4.2. XML.

Este lenguaje de marcado apareció impulsado por el World Wide Web Consortium (W3C) a finales de los años 90, y se puede decir que ha supuesto una revolución en el mundo de la informática, ya que está siendo aplicado en multitud de campos y para diversos fines[1]. Por citar sólo algunos: archivo electrónico y gestión de contenidos, publicación Web, intercambio electrónico de documentos, formato interno de herramientas, software, comercio electrónico, educación, química, sanidad, turismo, derecho, etc. XML es un estándar que nació dentro del mundo Web para representar datos que van a circular por la red.

Cuando Tim Berners Lee se planteó la idea del World Wide Web tenía la necesidad de un lenguaje en el que poder expresar los contenidos que se iban a distribuir por la red por lo que tomó como base SGML, que es un lenguaje muy general, y lo adaptó a sus necesidades dando lugar al HTML que ha constituido un éxito impresionante. Dicho éxito ha significado un crecimiento enorme y desordenado, incorporando cada vez más capacidades para tratar de cubrir las necesidades de representación de la información de innumerables colectivos con requisitos muy diversos: matemáticos (ecuaciones, integrales quebrados, etc.), químicos (fórmulas), músicos (partituras), etc. Esto hizo que se volviera a los orígenes, al SGML, para intentar buscar un remedio; pero SGML es un lenguaje complejo, con manuales complicados y que además no tiene en cuenta las últimas teorías de lenguajes formales y autómatas, es decir, no tiene en cuenta el estado del arte para los procesadores de lenguajes [Sánchez,02]. Se buscaba una versión simplificada de SGML, que permitiera definir lenguajes concretos de una forma flexible, y que permitiera definir procesadores para esos lenguajes fácilmente, y así surgió XML. Fue desarrollado por un Grupo de Trabajo de XML (originalmente conocido como el comité de revisión editorial de SGML) formado bajo el auspicio del World Wide Web Consortium (W3C) en 1996. Estaba presidido por Jon Bosak de Sun Microsystems con la participación activa de un Grupo Especial de Interés en XML (previamente

[1] Cover pages. Disponible en: http://xml.coverpages.org/xml.html (25/06/04)

conocido como el grupo de trabajo de SGML) también organizado por el W3C.

Ahora un matemático, un químico, o un músico pueden definir su propio lenguaje de marcas (siguiendo un conjunto breve de reglas simples) que cubra las necesidades de representación del contenido que quiere transmitir, y acompañarlo de un archivo DTD (Document Type Definition), notación estandarizada por el W3C el 10 de Febrero de 1998, o alternativamente, de un archivo XML Schema, notación estandarizada en Mayo de 2001, que es mucho más expresiva, potente y completa para especificar esos lenguajes concretos que usa cada colectivo o editor.

De una manera algo más formal XML consiste en una serie de reglas, pautas o convenciones, para planificar formatos texto para mostrar datos, de manera que produzcan archivos que sean fácilmente generados y leídos (por un ordenador) que sean inequívocos, y que evitan escollos comunes como la falta de extensibilidad, falta de soporte para la internacionalización o localismo, y la dependencia de una determinada plataforma [Gutiérrez,99].

4.2.1. Documentos XML.

Un objeto de datos es un Documento XML si es bien formado, tal y como es definido en esta especificación [Bray,98]. Un documento XML bien formado puede ser adicionalmente válido si cumple con algunas restricciones adicionales.

Cada documento XML tiene una estructura tanto lógica como física. Físicamente, el documento está compuesto de unidades llamadas entidades. Una entidad puede referirse a otras entidades con el fin de causar su inclusión en el documento. Un documento comienza en una "raíz" o entidad documento. Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencias de carácter e instrucciones de proceso. Todo lo anterior está indicado en el documento mediante marcas explícitas. Las estructuras lógicas y físicas deben anidarse apropiadamente.

Según la especificación un documento XML es bien formado si cumple las siguientes reglas:

- Tomado como un todo, cumple la regla denominada "document".

- Respeta todas las restricciones de buena formación dadas en la especificación.
- Cada una de las entidades analizadas que se referencia directa o indirectamente en el documento está bien formada.

Cumplir la regla "document" antes mencionada significa:

- Que contiene uno o más elementos (Regla nº1).
- Hay exactamente un elemento, llamado raíz, o elemento documento, del cual ninguna parte aparece en el contenido de ningún otro elemento (Regla nº 2).
- Para el resto de elementos, si la etiqueta de comienzo está en el contenido de algún otro elemento, la etiqueta de fin está en el contenido del mismo elemento. Es decir, los elementos delimitados por etiquetas de principio y final se anidan adecuadamente mutuamente (Regla nº3).

El siguiente ejemplo no es un documento XML bien formado:

Mi primer documento XML

ya que no contiene ningún elemento y, por tanto, está incumpliendo la regla número 1.

En cambio:

```
<p>Mi primer documento XML</p>
```

sí que lo es, al contener al menos el elemento "p". La principal razón por la que el procesador comprueba los elementos es para determinar si el documento tiene estructura de datos que pueda extraer. Un documento que carece de elementos no tiene estructura de datos. Un documento con al menos un elemento tiene estructura de datos.

Sin embargo:

```
<p>Mi primer documento XML</p>
<p>Mi primer documento XML</p>
```

no es un documento XML bien formado al incumplir la regla número 2, según la cual sólo puede existir un único elemento raíz.

Aunque escrito de la siguiente manera si que es correcto:

```
<documento>
        <p>Mi primer documento XML</p>
        <p>Mi primer documento XML</p>
</documento>
```

al convertirse el elemento "documento" en el elemento raíz, ser único y no formar parte del contenido de ningún otro elemento.

En cambio, el siguiente ejemplo:

```
<documento>
        <p>Mi primer <destacar>documento XML</p></destacar>
        <p>Mi primer documento XML</p>
</documento>
```

es incorrecto al incumplir la regla 3, ya que la etiqueta inicio del elemento "destacar" está dentro del contenido del elemento "p", pero su etiqueta final está fuera.

La forma correcta sería la siguiente:

```
<documento>
        <p>Mi primer <destacar>documento XML</destacar></p>
        <p>Mi primer documento XML</p>
</documento>
```

4.2.2. Sintaxis XML.

Además de las reglas anteriormente mencionadas, para escribir documentos XML bien formados, tenemos que conocer perfectamente la sintaxis del lenguaje XML y algunas restricciones que la especificación impone.

Es como en cualquier lenguaje: tenemos que conocer la sintaxis de cómo se escriben los elementos, atributos y entidades, y si las incumplimos el procesador del lenguaje o "parser" dará un error de mala formación. Como no se trata de escribir ahora un manual de XML vamos a ver mediante un ejemplo[2] algunas de las reglas sintácticas. Supongamos el siguiente texto:

```
<?xml version="1.0"?>
<documento>
   <p>Mi Primer <destacar importancia=1>documento XML</destacar></p]
   <p>Comienza con la etiqueta <documento&gt;</p>
   <p>A continuacion colocamos un elemento sin contenido</p>
   <imagen fichero="imagen.gif">
</documento>
```

[2] Bravo, J. Manual de XML. Disponible en: http://www.programación.net/html/xml (25/06/04)

Se trata de un documento XML que es, simplemente, un conjunto de cadenas de caracteres, en el que, al igual que en el HTML, podemos diferenciar dos tipos de construcciones: el marcado y los datos de carácter.
El texto incluido entre los caracteres menor que "<" y mayor que ">" o entre los signos "&" y ";" es el marcado. Son exactamente las partes del documento que tiene que entender el procesador de XML.
El marcado entre los signos "<" y ">" se denominan etiqueta.El resto no son más que datos de carácter, que se corresponde con lo que sería el contenido del documento: es decir, la parte imprimible de éste.Si un parser procesara dicho documento informaría, al menos, de 4 errores:
El valor del atributo "importancia", no está entrecomillado. En HTML es posible no entrecomillar el valor de los atributos, pero en XML es obligatorio. Tendríamos que haber escrito:

```
...<destacar importancia="1">documento XML</destacar>...
```

La etiqueta final, del elemento "p" está mal cerrada. En lugar del carácter "]" , tendríamos que poner el símbolo mayor que ">".

```
<p>Mi Primer <destacar importancia=”1”>documento XML</destacar></p>
```

Estamos utilizando el símbolo menor que "<" sin que forme parte de la definición de una etiqueta. Al ser un carácter reservado, tendríamos que escribirlo como la entidad predefinida <.

```
<p>Comienza con la etiqueta &lt;documento&gt;</p>
```

Estamos escribiendo el elemento vacío "imagen" de forma incorrecta. Al ser un elemento sin contenido tendríamos que haberlo escrito con una etiqueta de elemento vacío:

```
<imagen fichero="imagen.gif"/>
```

o también de la siguiente manera:

```
<imagen fichero="imagen.gif"></imagen>
```

Ambas son correctas, aunque recomendable la primera.
Por tanto, para que el ejemplo anterior corresponda a un documento XML bien formado, debería escribirse así:

```
<?xml version="1.0"?>
<documento>
  <p>Mi Primer <destacar importancia="1">documento XML</destacar></p>
  <p>Comienza con la etiqueta &lt;documento&gt;</p>
  <p>A continuacion colocamos un elemento sin contenido</p>
```

```
    <imagen fichero="imagen.gif"/>
</documento>
```

Otras reglas que debemos tener en cuenta son:

XML es sensible a la utilización de mayúsculas y minúsculas.

En el siguiente ejemplo:

```
<p>Mi primer documento XML</p>
<P>Mi primer documento XML</P>
```

los elementos "p" y "P" son diferentes.

Hay que tener mucho cuidado con esta regla, ya que su incumplimiento es habitual y suele ser la causa de la mayor parte de los errores. Es recomendable antes de empezar a escribir un documento XML establecer un criterio al respecto.

El nombre de la etiqueta de inicio y final debe ser el mismo.

El siguiente ejemplo es incorrecto.

```
<p>Mi primer documento XML</P>
```

ya que al hacer diferencia entre mayúsculas y minúsculas, el parser no entiende ambas etiquetas como del mismo elemento.

Ningún nombre de atributo puede aparecer más de una vez en la misma etiqueta de inicio o de elemento vacío.

El siguiente ejemplo es incorrecto:

```
...<destacar importancia="1" importancia="2">documento XML</destacar>...
```

4.2.3. Extensiones a XML.

El éxito de XML ha originado que se demanden nuevas funcionalidades [Ruiz,03], que se abordan definiendo extensiones adicionales para:

- **Estructurar documentos (XML Schema)**: un documento XML estructura los datos mediante un modelo jerárquico que representa un determinado esquema semántico a través de los elementos y atributos convenientes.

El aspecto que tiene un fichero XMLS es:

```
<schema targetNamespace=”http://www.bd.es/schema”
xmlns=”http://www.w3.org/2001/XMLSchema”
xmlns:bd=”http://www.bd.es/schema” >
<element name=”Articulo” type=”bd:tArticulo” />
<complexType name=”tArticulo”>
<element name=”Cabecera” type=”bd:tCabecera”/>
```

```
<element name="Cuerpo" type="bd:tCuerpo"/>
<element name="Final" type="bd:tFinal"/>
</complexType>
<complexType name="tCabecera">
<element name="Titulo" type="string"/>
<element name="Autor" type="string"/>
</complexType>
.....
```

- **Enlaces y direccionamiento (Xpath, Xlink, Xpointer).**

XML Path Language (XPath) es un lenguaje declarativo para localizar nodos y fragmentos (texto, elementos, atributos ...) en el árbol de un documento XML. Es utilizado por otras normas para direccionamiento (XLink, XPointer y XSLT) y "Pattern matching" (XSLT y XQuery). Se basa en el XPath Data Model para el que un documento XML se representa como un árbol jerárquico con siete tipos de nodos (raíz, elemento, texto, atributo, espacio de nombres, instrucción de procesamiento y comentario).

XPath en sí es un lenguaje sofisticado y complejo, pero distinto de los lenguajes procedurales que se suelen usar (C, C++, Basic, Java...). Además, como casi todo en el mundo de XML, aún está en estado de desarrollo, por lo que no es fácil encontrar herramientas que incorporen todas sus funcionalidades.

XPath es a su vez la base sobre la que se han especificado nuevas herramientas que aprovehcar para el tratamiento de documentos XML. Herramientas tales como XPointer, XLink y XQL (el lenguaje que maneja los documentos XML como si de una base de datos se tratase), que también están en estado de desarrollo, pero que sin duda cambiarán el modo en que actualmente concebimos la navegación por la Web. Así, XPath sirve para decir cómo debe procesar una hoja de estilo el contenido de una página XML, pero también para poder poner enlaces o cargar en un navegador zonas determinadas de una página XML, en vez de toda la página[3].

Un documento XML es procesado por un analizador (o *parser*) construyendo un árbol de nodos. Este árbol comienza con un elemento raíz, que se diversifica a lo largo de los elementos que cuelgan de él y acaba en nodos hoja, que contienen solo texto, comentarios, instrucciones de proceso o

[3] Rivas Santos, V. http://geneura.ugr.es/~victor/cursillos/xml/XPath/

incluso que están vacíos y solo tienen atributos. La forma en que XPath selecciona partes del documento XML se basa precisamente en la representación arbórea que se genera del documento. De hecho, los "operadores" de que consta este lenguaje recuerdan la terminología que se utiliza a la hora de hablar de árboles en informática: raíz, hijo, ancestro, descendiente, etc...

Un caso especial de nodo son los nodos atributo. Un nodo puede tener tantos atributos como desee, y para cada uno se le creará un nodo atributo. No obstante, dichos nodos atributo NO se consideran como hijos suyos, sino más bien como etiquetas añadidas al nodo elemento.

A continuación se muestra un ejemplo de cómo se convierte en árbol un documento XML. En primer lugar se muestra el documento XML y a continuación el árbol que genera.

```
<libro>
  <titulo>Dos por tres calles</titulo>
  <autor>Josefa Santos</autor>
  <capitulo num="1">
    La primera calle
    <parrafo>
    Era una sombría noche del mes de agosto...
     </parrafo>
    <parrafo destacar="si">
    Ella, inocente cual
       <enlace href="http://www.enlace.es">mariposa</enlace>
       que surca el cielo en busca de libaciones...
    </parrafo>
  </capitulo>
  <capitulo num="2" public="si">
    La segunda calle
    <parrafo>Era una obscura noche del mes de septiembre...</parrafo>
    <parrafo>
    Ella, inocente cual
    <enlace href="http://www.abejilla.es">abejilla</enlace>
    que surca el viento en busca del néctar de las flores...
    </parrafo>
  </capitulo>
  <apendice num="a" public="si">
    La tercera calle
    <parrafo>
    Era una densa noche del mes de diciembre...
    </parrafo>
```

```
    <parrafo>
    Ella, cándida cual
    <enlace href="http://www.pajarillo.es">abejilla</enlace>
    que surca el espacio en busca de bichejos para comer...
    </parrafo>
  </apendice>
</libro>
```

Y el árbol generado es:

```
/
|
+---libro
    |
    +---titulo
    |   |
    |   +---(texto)Dos por tres calles
    |
    +---autor
    |   |
    |   +---(texto)Josefa Santos
    |
    +---capitulo [num=1]
    |   |
    |   +---(texto)La primera calle
    |   |
    |   +---parrafo
    |   |   |
    |   |   +---(texto)Era una sombría noche ...
    |   +---parrafo
    |       |
    |       +---(texto)Ella, cual inocente mariposa...
    |
    +---capitulo [num=2]
        |
        +---(texto)La segunda calle
        |
        +---parrafo
        |   |
        |   +---(texto)Era una obscura noche ...
        +---parrafo
            |
            +---(texto)Ella, cual inocente abeja...
```

Como ya hemos comentado existen distintos tipos de nodos en un árbol generado a partir de un documento XML, a saber: raíz, elemento, atributo,

texto, comentario e instrucción de procesamiento (respectivamente; *root, elements, attribute, text, comment y processing instruction*).

Una instrucción en lenguaje XPath se denomina expresión (XPATH no es un lenguaje declarativo, por lo que el término instrucción no tiene el sentido de otros lenguajes de programación).

Un *location path* es la más importante de los tipos de expresiones que se pueden especificar en notación XPath. La sintaxis de un *location path* es similar a la usada a la hora de describir los directorios que forman una unidad de disco en Unix o Linux (y similar a la de los sistemas basados en MS-DOS y Windows, si exceptuamos la unidad de disco -C:, A:- y que las barras usadas son / en vez de las típicas \ de estos últimos sistemas operativos).

Sin embargo, solo la sintaxis es lo similar al sistema de archivos. El significado de las expresiones es totalmente diferente.

Por ejemplo, el siguiente *path* en Unix:

```
/usr/home/pepeillo/docs
```

hace referencia a un único directorio: docs el cual cuelga de el conjunto de directorios /usr/home/pepeillo.

Sin embargo, la siguiente expresión en XPath:

```
/libro/capitulo/parrafo
```

hace referencia a TODOS los elementos "parrafo" que cuelguen directamente de CUALQUIER elemento capitulo que cuelgue de CUALQUIER elemento libro que, finalmente, cuelguen del nodo raíz, /. Hay que tener en cuenta que una expresión en XPath no devuelve los elementos que cumplen con el patrón que representa dicha expresión, sino que devuelve una referencia a dichos elementos; es decir, una expresión XPath nos devuelve una lista de apuntadores a los elementos que encajan en el patrón. Dicha lista puede estar vacía o contener uno o más nodos.

Más ejemplos de XPATH:

- Seleccionar nombres de ingredientes de receta que se utiliza media taza:

```
//ingrediente[@cantidad='0.5' and @unidad=taza]/@nombre
```

- Seleccionar todos los capítulos públicos que tengan algún párrafo que contenga algún elemento con atributo href:

```
//capitulo[parrafo/*[@href]][@public='si']
```

XPointer describe cómo se puede apuntar a un lugar específico de un determinado documento XML. Es una extensión de XPath que permite asociar a una dirección URI con una expresión XPath con algunas propiedades extras.

```
http://www.sitio.es/doc.xml#xpointer( /libro/capitulo[@public] )
```

XLink (XML Linking Language) define la forma en la que los documentos XML se pueden relacionar entre sí definiendo nuevos tipos de elementos XML que representan enlaces (links).

- Transformación y presentación (familia XSL, CSS2).

XSL (eXtensible Stylesheet Language) no sólo permite definir el estilo a aplicar a cada elemento XML. También es un lenguaje de programación para transformar documentos XML. El resultado puede ser un documento HTML, WML (para WAP), texto plano, RTF, PDF, o incluso otro documento XML. Una hoja de estilo XSL es una serie de reglas que determinan como va a ocurrir la transformación.Cada regla se compone de un patrón de localización (pattern) y una plantilla (template). Por ejemplo:

```
<xsl:template match="/">
<HTML>
<BODY>
<xsl:for-each select="/LIBROS/LIBRO">
Título:
<xsl:value-of select="TITULO"/><BR/>
Autor:
<xsl:value-of select="AUTOR"/><BR/>
Precio:
<xsl:value-of select="PRECIO"/> pesetas<BR/>
</xsl:for-each>
</BODY>
</HTML>
</xsl:template>
```

- **Consultas (XQUERY).**

XQuery proporciona un modo flexible de consulta para extraer datos de los documentos XML. Los archivos XML pueden ser reales o virtuales, es decir, otras fuentes (hojas de cálculo, ASCII, bases de datos, ...) . Se pretende que desempeñe un papel similar al SQL en las BD relacionales: las colecciones de documentos XML podrán ser accedidas como si fueran una base de datos.

Ejemplo de consulta XQUERY:

Obtener el año y título de todos los libros publicados por Addison-Wesley después de 1991.

```
<bib>
{
for $b in doc("http://www.bn.com/bib.xml")/bib/book
where $b/publisher = "Addison-Wesley" and $b/@year > 1991
return
<book year="{ $b/@year }">
{ $b/title }
</book>
}
</bib>
```

- **Programación (DOM y SAX).**

Son API's (Application Program Interface) que facilitan un conjunto estándar de llamadas a funciones para manipular documentos XML desde programas.

- **Otros** (Namespaces, Xinclude, Xbase,...).

Los namespaces son un método para cualificar elementos y nombres de atributos de documentos XML, asociándolos con espacios de nombres (namespaces) identificados por referencias URI. Sirven, entre otras cosas, para evitar las colisiones en los nombres de los elementos y atributos.

```
<x xmlns:edi='http://ecommerce.org/schema'>
</x>
```

4.2.4. Ventajas e inconvenientes de XML.

Para acabar se enumeran una serie de ventajas e inconvenientes que, como en todo, tiene el uso de XML:

Ventajas	**Inconvenientes**
Universalidad	Complejidad de algunos estándares (XSLT)
Neutralidad	Rendimiento (comparado con un SGBD)
Legible por seres humanos	Lentitud en la aparición de estándares (Schemas)
Adaptable a cualquier lenguaje de	Estandarización de los lenguajes

programación	derivados
APIs sencillas	Explosión de estándares
Modelo extensible	Herramientas relativamente inmaduras
Modelo de transformaciones integrado	

Tabla 3. Ventajas e Inconvenientes de XML

4.3. RDF.

Para añadir contenido semántico se requiere un lenguaje de representación del conocimiento. Tradicionalmente, estos lenguajes han sido desarrollados en el campo de la inteligencia artificial y fijan su base formal sobre diversos paradigmas: el cálculo de predicados de primer y segundo orden, la lógica de descripciones o la orientación a objetos. Son las propiedades computacionales y expresivas las que diferencian, a la postre, estos lenguajes. Entre los más destacados se encuentran Ontolingua, Loom, OCML y Flogic. Con la aparición de XML como estándar para el intercambio de datos entre aplicaciones, aparecen lenguajes de representación del conocimiento para la Web, basados en XML, de entre los que destaca RDFS.

Así pues, empezamos diciendo que RDF (Resource Description Framework) es una recomendación del W3C (10/02/04), basado en XML, que proporciona la tecnología para escribir metadatos que describen recursos en la Web, proporcionando interoperatividad entre aplicaciones que intercambian información comprensible por la máquina y facilidades para el procesamiento automático de esos recursos [Lassila,99].

Se pueden destacar tres aspectos de la semántica funcional del formato RDF [Méndez]: un modelo de datos, una sintaxis y un esquema.

Un objeto de información o recurso se describe a través de un conjunto de propiedades denominadas "descripción RDF" (<rdf:description>). La esencia de RDF es pues, un modelo formal para la representación de las propiedades y los valores de esas propiedades, que se pueden entender como atributos de los recursos y, en este sentido, corresponden a los pares tradicionales de atributo-valor. Además estas propiedades también representan las relaciones entre los distintos recursos de información, de tal forma que este modelo puede parecer un esquema entidad-relación de las bases de datos relacionales. El modelo RDF se puede relacionar también con el diseño orientado a objetos donde los recursos corresponden a objetos y las propiedades corresponden a ejemplos de variables.

Según esto, el modelo de datos que propone RDF consiste en tres tipos de objetos:

- Recursos: cualquier objeto Web identificable unívocamente por un URI, es decir, un identificador uniforme de recursos como un URL. Un recurso puede ser un documento HTML o una parte de una página Web como, por ejemplo, un elemento HTML o XML dentro de un documento fuente, una colección de páginas, un sitio Web completo y en síntesis, cualquier recurso entendido como objeto de información.
- Propiedades: son aspectos específicos, características, atributos o relaciones utilizadas para describir recursos. Cada tipo de propiedad tiene sus valores específicos, define los valores permitidos, los tipos de recursos que puede describir y las relaciones que existen entre las distintas propiedades.
- Descripciones: Son el conjunto de un recurso, un nombre de propiedad y el valor de esa propiedad (sujeto, predicado y objeto, respectivamente)

Veamos un ejemplo extraído del código fuente de la propia propuesta de recomendación del esquema de RDF:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML xmlns:rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:rdfs = "http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
        xmlns:dc = "http://purl.org/dc/elements/1.0/">
  <HEAD>
<TITLE>Resource Description Framework (RDF) Schema Specification</TITLE>
<STYLE TYPE="text/css"> .EXAMPLE { margin-left: 1em } </STYLE>
<LINK rel="stylesheet" type="text/css" media="screen" href="/StyleSheets/TR/W3C-PR">
  <rdf:RDF>
<rdf:Description about=""
   xmlns:ddc="http://purl.org/net/ddc#"
dc:Title="Resource Description Framework (RDF) Schema Specification"
dc:Description="The Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources."
dc:Publisher="World Wide Web Consortium"
dc:Date="1999-03-03"
dc:Format="text/html"
dc:Type="technical specification"
dc:Language="en">
<dc:Subject resource="http://purl.org/net/ddc/025.30285"/>
```

```
    <dc:Subject resource="http://purl.org/net/ddc/025.316"/>
    <dc:Subject ddc:Class="025.302855741" ddc:Heading="Applications of computer file organization and access methods"/>
    <dc:Creator>
    <rdf:Bag rdf:_1="Dan Brickley"
             rdf:_2="R.V. Guha" /></dc:Creator>
    <rdfs:seeAlso rdf:resource="http://www.w3.org/1999/.status/PR-rdf-schema-19990303/status"/>
    </rdf:Description>
</rdf:RDF>
</HEAD>
```

El elemento Description <rdf:description> con el atributo "about", se utiliza para identificar (URI/URL) el recurso que se está describiendo que, en este caso, es además implícitamente el propio documento de la propuesta de recomendación del esquema.

Dentro de las etiquetas <rdf:Description>...</rfd:Descripion> se encuentran todas las propiedades (con el prefijo DC, según la declaración previa del namespace) con sus valores. Dentro de la descripción se declara otro namespace (xmlns:ddc ="http://purl.org/net/ddc#") que cualificará a su vez el elemento DC:subject, según la Clasificación Decimal de Dewey (DDC) .

Otro aspecto que merece la pena resaltar es el elemento rdf:bag_n. Un Bag es un objeto contenedor que sirve para consignar un conjunto de múltiples valores para la misma propiedad, cuyo orden es indiferente. En este caso se usa para repetir la misma propiedad <dc:creator> para matizar que ambos autores comparten la propiedad intelectual del documento. Si, por ejemplo, uno de los autores fuese más importante, se utilizaría el tipo de propiedad <rdf:Seq>.

4.3.1. RDFS

Las comunidades de descripción de recursos necesitan la habilidad para decir ciertas cosas sobre ciertas clases de recursos, ya que el modelo y la sintaxis RDF no facilitan los mecanismos para definir esas propiedades ni las relaciones entre esos predicados y otros recursos o sujetos. Para describir recursos bibliográficos, por ejemplo, son habituales atributos descriptivos tales como autor ["author"], título ["title"], y materia ["subject"]. Para la certificación digital se necesitan muchas veces atributos tales como un esquema simple de detección de errores ["checksum"] y autorización

["authorization"]. La declaración de estas propiedades (atributos) y su semántica correspondiente se definen, en el contexto de RDF, como un Esquema RDF [4]. Un esquema define no sólo las propiedades de un recurso (ej. título, autor, materia, tamaño, color, etc.) sino que puede también definir los tipos de recursos que se describirán (libros, páginas Web, personas, empresas, etc.).

Una de las aplicaciones más claras para RDF es la descripción de páginas Web. Este es uno de los objetivos básicos de la iniciativa de metadatos Dublin Core, que ha constituído la mayor influencia en el desarrollo de RDF. Una consideración importante en el desarrollo del Dublin Core, fue no permitir sólo descripciones simples, sino también proporcionar la posibilidad de cualificar descripciones para proporcionar la elaboración específica de un dominio y la precisión descriptiva.

La especificación del esquema RDF proporciona un sistema entendible por la máquina para definir esquemas para vocabularios específicos como el Dublin Core. Esto permite a los diseñadores especificar clases de tipos de recursos y propiedades para dar a conocer descripciones de esas clases, relaciones entre esas propiedades y clases, y las restricciones en las combinaciones permitidas de clases, propiedades y valores.

La figura 6 ilustra el concepto de clase, subclase, y recurso. Una clase se describe por un rectángulo redondeado; un recurso se describe por un punto grande. Una subclase se presenta en un rectángulo redondeado completamente incluido en otro (la superclase). Si un recurso está dentro de una clase, entonces existe una propiedad “rdf:type”.

[4] Resource Description Framework (RDF) Schema Specification 1.0. W3C. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/ (29/06/04).

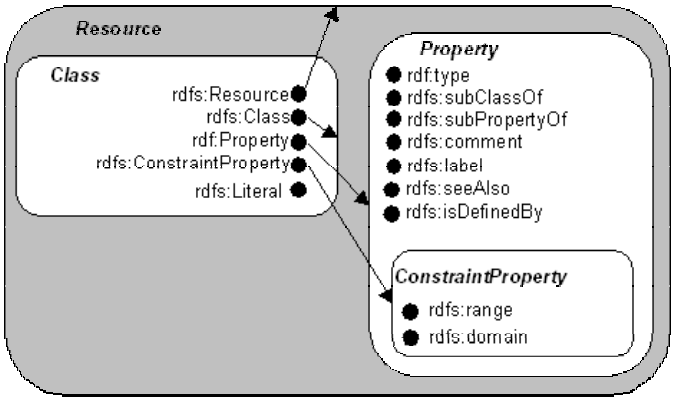

Figura 6. Clases y recursos como conjuntos y elementos.

La figura 7 muestra la misma información sobre la jerarquía de clase que en la figura 6, pero lo hace utilizando la representación gráfica de "nodos y arcos" del modelo de datos RDF. Si una clase es un subconjunto de otra, hay un arco rdfs:subClassOf desde el nodo que representa la primera clase hasta el nodo que representa la segunda. De forma similar, si un recurso es un objeto específico "instance" de una clase, entonces hay un arco rdf:type desde el recurso hasta el nodo que representa la clase.

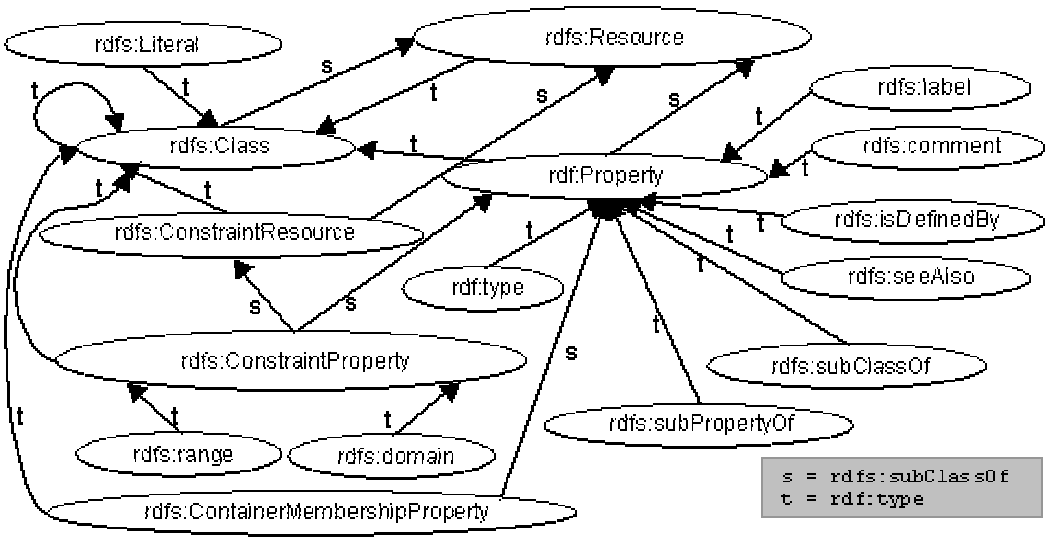

Figura 7. Jerarquía de clase para el Esquema RDF

Veamos un sencillo ejemplo de una jerarquía de clase. En primer lugar definimos la clase MotorVehicle. Después definimos tres subclases de MotorVehicle: PassengerVehicle (vehículo de pasajeros), Truck (camión) y Van (furgoneta). Después definimos la clase Minivan (minifurgoneta) que es una subclase de ambas Van y PassengerVehicle.

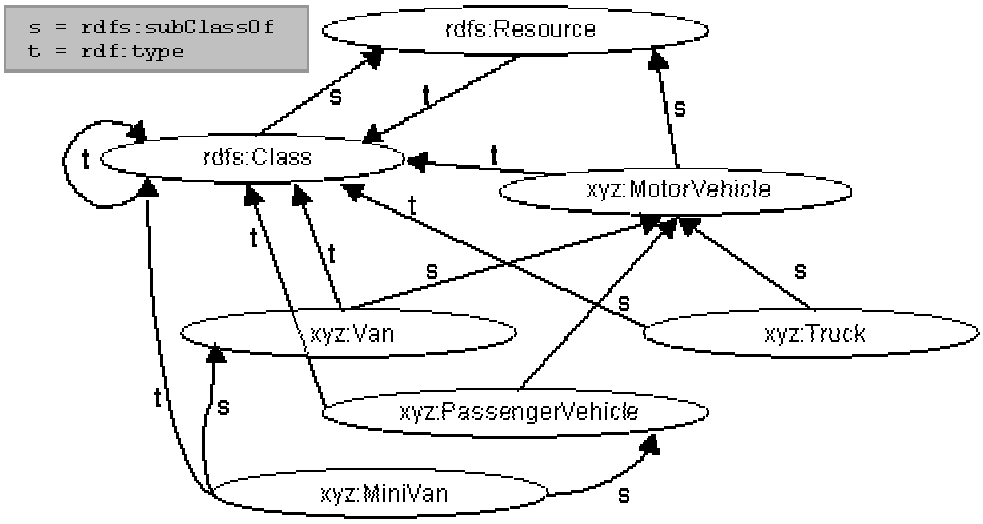

Figura 8. Jerarquía de clase para la clase MotorVehicle.

La representación del esquema anterior mediante el lenguaje RDFS es la siguiente:

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<!-- Note: this RDF schema would typically be used in RDF instance data
    by referencing it with an XML namespace declaration, for example
    xmlns:xyz="http://www.w3.org/2000/03/example/vehicles#".  This allows
    us to use abbreviations such as xyz:MotorVehicle to refer
    unambiguously to the RDF class 'MotorVehicle'. -->
<rdf:Description ID="MotorVehicle">
 <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"/>
</rdf:Description>
<rdf:Description ID="PassengerVehicle">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
<rdf:Description ID="Truck">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
<rdf:Description ID="Van">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

```
<rdf:Description ID="MiniVan">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Van"/>
 <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>
</rdf:RDF>
```

Además RDF utiliza una serie de restricciones que aumentan las posibilidades de descripción de los recursos, sus propiedades y sus relaciones:

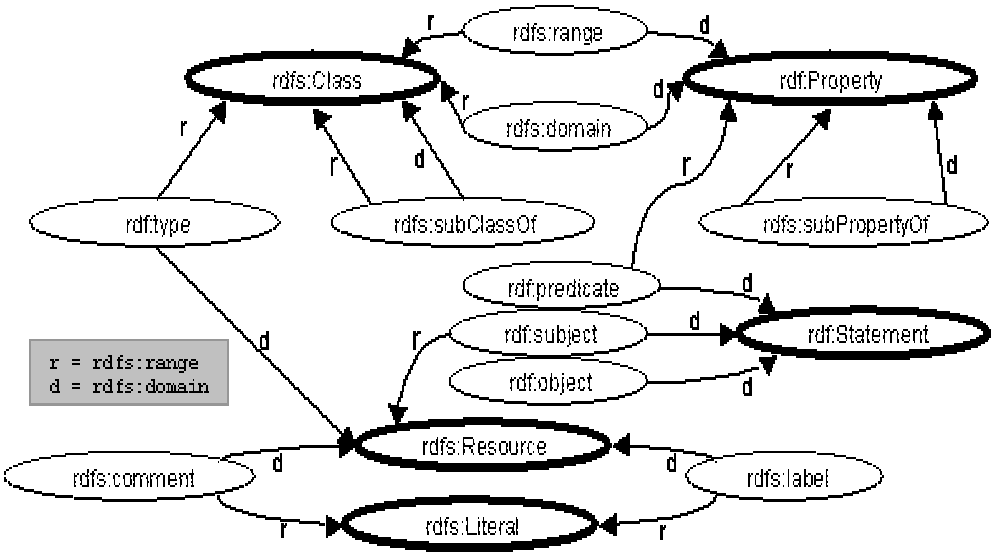

Figura 9. Restricciones en el Esquema RDF

RDFS también proporciona facilidades adicionales para soportar la evolución de los vocabularios RDF particulares (RDFvocabulary), documentación (rdfs:comment, rdfs:label, etc.).

En resumen, RDF permite el procesamiento de metadatos, proporcionando interoperabilidad entre aplicaciones que intercambian información legible por máquina en la Web. RDF utiliza XML para intercambiar descripciones de recursos Web, y puede utilizarse en diferentes áreas de aplicación, por ejemplo: en la recuperación de recursos para proporcionar mejores capacidades a los motores de búsqueda, en catalogación para la descripción de contenido y sus relaciones de contenido accesibles en un sitio Web particular, en una página, en una biblioteca digital, a través de agentes de software inteligente para facilitar que el conocimiento se comparta e intercambie, en la valoración de contenido, en la descripción de colecciones de páginas que representan un documento lógico individual, para la

descripción de los derechos de propiedad intelectual de las páginas Web, y para expresar las preferencias de privacidad de un usuario así como las políticas de privacidad de un sitio Web.

[Sánchez,02] Sánchez, L. Delgado, Carlos. XML el ASCII del siglo XXI. Novática, nº 158 (2002). Disponible en: http://www.ati.es/novatica/2002/158/nv158sum.html (25/06/04).

[Gutiérrez,99] Gutiérrez, E. XML en 10 puntos. Traducido de la versión original, disponible en:http://www.w3.org/XML/1999/XML-in-10-points.html (25/06/04).

[Bray,98] Bray, Tim et al. El lenguaje extensible de marcas (XML) 1.0. Recomendación del W3C (1998). Disponible en: http://www.w3.org/TR/1998/REC-xml-19980210 (25/06/04).

[Ruiz,03] Ruiz, F. XML y Derivados. Grupo Alarcos, Universidad de Castilla La Mancha. Disponible en: http://alarcos.inf-cr.uclm.es/per/fruiz/cur/conf/xml/xml-bn.pdf (25/06/04)

[Lassila,99] Lassila, O & Swick,R. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, World Wide Web Consortium, Cambridge (MA), February 1999. Disponible en: http://www.w3.org/TR/REC-rdf-syntax/ (29/06/04)

[Méndez,99] Méndez, E. Rdf: un modelo de metadatos flexible para las bibliotecas digitales del próximo milenio. 7ª Jornadas Catalanas de Documentación. Disponible en: http://www.bib.uc3m.es/~mendez/publicaciones/7jc99/nota5 (29/06/04).