cuales se ha conseguido un matching con los datos.
- **construct-from-where**: Devuelve un subgrafo de los datos con los que se ha conseguido un matching. Permiten el uso de queries compuestas.

Otros recursos

- [Manual de SeRQL](#)
- [Ejemplos de queries](#)
- [Artículo "SeRQL: A Second Generation RDF Query Language"](#)
- [Artículo "SeRQL: An RDF Query and Transformation Language"](#)

Ejemplos de SeRQL

Para ejecutar las queries, necesitamos un conjunto de datos sobre el que recuperar la información. Para ello, tomemos por ejemplos el documento en RDF, que describe a una persona trabajando para una compañía:

```
<?xml version=?1.0??>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:foo="http://www.mycompany.smthg/company#">
<rdf:Description about="http://www.mycompany.smthg/company/Person">
<foo:worksFor> rdf:resource="http://www.mycompany.smthg/company/Company"
</foo:worksFor>
</rdf:Description>
<rdf:Description about=?http://www.mycompany.smthg/company/Company?>
<rdf:type>rdf:resource="http://www.mycompany.smthg/company/
CompanySchema#ITCompany"</rdf:type>
</rdf:Description>
</rdf:RDF>
```

Supongamos que queremos recuperar de este grafo de RDF personas que trabajan para compañías de IT. Podemos entender este proceso como encontrar el siguiente patrón en los datos:

```
{Person} ex:worksFor {Company} rdf:type {ex:ITCompany}
```

En esta línea, las partes rodeadas por llaves representan los nodos en el grafo de RDF, y los textos entre ellas representan los enlaces dentro del grafo. La dirección de los arcos (propiedades) en SeRQL es siempre de izquierda a derecha.

Expresiones de camino múltiple pueden ser especificadas separándolas con comas. Por ejemplo, la expresión anterior puede expresarse con dos expresiones menores:

```
{Person} ex:worksFor {Company}, {Company} rdf:type {ex:ITCompany}
```

Los nodos y enlaces en las expresiones de camino pueden ser variables, URIs o literales. También, un nodo puede dejarse vacío en caso de que su valor no interese al usuario. Algunos ejemplos serían:

```
{Person} ex:worksFor {} rdf:type {ex:ITCompany}
```

O en otros contextos:

```
{Painting} ex:painted_by {} ex:name {"Picasso"}
{comic:RoadRunner} SomeRelation {foo:WillyECoyote}
```

Queries

El lenguaje de recuperación SeRQL soporta dos modos de query. El primero, `?select queries?`, se caracteriza por devolver una tabla con un conjunto de uniones variable-

valor. El segundo, `?construct queries?`, devuelve un grafo de RDF, que puede ser un subgrafo del grafo completo de datos, u otro grafo con información derivada.

Una consulta SeRQL usualmente se compone de una a seis palabras clave. Estas son:

- **SELECT** o **CONSTRUCT**: determina qué se construye con los resultados encontrados. En **SELECT**, puede indicarse las variables para las que se quiere obtener en los resultados. En **CONSTRUCT**, pueden especificarse los tripletes a ser devueltos.
- **FROM**: Opcional. Contiene expresiones de camino. Define patrones de tripletes en un grafo de RDF que son relevantes para la consulta.
- **WHERE**: Opcional. Contiene restricciones booleanas para los valores obtenidos en los resultados, permitiendo establecer filtros en estos.
- **LIMIT** y **OFFSET**: Opcionales. Pueden usarse de forma separada o conjunta para conseguir un subconjunto de los resultados. **LIMIT** impone un número máximo de respuestas, y **OFFSET** determina qué resultado se devolverá en primera posición.
- **USING NAMESPACE**: Opcional. Puede contener declaraciones de namespace, y asociaciones de estos a prefijos para un uso más cómodo.

Como ejemplo, pondremos la siguiente consulta **SELECT**:

```
SELECT DISTINCT *  
FROM {Country1} ex:borders {} ex:borders {Country2}  
USING NAMESPACE  
ex = <http://example.org/things#>
```

En la cual se indica que se obtengan todos los países con frontera con Country1 y Country2, sin repetir elementos (**DISTINCT**).

Un ejemplo de consulta **CONSTRUCT**:

```
CONSTRUCT {Parent} ex:hasChild {Child}  
FROM {Child} ex:hasParent {Parent}  
USING NAMESPACE  
ex = <http://example.org/things#>
```

Esta consulta define la inversa de la propiedad `foo:hasParent` como `foo:hasChild`. De este modo, se consigue un grafo con información no contenida en los datos originales, pero derivada de ellos.

SPARQL

SPARQL (pronunciado "sparkle") es un lenguaje de recuperación basado en RDF; su nombre es un acrónimo recursivo del inglés *SPARQL Protocol and RDF Query Language*. Se trata de una recomendación para crear un lenguaje de consulta dentro de la Web semántica que está ya implementada en muchos lenguajes y bases de datos. Desde 2005 está en proceso de estandarización por el RDF Data Access Working Group (DAWG) del W3C; en abril del 2006 se anunció el paso de su especificación a Candidate Recommendation, aunque volvió al estado de [Working Draft](#) en octubre de 2006.

Con **SPARQL** los desarrolladores y usuarios finales pueden representar y utilizar los resultados obtenidos en las búsquedas a través de una gran variedad de información como son datos personales, redes sociales y metadatos sobre recursos digitales como música e imágenes.

Especificación

SPARQL consiste en tres especificaciones separadas, que contienen diferentes partes de su funcionalidad. En total, consiste en un lenguaje de query, un formato para las respuestas, y un medio para el transporte de consultas y respuestas:

- [SPARQL Query Language](#): Núcleo de SPARQL. Explica la sintaxis para la composición de sentencias y su concordancia.
- [SPARQL Protocol](#): Formato utilizado para la devolución de los resultados de las búsquedas (queries SELECT o ASK), a partir de un esquema de XML.
- [SPARQL Query XML Results Format](#): Describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores. Utiliza WSDL para definir protocolos remotos para la consulta de bases de datos basadas en RDF.

Otros recursos

- [Guía rápida de SPARQL](#)

Ejemplos de uso de SPARQL

Para dar unos ejemplos del uso de **SPARQL** para las búsquedas sobre RDF, tomaremos un conjunto de datos inicial. Los datos elegidos son la descripción en RDF de la tabla periódica. Un pequeño fragmento de esta descripción puede verse ahora, con las propiedades asociadas al elemento "Sodio":

```
<Element rdf:ID="Na"
xmlns="http://www.daml.org/2003/01/periodictable/PeriodicTable#">
<name>sodium</name>
<symbol>Na</symbol>
<atomicNumber>11</atomicNumber>
<atomicWeight>22.989770</atomicWeight>
<group rdf:resource="#group_1"/>
<period rdf:resource="#period_3"/>
<block rdf:resource="#s-block"/>
<standardState rdf:resource="#solid"/>
<color>silvery white</color>
<classification rdf:resource="#Metallic"/>
<casRegistryID>7440-23-5</casRegistryID>
</Element>
```

Tripletes en SPARQL

Así como RDF se compone de tripletes, también lo hace **SPARQL**, consistentes en sujeto-predicado-objeto, y terminadas con un punto. Los tripletes en este lenguaje representarán patrones mediante los cuales se buscarán tripletes coincidentes en los datos. Un triplete para el ejemplo anterior expresado en **SPARQL** sería:

```
<http://www.daml.org/2003/01/periodictable/PeriodicTable#Na> table:name "sodium".
```

Las URIs se escriben dentro de símbolos < y >, y las cadenas entre comillas simples o dobles. Es importante la especificación de los namespaces, sea mediante URIs o con la notación namespace:elemento. **SPARQL** especifica una serie de abreviaciones para la fácil escritura de complejos tripletes.

En el anterior patrón, todos los datos sobre el triplete buscado se dan, lo cual no es verdaderamente útil pues sólo nos permitiría ver si ese triplete existe. Para buscar más de un triplete en los datos es necesario el uso de variables.

Tanto el sujeto, predicado y objeto de un triplete pueden ser sustituidos por variables. El siguiente ejemplo muestra variables en el sujeto y el objeto.

```
?element table:name ?name.
```

Una variable puede asociarse a cualquier valor concreto. De este modo, el patrón anterior devolverá todo recurso RDF con una propiedad name. En **SPARQL** todas las posibles asociaciones se consideran, con lo que si un recurso tiene varias instancias de una propiedad, varios resultados serán devueltos para ese recurso.

Queries en SPARQL

Mostraremos un ejemplo ahora de query completa:

```
PREFIX table: <http://www.daml.org/2003/01/periodictable/PeriodicTable#>
```

```
SELECT ?name
```

```
FROM <http://www.daml.org/2003/01/periodictable/PeriodicTable.owl>
```

```
WHERE { ?element table:name ?name. }
```

En la primera línea encontramos la palabra clave PREFIX. Su función es equivalente a la declaración de namespaces en XML: asocia una URI a una etiqueta, que se usará en adelante para describir el namespace. Pueden incluirse varias de estas etiquetas en una misma query.

El comienzo de la query queda marcado por la palabra clave SELECT. Semejante a su uso en SQL, sirve para definir los datos que deben ser devueltos en la respuesta. En el ejemplo tan sólo se devolverá el nombre del elemento.

La palabra clave FROM identifica los datos sobre los que se ejecutará la consulta. En este ejemplo, se limitará al RDF de la tabla periódica. Una consulta puede incluir varios FROM.

La palabra clave WHERE indica el patrón sobre el que se filtrarán los tripletes del RDF. El patrón del ejemplo es el comentado en la sección anterior.

El resultado que se obtendrá es una tabla como la siguiente:

row	name
1	sodium
2	neon
3	iron

SPARQL aporta otra serie de palabras clave que pueden ser utilizadas para refinar nuestras consultas. Consultas UNION pueden permitirnos combinar consultas alternativas. También pueden ordenarse los resultados, limitarse mediante LIMIT, o conseguir desplazamientos mediante OFFSET.