UNIVERSITAT OBERTA DE CATALUNYA

Ingeniería Técnica en Informática de Gestión

Estudio del modelo de representación XML/RDF

Alumno: David Fernández Medina
Dirigido por: Carlos Granell Canut

CURSO 2003-04

Resumen

Estudio del modelo de representación XML/RDF

Este estudio tiene como objetivo introducir al lector en dos tecnologías complementarias, XML y RDF. El estudio muestra los inicios de los lenguajes de marcado explicando para qué fueron concebidos. De ellos destacan el SGML como estándar internacional de almacenamiento e intercambio de datos y HTML como aplicación SGML creada para el WWW. El estudio también pretende mostrar las limitaciones de HTML que provocaron la aparición del XML, un metalenguaje de marcado subconjunto de SGML y que proporciona una forma mejor de estructurar y expresar los contenidos en Internet, aunque su uso no está limitado a este ámbito. Aquí se explica como funciona XML y sus tecnologías asociadas. Sin embargo las limitaciones de XML con los metadatos hacen que se relacione con otra tecnología, el RDF, que proporciona un mayor significado a los datos, una forma de interpretar su semántica. El estudio concluye con la presentación de esta asociación de tecnologías, la representación XML/RDF donde se representa el modelo de datos RDF utilizando la sintaxis XML. Esto, llamado serialización XML/RDF, es lo que puede dotar de mayor inteligencia a la WWW y a otros ámbitos implicados. Aprovechando la expresividad universal e interoperabilidad sintáctica de XML junto con la interoperabilidad semántica que aporta RDF, el extenso y creciente mundo de Internet puede encontrar en la representación XML/RDF la base para una mejor recuperación y procesamiento de su información.

Índice

Índice de figuras

Glosario

Lenguaje de marcado: lenguaje que permite añadir marcas a un documento de texto para dar semántica o describir como presentar el contenido del documento.

Metalenguaje: en el mundo de la informática, un lenguaje informático que se utiliza para definir otros lenguajes.

GML: *Generalized Markup Languaje* (Lenguaje de marcado generalizado).Primer metalenguaje moderno de marcado.

SGML: *Standard Generalizaed Markup Language*. Metalenguaje para definir lenguajes de marcado predecesor de XML.

ISO: *International Standardization Organitation* (Organización Internacional de Normalización).Es una organización no gubernamental constituida en 1947 por las organizaciones de normalización de 133 países. La misión de ISO es promover el desarrollo de la normalización y actividades afines en el mundo con el propósito de facilitar el intercambio internacional de bienes y servicios, desarrollando a su vez, la cooperación internacional en los ámbitos de la actividad económica, científica, intelectual y tecnológica .

DTD: *Document Type Definition* (Definición de Tipo de Documento). Es un formato que permite definir la estructura y elementos de documentos SGML y XML.

CDATA: *Character DATA.*Una etiqueta XML predefinida para indicar al parser que no interprete los caracteres contenidos en ella.

HTML: *HyperText Markup Language*. Es un lenguaje de marcado utilizado para la creación de documentos que se quieran publicar en la Web estandarizado por el W3C.

XML: *Extensible Markup Language* (Lenguaje Extensible de Marcas). Es un metalenguaje para definir lenguajes de marcado. Estandarizado por el W3C.

W3C: *World Wide Web Consortium*. Es el organismo que se encarga de desarrollar los estándares relacionados con el Web.

WWW o **W3:** *Word Wide Web*. Sistema de información distribuido, basado en hipertexto, creado a principios de los años 90 por Tim Berners-Lee.

CSS: *Cascading Style Sheets* (Hojas de Estilo en Cascada). Hojas de estilo que permiten definir como presentar documentos XML y también documentos HTML.

XSL: *eXtensible Stylesheet Language* (Lenguaje Extensible de Hojas de Estilo). Lenguaje de hojas de estilo para XML estandarizado por el W3C. Se compone de XSLT y XSL-FO.

XSLT: *XSL Transformations* (Transformaciones XML). Parte de XSL que permite definir como transformar un documento XML en otro documento en formato XML, HTML o texto plano.

XSL-FO: *XSL Formatting Objects* (Objetos de Formateado de XSL). Parte de XSL que permite definir como presentar un documento XML.

Parser: Analizador sintáctico

URI: *Uniform Resource Identifier* (Identificador Uniforme del Recurso) Es el conjunto genérico de todos los nombres y direcciones que se refieren a un recurso.

URL: *Uniform Resource Locator* (Localizador Uniforme del Recurso). Es un tipo de URI para identificar una ubicación exacta en el World Wide Web.

Namespace: (Espacio de nombres). Permite identificar una fuente que define un conjunto de elementos y atributos que son utilizados en un documento XML.

XML Schema: (Esquema XML) Al igual que las DTDs, permite definir como es una cierta aplicación XML, pero de una forma mucho más precisa.

Metadato: Es un dato que se utiliza para describir o añadir información sobre otro dato.

RDF: *Resource Description Framework*. (Infraestructura para la Descripción de Recursos) Es la propuesta del W3C para definir metadatos en el Web y la base para el procesamiento de metadatos: proporciona interoperabilidad semántica entre aplicaciones que intercambian información entendible por máquina. RDF es simplemente un modelo de datos que permite crear metadatos legibles y entendibles por máquina. La interoperabilidad semántica de sistemas de metadatos implica significados compartidos y gramáticas compartidas.

Rdfs o RDF Schema: (Esquema RDF) definición de propiedades y relaciones entre recursos identificados en la WWW.

Serialización: representación de un origen en otro formato distinto al original.

Introducción

El presente estudio pretende realizar una introducción a dos tecnologías que permiten dotar de inteligencia a Internet. Por una lado está XML, un metalenguaje de marcado, una sintaxis para expresar y transportar los documentos. Y por otro RDF, un modelo de datos para la descripción de recursos definiendo metadatos. Ambas tecnologías son independientes pero presentan algunas limitaciones y necesidades que hacen interesante su complementación, el modelo de representación XML/RDF. En este ámbito se definen los objetivos del estudio, explicar ambas tecnologías y sus aportaciones, y mostrar cómo y porqué se relacionan. Además de los conceptos teóricos se han incorporado ejemplos prácticos de uso con el fin de mostrar aplicaciones concretas que se benefician de ambas tecnologías. No se ha pretendido realizar una explicación exhaustiva de estas tecnologías y su entorno, sino ofrecer una visión introductoria y clara para ser entendida por lectores sin conocimientos previos de este ámbito, tal como era el caso del autor de este estudio. Para ello se toma como referencia la documentación que ofrecen los organismos y autores implicados en estas tecnologías, desde sus especificaciones hasta artículos más informales, y de esta documentación se ha extraído la información más significativa para ser expuesta en el presente documento.

1- Inicios de los lenguajes de marcado

Los lenguajes de marcado surgen a partir del nacimiento de SGML (Standard Generalized Markup Languaje) que es una norma de estandarización derivada de GML (Generalized Markup Languaje) [1] , la cual define que el etiquetado debe describir la estructura del documento electrónico y otros atributos de forma que puedan ser interpretados y procesados por una máquina. Así mediante marcas de delimitan y describen los elementos que componen el documento electrónico. Los lenguajes de marcado pueden utilizar de forma distintas las marcas, algunos las utilizan para estructurar los elementos del documento mientras otros las usan para definir el formato de presentación estos elementos.

1.1- SGML (Standard Generalized Markup Language)

En 1969 tres señores de IBM Research ,Charles F. Goldfarb, Ed Mosher y Ray Lorie, inventaron el primer lenguaje moderno de marcado GML (Generalizad Markup Languaje). Este describía cómo acotar mediante marcas la estructura arbitraria de un conjunto de datos, y llegó a convertirse en un metalenguaje, Un lenguaje que puede ser usado para describir otros lenguajes, sus gramáticas y sus vocabularios. GML se convirtió en SGML [2], y en 1986 fue adoptado por la ISO como un estándar internacional de almacenamiento e intercambio de datos (ISO8879:1986) . SGML proporciona un modo consistente y preciso de aplicar etiquetas para describir las partes que componen un documento, permitiendo además el intercambio de documentos entre diferentes plataformas gracias a que está codificado únicamente con caracteres ASCII. Las marcas determinan la estructura no la presentación. Los elementos que pueden aparecer en un tipo de documento, sus características y el orden en el que deben escribirse se definen en un documento aparte llamado DTD (Document type definition), de esta forma los documentos se consideran instancias de un tipo de documento específico que define su estructura válida. El documento debe incluir una referencia a la DTD a partir de la cual se ha definido y cuyas restricciones debe cumplir. SGML también tiene capacidad de hipertexto donde los enlaces unen un elemento origen con un elemento destino. Para poder imprimir o visualizar por pantalla un documento SGML se le debe aplicar un formato mediante una hoja de estilos donde se indica como formatear cada elemento del documento. Estas características hicieron que SGML se convirtiera en norma de facto para el intercambio de documentos extensos, complejos y de tipos muy diferentes. Sin embargo tiene un gran inconveniente, su complejidad.

1.2- HTML (HyperText Markup Language)

Tim Berners-Lee cuando desarrolló el World Wide Web a finales de los 80 necesitaba un lenguaje para expresar los contenidos que iban a circular por la red y presentarse a través del navegador. Entonces utilizó el metalenguaje SGML como base para definir el lenguaje HTML original [3]. Así se concibe HTML como una aplicación SGML para codificar documentos y distribuirlos en el World Wide Web. Su sencillez

le proporciona éxito inmediato (revolución en Internet) pese a ello presenta algunas limitaciones de las que destacan las siguientes:

- Limitación para describir documentos complejos o datos.
- Define más la presentación que el contenido.
- No es fácilmente procesable por máquinas.
- Problemas de internacionalización.
- Falta estructuración y validación.
- No es reutilizable ni extensible.
- Interpretación ambigua según software utilizado.
- Sólo usable para Web.
- No dispone de un mecanismo de enlaces robusto.
- Tecnologías (Java, lenguajes de script, etc.) muy potentes cuya capacidad está siendo infrautilizada por las limitaciones del formato HTML.

Algunas de estas limitaciones se han querido subsanar con la incrustación de scripts, javascripts, Active X, HTML dinámico, hojas de estilo en cascada (CSS). Todo esto es insuficiente para crear una arquitectura abierta de tipo cliente/servidor, con lo que el W3C (Word Wide Web Consortium), organismo que vela por el desarrollo de la Word Wide Web, se ha replanteó crear un nuevo estándar llamado XML (eXtensible Markup Language) que ofreciese las ventajas que aportaba SGML pero evitando sus complejidades.

2- XML (eXtensible Markup Language)

La versión 1.0 del lenguaje XML es una recomendación [4] del W3C desde Febrero de 1998, pero se ha trabajado en ella desde un par de años antes. Está basado en el anterior estándar SGML, de él se derivó XML como subconjunto simplificado.
HTML es simplemente un lenguaje, una aplicación SGML, mientras que XML como SGML es un metalenguaje, esto es, un lenguaje para definir lenguajes.
XML no se presenta como sustituto de HTML, XML no ha nacido sólo para su aplicación en Internet, sino que se propone como lenguaje para intercambio de información estructurada entre diferentes plataformas.
Mientras que HTML especifica lo que cada etiqueta y atributo significan (y frecuentemente la apariencia que presentará en un navegador el texto que hay entre ellos) XML usa las etiquetas sólo para delimitar piezas de datos, y deja la interpretación de los datos, completamente, a la aplicación que los lee, pero las reglas para los archivos XML son más estrictas que para los archivos HTML.

Algunos de los objetivos planteados por el Grupo de Trabajo XML y el W3C son:

- XML debe ser directamente utilizable sobre Internet.
- XML debe soportar una amplia variedad de aplicaciones.

- XML debe ser compatible con SGML.
- Debe ser fácil la escritura de programas que procesen documentos XML.
- El número de características opcionales en XML debe ser absolutamente mínimo, idealmente cero.
- Los documentos XML deben ser legibles por los usuarios de este lenguaje y razonablemente claros.
- El diseño de XML debe ser formal, conciso y preparado rápidamente.
- Los documentos XML deben ser fácilmente creables.
- La brevedad en las marcas XML es de mínima importancia.

2.1- Comparativa HTML y XML

El siguiente cuadro muestra algunas de las características que diferencian XML y HTML.

XML	HTML
-Subconjunto propio de SGML (Metalenguaje).	-Aplicación de SGML (Lenguaje).
-Solo se codifica estructura de datos, no forma de presentación.	-Mezcla información estructural y de presentación.
-El significado de las marcas puede cambiar.	-Marcas sintácticas con un significado fijo.
-No especifica etiquetas, sino cómo deben definirse conjuntos de etiquetas aplicables a un tipo de documento.	-Conjunto limitado de etiquetas y un único tipo de documento.
-Modelo de hiperenlaces complejo (múltiples destinos, fijos y relativos, etc.).	-Modelo de hiperenlaces simple (unidireccionales y fijos).
-Gran capacidad para procesar documentos, el navegador es una plataforma para el desarrollo de aplicaciones.	-Escasa capacidad de procesamiento, el navegador es un mero visor de páginas
-Fin de la guerra de los navegadores y etiquetas propietarias.	-El problema de la 'no compatibilidad' y las diferencias entre browsers ha alcanzado un punto en el que la solución es difícil.

Figura 1: Tabla comparativa entre XML y HTML

En el siguiente cuadro se muestra un mismo ejemplo empleando XML y HTML donde queda patente la falta de estructuración de la información en HTML contra la estructuración jerárquica de XML.

Ejemplo HTML	**Ejemplo XML**
David Fernández Medina c/ Sin nombre 15, 6 Badalona 08912rcelona 	<dirección> <destinatario> <nombre>David</nombre> <apellidos>Fernández Medina</apellidos> </destinatario> <calle> c/ Sin nombre </calle> <numero>15</numero> <piso>6</piso> <ciudad>Badalona</ciudad> <código postal>08912ódigo postal> <provincia>Barcelona</provincia> </dirección>

Figura 2: Ejemplo comparativo de XML y HTML

2.2- Características de XML

Esta son las principales características de XML:

- Es un dialecto simplificado de SGML pero con las propiedades más importantes y deseables de éste.
- XML es un Metalenguaje: es un lenguaje de marcas permite la creación de otros lenguajes de marcas.
- XML ofrece una representación estructural de los datos que se puede implementar ampliamente y es fácil de distribuir.
- XML garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes. Interoperabilidad.
- Intercambio de datos e información estructurada a través de Internet y el WWW (Prácticamente se ha convertido en el estándar de facto para el intercambio de información en Internet.).
- Facilita la Integración de datos procedentes de fuentes heterogéneas.
- Fácilmente procesable por humanos y máquinas.
- Separa información o contenido de presentación y formato.
- Utilizable con cualquier lenguaje o alfabeto.

- Codifica datos de acuerdo a un esquema semántico.
- Basado en marcas sintácticas que cumplen reglas léxicas y sintácticas. Se etiquetan los componentes del documento de acuerdo con su semántica.
- Se le puede asociar una gramática para definir un modelo de estructura de documento (DTD - Esquema).
- Extensible, etiquetas no predefinidas.
- Pretende ser razonablemente simple.

2.3- Estructura del documento XML

Un documento XML debe incluir un **prólogo y la instancia documento.**

El prólogo puede estar formado por tres secciones:

- Declaración XML
- Declaración de tipo de documento.
- Referencia a una hoja de estilo externa.

El siguiente ejemplo muestra las distintas partes del documento XML

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE Libro SYSTEM "libro.dtd">
<?xml-stylesheet href="librohtml.xsl" type="text/xsl"?>

<Librería>
  <Libro>
     <Titulo>El Quijote</Titulo>
     <Autor>Miguel de Cervantes Saavedra</Autor>
     <ISBN>11111-11111-11111</ISBN>
  </Libro>
<Librería>
```

Declaración XML
Declaración de tipo de documento
Hoja de estilo externa
} Prólogo

} Instancia documento

2.3.1- Prólogo

La **declaración XML** puede contener tres partes

Versión (version)

Indica qué versión de la especificación XML utiliza el documento.

Codificación (Encoding)

Juego de caracteres que se utiliza en el documento. Es opcional.

Declaración de documento aislado (standalone)

Dice si este documento referencia o no a una entidad externa o a una especificación de datos externa. Si no hay referencias externas "yes" es el valor apropiado. Es opcional.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
```

Declaración de tipo de documento (Opcional)

• El tipo de documento define la estructura y sintaxis que deben seguir los documentos que sean de ese tipo.
• <!DOCTYPE nombre … > donde nombre será sustituido por el nombre del tipo de documento en cuestión.

```
<!DOCTYPE Libro SYSTEM "libro.dtd">
```

• Tras el nombre del tipo de documento, la declaración de tipo de documento puede contener:
a) la definición del tipo de documento o DTD
b) una referencia a un recurso externo que contiene la DTD de ese tipo de documento.
c) una referencia a un recurso externo con la DTD, y la definición de parte de la DTD aplicable únicamente a ese documento.

Referencia a una hoja de estilo externa (opcional).
Indica cómo representar el documento, instrucción de procesamiento. Para ello se puede optar por dos tecnologías, las hojas de estilo en cascada (CSS) o las hojas de transformaciones XSL. Aquí se muestra un ejemplo de declaración de cada una:

```
<?xml-stylesheet href="libro.css" type="text/css"?>
<?xml-stylesheet href="libro.xsl" type="text/xsl"?>
```

2.3.2- Instancia documento

La instancia documento es el contenido del documento propiamente dicho. La instancia comienza con la marca de inicio del elemento documento o raíz, y concluye con la marca de fin de este mismo elemento.
El elemento raíz puede contener más elementos formando un estructura jerárquica en forma de árbol.
Un par de etiquetas define un elemento cuyo contenido puede ser:

-texto: <par> Contenido de un párrafo </par>
-elementos:
<chapter>
<section> blablabla... </section>
<section> blablabla... </section>
</chapter>
-mixto:
<par> Texto con <footnote>pie</footnote> </par>

Dentro de la instancia documento también se puede hacer referencia a un esquema XML, este esquema es una tecnología alternativa a la DTD y al igual que la DTD sirve para definir la gramática que debe seguir el documento. El siguiente ejemplo muestra su uso:

```
<?xml version="1.0"?>
<Libreria xmlns="http://www.libros.org"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.libros.org libreria.xsd">
    <Libro>
         <Titulo>El Quijote</Titulo>
         <Autor>Miguel de Cervantes Saavedra</Autor>
         <ISBN>11111-11111-11111</ISBN>
   </Libro>
</Librería>
```

2.4- Documentos bien formados y válidos

- **Documentos válidos**: Está bien formado y de acuerdo a una definición de lenguaje (gramática): DTD o XMLSchema. De la validación de encarga un procesador o parser XML.
- **Documentos bien formados**: Sin errores de sintaxis. Cumple reglas de la especificación XML 1.0.

Hay pues diferencia entre los parsers que procesan documentos XML sin comprobar que siguen las reglas marcadas por una gramática (sólo comprueban que está bien formado), que se llaman parsers no validadores, y los que sí lo hacen, que son parsers validadores (comprueba que además de bien formado se atiene a su DTD o XML Schema y por tanto es válido).

2.4.1- Documentos bien formados:

Un objeto textual o documento XML se dice que está bien formado si, considerándolo como conjunto, encaja con las especificaciones XML de producción, lo que implica:

- Todo documento XML se compone de elementos. Cada elemento está delimitado por una etiqueta de comienzo y otra de fin, a no ser que sea vacío. Los elementos vacíos constan de una única etiqueta. Los nombres de las etiquetas son arbitrarios y no pueden contener espacios.
- Siempre hay un elemento raíz, cuya etiqueta de inicio ha de ser la primera de todas y la de cierre la última de todas.
- Cada elemento puede contener datos de carácter, elementos, ambas cosas a la vez o puede estar vacío (en tal caso debe terminar con '/>').
- No se puede mezclar la anidación de las etiquetas: los elementos deben abrirse y cerrarse por orden.
- Los elementos pueden tener atributos (propiedades) que nos ofrecen información sobre ellos. Los valores de los atributos deben ir entrecomillados. Tanto atributos como valores son arbitrarios.
- Mayúsculas y minúsculas no son intercambiables.
- El espacio en blanco es libre, se puede utilizar para leer mejor el documento.
- Se pueden utilizar comentarios, que el analizador no tratará, en cualquier sitio excepto dentro de las declaraciones, etiquetas y otros comentarios.
- Las secciones CDATA sirven para introducir texto que el analizador tendrá en cuenta como datos de carácter, sin interpretarlo como XML.

2.4.2- Documentos validados por DTD

En ella definimos los elementos que conformarán ese tipo de documentos y como tienen que estar organizados para que sea correcto. La DTD puede determinar:

– Los elementos que pueden aparecer en el documento
– Los atributos que pueden utilizarse junto a cada elemento
– Cómo se pueden anidar los elementos (padres e hijos)
– El orden en el que deben aparecer los elementos hijos de un mismo padre
– El número permitido de elementos hijos
– Si un elemento puede ser vacío o no
– Tipos de datos para elementos y atributos
– Valores por defecto y fijos para elementos y atributos

El DTD puede estar incluido en el propio documento XML o bien ser un documento externo.
Un documento XML se asociará con su DTD mediante la etiqueta <!DOCTYPE > en el prólogo del mismo. En esta etiqueta se podrá especificar una DTD completa (DTD interna), o la URL donde encontrar la DTD (DTD externa) que permite validar el XML.

Interno:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE doc [<!ELEMENT doc (#PCDATA)>]>
```

Externo:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE doc SYSTEM "mitipo.dtd" >
```

Entre las ventajas se podrían enumerar que DTD es un estándar robusto y maduro, lo que le aporta mucha experiencia, esto implica que existen gran cantidad de desarrollos compatibles (programas de edición, parsers, transformadores, ejemplos , etc.).
Por el contrario se podría decir que las DTDs son SGML, no XML, por lo que no recoge conceptos recientes como son los Namespaces. También se le puede reprochar que la definición de tipos es bastante limitada. Ante estas limitaciones surge XML Schema [5] desarrollado con XML y como una alternativa a las DTD.

2.4.3- Documentos validados por XML Schema

Su finalidad, como la de las DTDs, es la de describir la estructura de documentos XML. Entre sus principales características destacan:

-XML Schemas están escritos en XML y por lo tanto son extensibles.
-Son más ricos y útiles que los DTDs.
-Soportan tipos de datos y no solo #PCDATA.
-Soportan espacios de nombres (namespaces)
-Procesables igual que los documentos XML

El siguiente ejemplo muestra parte de un documento XML que declara un esquema XML:

```
<?xml version="1.0"?>
        <note xmlns="http://www.w3schools.com"
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:schemaLocation= "http://www.w3schools.com/schema/note.xsd">
        ...
 </note>
```

2.5- Los espacios de nombres (Namespaces)

Un espacio de nombres (namespaces) [6] XML es un conjunto de nombres, identificados por una referencia URI, que se utilizan en documentos XML como tipos de elemento y nombres de atributo. Resuelven los problemas que pueden surgir cuando en un documento XML se combinan elementos procedentes de esquemas diferentes, y/o los esquemas contienen elementos o atributos con idéntico nombre. En un documento que combine dos o más esquemas, se utilizarán tantos namespaces como esquemas y deberán estar declarados en la marca de inicio del elemento raíz del documento. La declaración consiste en las letras xmlns: seguidas de un identificador para el namespace, tras él se indica un URI único para el namespace. Todos los nombres de elementos y de atributos que se utilizan en un documento XML en el que se hayan declarado namespaces deben ir cualificados por el prefijo declarado para su namespace.

Este es un ejemplo básico de uso de un namespace:

```
<v:vuelo-aéreo xmlns:v =“http://www.myserver.com/schemes/vuelo”
</v:vuelo-aéreo>
```

2.6- Presentación de XML

Para poder publicar un documento XML es necesario vincularlo a una hoja de estilo que contenga información relativa a cómo se deben formatear el documento para imprimir o visualizar en pantalla. Para crear hojas de estilo para documentos XML disponemos de dos alternativas:

- Utilizar hojas de estilo CSS (Cascading Style Sheets) [7], heredadas del lenguaje HTML, para el cual se diseñaron.
- Utilizar hojas de estilo XSL [8], específicamente diseñadas para documentos XML.

2.6.1- CSS (Cascading Style Sheets)

Las hojas de estilo en cascada es una forma muy popular de dar formato a archivos de lenguaje de marcado como HTML. Aprovechando precisamente esta utilidad podemos usarlas para representar nuestros documentos XML en un navegador Web. Esto se realiza añadiendo una instrucción de procesado a nuestro XML en el prólogo del mismo:

<?xml-stylesheet href="MI_HOJA_DE_ESTILO.CSS" type="text/css" ?>

Ello hará que el navegador al visualizar el contenido del XML, acceda a la URL "MI_HOJA_DE_ESTILO.CSS" y la use para procesar el documento.

Básicamente un fichero CSS, es un fichero de texto plano en el que especificamos reglas CSS con la siguiente sintaxis:

selector {propiedad_1:valor_1 ;....; propiedad_n:valor_n; }

Desventajas del uso de CSS

Aunque el uso de CSS nos permite dotar a nuestros documentos XML de una "visualización visible", presenta una serie de inconvenientes que obligaron al W3C a buscar otra alternativa:

- Un fichero CSS no tiene sintaxis XML. Si estamos utilizando tecnologías XML carece de sentido que usemos algo no XML. Es el mismo caso que pasaba con DTD y XML Schema.
- Es un simple mecanismo de Visualización. Sólo podemos usarlo en un navegador WEB que soporte CSS.
- No existe interacción con el documento XML.

2.6.2- XSL (eXtensible Stylesheet Language)

XSL es el acrónimo de eXtensible Stylesheet Language, y lo componen dos estándares abiertos del W3C: XSLT (XSL Transformations) y XSL FO (Formatting Objects). XSLT presenta una forma de transformar documentos XML de forma que puedan ser interpretados por XSL FO (de forma similar a CSS) para su presentación en otros formatos y dispositivos (HTML, PDF, RTF, Word, WAP, PDA etc.).

XSL está basado en XML, no como sucedía con CSS. Por tanto es un lenguaje declarativo, claro, legible y extensible.

En el prólogo del documento XML se hace referencia a la hoja de estilo, un ejemplo sería:

<?xml-stylesheet href=" MI_HOJA_DE_ESTILO.XSL " type="text/xsl" ?>

2.7- Procesamiento de un documento XML

En la siguiente figura se muestra el procesamiento del documento XML. Un documento XML será validado por un parser validador cuando haga referencia a una DTD o XML Schema. Este parser comprobará si está bien formado (cumple con la especificación XML 1.0) y si cumple con la gramática definida en la DTD o XML Schema. Si el documento XML no indica ninguna gramática para validarlo entonces un parser no validador comprobará que esté bien formado. El documento XML también puede estar vinculado a una hoja de estilo (o a varias) y en tal caso será procesado para su presentación. Las hojas de estilo pueden ser CSS o XSL, en el primer caso el formato de salida será un documento HTML, mientras que con la XSL se puede obtener como salida un documento HTML, PDF, Word, etc.

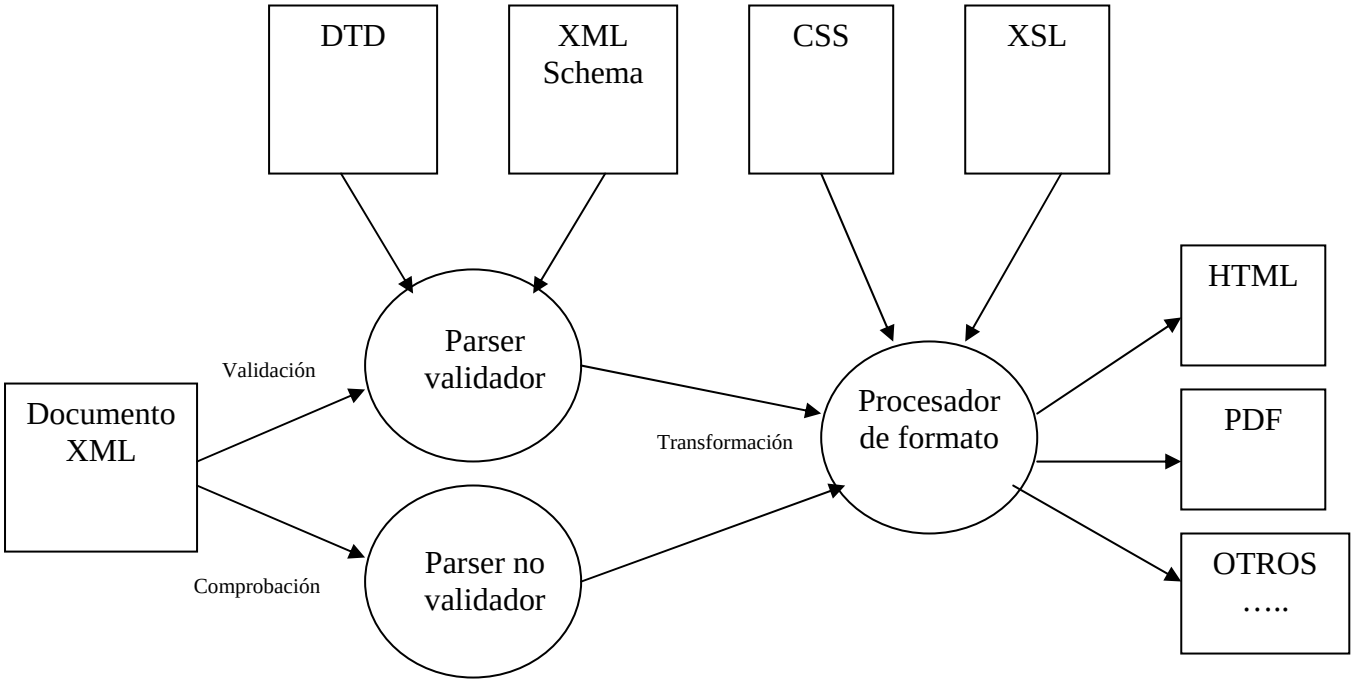

Figura 3: Procesamiento de documento XML

2.8- Limitaciones de XML con los metadatos

Los metadatos se pueden definir como datos sobre datos, es decir, descripciones de características y propiedades sobre datos. Estas descripciones son de gran ayuda en la catalogación y recuperación de datos. XML presenta algunas deficiencias para la creación de metadatos. Básicamente, la falta de flexibilidad y escalabilidad, ya que el orden en el cual los elementos aparecen en un documento XML es significativo y muchas veces necesario. Esto es altamente antinatural en el mundo de los metadatos, no es relevante si en un listado de propiedades de un libro se pone el ISBN antes que la fecha de publicación, lo importante es poder acceder a los datos. Además es difícil y costoso el mantenimiento del orden sobretodo

cuanto mayor son los elementos a tratar [9]. La limitación está en la estructura jerárquica y en forma de árbol del modelo de datos XML. Los árboles XML son complicados de combinar entre sí. Además los accesos en estos árboles es problemático debido a los múltiples caminos que puede haber para llegar hasta un dato determinado [10]. Este problema se acentúa cuanto mayor es el árbol, y por ello no ofrece buena escalabilidad en sus accesos. Por otro lado las gramáticas XML permiten validar la estructura de los documentos XML pero no permiten interpretar de forma general los datos que contienen. Para ello se debe implementar una aplicación que proporcione los significados de los datos. Ante estas limitaciones aparece RDF con un modelo de datos más flexible basados en triples (sujeto + predicado + objeto) y cuyos elementos están identificados con URIs lo que permite el acceso a su semántica y a la interpretación de los datos.

3- RDF (Resource Description Framework)

RDF [11] fue creado en agosto de 1997 bajo los auspicios del World Wide Web Corsortium (W3C) con el fin de crear un formato que permitiera alcanzar la compatibilidad entre los diversos sistemas de metadatos, suministrando para ello una arquitectura genérica de metainformación.

El objetivo general de RDF es definir un mecanismo para describir recursos que no cree ninguna asunción sobre un dominio de aplicación particular, ni defina (a priori) la semántica de algún dominio de aplicación. La definición del mecanismo debe ser neutral con respecto al dominio, sin embargo el mecanismo debe ser adecuado para describir información sobre cualquier dominio. La capacidad que tiene RDF para procesar metadatos facilita la interoperabilidad entre diversas aplicaciones, proporcionando un mecanismo perfecto de intercambio de información a través del Web.

Existen varios conceptos que pueden definir el modelo RDF, entre los cuales caben mencionar:

• Sistema que permite la interoperabilidad entre aplicaciones mediante el intercambio de información legible por ordenador a través del Web (Brickley, Dan y Guha, R. V., 2000).

• Mecanismo que facilita la automatización de procesos susceptibles de ser realizados con recursos Web (Lassila, Ora, 1998).

• Infraestructura que permite la codificación, intercambio y reutilización de metadatos estructurados (Miller, Eric, 1998). Es capaz, además, de fusionar diferentes sistemas de metadatos utilizados para la descripción de recursos Web (Iannella, Renato, 1999).

• Es una forma de expresar relaciones entre objetos (Hjelm, Johan, 2001).

En general RDF se enfoca en establecer un mecanismo que permita describir recursos, entendidos estos como objetos, que tengan como principios la multiplataforma (es decir, independencia de software y/o sistema operativo) y la interoperatividad de metadatos (que posibilite fusionar diferentes descripciones de recursos realizadas con distintos conjuntos de metadatos). RDF posee semánticas que generan una base para razonar sobre el significado de una expresión RDF. Además posee un vocabulario extensible, basado en URIs.

3.1- El modelo de datos RDF

- Un **recurso** es cualquier cosa que pueda ser identificada unívocamente por un URI (Uniform Resource Identifier). Identificadores universales para cualquier recurso de la red o de fuera de ella. Puede ser un documento HTML, una parte de una página, una colección de páginas, un sitio Web completo, una imagen...
- Una **propiedad** es un recurso que tienen un nombre y que puede usarse como una propiedad, por ejemplo autor o titulo. En muchos casos todo lo que nos importa en realidad es el nombre, pero una propiedad necesita ser un recurso de forma tal que pueda tener sus propias propiedades.
- Un **valor** es la representación que toma la propiedad.

Una **sentencia o descripción** consiste en la combinación de un recurso, una propiedad y un valor. Estas partes son conocidas como el sujeto, predicado y el objeto de la sentencia. Sentencia es por ejemplo *"El autor de http://www.myWeb.com/Documentos/RDF.html es David"*. El valor puede ser una cadena de caracteres (literal) por ejemplo "David" o puede ser otro recurso por ejemplo *"El home page de http://www.myWeb.com/Documentos/RDF.html es http://www.myWeb.com ."*

3.2- Representación del modelo de datos: El grafo RDF

Si se obvia el elemento description (al ser el que aglutina a los tres principales), se encuentra con que la base del modelo RDF es un triple donde un sujeto (el recurso) tiene un predicado (propiedad) con un objeto determinado (recurso o literal). Eso se podría representar en forma de gráfico de nodos y flechas. Los recursos se representan por óvalos, las propiedades son las flechas o arcos etiquetados que unen los nodos (sujeto-objeto) y los objetos son nodos representados por óvalos (si es un recurso) o por un rectángulo (si es un literal).

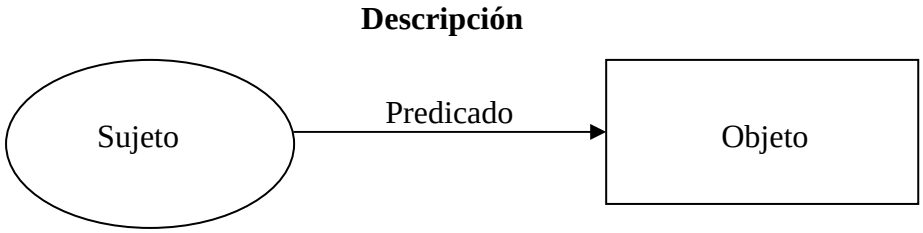

Figura 4: Grafo RDF simple

Este grafo RDF está formado una tripleta que muestra una descripción.

Usando el grafo anterior podemos representar la siguiente descripción:

"El autor de http://www.myWeb.com/Documentos/RDF.html es David"

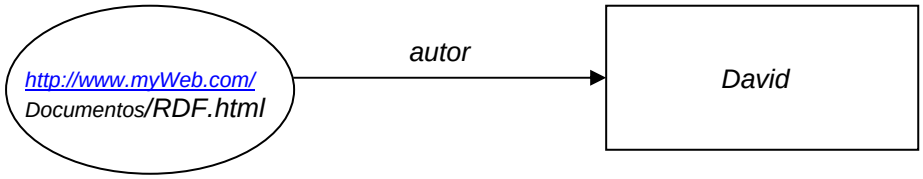

Figura 5: Grafo RDF ejemplo simple

El grafo puede ir creciendo conforme se le añadan más elementos a la descripción del recurso.

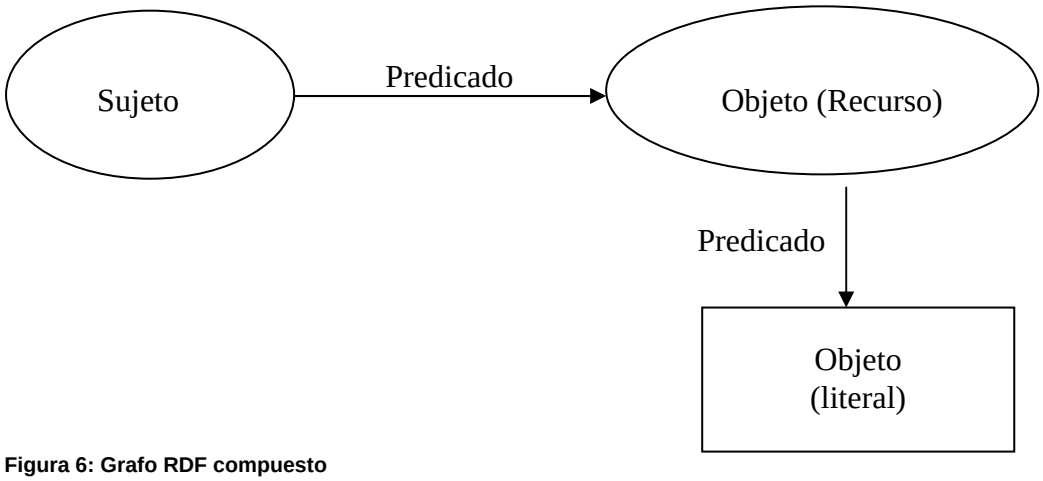

Figura 6: Grafo RDF compuesto

Este grafo RDF está formado por dos tripletas, la primera tripleta formada por el sujeto y un predicado cuyo valor (objeto) es a su vez otro recurso. Este último recurso contiene su propia propiedad utilizando un predicado y su correspondiente objeto para expresarla formando una segunda tripleta.

Con este grafo podemos expresar las siguientes sentencias o descripciones:

"El autor de http://www.myWeb.com/Documentos/RDF.html es David"

"El home page de http://www.myWeb.com/Documentos/RDF.html es http://www.myWeb.com ."

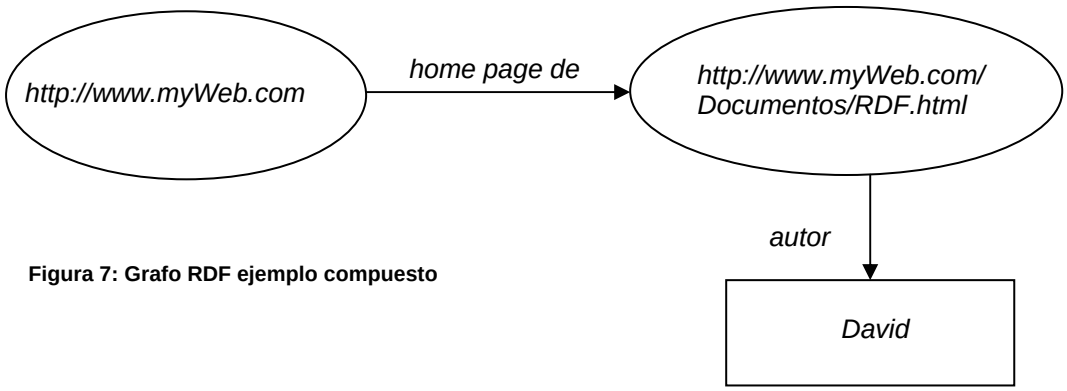

Figura 7: Grafo RDF ejemplo compuesto

Un recurso también puede ser anónimo, en tal caso no tiene una URI asociada y se representa mediante un óvalo sin etiquetar. La siguiente descripción *"El individuo cuyo nombre es David y email David@myWeb.org es el autor de http://www.myWeb.com/Documentos/RDF.html"* se puede representar mediante el siguiente grafo:

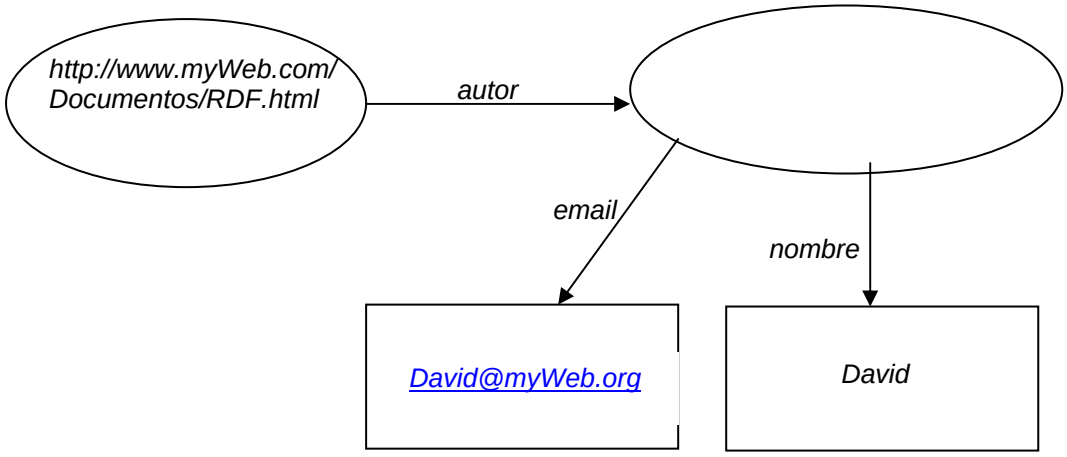

Figura 8: Grafo RDF con recurso anónimo

Los predicados son recursos no anónimos, y por tanto están referenciados por URIs y pueden tener sus propias propiedades.
Podemos usar el predicado *autor* del ejemplo anterior para realizar la siguiente descripción:

"Un autor es una persona"

Figura 9: Grafo RDF predicado con propiedad

Mediante esta nueva descripción un sistema podría inferir una nueva información utilizando la lógica de predicados ya que si sabemos que:

1-*"Un autor es una persona" (si A es un autor entonces A es una persona)*

2-*"El autor de http://www.myWeb.com/Documentos/RDF.html es David" (David es un autor)*

Por deducción natural tenemos:

David es una persona

Además siendo *persona* un recurso que puede tener definidas unas propiedades, podemos obtener propiedades del sujeto *David* sabiendo que las propiedades que tiene *persona* también las tiene *David*.

La forma de realizar estas relaciones y obtener sus propiedades se hace mediante el acceso a sus correspondientes URIs. Estas URIs identifican los recursos y propiedades.

3.3- RDF Schema

El modelo de datos de RDF define un modelo simple para describir las relaciones entre recursos en términos de propiedades y valores designados. Las propiedades RDF pueden entenderse como atributos de un recurso y en este sentido corresponden con los tradicionales pares atributo-valor. Las propiedades RDF también representan relaciones entre recursos. De esta forma, el modelo de datos RDF puede parecer un diagrama entidad-relación. El modelo de datos RDF, sin embargo, no proporciona mecanismos para declarar estas propiedades, ni proporcionan ningún mecanismo para definir las relaciones entre estas propiedades y otros recursos. Este es el papel o la función que desempeña el Esquema RDF [12].

Un Esquema RDF proporciona información sobre la interpretación de una sentencia dada en un modelo de datos RDF. Mientras una DTD o un Esquema XML pueden utilizarse para validar la sintaxis de una expresión XML, un esquema sintáctico sólo no es suficiente para los objetivos de RDF. Los Esquemas RDF pueden también especificar restricciones que deben seguirse por estos modelos de datos. El trabajo futuro en torno al Esquema RDF y al Esquema XML podría facilitar la sencilla combinación de reglas sintácticas y semánticas para ambos.
La *RDF Schema* es una extensión de RDF que proporciona primitivas adicionales. Enriquece el modelo básico, proporcionando un vocabulario para RDF, que se asume tiene una cierta semántica. Permite a los diseñadores especificar una jerarquía explícita de clases de recursos y propiedades que describen estas clases, junto con las restricciones sobre las combinaciones permitidas de clases, propiedades y valores.

En general, RDF Schema permite:

-Definir no sólo las propiedades de un recurso (ej. título, autor, materia, tamaño, color, etc.) sino que también definir los tipos de recursos que se describirán (libros, páginas Web, personas, empresas, etc.).
-proporciona información sobre la interpretación de una sentencia dada en un modelo de datos RDF (semántica).
-Definir la herencia de clases para crear una taxonomía del modelo; Esta es una poderosa característica de RDF Schema dado que en ella radica la extensibilidad en cuanto a elaboración de nuevos esquemas.
-Definir las relaciones entre recursos y propiedades, lo cual ayudará a inferir información del modelo y a la vez mejorar los procesos de búsqueda.
-Especificar restricciones que deben seguirse por estos modelos de datos.

El núcleo del vocabulario del esquema se define en un namespace, denominado aquí informalmente 'rdfs', e identificado por el URI de referencia http://www.w3.org/2000/01/rdf-schema#. Esta especificación utiliza también el prefijo 'rdf' para referirse al namespace principal RDF http://www.w3.org/1999/02/22-rdf-syntax-ns# que es un esquema RDF del modelo de datos RDF. Estos vocabularios definen un conjunto de recursos RDF (incluyendo clases y propiedades), y las restricciones en sus relaciones.

3.3.1- Clases, propiedades y restricciones

Clases:

Los recursos siguientes son las clases principales que se definen como parte del vocabulario del Esquema RDF. Cada modelo RDF que se traza sobre el namespace del Esquema RDF (implícitamente) los incluye. Estas son las principales:

rdfs:Resource: Cualquier cosa descrita por una sentencia RDF se considera una instancia de la clase rdfs:Resource, para ello debe poseer URI que lo identifique y permita el acceso a su descripción. Representa a los recursos definidos en el modelo de datos.

rdf:Property es la clase de todas las propiedades utilizadas en la caracterización de las instancias de rdfs:Resource. Son utilizados como predicados de los triples, la semántica de un triple depende de la 'property' utilizada como predicado.

rdfs:Class: este corresponde con el concepto genérico de un tipo o categoría semejante a la noción de Clase en los lenguajes de programación orientados a objetos tales como Java. Cuando un esquema define una nueva clase, el recurso que representa esa clase debe tener una propiedad rdf:type cuyo valor es el recurso rdfs:Class. Las clases RDF pueden definirse para representar cualquier cosa, como páginas Web, personas, tipos de documentos, bases de datos o conceptos abstractos.

Propiedades:

Son objetos específicos de una categoría *(instancias)* de la clase **rdf:Property** y proporcionan un mecanismo para expresar las relaciones entre las clases y sus objetos específicos de categoría *(instancias)* o superclases. Estas son las principales:

rdf:type Esta propiedad indica que un recurso es miembro de una clase, de tal forma que tiene todas las características que se prevén de un miembro de esa clase. Es un objeto específico de la categoría *(instancia)* de la clase especificada.

Rdfs:subClassOf modela la jerarquía de clases, donde una clase puede ser subclase de otras subclases. La propiedad **rdfs:subClassOf** es transitiva. Si la claes A es una subclase de otra clase B más amplia, y B es una subclase de C, entonces A es también implícitamente una subclase de C. Por lo tanto, los recursos que son objetos específicos de una categoría (instancias) de la clase A serán también ***instancias*** de C, puesto que A es un subconjunto de ambas, tanto de B como de C.

rdfs:subPropertyOf**:** es un objeto específico de una categoría (instancias) de **rdf:Property** que se utiliza para especificar que una propiedad es una especialización de otra. En tal caso se aplica la propiedad de transitividad por tanto si una propiedad P2 es una subpropiedad de (*rdfs:subPropertyOf*) otra propiedad P1, y si un recurso R tiene una propiedad P2 con valor V, esto implica que el recurso R también tiene la propiedad P1 con valor V.

rdfs:label: es para dar un nombre al sujeto legible por humanos.

rdfs:comment: es para proveer de una descripción más larga del recurso.

rdfs:seeAlso:Especifica un recurso que podría proporcionar información adicional sobre el recurso sujeto.

rdfs:isDefinedBy: La propiedad rdfs:isDefinedBy es una subpropiedad de rdfs:seeAlso, e indica el recurso que define el recurso sujeto.

Restricciones

La especificación de rdfs presenta un vocabulario RDF para hacer sentencias sobre restricciones en el uso de las propiedades y clases en datos RDF. Por ejemplo, un esquema RDF podría describir las limitaciones de los tipos de valores que son válidos para una propiedad, o las clases para las que tiene sentido asignar tales propiedades. El Esquema RDF proporciona un mecanismo para describir tales restricciones, pero no dice si, ni cómo, una aplicación puede procesar la información restringida.

rdfs:ConstraintResource define la clase de todas las restricciones.

rdfs:ConstraintProperty es un subconjunto de rdfs:ConstraintResource. Tiene dos instancias***: rdfs:range*** y ***rdfs:domain.***

rdfs:range: se usan para restringir el rango (un conjunto de valores válidos para la propiedad). No se permite expresar más de una restricción de rango sobre una propiedad.
rdfs:domain: usan para restringir el dominio (el conjunto de recursos que pueden tener una determinada propiedad). En dominios si se permite expresar más de una restricción de dominio, y se interpreta como una unión de dominios.

Otras clases

rdfs:Literal Valores atómicos tales como conjuntos de caracteres (strings) textuales, son ejemplos de literales RDF.

rdf:Statement: Corresponde con el conjunto denominado sentencia en el modelo formal para RDF presentado.

rdf:subject: Corresponde con la propiedad denominada sujeto en el modelo formal para RDF. Su rdfs:domain es rdf:Statement y rdfs:range es rdfs:Resource.

rdf:predicate: Corresponde con la propiedad denominada predicado en el modelo formal para RDF.

rdf:object Corresponde con la propiedad denominada objeto en el modelo formal para RDF.

rdfs:Container Esta clase se utiliza para representar las clases contenedoras. Es un objeto específico de la categoría (instancia) de rdfs:Class y rdfs:subClassOf de rdfs:Resource.

rdf:Bag Es un objeto específico de la categoría (instancia) rdfs:Class y rdfs:subClassOf rdfs:Container.
Un bag es una lista de recursos o literales sin orden.

rdf:Seq Es un objeto específico de la categoría (instancia) rdfs:Class y rdfs:subClassOf rdfs:Container.
secuencia es una lista ordenada de recursos o literales. Se permiten valores duplicados.
rdf:Alt Es un objeto específico de la categoría (instancia) rdfs:Class y rdfs:subClassOf rdfs:Container. Es una secuencia de recursos o literales para un "único" valor o propiedad. De la lista de recursos o literales debe elegirse uno.

rdfs:ContainerMembershipProperty Esta clase tiene como miembros las propiedades _1, _2, _3 ... utilizadas para indicar la agrupación contenedor.

rdf:value: Identifica el principal valor (normalmente un string) de una propiedad cuando el valor propiedad es un recurso estructurado.

3.3.2- Relaciones entre recursos

La Figura 10 ilustra el concepto de clase, subclase, y recurso. Una clase se describe por un rectángulo redondeado; un recurso se describe por un punto grande. En la figura abajo, se han dibujado flechas desde un recurso a una clase que define. Una subclase se presenta en un rectángulo redondeado (la subclase) completamente incluido en otro (la superclase). Si un recurso está dentro de una clase, entonces existe una propiedad rdf:type o explícita o implícita de aquel recurso cuyo valor es un recurso que define la

clase que contiene. (Estas propiedades se muestran como arcos en la representación gráfica directamente etiquetada en la figura 11).

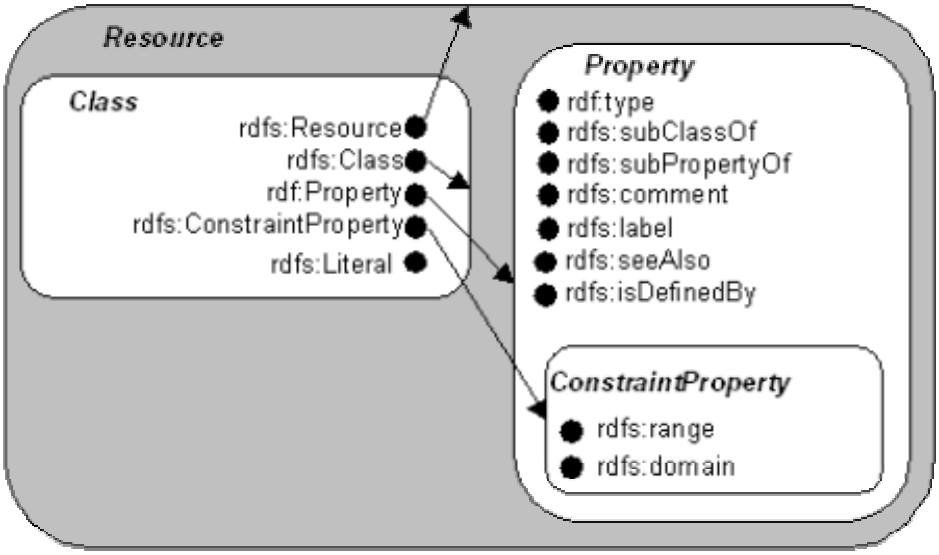

Figura 10: Clases y recursos como conjuntos de elementos

La Figura 11 muestra las relaciones jerárquicas entre los recursos (clases, propiedades y restricciones). Como se observa las clases y propiedades son subclases de recursos. Las propiedades a su vez son instancias de una clase.

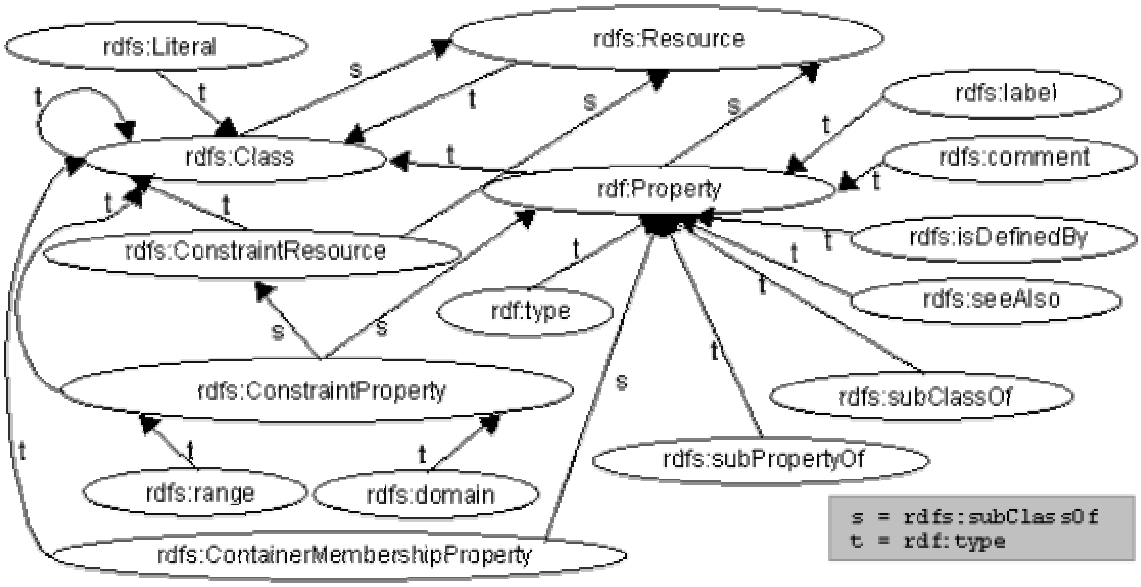

Figura 11: Jerarquía de clase para el Esquema RDF

Las restricciones se muestran en la figura 12. Los nodos destacados en **negrita** son objetos específicos (instancias) de **rdfs:Class**.

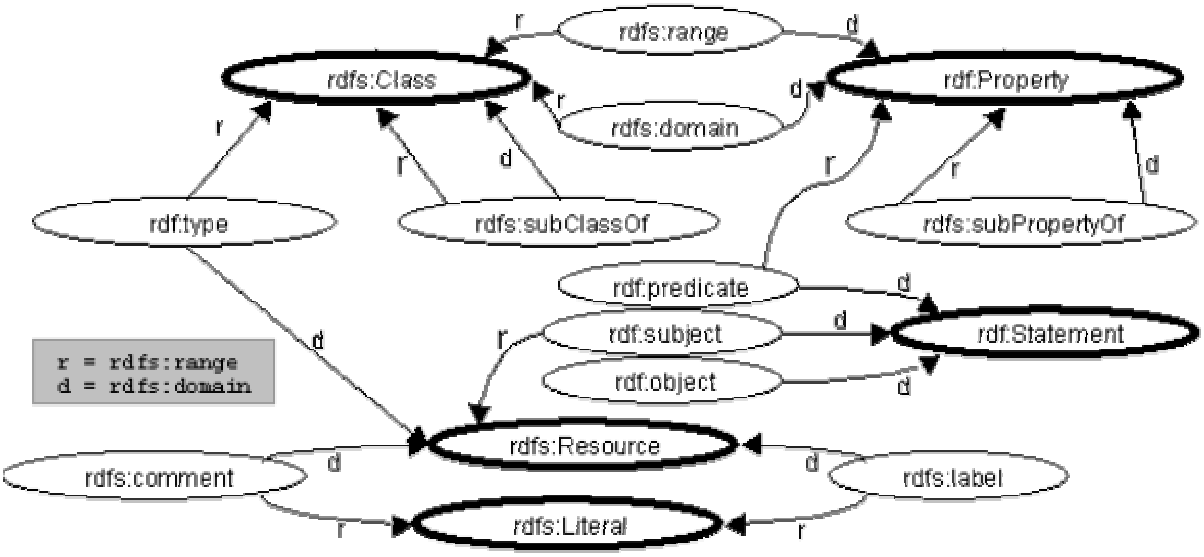

Figura 12: Restricciones en el Esquema RDF

Podemos crear nuevos vocabularios mediante estos y otros esquemas usando su namespace. También podemos describir recursos combinando varios vocabularios. El siguiente ejemplo ilustra una descripción de un documento Web usando los esquemas rdf y rdfs. Además se usa un esquema ficticio cuyo namespace podría ser *http://www.myWeb.org/schema#* y al que se hace referencia mediante el prefijo 's'

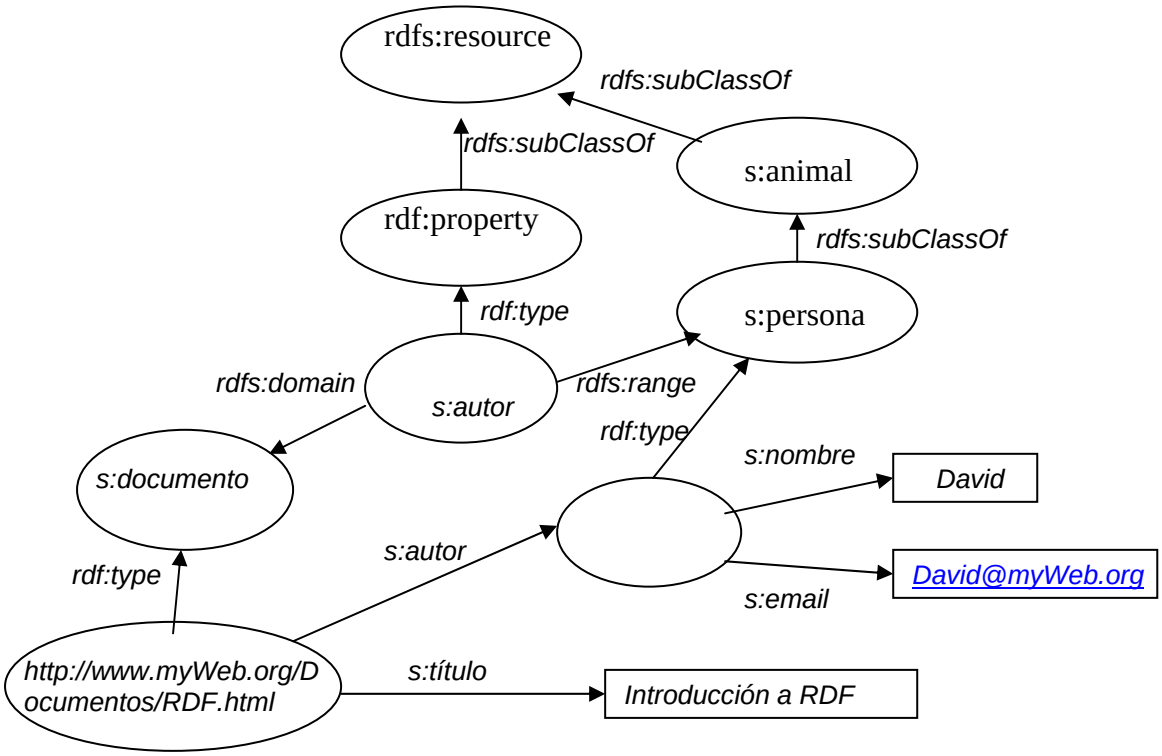

Figura 13: Grafo RDF combinando esquemas RDF

3.4- El documento RDF

Un documento RDF está formado por la declaración de los esquemas RDF a utilizar (namespaces con su correspondiente prefijo) tras lo que le siguen las descripciones formadas por la URI del sujeto (recurso) y el listado de pares predicado-objeto (propiedad-valor) que lo describen.
La siguiente figura muestra la estructura del documento RDF.

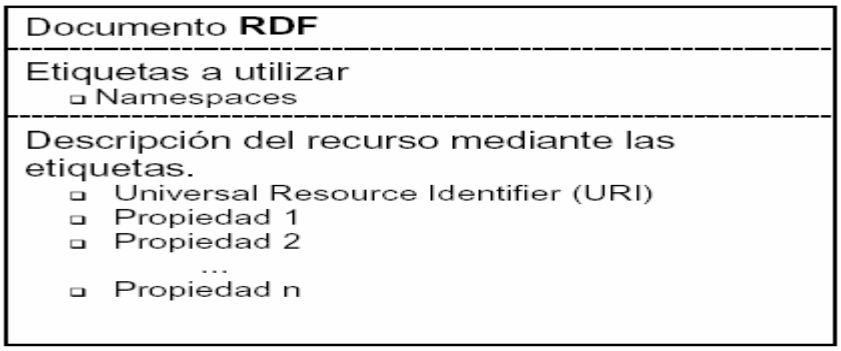

Figura 14: Estructura del documento RDF

La descripción del documento Web del anterior ejemplo se podría documentar utilizando una sintaxis ficticia así:

Declaración de namespaces:
rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs = http://www.w3.org/2000/01/rdf-schema#.
s = http://www.myWeb.org/schema#
Descripción sobre *http://www.myWeb.com/Documentos/RDF.html*
Propiedades
rdf:type= s:documento
s:título= Introducción a RDF
s:autor=
s:nombre=David
s:email=David@myWeb.org

Los esquemas RDF también son documentos formados por la declaración de los namespaces utilizados y las descripciones de los términos del vocabulario.
Este podría ser un ejemplo de parte del documento que define el esquema ficticio usado en el ejemplo anterior y con sufijo y namespace s: http://www.myWeb.org/schema#:

Declaración de namespaces:
rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs = http://www.w3.org/2000/01/rdf-schema#.
s = http://www.myWeb.org/schema#

Descripción sobre *http://www.myWeb.com/schema/animal#*

Propiedades

rdfs:subClassOf= rdfs:resource

Descripción sobre *http://www.myWeb.com/schema/persona#*

Propiedades

rdfs:subClassOf= s:animal

Descripción sobre *http://www.myWeb.com/schema/autor#*

Propiedades

rdf:type= rdf:property

rdfs:range= s:person

rdfs:domain= s:documento

s:nombre <nombre del autor>

s:email <email del autor>

3.5- Una sintaxis para RDF

El modelo de datos RDF proporciona un marco abstracto y conceptual para definir y utilizar metadatos. Necesita también una sintaxis concreta para crear e intercambiar metadatos. En este apartado se muestran algunas formas de representar el modelo de datos de RDF para su comparación y partiendo de un ejemplo con la siguiente descripción:

David es una persona

3.5.1- Grafo RDF

El grafo RDF nos permite representar la descripción usando el modelo de datos explicado anteriormente. Ofrece una forma clara y visual de representar descripciones o tripletas.

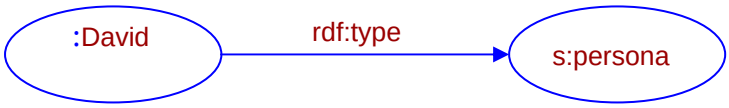

Figura 15: Grafo RDF representación ejemplo

3.5.2- Notation 3

El documento Notation 3 [13] se codifica en UTF-8 bytes [14] y su gramática está formada por sentencias que pueden ser directivas o declaraciones.

Las directivas usan el prefijo "@prefix" y proveen información del término referenciado por el documento (acceso al vocabulario mediante namespace).
La declaración expresa el dato comunicado por el documento y está formada por sujeto, predicado y objeto, y pueden hacer referencia a una URI. Sólo el objeto puede ser además un literal.

```
@prefix : < http://www.myWeb.org/David#> .
@prefix s: < http://www.myWeb.org/schema#>.
:David a s:persona;
```

Notation 3 o N3 tiene definidos algunos predicados. En este caso se usa el predicado 'a' que hace referencia a *rdf:type (*http://www.w3.org/1999/02/22-rdf-syntax-ns#type)

3.5.3- N-Triples

El W3C desarrolló N-triples [15] , un formato para una representación de RDF que está especialmente preparado para las colecciones de la prueba ante las dificultades que provoca la flexibilidad de XML como sintaxis (XML/RDF) al querer comparar resultados de procesos de testing automatizado. Los N-triples son un formato del texto plano (sucesión de caracteres de ASCII) para poner en código un gráfico de RDF. Fue diseñado para ser un subconjunto fijo de N3. Un documento de N-triples es una sucesión de triples de RDF formada del sujeto, predicado y objeto. La sintaxis de N-triples ofrece dos variantes, con prefijo o sin él (formato estándar):

Con @prefix:

Esta variante es muy semejante a la sintaxis N3, primero se declaran los esquemas a utilizar mediante prefijo y namespaces, para posteriormente mostrar las descripciones usando los prefijos declarados. La descripción también utiliza la estructura sujeto + predicado + objeto, aunque en este caso el predicado hace referencia explícita al esquema y recurso utilizado.

```
@prefix : < http://www.myWeb.org/David# > .
@prefix s: < http://www.myWeb.org/schema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
:David rdf:type s:persona;
```

En formato estándar:

El formato estándar ofrece una forma más sencilla de traducción de los grafos ya que los tripes se traducen siguiendo la siguiente estructura:

Si el objeto es un recurso
<URI sujeto><URI predicado><URI objeto>

Si el objeto es un literal
<URI sujeto><URI predicado>”objeto”

Esta sería la representación del ejemplo utilizando esta sintaxis:

<http://www.myWeb.org/David#David ><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://www.myWeb.org/schema#persona>

3.5.4- Prolog

Prolog es un lenguaje de programación lógica (de ahí su nombre). Esto significa que está basado en la lógica de predicados, en concreto en un subconjunto de esta lógica denominado cláusulas de Horn.
En Prolog, un programa es un conjunto de hechos y reglas que representan el problema que se pretende resolver. Ante una determinada pregunta sobre el problema, el Prolog utilizará estos hechos y reglas para intentar demostrar la veracidad o falsedad de la pregunta que se le ha planteado.

Los elementos fundamentales de un programa PROLOG son los siguientes:

Hechos: Expresan relaciones entre objetos, primero se escribe la relación, y luego los objetos separados por comas y encerrados entre paréntesis. Al final de un hecho debe ir un punto (".").
Variables: Representan objetos que el mismo PROLOG determina.
Reglas: Las reglas se utilizan en PROLOG para significar que un hecho depende de uno o más hechos. Son la representación de las implicaciones lógicas del tipo p ---> q (p implica q).
Su aplicación es importante en matemáticas (demostración teoremas), inteligencia artificial y consultas a bases de datos (permite inferir relaciones no especificadas a priori).

Para representar RDF existen varias formas [16].

-Realizando las descripciones como si fueran hechos utilizando la estructura *predicado(sujeto,objeto)*

El ejemplo se representaría como es(David, humano), aunque humano(David) también sería válida considerando la propiedad ‘es ‘ implícita en la estructura *objeto(sujeto)*.
También es posible utilizar los namespaces en la descripción, aquí se muestran dos formas

'http://www.w3.org/1999/02/22-rdf-syntax-ns#type'('http://www.myWeb.org/David# David ', ‘http://www.myWeb.org/schema#persona’).

ns(d_, "http://www.myWeb.org/David#David ").
ns(s_, "http://www.myWeb.org/schema#").
s_humano(d_David).

Mediante prolog también puede ser interesante distinguir los hechos RDF del resto, en tal caso se podría utilizar el siguiente formato
rdf-triple(predicado, sujeto, objeto), así quedaría el ejemplo:

```
rdf_triple('http://www.w3.org/1999/02/22-rdf-syntax-ns#type','http://www.myWeb.org/David#David',
'http://www.myWeb.org/schema#humano').
```

3.5.5- XML (no RDF/XML)

XML permite la representación del modelo de datos utilizando su expresividad e interoperabilidad sintáctica. XML permite la generación de etiquetas, por tanto podemos fácilmente representar cualquier información y estructura. Para representar la descripción del ejemplo podemos utilizar la siguiente estructura:

```
<persona>
<uri>http://www.myWeb.org/David#David</uri>
</persona>
```

En este caso se puede considerar que la etiqueta <uri> delimita la URI del recurso sujeto, y que pertenece al tipo *persona* pues está incluido dentro de su declaración. Sin embargo no hay forma de reconocer algún aspecto semántico de esta declaración ya que XML nos ayuda a estructurar el documento pero no nos dice nada de como interpretar de forma general los datos contenidos en él. Para interpretar estos datos se debe implementar una aplicación determinada que proporcione los significados de los datos.

La siguiente representación usa una DTD interna para declarar los prefijos y namespaces de los esquemas RDF y del recurso descrito. Existen otras DTDs y esquemas XML para realizar esta declaración. Tras ella se estructura el documento utilizando la etiqueta <Graph> para delimitar el grafo y dentro se incluye el triple (descripción) delimitado por la etiqueta <Triple>. Al igual que el modelo de datos el triple está formado por sujeto, predicado y objeto, y esto se indica con sus correspondientes etiquetas. También se indica mediante la etiqueta <uri> que estos elementos del modelo de datos corresponden a URIs.

```
<!DOCTYPE Graph [
 <!ENTITY rdf    "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <!ENTITY s    " http://www.myWeb.org/schema#">
 <!ENTITY d    "http://www.myWeb.org/David#">
]>
<Graph>
  <Triple>
    <subject><uri>&d;David</uri></subject>
    <predicate><uri>&rdf;type</uri></predicate>
    <object><uri>&s;persona</uri></object>
  </Triple>
</Graph>
```

Esta sintaxis XML permite fácilmente la declaración de los namespaces y la representación de los grafos y sus triples. Aun así esta representación sigue presentando una clara limitación en cuanto a la interoperabilidad semántica ya que sigue sin proporcionar una forma de acceder a los significados de sus componentes (etiquetas) por lo que se hace necesaria una aplicación específica que proporcione la semántica completa de los datos incluidos.

3.5.6- RDF/XML

La sintaxis RDF/XML es la que ofrece mayor interoperabilidad semántica pues todos los datos permiten el acceso a su significado mediante su correspondiente namespace. Además al usar la sintaxis XML hereda la buena expresividad e interoperabilidad sintáctica. Esta sería la representación del ejemplo:

```
<rdf:RDF xmlns="http://www.myWeb.org/David#"
   xmlns:s=" http://www.myWeb.org/schema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <s:persona rdf:about="#David">
   </s:persona>
</rdf:RDF>
```

4-Serialización RDF/XML

Se definen dos sintaxis XML [17] para codificar una instancia (objeto específico de una categoría) del modelo de datos. La *sintaxis serializada* expresa las capacidades totales del modelo de datos de una forma muy regular. La *sintaxis abreviada* incluye términos adicionales que proporcionan una forma más compacta para representar un subconjunto del modelo de datos. Los intérpretes de RDF se han anticipado a implementar ambas sintaxis, la serializada completa y la abreviada. Así, los autores de metadatos pueden mezclar ambas libremente.

4.1-Sintaxis serializada básica

Una sentencia RDF raramente aparece en forma aislada. Lo normal es que varias propiedades de un recurso sean indicadas simultáneamente. La sintaxis RDF/XML ha sido diseñada para permitir agrupar varias sentencias sobre un mismo recurso en un elemento "Description". El elemento "Description" menciona en un atributo "about" el recurso al que se aplican las sentencias. Si el recurso todavía no existe, el elemento "Description" puede asignarle un identificador en el momento usando un atributo ID.

Ora Lassila es el creador [autor] del recurso http://www.w3.org/Home/Lassila.

se representa en RDF/XML:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Aquí el prefijo del namespace '*s*' se refiere a un prefijo específico elegido por el autor de esta expresión RDF y definido en una declaración XML del namespace como ésta:

```
  xmlns:s="http://description.org/schema/"
```

Esta declaración del namespace podría incluirse normalmente como un atributo XML en el elemento ***rdf*:RDF** pero también puede incluirse con un elemento **Description** específico o incluso una expresión propertyElt concreta. El URI del nombre del namespace, en la declaración del namespace, es un identificador único universal para un esquema particular que la persona que define los metadatos [este autor de los metadatos] utiliza para definir el uso de la propiedad Creator. Otros esquemas pueden definir igualmente la propiedad denominada **Creator** y las dos propiedades se diferenciarán gracias a sus identificadores de esquema. Nótese también que un esquema normalmente define también varias propiedades; una única declaración de namespace será suficiente para crear un amplio vocabulario de propiedades que podrán usarse.

El documento XML completo que contiene la descripción citada anteriormente, podría ser:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

4.2- Sintaxis abreviada básica

Aunque la sintaxis serializada muestra la estructura de un modelo RDF más claro, normalmente es mejor utilizar una forma XML más compacta. Esto se lleva a cabo a través de la *sintaxis abreviada* de RDF. Como valor añadido, la sintaxis abreviada permite a los documentos seguir DTDs de XML bien estructuradas que se interpretan como modelos RDF.

Se definen tres formas de abreviación para la sintaxis básica serializada.

1ª forma de abreviación

La primera se puede utilizar para propiedades no repetidas dentro de un elemento **Description** donde los valores de dichas propiedades son literales. En este caso, las propiedades se pueden expresar como atributos XML del elemento Description. El ejemplo anterior se convierte pues en:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila"
                  s:Creator="Ora Lassila" />
</rdf:RDF>
```

Aquí se presenta otro ejemplo del uso de esta misma forma abreviada:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org">
    <s:Publisher>World Wide Web Consortium</s:Publisher>
    <s:Title>W3C Home Page</s:Title>
    <s:Date>1998-10-03T02:27</s:Date>
  </rdf:Description>
</rdf:RDF>
```

equivalente para los fines de RDF a:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org"
      s:Publisher="World Wide Web Consortium"
      s:Title="W3C Home Page"
      s:Date="1998-10-03T02:27"/>
</rdf:RDF>
```

2ª forma de abreviación

La segunda forma abreviada trabaja sobre elementos **Description**. Esta forma abreviada puede emplearse para sentencias específicas donde el objeto de la declaración es otro recurso y el valor de cualquier propiedad dada "in-line" para este segundo recurso son strings (cadenas de caracteres manipuladas como grupo). En este caso se usa una transformación similar de elementos XML que se nombran como atributos XML: las propiedades del recurso en el elemento **Description** anidado puede escribirse como atributos XML del elemento ***propertyElt*** en el que se comprende **Description**

El individuo al que se refiere el identificador de empleado id 85740 se llama Ora Lassila y tiene la dirección de correo lassila@w3.org. Ese individuo creó el recurso http://www.w3.org/Home/Lassila

se escribe en RDF/XML utilizando la forma serializada explícita como:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
```

```
    <rdf:Description about="http://www.w3.org/staffId/85740">
      <v:Name>Ora Lassila</v:Name>
      <v:Email>lassila@w3.org</v:Email>
    </rdf:Description>
  </s:Creator>
 </rdf:Description>
</rdf:RDF>
```

Utilizando esta segunda sintaxis abreviada, el elemento interior **Description** y las expresiones de las propiedades que contiene pueden escribirse como atributos del elemento **Creator**:

```
<rdf:RDF>
 <rdf:Description about="http://www.w3.org/Home/Lassila">
  <s:Creator rdf:resource="http://www.w3.org/staffId/85740"
    v:Name="Ora Lassila"
    v:Email="lassila@w3.org" />
 </rdf:Description>
</rdf:RDF>
```

3ª forma de abreviación

La tercera forma de sintaxis abreviada básica se aplica a el caso común de un elemento **Description** que contenga un propiedad **type**. En este caso, el tipo de recurso definido en el esquema que corresponde al valor de la propiedad **type** puede utilizarse directamente como un nombre de elemento. Por ejemplo, utilizando el fragmento RDF anterior si queremos añadir que el recurso http://www.w3.org/staffId/85740 representa una instancia [objeto específico de la categoría] de una persona, podríamos escribirlo en una sintaxis serializada completa, así:

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:s="http://description.org/schema/">
 <rdf:Description about="http://www.w3.org/Home/Lassila">
  <s:Creator>
   <rdf:Description about="http://www.w3.org/staffId/85740">
    <rdf:type resource="http://description.org/schema/Person"/>
    <v:Name>Ora Lassila</v:Name>
    <v:Email>lassila@w3.org</v:Email>
   </rdf:Description>
  </s:Creator>
 </rdf:Description>
</rdf:RDF>
```

y utilizando la tercera forma abreviada, así:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <s:Person about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
      </s:Person>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>
```

5- Aplicaciones con XML y RDF

En este apartado se muestran algunas aplicaciones que aprovechan las capacidades de XML y RDF de forma conjunta o por separado.

Aplicaciones con XML

Entre las capacidades de XML destacan por un lado la estructuración del contenido en un documento XML y su separación de la presentación, y por otro la posibilidad de definir gramáticas (DTD o XML Schema) que permiten que un parser valide un documento XML. Aprovechando estas capacidades se muestran dos ámbitos de aplicación de XML:

Creación de Webs y documentos electrónicos

La estructuración que ofrece XML es interesante para la creación Web y documentos electrónicos ya que en ambos casos la estructuración contenidos facilita su localización por parte de personas o aplicaciones como los motores de búsqueda en Internet. También es muy útil la separación del contenido de la presentación pues facilita de forma sustancial el mantenimiento de una Web ya que se puede remodelar totalmente su presentación simplemente aplicándole diferentes hojas de estilo sin necesidad de tocar su contenido. Esto también permite una mayor independencia entre los desarrolladores de contenido y los diseñadores gráficos. Otro beneficio de la aplicación de hojas de estilo es la de poder obtener fácilmente distintos formatos de salida de un mismo documento XML, así por ejemplo una tienda virtual puede tener su catálogo en un documento XML y mediante distintas hojas de estilo transformarlo en HTML para su visionado Web, en PDF para su impresión o en otros formatos compatibles para visualizar por teléfono móvil, etc.

Comunicación entre aplicaciones

XML permite la comunicación entre aplicaciones y sistemas heterogéneos gracias a su capacidad para codificar cualquier estructura de documento y a la definición de gramáticas (DTD o XML Schema) para su validación. Esto posibilita que se puedan exportar datos desde cualquier de aplicación a un documento XML. Este documento será validado mediante una gramática que verifique que los datos del documento XML sean adecuados tanto en contenido como en estructura para ser importados a otra aplicación. Más concretamente dos empresas pueden definir una gramática común para la gestión de pedidos independientemente de sus aplicaciones y sistemas. Así la empresa demandante exporta los datos desde su aplicación de pedido a un documento XML que una vez validado será recibido por la segunda empresa que importará los datos del pedido desde el documento XML a su aplicación de pedidos. De esta misma forma una empresa puede comunicar sus aplicaciones internas de forma que puede extraer los datos de una base de datos y transportarlos a su aplicación de contabilidad, gestión, etc. mediante el uso de documentos XML y gramáticas los validen. Esto permite la automatización de sus procesos de negocio con el consecuente aumento de agilidad y la disminución de errores.

Aplicaciones con RDF

La capacidad de RDF para la descripción de recursos lo hace idóneo para la recuperación de información en la red. Mediante RDF se pueden describir las Webs y documentos electrónicos de forma que faciliten su acceso a los motores de búsqueda. Así si los recursos se describen correctamente los motores de búsqueda trabajarán de forma mucho más eficiente ya que en lugar de buscar las coincidencias dentro de los documentos podrán acceder directamente a su descripción y localizar aquí los conceptos buscados. Además la semántica que aportan los vocabularios (esquemas RDF) permite que se tengan en cuenta los significados de los conceptos y poder establecer correspondencias entre ellos. De esta forma si en un motor de búsqueda basado en búsquedas sobre descripciones RDF se le indica que se requiere información sobre una marca X de coches, simplemente escribiendo en el buscador “coche X”, el buscador podrá saber accediendo a un vocabulario RDF que queremos información de esta marca X de coches, pero que puede estar descrita como “coche”, “automóvil”, “turismo”, etc. por tanto puede ser interesante tener en cuenta las distintas variantes de un mismo concepto.

También la descripción de los documentos y Webs permiten su aprovechamiento por parte de los navegadores a la hora de determinar si en contenido al que pretende acceder un menor es adecuado o se le debe impedir su acceso. Estas son por tanto algunas de las múltiples posibilidades que permiten las descripciones de recursos de la red mediante RDF.

En ámbitos más concretos y controlables podemos usar RDF para la catalogación describiendo el contenido y las relaciones de contenido dentro de una Web, una biblioteca digital, una Intranet corporativa, etc. En estos ámbitos se puede establecer un vocabulario que abarque todos sus términos más significativos y utilizar aplicaciones que se basen en el vocabulario para realizar la catalogación. Por

ejemplo una biblioteca digital puede tener catalogados todos sus documentos mediante descripciones RDF que incluyan autor, título, temática, etc., y definir el vocabulario correspondiente donde se definan los términos y sus relaciones. Así se puede establecer que las temáticas pueden ser científica, histórica, etc., y dentro de estas temáticas indicar subgrupos como matemáticas, biología, etc.

Aplicaciones con XML/RDF

Con XML/RDF se aprovechan las capacidades de ambas tecnologías, partiendo de los ejemplos anteriores se pueden diseñar Webs y documentos electrónicos con XML/RDF donde sea interesante aprovechar la estructuración y la separación de la presentación que ofrece XML y a su vez aprovechar el aporte semántico de RDF en sus contenidos. Por otro lado una empresa puede tener su catálogo de productos con XML/RDF pudiendo describir estos productos gracias a RDF y a su vez transportar los datos a otras aplicaciones y sistemas aprovechando las ventajas de XML.

6- Ejemplos prácticos

En el siguiente apartado se muestran una serie de ejemplos prácticos de los conceptos tratados. Se trata de crear un documento con enlaces web y la descripción de algunos de sus elementos. Se utiliza la dirección ficticia www.myWeb.org para referenciar los distintos documentos.

Ejemplo 1:
Este ejemplo se compone de un documento xml donde se define la descripción de webs y un documento dtd para validar la gramática del documento xml.

webs.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE note SYSTEM "http://www.myWeb.org/dtd/Web.dtd">
<DescripcionDeWebs>
  <Web>
        <nombreWeb>W3C Wordl Wide Web consortium</nombreWeb>
        <url>http://www.w3.org</url>
        <creador>w3c</creador>
        <actualizacion>alta</actualizacion>
        <valoracion>buena</valoracion>
  </Web>
  <Web>
        <nombreWeb>My web</nombreWeb>
        <url>http://www.myWeb.org</url>
```

```
        <creador>David</creador>
<creador>Juan</creador>
        <actualizacion>baja</actualizacion>
        <valoracion>regular</valoracion>
  </Web>
</DescripcionDeWebs>
```

webs.dtd

```
<!ELEMENT DescripcionDeWebs (Web+)>
<!ELEMENT Web (creador*,actualizacion?,valoracion?)>
<!ATTLIST Web nombreWeb CDATA #REQUIRED
              url CDATA #REQUIRED>
<!ELEMENT creador (#PCDATA)>
<!ELEMENT actualizacion (alta|baja)>
<!ELEMENT valoracion (buena|regular|mala)>
```

Ejemplo 2:

Este ejemplo se compone de un documento xml donde se define la descripción de webs y un documento xsd (xml schema) para validar la gramática del documento xml.

webs.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<DescripcionDeWebs xmlns="http://www.myWeb.org"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.myWeb.org webs.xsd">
  <Web>
        <nombreWeb>W3C Wordl Wide Web consortium</nombreWeb>
        <url>http://www.w3.org</url>
        <creador>w3c</creador>
        <actualizacion>alta</actualizacion>
        <valoracion>buena</valoracion>
  </Web>
  <Web>
        <nombreWeb>My web</nombreWeb>
        <url>http://www.myWeb.org</url>
        <creador>David</creador>
```

```
<creador>Juan</creador>
        <actualizacion>baja</actualizacion>
        <valoracion>regular</valoracion>
  </Web>
</DescripcionDeWebs>
```

webs.xsd

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- Defino un tipo especial para el atributo de tipo actualizacion -->
  <!—La actualización sólo puede se alta o baja -->
  <xsd:simpleType name="tactualizacion">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="alta"/>
      <xsd:enumeration value="baja"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- Defino un tipo especial para el atributo de tipo valoracion -->
  <!—La actualización sólo puede se buena, regular  o mala -->
  <xsd:simpleType name="tvaloracion">
    <xsd:restriction base="xsd:string">
     <xsd:enumeration value="buena"/>
     <xsd:enumeration value="regular"/>
     <xsd:enumeration value="mala"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- Defino un tipo especial para los elementos de clase web -->
  <xsd:complexType name="tweb">
    <xsd:sequence>
      <xsd:element ref="creador" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="actualizacion"/>
      <xsd:element ref="valoracion"/>
    </xsd:sequence>
    <xsd:attribute name="nombreWeb"    type="xsd:string"     use="required"/>
    <xsd:attribute name="url"   type="xsd:anyURI" use="required"/>    "/>
```

```
 </xsd:complexType>

 <!-- Definición de los elementos teléfono, correo y nombre -->
  <xsd:element name="creador" type="xsd:string"/>
 <xsd:element name="actualizacion"   type="xsd:tactualizacion"/>
 <xsd:element name="valoracion"   type="xsd:tvaloracon"/>

 <!-- Definición del elemento de tipo descripcionDeWebs -->
 <xsd:element name=" descripcionDeWebs ">
   <xsd:complexType>
     <xsd:sequence>
          <xsd:element name="Web" type="tWeb" minOccurs="1" maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Ejemplo 3:

Este ejemplo se compone de un documento rdf/xml donde se define la descripción de webs y un esquema rdf donde se obtienen los significados de los elementos del documento rdf/xml.

webs.xml

```
<?xml version="1.0" encoding="ISO-8859-1">
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://www.myWeb.org/schema#">
  <rdf:Description about=" http://www.w3.org"
    s:nombreWeb=” W3C Wordl Wide Web consortium”
    s:creador= “w3c”
          s:actualizacion=”alta”
          s:valoración=”buena”
</rdf:Description>
  <rdf:Description about=" http://www.myWeb.org"
    s:nombreWeb=”myWeb”
    s:creador= “David”
    s:creador= “Juan”
          s:actualizacion=”baja”
```

```
        s:valoración=”regular”
</rdf:Description>
</rdf:RDF>
```

rdf-schema

```
<rdf:RDF xml:lang="es"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<!—Mi primer esquema rdf-->
<rdf:Description ID="Animal">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
         <rdfs:subClassOf  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
        <rdfs:comment>Clase para animal</rdfs:comment>
</rdf:Description >
<rdf:Description ID="Persona">
         <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
         <rdfs:subClassOf rdf:resource="#Animal"/>
        <rdfs:comment>Clase para persona</rdfs:comment>
</rdf:Description >
<rdf:Description ID="Creador">
         <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
         <rdfs:subClassOf rdf:resource="#Persona"/>
        <rdfs:comment>Clase para creador</rdfs:comment>
</rdf:Description >
<rdf:Description ID="Web">
         <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:comment>Clase para Web</rdfs:comment>
</rdf:Description >
<rdf:Description rdf:ID="nombre">
         <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
         <rdfs:comment>Nombre</rdfs:comment>
         <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
  </ rdf:Description>

<rdf:Description rdf:ID="nombreWeb">
         <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
         <rdfs:comment>Nombre para webs</rdfs:comment>
         <rdfs:domain rdf:resource="#Web"/>
```

```
        <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
  </ rdf:Description>
<rdf:Description rdf:ID="valoracion">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:comment>valoracion</rdfs:comment>
        <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
</ rdf:Description>
<rdf:Description rdf:ID="actualizacion">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:comment>actualizacion</rdfs:comment>
        <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
</ rdf:Description>
```

Conclusiones

Internet es hoy en día el medio de acceso a la información más importante en los países desarrollados. Su crecimiento y desarrollo avanza de forma imparable, y cada vez más usuarios y empresas utilizan la red para sus tareas. En definitiva Internet se está convirtiendo en la plataforma predominante de comunicación y de acceso a la información. Sin embargo el HTML como lenguaje para Internet se está viendo limitado para ofrecer soporte a este constante crecimiento. Por ello se hace necesaria una forma de expresar los contenidos de Internet de manera flexible, que se adapte a la multitud de plataformas y funcionalidades. En esto XML cumple claramente con los objetivos propuestos. Un metalenguaje que permite definir diferentes lenguajes y estructuras adaptadas a cualquier ámbito. Su expresividad e interoperabilidad sintáctica son sus mayores avales, además de separar el contenido de su presentación. Sin embargo presenta deficiencias en cuanto a la descripción de recursos (metadatos). Aquí es donde entra en escena RDF aportando un modelo de datos adecuado a las descripciones de forma flexible y escalable. Además permite la creación de vocabularios mediante la definición de esquemas lo que aporta una mayor interoperabilidad semántica. Así la unión de ambas tecnologías, la representación XML/RDF, se beneficia de sus capacidades sintácticas y semánticas para ofrecer un modelo de representación que permita una mejor recuperación y procesamiento de la información en Internet y una mayor integración entre aplicaciones.

Referencias

[1] Charles F. Goldfarb. *The Roots of SGML -- A Personal Recollection.* (1996) [http://www.sgmlsource.com/history/roots.htm, 10 de octubre de 2003]

[2] Martin Bryan. *An Introduction to the Standard Generalized Markup Language (SGML).* (1992) [http://xml.coverpages.org/bryanIntro1992.html, 10 de octubre 2003]

[3] Tim Berners-Lee. *Hypertext Markup Language (HTML)* (1993)

[http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt, 15 de octubre de 2003]

[4] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E. *Extensible Markup Language (XML) 1.0 (Second Edition)* (2000) [http://www.w3.org/TR/REC-xml, 20 de octubre de 2003]

[5] David C. Fallside. *XML Schema Part 0: Primer* (2001) [http://www.w3.org/TR/xmlschema-0, 22 de octubre de 2003]

[6] Tim Bray ,Dave Hollander, Andrew Layman. *Namespaces in XML* (1999) [http://www.w3.org/TR/REC-xml-names, 22 de octubre de 2003]

[7] Dave Raggett. *Adding a touch of style* (2002) [http://www.w3.org/MarkUp/Guide/Style, 25 de octubre de 2003]

[8] Sharon Adler[et al.] *Extensible Stylesheet Language (XSL) Version 1.0* (2001). [http://www.w3.org/TR/xsl/, 25 de octubre de 2003]

[9] Tim Bray. *What is RDF?* (2001) [http://www.xml.com/pub/a/2001/01/24/rdf.html, 26 de octubre de 2003]
[10] Tim Berners-Lee. *Why RDF model is different from the XML model* (1998). [http://www.w3.org/DesignIssues/RDF-XML.html, 26 de octubre de 2003]

[11] Frank Manola, Eric Millar. *RDF Primer* (2003) [http://www.w3.org/TR/rdf-primer, 30 de octubre de 2003]

[12] Dan Brickley, R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema* (2003) [http://www.w3.org/TR/rdf-schema, 26 de octubre de 2003]

[13] Tim Berners-Lee, Dan Connolly, Sandro Hawke. *Getting into RDF and the Semantic Web using N3 (* (2003) [http://www.w3.org/2000/10/swap/Primer.html, 10de noviembre 2003]

[14] Tim Berners-Lee. *Notation 3* (1998) [http://www.w3.org/DesignIssues/Notation3, 10 de noviembre de 2003]

[15] Dave Beckett. *N-Triples* (2001) [http://www.w3.org/2001/sw/RDFCore/ntriples, 10 de noviembre de 2003]

[16] Bijan Parsia. *An Introduction to Prolog and RDF* (2001) [http://www.xml.com/pub/a/2001/04/25/prologrdf/index.html, 20 de noviembre de 2003]

[17] Dave Beckett. RDF/XML Syntax Specification (Revised) (2003) [http://www.w3.org/TR/rdf-syntax-grammar, 15 de noviembre de 2003]