Ontologías y Web Semántica

RDF

Pablo R. Fillottrani

Depto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

marzo 2019

Introducción a RDF

Breve historia

- RDF (Resource Description Framework) es un lenguaje diseñado para describir recursos, o metadatos sobre la información en la Web
- mediante RDF se pueden describir estructuras comunes para ser usadas en interoperabilidad en el intercambio de datos
- RDF es una recomendación W3C, parte definida en 1999 y parte en 2000, y actualizada en febrero 2004

Documentos

- la especificación consta de los siguientes documentos:
  - RDF Primer objetivos y usos de RDF
  - RDF Concepts and Abstract Syntax introduce conceptos básico y sintaxis intermedia
  - RDF/XML Syntax define la sintaxis de RDF en XML
  - RDF Semantics define el significado formal
  - RDF Vocabulary Description Language cómo usar RDF para definir vocabularios para contenido en la Web
  - RDF Test Cases describe casos de prueba positivos y negativos

XML vs RDF

- no existe ningún significado definido para XML: los marcados y sus anidamientos son arbitrarios
- ejemplo:

```
<footballTeam name="Boca Juniors">
  <player>Carlos Tévez</player>
</footballTeam>
<footballPlayer>
 <name>Carlos Tévez</name>
 <team>Boca Juniors</team>
</footballPlayer>
<footballPlayer name="Carlos Tévez">
 <team>Boca Juniors</team>
</footballPlayer>
```

- RDF provee un modelo de datos para representar el significado de datos
- RDF es independiente del dominio, o sea los datos no necesariamente deben estar especificados en XML
- aumenta la posibilidad de interoperabilidad de datos
- para aplicaciones simples, donde no es necesario la persistencia ni la interoperabilidad de datos, es preferible usar XML sin RDF
- se puede hacer una analogía entre la relación XML con RDF, y la relación que existe en bases de datos jerárquicas y relacionales

Modelo

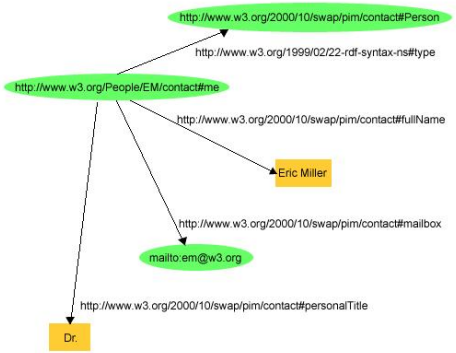
- el modelo RDF es un conjunto de sentencias que definen propiedades entre objetos
- cada propiedad es necesariamente una relación binaria
- se diferencia el dato origen de la propiedad, llamado sujeto, y el dato destino, llamado objeto (o sea las propiedades no necesariamente son simétricas)
- los datos objetos pueden ser recursos, identificados con URIs, o literales
- los datos sujetos sólo pueden ser recursos

- las propiedades también pueden ser identificadas por URIs
- si vemos los recursos y literales como nodos, las propiedades como arcos dirigidos, entonces el modelo de datos RDF es un grafo dirigido etiquetado
- las sentencias se representan con triplas, que representan cada arco del grafo

```
sujeto   predicato   objeto
```

- el grafo, o el conjunto de triplas, son representaciones equivalentes del modelo. El grafo tiene la ventaja de que un nodo interviniente en varios arcos no se repite.

- ejemplo



Recursos

- los recursos son siempre identificados por URIrefs absolutos
- son nodos del grafo que representan objetos y sujetos de las propiedades
- puede ser reemplazado por blank nodes (nodos en blanco)

Literales

- los literales sólo pueden ser objetos de las triplas
- pueden ser caracteres entre comillas

  `"doctor"`
- opcionalmente se les puede agregar un indicador de lenguaje

  `"dotor"\@es-AR`
- y también pueden estar etiquetados con el tipo de dato simple bajo el cual debe interpretarse la cadena de caracteres

  `"1"^^xs:integer`

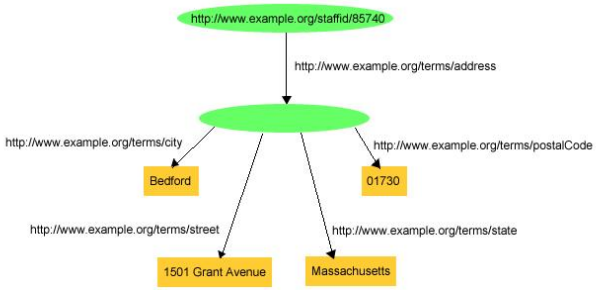
- el chequeo de tipos no forma parte de la semántica RDF
- los tipos de datos son los definidos por XML Schema
- esto provee un mecanismo extensible
- cada tipo de datos tiene un espacio léxico, un espacio de valores y un mapeo del espacio léxico a los valores

Propiedades

- también son identificadas por URIrefs absolutos
- corresponden a las etiquetas de los arcos del grafo
- son relaciones binarias entre recursos y recursos, o recursos y literales

Nodos en blanco

- sirven para representar recursos sin nombre, ie sin URI que los identifique
- la representación depende de la sintaxis elegida
- ejemplo



- la principal característica de este modelo es que la triplas pueden combinarse libremente, reteniendo su significado (por eso la necesidad de URIrefs absolutos)
- un subgrafo del grafo modelo está formado por un subconjunto de sus triplas; una mezcla de modelos es la unión de los respectivos conjuntos de triplas
- nodos en blancos en grafos distintos se interpretan distintos; URIrefs y literales se interpretan iguales si corresponden a la misma secuencia de caracteres

- una instancia de un grafo es un grafo que se obtiene al reemplazar un nodo en blanco por un recurso
- un grafo está instanciado totalmente (grounded) si no contiene nodos en blanco
- el vocabulario de un grafo es la colección de URIrefs usados en ese grafo

- se puede definir una relación de consecuencia entre grafos RDF: un grafo RDF es consecuencia de otro si todas las triplas que son válidas en el primero también son válidas en el segundo
- esto se aplica no sólo a las triplas explícitas, sino también a las implícitas
- se puede probar un lema de subgrafo: todo grafo tiene como consecuencia a sus subgrafos
- también un lema de instancias: todo grafo tiene como consecuencia a sus instancias
- finalmente, el lema de monotonía: si un grafo es consecuencia de un subgrafo de un grafo, el primer grafo es consecuencia también del último

Sintaxis para RDF

1. N3
2. N-triples
3. notación gráfica
4. RDF/XML

N3

- N3 es una notación independiente de RDF
- representa una tripla como

```
sujeto    predicato    objeto .
```

- URIrefs se escriben entre ángulos
- es posible usar espacios de nombre para simplificar las URIrefs, en este caso se omiten los ángulos
- no existe sintaxis específica para los nodos en blanco

- ejemplos:

```
<urn:isbn:0-345-67890-1>
   <http://purl.org/dc/elements/1.1/creator>
      autor .
autor   uns-dcic-prf:#nombre     "Pablo".
autor    uns-dcic-prf:#apellido  "Fillottrani".
```

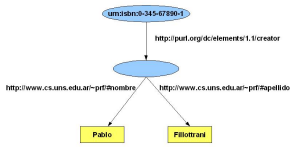
N-triples

- N-triples es un subconjunto de la notación N3
- es necesario separar los elemento con sólo un espacio o un caracter TAB
- hay sólo una tripla por línea
- las líneas que comienzan con el caracter # son comentarios
- los nodos en blanco comienzan con _ : seguidos del nombre

- ejemplo de un archivo N-triple:

```
# Mi libro en RDF
<urn:isbn:0-345-67890-1> dc:creator _:autor .
_:autor uns-dcic-prf:nombre "Pablo" .
_:autor uns-dcic-prf:apellido "Fillottrani" .
```

- ejemplo



RDF/XML

- un documento RDF/XML consiste de una especificación del elemento `rdf:RDF`
- puede ser parte de un documento XML más grande, o ser el elemento raíz del documento
- este tipo de elementos contiene una serie de elementos `rdf:Description`
- cada `rdf:Description` es la descripción de un recurso. Si tiene un atributo `rdf:about` entonces ése es el URI del recurso; si no lo tiene se trata de un nodo en blanco.

- el contenido de un elemento `rdf:Description` está formado por una serie de sentencias de propiedades
- cada sentencia de propiedad engloba un par predicado/objeto: el predicado corresponde con el nombre del elemento y el objeto con su contenido
- dado que los predicados se identifican con URIrefs, es necesario usar espacios de nombres para crear nombres válidos en XML
- si el contenido es texto, el objeto representado es un literal
- si el contenido es vacío entonces el objeto es otro recurso, identificado mediante el valor del atributo `rdf:resource`

- los elementos `rdf:Description` pueden aparecer anidados en el elemento predicado, permitiendo referencia triplas que salen del nodo objeto
- la secuencia de anidación de elementos: recursos-predicados-recursos-predicados-... se denomina striping (rayado)

Ejemplo

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:uns-dcic-prf="http://www.cs.uns.edu.ar/~prf/#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="urn:isbn:0-345-67890-1">
       <dc:creator>
           <uns-dcic-prf:nombre>Pablo</uns-dcic-prf:nombre>
           <uns-dcic-prf:apellido>Fillottrani</uns-dcic-prf:apellido>
       </dc:creator>
    </rdf:Description>
</rdf:RDF>
```

Algunos atributos especiales

- `rdf:parseType`, que pude tener valor `Literal` o `Resource`, permite incluir documentos XML como literales en RDF y definir recursos sin necesidad de `rdf:Description`
- `rdf:nodeID` permite definir nombres para nodos en blanco, haciendo que la sintaxis RDF/XML sea equivalente a la notación gráfica
- `rdf:ID` genera un nuevo URIref con el URI del documento base, y el valor del atributo como fragmento

Ejemplo

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:uns-dcic-prf="http://www.cs.uns.edu.ar/~prf/#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="urn:isbn:0-345-67890-1">
       <dc:creator rdf:nodeId="autor"/>
    </rdf:Description>
    <rdf:Description rdf:nodeId="autor">
       <uns-dcic-prf:nombre>Pablo</uns-dcic-prf:nombre>
       <uns-dcic-prf:apellido>Fillottrani</uns-dcic-prf:apellido>
    </rdf:Description>
</rdf:RDF>
```

- otro refinamiento de N3
- proveen un mecanismo para manejo de espacios de nombres
- permite abreviaturas para triplas con el mismo sujeto
- introduce también abreviaturas para colecciones

- abreviaturas para espacios de nombres:

  `@prefix nombre: <URI> .`
- abreviatura para el espacio de nombre por defecto:

  `@prefix : <URI> .`
- permite el uso de QNames (qualified names) en URIrefs

  `dc:title  rdf:type  :datatype`

- se puede agrupar triplas con el mismo sujeto

```
sujeto predicado1 objeto1 ;
       predicado2 objeto2 .
```

- los nodos en blanco como objeto se notan con `[]`

```
sujeto  predicado [] .
```

- los nodos en blanco como sujeto se notan incluyendo entre los corchetes el predicado y el objeto

```
[ predicato objeto ] .
```

Ejemplo

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix uns-dcic-prf: <http://www.cs.uns.edu.ar/~prf/#> .
@prefix dc: <http://purl.org/dc/elements/1.1/>

urn:isbn:0-345-67890-1
  dc:creator [ uns-dcic-prf:nombre
                        "Pablo" ;
               uns-dcic-prf:apellido
                        "Fillottrani" ] .
```

Contenedores

- los elementos contenedores se usan para agrupar recursos sobre los que queremos describir en conjunto. Cada uno de los elementos que agrupan se denomina miembro.
- RDF define tres marcados para contenedores
  - `rdf:Bag`
  - `rdf:Seq`
  - `rdf:Alt`
- para indicar que un recurso es un contender se puede usar la propiedad `rdf:type`
- pero no se especifica ninguna semántica (restricción formal) a los elementos de este tipo

- un `rdf:Bag` es un contenedor sin orden entre sus miembros, que pueden repetirse
- un `rdf:Seq` es un contenedor en donde el orden de los miembros es relevante, y también pueden repetirse
- un `rdf:Alt` es un conjunto de alternativas
- el contenido de estos elementos se nombran con los nombres `rdf:_1`, `rdf:_2`, etc
- cuando el orden no es relevante, puede usarse también `rdf:li`

Ejemplo

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">
   <rdf:Description rdf:about="http://example.org/courses/6.001">
      <s:students>
         <rdf:Bag>
            <rdf:li rdf:resource="http://example.org/students/Amy"/>
            <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
            <rdf:li rdf:resource="http://example.org/students/Johann"/>
            <rdf:li rdf:resource="http://example.org/students/Maria"/>
            <rdf:li rdf:resource="http://example.org/students/Phuong"/>
         </rdf:Bag>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

Colecciones

- una limitación de los contenedores anteriores, es que en ningún lado se especifica que los elementos declarados miembros de un contenedor son los únicos miembros.
- para sobrellevar esta limitación, la última versión de RDF incluyó el contenedor `rdf:List`, junto con las propiedades predefinidas `rdf:first` y `rdf:rest` y el recurso predefinido `rdf:nil`.
- permiten crear listas con la misma semántica de la programación funcional
- con el atributo `rdf:parseType="Collection"` se puede abreviar su escritura
- de vuelta, RDF no impone ninguna restricción al uso de este vocabulario

Ejemplo

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

<rdf:Description rdf:about="http://example.org/courses/6.001">
   <s:students rdf:nodeID="sch1"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch1">
   <rdf:first rdf:resource="http://example.org/students/Amy"/>
   <rdf:rest rdf:nodeID="sch2"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch2">
   <rdf:first rdf:resource="http://example.org/students/Mohamed"/>
   <rdf:rest rdf:nodeID="sch3"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch3">
   <rdf:first rdf:resource="http://example.org/students/Johann"/>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</rdf:Description>
</rdf:RDF>
```

Ejemplo

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">
   <rdf:Description rdf:about="http://example.org/courses/6.001">
      <s:students rdf:parseType="Collection">
            <rdf:Description rdf:about="http://example.org/students/Amy"/>
            <rdf:Description rdf:about="http://example.org/students/Mohamed"/>
            <rdf:Description rdf:about="http://example.org/students/Johann"/>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

Reificación

- es posible que una aplicación RDF describa propiedades sobre sentencias RDF, por ejemplo para decir quién o cuándo fue especificada
- RDF provee entonces vocabulario predefinido para describir sentencias RDF. Esto se denomina reificación
- este vocabulario consiste del tipo `rdf:Statement` y de los predicados `rdf:subject`, `rdf:predicate` y `rdf:object`
- de vuelta, es necesario usar con cuidado este vocabulario, porque puede llevar a confusiones
- el conjunto de las sentencias que reifican una sentencia es distinto a la sentencia, y ninguno implica el otro

Ejemplo

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:exterms="http://www.example.com/terms/"
            xml:base="http://www.example.com/2002/04/products">
  <rdf:Description rdf:ID="item10245">
     <exterms:weight rdf:ID="triple12345" rdf:datatype="&xsd;decimal">2.4
     </exterms:weight>
  </rdf:Description>
  <rdf:Description rdf:about="#triple12345">
     <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
  </rdf:Description>
</rdf:RDF>
```