Capítulo 2 :
Conceptos fundamentales

Capítulo 2

Conceptos fundamentales

En este capítulo se hace una revisión de los conceptos y de las tecnologías utilizadas en la realización de este proyecto. Es importante considerarlos ya que contribuyeron en gran medida en el diseño e implementación de las herramientas que se desarrollaron.

2.1 Los diferentes tipos de datos

Los datos pueden clasificarse de acuerdo a su nivel de estructura, de esta forma tenemos (1) datos no estructurados que se caracterizan por no tener una estructura definida y para los cuales se hace un fuerte uso de métodos de recuperación de información y de herramientas de indexamiento; (2) datos estructurados que tienen una estructura bien definida y un modelo prescriptivo que los define, haciendo uso de atributos fuertemente tipados; (3) datos semi-estructurados, los cuales se ubican entre los dos extremos anteriores y se caracterizan por tener asociado un modelo descriptivo más que prescriptivo [Proal 2003] [Sánchez et al 2004].

2.2 XML: eXtensible Markup Language

XML es el Lenguaje eXtensible de Marcado. A diferencia de HTML, XML fue concebido como un mecanismo que permite el modelado de la información sin hacer referencia alguna a la manera en que dicha información será visualizada.

De manera general podemos decir que las ventajas que XML ofrece son las siguientes [Gutiérrez & Martínez 2001]:

- Independencia de los datos respecto de las aplicaciones, dado que se ha convertido en un formato estándar para el intercambio de la información.
- Información sobre la información, un contexto, lo cual es de mucha utilidad a la hora de procesar dichos documentos.
- Elementos para describir la estructura de un documento.
- Un medio para organizar la información de la manera en que nosotros queramos.
- Capacidad para pasar información a las aplicaciones, ya que hoy en día muchos de los desarrollos de software hacen uso de XML como un medio para definir aspectos de configuración de las aplicaciones.

Para comprender aún más la utilidad de XML podemos verlo como un conjunto de estándares relacionados y que de alguna u otra manera se complementan. Las secciones 2.3, 2.4, 2.5, 2.6 y 2.7 de este capítulo están orientadas a dar una breve descripción sobre estos estándares y la manera en que se utilizan de forma complementaria.

2.3 XSL: eXtensible Style Language

XSL es un lenguaje que permite añadir información a un documento referente a cómo debe de visualizarse, ya que incluye capacidades de proceso y transformación.

XSL se subdivide en 3 partes:

1.- XSLT: un lenguaje para realizar transformaciones a un documento XML y obtener como resultado estructuras y estilos diferentes como XHTML, HTML o simplemente documentos XML con estructura diferente.

2.- XPath: Es un lenguaje que define una sintaxis estándar para localizar partes específicas dentro de un documento XML. La sintaxis que propone es sencilla ya que es similar a la usada para especificar rutas de directorios:

- Ejemplo de la sintaxis de XPath:

  Para visualizar la manera en que se utiliza XPath como lenguaje de consulta, consideremos el documento XML que se ilustra en la figura 2.1:

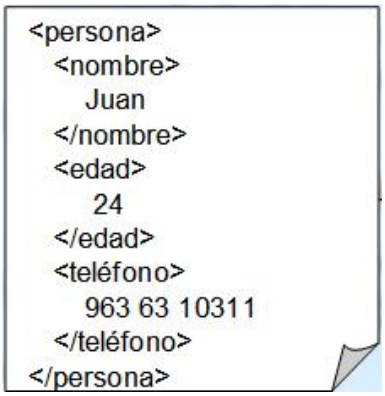

Figura 2.1. Ejemplo de un documento XML

Si quisiéramos obtener el nombre de la persona descrita en ese documento XML, haríamos la siguiente consulta:

/persona/nombre/text( )

Lo cual nos daría como resultado “Juan”. Ahora bien, si necesitáramos obtener al mismo tiempo el nombre, la edad y el teléfono de la persona ahí descrita, haríamos lo siguiente:

/persona/nombre/text( ) | /persona/edad/text( ) | /persona/teléfono/text( )

Lo cual nos daría como resultado:

“Juan”

“24”

“963 63 10311”

3. - XSL-FO: eXtensible Stylesheet Language Formatting Objects

Define un vocabulario para establecer un formato a los datos descritos en los documentos XML, pensando en una especie de presentación o “layout” que esos datos deben de tener a la hora de visualizarse en pantalla, al momento de imprimirlos o al utilizarse en otros medios multimediales.

Una manera muy popular de utilizar XSL-FO es cuando se quiere convertir un documento XML y presentarlo como un documento PDF.

2.4 XQuery: XML query language

La mejor manera para definir a XQuery es decir que XQuery es un lenguaje que fue especialmente diseñado para definir consultas sobre datos cuya estructura es igual o similar a la de un documento XML; ya sea que dichos datos sean documentos XML en forma “nativa” o que estén almacenados por ejemplo en alguna base de datos relacional y haya un mecanismo o “middleware” que permita traducir esas consultas en XQuery a su correspondiente sintaxis en SQL. Más adelante se explicará un poco más acerca de esto al describir un tipo en particular de bases de datos XML, llamadas “habilitadas”.

XQuery hace uso de la sintaxis proporcionada por XPath e incluso comparten muchas de las funciones que se especifican para el lenguaje XPath; pero además de eso, XQuery

permite definir una estructura de presentación a las respuestas que se obtienen al momento de hacer una consulta. Esto es de mucha utilidad ya que al mismo tiempo que se indica una consulta, también se establece la presentación que se quiere obtener con los resultados que la consulta arroje, pudiendo establecer por ejemplo, que los resultados se plasmen en un documento HTML. En la figura 2.2 puede verse un ejemplo de una consulta con XQuery que permite obtener ciertos datos de un documento XML que contiene un catálogo de discos, e indicar la forma (sintaxis HTML) en que se visualizarán los resultados:

```
<table>
{
  <tr><td>Autor</td><td>Titulo</td><tr>
  for $x in doc("cdCatalog.xml")/catalogo/cd
  order by $x/autor
  return <tr>
         <td>{$x/autor/text()}</td>
         <td>{$x/titulo/text()}</td>
         </tr>
}
</table>
```

Figura 2.2. Ejemplo de una consulta con XQuery

El resultado de esta consulta se ilustra en la figura 2.3:

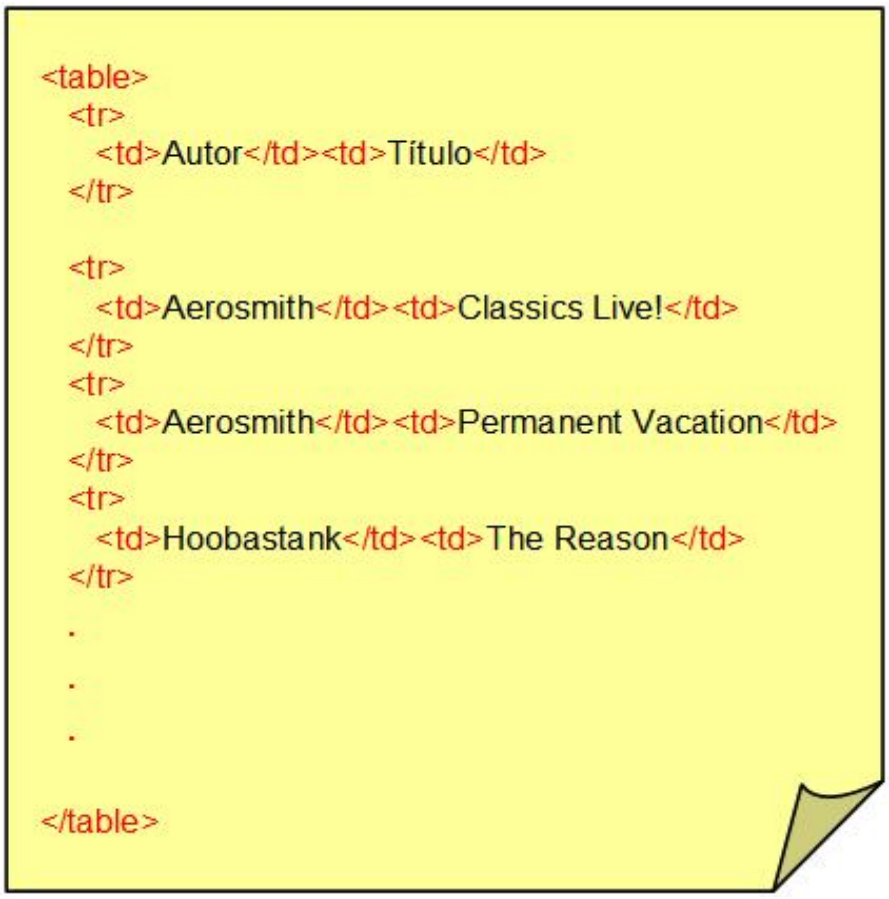

Figura 2.3. Resultado de la consulta con XQuery de la figura 2.2

2.5 Separación entre contenido y presentación

Al hablar de XML no podemos pasar por alto la capacidad que brinda para establecer una separación entre la información que describe y la forma en que esa información será presentada. Esto es de gran utilidad ya que podemos tener por un lado, una forma única de representar cierta información, mientras que de otro lado podemos darle la presentación o formato deseado a esa información.

Para entender mejor lo descrito en el párrafo anterior, pensemos en un documento XML y la variedad de formas en que puede presentarse la información descrita en él. Podemos, por ejemplo, obtener todos o un subconjunto de los datos y crear con ellos

documentos HTML, PDF, CVS (contenido separado por comas y que es reconocido por las hojas de cálculo más populares), TXT u otro documento XML cuya estructura es diferente al original. Esto puede visualizarse en la figura 2.4:

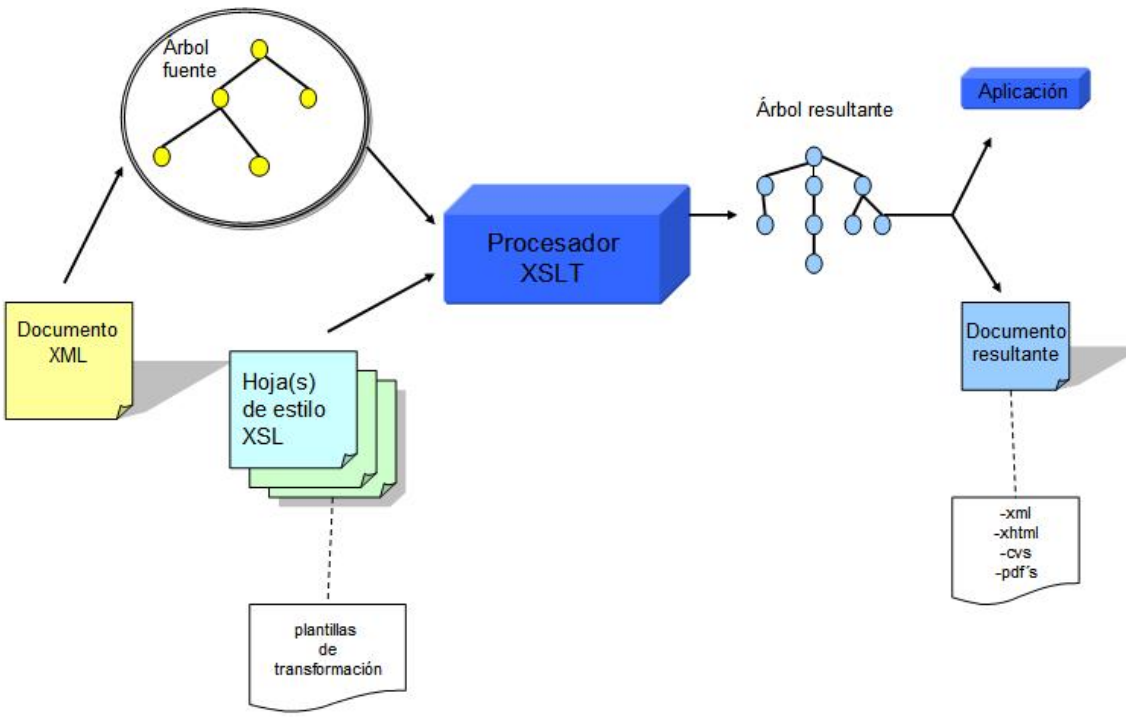

Figura 2.4. Modelo abstracto del procesamiento de un documento XML usando XSLT (adaptada de [Xalan 2005])

De la figura 2.4 podemos ver que la herramienta encargada de llevar a cabo la transformación requerida, es el Procesador XSLT; el cual toma de entrada el contenido y por supuesto la estructura de un documento XML y sus correspondientes transformaciones (especificadas en una hoja de estilo o XSL Style Sheet a manera de

“plantillas de transformación”). Con esto se produce un documento que puede ser totalmente distinto al original.

2.6 Bases de datos XML

Es muy común el pensar en utilizar documentos XML como una base de datos, dado que éstos tienen estructura y permiten definir nuestras propias marcas o “tags” para dar un contexto o meta-información a la información que en ellos se expresa. Sin embargo, mantener un directorio de archivos XML como nuestra base de datos se torna ineficiente cuando lo que se necesita es contar con todas las facilidades que las bases de datos de hoy en día ofrecen: transacciones, seguridad, acceso multi-usuario o concurrencia, consistencia, entre otras. Es por ello que en los últimos años han surgido las llamadas bases de datos XML (XMLDbs), las cuales nos brindan muchas de las facilidades dadas por bases de datos relacionales u orientadas a objetos, pero con un enfoque más especializado hacia el manejo de datos semi-estructurados modelados como documentos XML.

Es así como podemos distinguir 2 tipos de XMLDbs: habilitadas y nativas. Las primeras son muy utilizadas cuando lo que se quiere es usar XML como un formato para el intercambio de datos. Por lo regular son bases de datos relacionales con funcionalidades agregadas que permiten obtener los datos descritos en un documento XML, vaciarlos a sus correspondientes tablas y finalmente, cuando así se desee, obtener nuevamente los datos almacenados en tablas e intentar reconstruir el documento XML original. Las XMLDBs nativas por el contrario, permiten almacenar documentos XML y recuperarlos a exactitud ya que utilizan el modelo de datos de XML para representar todos los componentes de un documento, así como su estructura. [Steegmans et al 2004].

2.6.1 Iniciativa XML:DB

La iniciativa XML:DB[3] es una organización formada con el objetivo de establecer estándares y especificaciones para el manejo de datos utilizando bases de datos XML (de forma más específica, bases de datos XML nativas). Propone una interfaz de programación (API) llamada XML:DB, cuya utilidad es similar al JDBC API utilizado por bases de datos relacionales. XML:DB se usa cuando lo que se requiere es tener acceso a bases de datos utilizando Java como ambiente de desarrollo.

2.7 Formatos de metadatos

En esta sección se explicará brevemente qué son los formatos de metadatos, cuál es su utilidad y se hará mención de algunos de los formatos más utilizados.

Los formatos de metadatos establecen una manera de representar, de manera uniforme, los descriptores de un recurso; ya sea un documento, imágenes, sitios Web, entre otros. Su utilidad se ha popularizado ya que muchos acervos digitales están ocultos para la mayoría de los motores de búsqueda que se tienen a través de Internet.

Esto se debe a que muchos de estos recursos se encuentran almacenados en bases de datos o son generados de forma dinámica y no están a la vista como simples páginas HTML estáticas. Dichos recursos constituyen lo que muchos llaman “Web oculta”, cuyo contenido se estima en 400 ó 500 veces más que el contenido accesible de forma directa a través del Web [Perkins 2001].

[3] The XML:DB Initiative for XML Databases: http://xmldb-org.sourceforge.net/

Utilizando formatos de metadatos pueden establecerse mecanismos para el intercambio de la información y de esta manera poner a la vista todos esos recursos en cierta forma ocultos para los motores de búsqueda tradicionales.

Algunos de los formatos de metadatos más utilizados dentro del contexto de bibliotecas digitales son: MARC, Dublín Core, BibText y RDF, entre otros. Uno de los que ha tenido mayor aceptación es Dublin Core[4], ya que tiene un enfoque multidisciplinario. Consiste en un conjunto de 15 campos o elementos para describir a un documento (ver Apéndice A).

En la figura 2.5 se muestra un ejemplo de este formato de metadatos describiendo a un documento de tesis de la colección de Tesis Digitales:

```
- <metadata>
  - <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:dc="http://purl.org/dc/elements/1.1/"
      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
      <dc:title>Un Marco de Comunicación Inter-Agentes en una Biblioteca Digital</dc:title>
      <dc:creator>Adolfo Barceinas Guevara</dc:creator>
      <dc:subject>Digital libraries</dc:subject>
      <dc:subject>Information storage and retrieval systems</dc:subject>
      <dc:subject>User interfaces</dc:subject>
      <dc:description>Este proyecto de tesis se enfoca a resolver el problema de la comunicacion entre los componentes de una
        biblioteca digital altamente distribuida. Especificamente, se considero la arquitectura de la Biblioteca Digital Floristica (FDL), la
        cual tiene una orientacion hacia servicios a usuarios, incluyendo servicios de agentes.</dc:description>
      <dc:publisher>Universidad de las Americas, Puebla</dc:publisher>
      <dc:contributor>Jose Alfredo Sanchez Huitron</dc:contributor>
      <dc:date>1998-05-19</dc:date>
      <dc:type>Licenciatura, Ingenieria en Sistemas Computacionales</dc:type>
      <dc:format>text/html</dc:format>
      <dc:format>application/pdf</dc:format>
      <dc:identifier>http://www.udlap.mx/~tesis/lis/barceinas_g_a</dc:identifier>
      <dc:source>null</dc:source>
      <dc:language>ES</dc:language>
      <dc:relation>http://www.udlap.mx/~tesis/lis/barceinas_g_a</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/indice.html</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/resumen.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/capitulo1.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/capitulo2.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/capitulo3.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/capitulo4.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/capitulo5.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/capitulo6.pdf</dc:relation>
      <dc:relation>http://www.pue.udlap.mx/~tesis/lis/barceinas_g_a/bibliografia.pdf</dc:relation>
      <dc:coverage>null</dc:coverage>
      <dc:rights>http://www.udlap.mx/reglamte/capitulo2/t2c4.html</dc:rights>
    </oai_dc:dc>
  </metadata>
```

Figura 2.5. Ejemplo de uso de Dublin Core para representar los metadatos de un documento de tesis.

[4] Dublin Core Metadata Initiative (DCMI): http://dublincore.org/

2.8 OAI-PMH

En esta sección se da una descripción a detalle del Protocolo OAI-PMH (Open Archives Initiative-Protocol for Metadata Harvesting), así como del marco de interoperabilidad que propone. Se definen los conceptos base que se utilizan en la definición del protocolo y se describen sus principales características.

2.8.1 Orígenes

Los orígenes de OAI se centran en los archivos llamados *E-Prints* (documentos accesibles vía Web) sin embargo, desde hace algunos años se ha orientado hacia cualquier tipo de recurso que pueda ser descrito (documentos, imágenes, antigüedades en museos, etc.), con énfasis en lo que podría considerarse como publicaciones académicas [Lagoze & Van de Sompel 2001].

OAI estableció un protocolo de propósito general como mecanismo de interoperabilidad (independiente de los tipos de contenido que podría abarcar, modelos económicos que rodeaban esos contenidos; así como propiedades intelectuales ligadas a esos contenidos), al cual se denominó *Open Archives Initiative Protocol for Metadata Harvesting* (OAI-PMH). El objetivo central de OAI no se orientó al surgimiento de servidores que permitieran el libre acceso a los metadatos de sus repositorios; sino a poder lograr una federación de repositorios registrados ante dicha iniciativa y de esta manera dar pauta a una variedad de servicios con valor agregado que podrían surgir.

Los repositorios que decidieran implementar este protocolo como mecanismo para la exposición de sus metadatos tomarían la decisión de proporcionar o no una liga al contenido completo descrito por los metadatos que publicaran. Con esto OAI se orientó a promover de forma más general estándares de interoperabilidad que abarcan un gran número de comunidades y al establecimiento de una federación de repositorios independientemente del tipo de contenido que cada uno provea y los aspectos económicos para su acceso que se establezcan.

2.8.2 Entidades participantes

OAI-PMH es un protocolo únicamente para el intercambio de metadatos; no provee un mecanismo directo para exponer el contenido completo (documentos, imágenes, etc.). Sin embargo, no se descarta la posibilidad de que en un futuro el protocolo provea facilidades para el intercambio de contenido completo.

OAI-PMH está basado en un modelo que establece una clara división entre dos tipos de participantes:

- Proveedores de datos: Son entidades que exponen metadatos de la información que poseen mediante la utilización del protocolo. Dicha información puede estar organizada en uno o más repositorios.

Los repositorios son esencialmente servidores accesibles vía web ofrecidos por los Proveedores de datos. Dichos repositorios permiten la exposición de metadatos a través de registros sobre el contenido que poseen. Un Repositorio puede organizar de forma

opcional su contenido en diferentes conjuntos, permitiéndoles a los usuarios hacer una recolección de metadatos más selectiva.

- Proveedores de servicios: Entidades que obtienen metadatos mediante la utilización del protocolo, con la intención de proveer algún servicio. Un ejemplo claro de un proveedor de servicio es aquel que permite búsquedas simultáneas en diversos repositorios.

La división anterior no restringe el hecho de que una entidad pueda fungir como Proveedor de datos y como Proveedor de servicios a la vez. El mecanismo de interacción entre estas entidades se da vía el protocolo HTTP, a través de sus métodos GET y POST. Un Proveedor de servicios manda una petición a uno o varios Proveedores de datos. Una petición puede ser de alguno de los 6 tipos que define el protocolo.

Lo explicado en el párrafo anterior puede visualizarse más claramente mediante la figura 2.6:

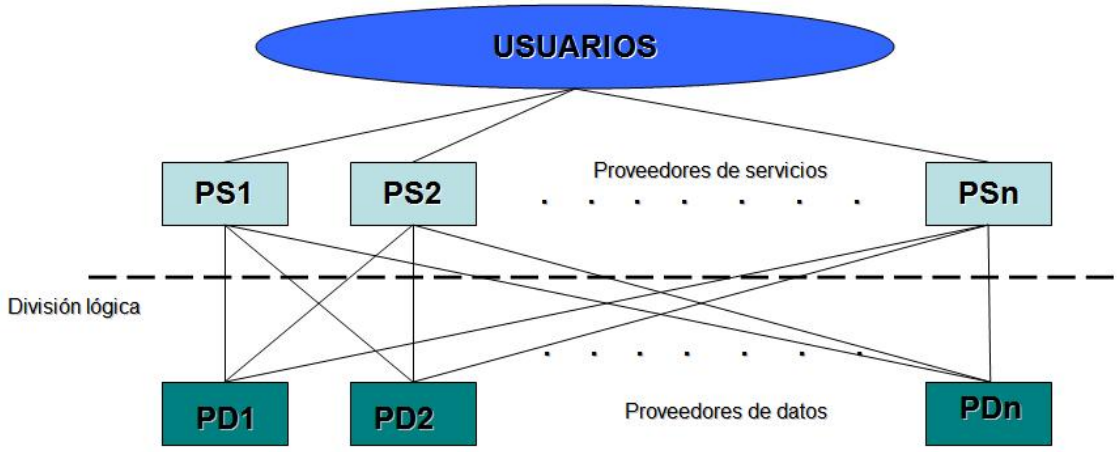

Figura 2.6. Proveedores de datos y Proveedores de servicios utilizando OAI-PMH

(Adaptada de [Warner 2001])

2.8.3 Metadatos

OAI permite el uso de varios formatos de metadatos; incluso da pauta a la utilización de formatos acordes a las necesidades de ciertas comunidades pero con la condición de que todos los formatos a utilizar tengan asociado un esquema en XML que defina su estructura. Sin embargo, esta iniciativa exige la utilización, para todas las entidades participantes, del estándar Dublin Core (DC) y mediante su utilización se establece una base para la interoperabilidad. En el apéndice A de este documento se anexa el esquema que define a este formato de metadatos.

El comité de OAI tomó la decisión de utilizar el formato no calificado (unqualified) DC con el propósito de dar mayor flexibilidad de acuerdo a las necesidades específicas de cada comunidad. De esta manera las comunidades eligen, de acuerdo a la lista de elementos especificados en un esquema en XML que define al formato Dublín Core (ver apéndice A), cuáles elementos utilizar para la descripción de la información.

2.8.4 Registros

Un registro es una codificación en XML y puede hacer referencia a uno o más formatos de metadatos especificados por un elemento <*metadata*>. Está formado por tres elementos o contenedores: encabezado, contenedor de metadatos y contenedor "acerca de".

- Encabezado: contiene información que es común a todos los registros, es independiente del formato de metadatos diseminado en el registro. Es necesario para propósitos de búsquedas selectivas. La información que se especifica dentro de este elemento es, el identificador del registro y la fecha de creación o modificación del registro, así como los conjuntos a los que pertenece el registro.

- Contenedor de Metadatos: contiene metadatos en un formato. Como se especificó anteriormente, todos los Proveedores de datos deben exponer metadatos en el formato Dublin Core. Sin embargo, otros tipos o formatos de metadatos son permitidos, siempre y cuando se proporcione un esquema en XML que los defina.

- Contenedor "Acerca de": es un contenedor opcional que describe la parte de metadatos incluida en cada registro. La estructura interna de este contenedor no está definida por el protocolo. Dicha estructura es definida por cada Proveedor de datos y lo único que se requiere es la definición de un esquema en XML que respalde dicha estructura. Muchos lo utilizan para indicar términos y condiciones de uso y derechos de autor.

Después de haber definido los subelementos o contenedores de cada registro, podemos visualizar de forma completa su estructura en la figura 2.7. En la esquematización de la estructura de un registro las líneas punteadas indican que el elemento en cuestión puede ser opcional, ya que puede aparecer o no como parte de un registro. En el Apéndice C se explica más a detalle la herramienta utilizada para la generación de este tipo de gráficas.

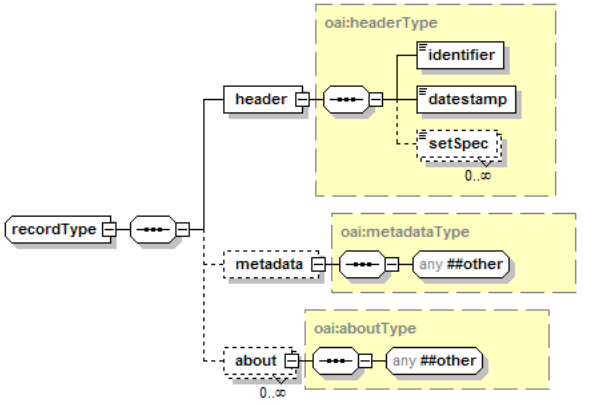

Figura 2.7. Estructura completa de un Registro

De forma abstracta, un repositorio contiene *"unidades de información"* y cada una de ellas puede tener relacionado uno o más registros dado que pueden diseminarse varios formatos de metadatos e incluirlos en cada registro. La naturaleza de cada *unidad de información* está fuera del alcance del protocolo.

Lo anterior puede explicarse de la siguiente manera: Un repositorio tiene un registro de la información que posee normalmente en una base de datos, mientras que una *unidad de información* es generado dinámicamente de acuerdo al formato o los formatos de metadatos solicitados y que pueden ser diseminados por esa *unidad*.

El identificador de cada elemento contenido en la base de datos de un repositorio es diferente al identificador de cada *unidad de información* generada. Si un elemento (p. e.

un documento almacenado) tiene como identificador "*is106278*", la *unidad* relacionada a ese elemento en particular podría tener un identificador de la siguiente forma: "*oai:ThesisUdla:is106278*" (la nomenclatura de este identificador es definida por OAI acorde al registro del repositorio ante esa organización). Donde el primer elemento especifica que dicho repositorio está registrado ante OAI, el segundo es el nombre del Repositorio como quedó registrado y el tercer elemento normalmente es el identificador que relaciona a la *unidad de información* con el elemento que describe.

Un ejemplo en XML de un registro se muestra en la figura 2.8 (por simplicidad sólo se muestra su estructura y no se despliega la sección de metadatos, que es donde se especifican los "tags" o elementos del formato de metadatos solicitado). En esta imagen podemos ver que el identificador de la *unidad de información* (generada como respuesta a la consulta) es "oai:UdlaXMLThesis:1", mientras que el identificador del recurso descrito (en este caso un documento de tesis) es "1".

Figura 2.8. Codificación en XML de un registro

2.8.5 Recolección selectiva de metadatos

Es de utilidad el hecho de poder realizar peticiones de acuerdo a ciertos criterios por ejemplo, relacionados con la manera en que está organizado el contenido de un

repositorio o acordes a fechas específicas en que el contenido está registrado. Es por ello que el protocolo de OAI permite la recolección de metadatos de forma selectiva, evitando el obtener respuestas innecesarias para ciertos propósitos. Tal mecanismo permite la obtención de un subconjunto de registros como respuesta a las peticiones que se hacen. Muchos protocolos permiten especificar criterios de selección en forma de consultas a bases de datos pero OAI no optó por ello dada la interoperabilidad y sencillez que se necesita en el protocolo. En vez de ello escogió los dos criterios siguientes de selección:

- Rango de fechas: este criterio de selección permite la obtención de registros creados o modificados en un rango de fecha dado.
- Especificación de conjuntos: un repositorio puede elegir el organizar su información en diversos conjuntos, incluso éstos pueden organizarse de forma jerárquica. Cuando un repositorio ofrece esta facilidad, permite a los clientes que obtienen metadatos la posibilidad de hacer peticiones acordes a la organización de los elementos del repositorio.

2.8.6 Tipos de peticiones

El protocolo para la recolección de metadatos define 6 peticiones o verbos con la siguiente estructura:

- *URL*: especifica la dirección del servidor que actúa como repositorio así como el nombre del manejador de las peticiones (p.e. servlet o jsp).

- *Argumentos*: Consiste de una lista de parejas “argumento-valor”. Como mínimo cada petición debe tener un “argumento = valor” que especifica el nombre de la petición que se hizo.

Los seis tipos de peticiones o verbos son los siguientes:

- GetRecord
- Identify
- ListIdentifiers
- ListMetadataFormats
- ListRecords
- ListSets

Las respuestas a cada una de las peticiones son codificadas en XML y pueden ser validadas a través de un esquema general que especifica la estructura correcta de dichas respuestas (ver Apéndice B). Los detalles de cada petición se indican en el Apéndice C de este documento.

2.9 Conclusiones

En este capítulo se hizo una revisión de los conceptos y las tecnologías utilizadas para la realización de este proyecto. En forma general vimos cuáles son los diferentes tipos de datos de acuerdo a su estructura, se describió qué es XML y una serie de tecnologías relacionadas con éste. Se describió también qué son las bases de datos XML y cuáles son los 2 tipos que existen hoy en día. Se mencionó también qué es la iniciativa XML:DB y cuál es el objetivo que persigue. Posteriormente se definieron los formatos

de metadatos y se explicó el por qué de su utilización; brindando un ejemplo del formato de metadatos Dublin Core.

Finalmente se abordó a detalle el Protocolo para la Recolección de Metadatos de la Iniciativa de Archivos Abiertos. Se vio que su campo de aplicación es mucho más amplio que únicamente lo relacionado a los archivos llamados *E-Prints*, abarcando cualquier tipo de contenido y dando pauta a la formación de repositorios registrados ante OAI. Concluyendo que el Protocolo para la Recolección de Metadatos establece un mecanismo sencillo de implementar y sobre todo una base firme para la interoperabilidad entre diversos sistemas, haciendo énfasis en el hecho de que la heterogeneidad de la información y los posibles mecanismos económicos que rodean el acceso a dicha información está fuera del alcance de dicho protocolo.