UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGON

El Lenguaje Extensible de Marcado (XML)

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN CUMPUTACION

P R E S E N T A :

GONZALEZ LUNA JUAN DAVID

No. Cta. 9014325-4

Asesor: Ing. Norma Raquel Soto Arredondo

SAN JUAN DE ARAGON 2006

AGRADECIMIENTOS

A mi alma mater, la Universidad Nacional Autónoma de México, la cual me dio la gran oportunidad de ser universitario, mismo que llevaré con orgullo durante toda mi vida.

A mi querida escuela la Facultad de Estudios Superiores Aragón.

A mi asesor de tesis Ingeniera Norma Raquel Soto Arredondo por su invaluable ayuda para la culminación del presente trabajo.

A mis revisores de tesis, Ingeniero Roberto Blanco Bautista, Ingeniero Enrique García Guzmán, Ingeniero Rodolfo Zaragoza Buchaín, Maestro en C. Marcelo Pérez Medel.

A mis padres: Dr. Juan González Castañeda (†) y Galdy Luna Figueroa que con su gran ejemplo e inagotable amor me otorgaron las herramientas para hacerle frente a la vida. Gracias mamá por tu apoyo constante en el proceso de mi titulación, el cual siempre me animó y me recordó en todo momento que todavía tenía un objetivo por cumplir. Gracias papá por la formación que me diste con tu ejemplo como persona, profesionista y tu cariño como padre y amigo.

A mis hermanas: Maestra en A. T. Lirio Azahalia González L., Lic. Dalia Alhelí González L., Contadora Orquídea Gardenia González L. y Contadora Rosalinda González L. por su gran cariño y apoyo. Los mejores recuerdos de mi niñez y juventud son gracias a ustedes.

A mis amigos: Ing. Fredy Cervantes e Ing. Oscar García por regalarme ese gran tesoro que es su amistad.

CONTENIDO

Introducción

En los años 70's IBM inventó un lenguaje llamado GML (Lenguaje General de Marcado), con el objetivo de ser utilizado para almacenar grandes cantidades de datos de diversos temas. En esa época, IBM generaba una gran cantidad de información sobre todas las áreas en las que trabajaba e investigaba, por esa razón, necesitaba una manera de guardarla y manipularla. El propósito de GML es el de clasificar y escribir cualquier documento para que posteriormente pueda ser procesado de manera conveniente.

Este lenguaje fue retomado por ISO (Organización Internacional de Normalización) para crear un nuevo estándar. En 1986 ISO trabajó para normalizar el lenguaje, creando de esta forma el SGML (Lenguaje Estándar General de Marcado). IBM, DEC e IRS (Servicio Interno de Crédito de los Estados Unidos), y el departamento de defensa de los Estados Unidos fueron las empresas que más contribuyeron en la especificación desarrollada.

SGML fue la primera tecnología estandarizada que permitía a los usuarios separar los datos de los procesos que los utilizaban. Con SGML, los usuarios pueden ir a través de un proceso llamado análisis de la información, para descubrir la estructura y el contenido de los datos. Un vocabulario llamado definición de tipo de documento (DTD), es desarrollado para hacer el análisis. El DTD define una información de clase, de tal forma que cada DTD es propio por cada conjunto de datos. El DTD indica el contenido de los objetos en el conjunto de información usando una sintaxis precisa llamada modelo de contenido. Debido a que cada conjunto de información tiene diferentes requerimientos y diferentes objetos, el DTD que describe cada conjunto también es diferente.

SGML es un lenguaje que tiene invertido mucho trabajo, capaz de adaptarse a un gran abanico de problemas y a partir de él se han creado los siguientes sistemas para almacenar información; otorga a las compañías la habilidad de obtener, de un mismo documento, diferentes tipos de información aplicando diferentes procesos. Un documento SGML consiste en un archivo de código ASCII con etiquetas y contenido. Una aplicación que procesa el documento, lee el documento y determina la estructura de la información definiendo las etiquetas y observando el contenido

Aunque la intención del uso de SGML es buena, éste fue creado en una era de computadoras caras y lentas, por lo que uno de los objetivos era el de utilizar un conjunto de características que hicieran a los documentos SGML lo más pequeños posibles. Esta minimización hace que SGML sea complejo en su procesamiento y ocasiona que las aplicaciones que leen archivos SGML bajen su rendimiento.

Por otro lado, en el año 89, para el ámbito de la red Internet, un usuario llamado Berners-Lee que había conocido el lenguaje de etiquetas y los hiperenlaces creó, con ayuda de SGML, un nuevo lenguaje llamado HTML, que fue utilizado para un nuevo servicio de Internet que en nuestros días es muy popular: la WWW. Este lenguaje fue adoptado rápidamente por la comunidad. Varias organizaciones comerciales crearon sus propios visores de HTML y compitieron entre ellos para hacer el visor más avanzado, inventando etiquetas como su propia voluntad les indicaba. Desde el 96 hasta hoy una organización llamada W3C ha tratado de poner orden en el HTML y establecer sus reglas y etiquetas para que sea un estándar. Sin embargo el HTML creció de una manera un poco descontrolada.

HTML es muy utilizado para el intercambio de documentos electrónicos a través de la WWW, y aunque es aceptado y utilizado mundialmente, su propósito no es el de intercambiar información entre aplicaciones, sino, el de entregar información al usuario final, utilizar hiperenlaces y presentar dicha información en un formato específico.

Pronto se tuvo la necesidad de un estándar que fuera más sencillo, rápido y práctico tal como HTML, pero extensible como SGML. Además de que fuera compatible con las herramientas, protocolos y técnicas que ya se estaban usando.

En el año de 1998 el W3C empezó el desarrollo de XML (Lenguaje Extensible de Marcado). En este lenguaje se ha pensado mucho más y muchas personas con grandes conocimientos en la materia están trabajando todavía en su gestación. Este grupo diseño XML teniendo 10 objetivos en mente:

1. XML será usado directamente en Internet
2. XML podrá soportar una gran variedad de aplicaciones.
3. XML será compatible con SGML.
4. Será fácil escribir programas que procesen documentos XML
5. El número de características opcionales en XML tendrá que mantenerse absolutamente mínima, idealmente ninguna.
6. Los documentos XML serán legibles y claros para los humanos.
7. El diseño XML deberá prepararse rápidamente.
8. El diseño de XML deberá ser formal y conciso.
9. Los documentos XML deberán ser fáciles de crear.
10. Hacer marcadores breves en XML será de mínima importancia.

XML permite a los usuarios crear sus propios lenguajes de marcado. Los lenguajes que son usados para crear lenguajes de marcado son comúnmente conocidos como metalenguajes de marcado.

XML es una recomendación técnica de W3C. XML pertenece a W3C, no a una compañía en particular, por lo que los usuarios no se encuentran sujetos a una sola plataforma o a un lenguaje de desarrollo en particular. Además es gratis, esto quiere decir que su uso no implica el pago de ningún derecho de autor.

XML es fácil de usar y aprender. Se encuentra ideado para ser utilizado sobre Internet. Como con los documentos SGML, los archivos XML son totalmente hechos cadenas de texto, lo que implica que pueden pasar fácilmente entre muros de fuego y puedan ser enviados sobre las redes existentes.

XML es una versión simplificada de SGML, optimizada para ser utilizada en Internet. Al igual que SGML, XML permite crear un conjunto propio de elementos (etiquetas o marcadores) cuando se esta escribiendo un documento. También, al igual que SGML, un individuo o un comité de estándares pueden definir una aplicación XML (también denominada vocabulario), el cual es un conjunto de elementos y una estructura de documento de propósito general, que peden utilizarse para describir documentos de un determinado tipo (como documentos que contienen fórmulas matemáticas u órdenes de compra).

La simplicidad de XML radica en que su Sintaxis ofrece menos opciones que SGML, haciendo de esta manera que la lectura de documentos XML se más sencilla para los humanos.

Hay que tener en cuenta que XML no hace nada por sí mismo. Es un metalenguaje que permite definir lenguajes específicos para cada aplicación (para cada uso).
XML simplemente proporciona un mecanismo que permite almacenar e intercambiar la información de una forma estructurada y en un formato comprensible por aplicaciones situadas en sistemas heterogéneos, tal y como se muestra en la figura 1:

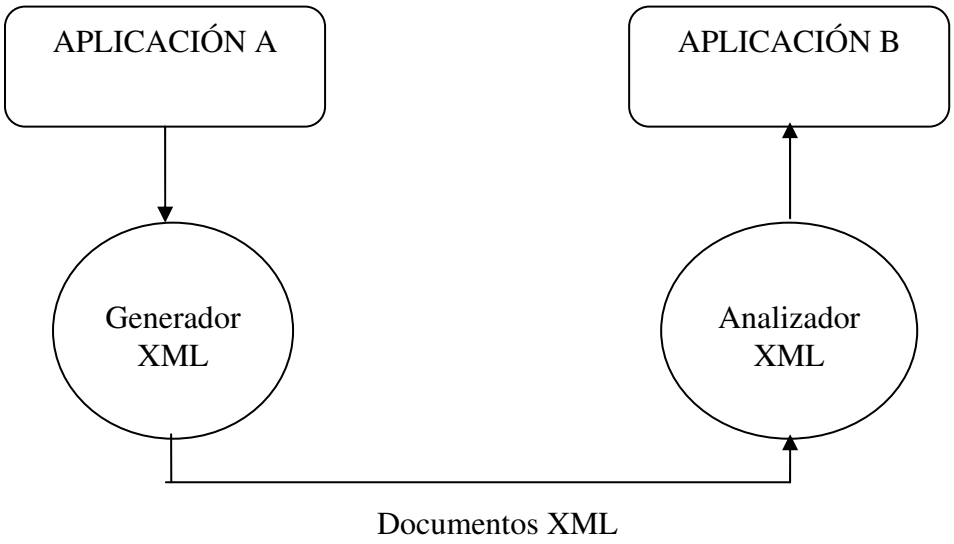

Figura 1: Comunicación de dos aplicaciones por medio de documentos XML

El concepto de documento XML es ligeramente distinto del concepto tradicional de documento (texto básicamente). Un documento XML puede contener cualquier tipo de información y estructurada de muy diversas formas, siguiendo siempre las reglas impuestas para cada dialecto o lenguaje XML. Es decir un documento XML puede ser una base de datos con información sobre clientes, otro documento XML puede ser una ecuación matemática, otro puede ser un libro, etc.

Las reglas que marcan las posibles estructuras de los documentos XML (a partir de la especificación XML) son un compromiso entre flexibilidad (permite construir estructuras muy complejas si así se desea) y sencillez de procesamiento (lo que permite desarrollar software rápido y barato para validar, analizar y extraer información de los documentos XML).

Capítulo I: Creación de documentos bien formados

Antes de que un documento XML pueda ser utilizado, lo primero que debe de cumplir es el acatamiento de ciertas reglas que indican que el documento es apto para su procesamiento. Esta es la primera condición para que un documento XML sea procesado dentro del contexto de una aplicación.

Reglas de XML para la creación de documentos bien formados

Para que un documento XML se encuentre bien formado, debe apegarse a ciertas reglas que indican las partes que componen al documento. Estas reglas no influyen en la flexibilidad de los documentos, están enfocadas a reglamentar la estructura y Sintaxis que éstos deben de cumplir, y son las siguientes:

- Las etiquetas XML tendrán que cerrarse en el mismo orden en que se hayan ido creando.
- Todas las etiquetas XML se constituyen por una etiqueta de apertura y una de finalización, que tendrá idéntico nombre que la de apertura, pero comenzando por el carácter de diagonal (/). Es posible crear etiquetas XML únicas que actúen como apertura y finalización al mismo tiempo. Estas etiquetas deberán terminar con el carácter de diagonal, por ejemplo: <libro nombre="Cumbres borrascosas" />
- Los identificadores de etiquetas son sensibles a la utilización de caracteres en minúsculas y mayúsculas, por esta razón, la etiqueta de apertura y finalización de un mismo elemento, como todas las etiquetas iguales de un documento deberán guardar esta similitud en lo que respecta al uso de letras mayúsculas y minúsculas.
- Todo documento se basa en una estructura jerárquica, en forma de árbol, en la que siempre se encuentra un primer elemento llamado raíz. El hecho de indicar que los documentos se basan en una estructura jerárquica quiere decir que el conjunto de etiquetas que componen el documento se anidan unas dentro de otras en una estructura de árbol.
- Las etiquetas XML siempre aparecen marcadas por un carácter de inicio (<) y un carácter de finalización (>)
- Todas las etiquetas XML se constituyen por una etiqueta de apertura y una de finalización, que tendrá idéntico nombre que la de apertura, pero que comenzará por el carácter de diagonal (/).
- Todos los elementos que constituyen un documento XML, deberán estar marcados por una etiqueta.
- Las etiquetas de XML se componen del nombre de la etiqueta y de una serie de atributos. Esta es una forma de expandir las posibilidades de XML, ya que dan la vuelta a un modelo puramente jerárquico, permitiendo que cada etiqueta pueda contener una serie de valores dentro de ella sin tener que construir para ello una nueva etiqueta anidada.
- El valor de los atributos de una etiqueta debe ir expresado entre comillas, pudiendo ser estas comillas dobles o comillas sencillas.

Partes de documentos bien formados

Un documento XML bien formado esta compuesto por dos partes principales: el prólogo (aunque éste es un elemento opcional) y el elemento documento o elemento raíz.

El prólogo contiene información sobre el documento, como lo es la versión XML utilizada.

El elemento raíz es la sección que contiene a toda la información del documento y corresponde a la etiqueta que se encuentra en el nivel más alto de la jerarquía de elementos.

En el siguiente ejemplo podemos ver el listado de un documento XML válido que representa un inventario de libros que se maneja en una librería

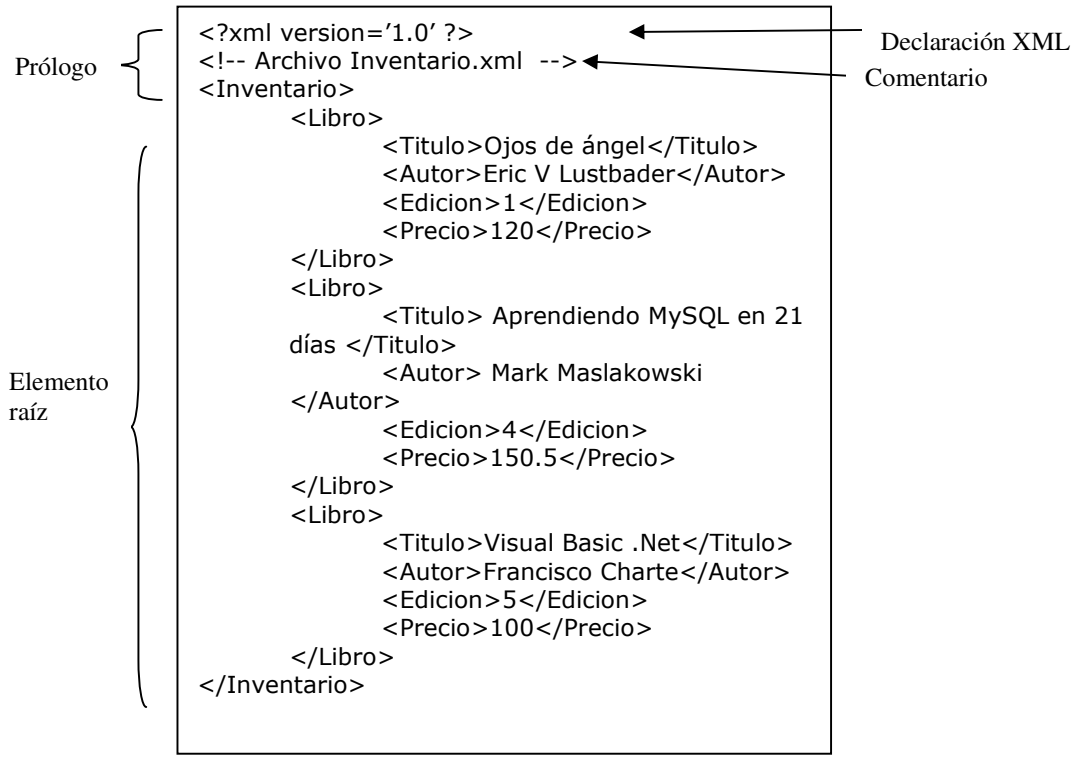

Figura 2. Partes de un documento XML bien formado

En la figura 2 se observa que en la parte del prólogo existe la etiqueta `<?xml version='1.0'>`, el atributo indica la versión de XML del documento. Además del atributo de la versión de XML, es posible incluir otros dos atributos: a) *encoding*, que sirve para especificar la codificación empleada para el documento y, b) *standalone*, con este atributo se indica la relación del documento con respecto a su definición de tipo de documento (en el siguiente capítulo se verá a detalle en qué consiste la definición de tipo de documento o DTD); toma el valor de *yes* cuando la especificación de tipo de documento se encuentra dentro del propio documento, y *no* cuando el documento tiene una definición de tipo de documento externo, es decir, en otro archivo.

El elemento raíz corresponde, en este caso, a la etiqueta "Inventario". Dentro del elemento raíz se encuentran más elementos, que corresponden a cada libro del inventario, cada uno de estos, con los elementos Titulo, Autor, Edición y Precio.

Cabe mencionar que en un documento XML pueden existir espacios en blanco, estos se encuentran formados por uno o más espacios, tabuladores, retornos de carro o caracteres de avance de línea. Para diseñar un documento que sea más legible al ser humano, se pueden añadir libremente espacios en blanco entre las marcas XML (como marcadores de inicio,

marcadores de fin, comentarios e instrucciones de procesamiento) y también pueden ir entre los atributos de una etiqueta. La aplicación que procese el documento ignorará por completo estos espacios en blanco, a menos que se encuentre dentro de un elemento que directamente contenga datos de caracteres, en ese caso, la aplicación que procese el documento pasará el espacio en blanco a la aplicación que lo utiliza, como parte de los datos de caracteres del elemento.

El prólogo del documento XML de la figura 2 contiene un ejemplo de algunos de los elementos permitidos dentro del prólogo. Sin embargo, todos los elementos del prologo son opcionales, aunque la especificación de XML indica que se deberá incluir la declaración de XML. De ahí que el propio prólogo sea opcional y que un documento que solo contiene un elemento de documento (llamado documento minimalista), sea considerado como un documento bien formado. Un ejemplo de un documento minimalista es el siguiente:

<elemento>Documento minimalista </elemento>

Como se observa en la figura 2, los elementos se encuentran anidados, en este caso solo existe un máximo de tres niveles, pero no existe ninguna restricción a este respecto. Lo que si se respeta (y que deberá respetarse en todos los documentos XML) es que contiene exactamente un elemento de nivel superior (el elemento documento o elemento raíz), en este caso la etiqueta Inventario.

Otro de los aspectos que podemos notar en la figura 2, es que los elementos se encuentran adecuadamente anidados, es decir, si un elemento (que se encuentra delimitado por un marcador de inicio y un marcador de fin) comienza dentro de otro elemento, también deberá finalizar dentro del mismo elemento.

La especificación de XML requiere que siempre se incluyan los marcadores de inicio y de fin, la única excepción es un elemento sin contenido, para el cual se utiliza un marcador de elemento vacío, como se muestra a continuación:

<libro titulo=”Aprendiendo MySQL en 21 días” autor=”Mark Maslakowski” edicion=”4” precio=”150.5”/>

O su equivalente:

<libro titulo=”Aprendiendo MySQL en 21 días” autor=”Mark Maslakowski” edicion=”4” precio=”150.5”> </libro>

Cuando se añada un elemento al documento XML, se podrá seleccionar cualquier nombre del tipo que se desee, siempre y cuando se cumplan las siguientes reglas:

- El nombre debe comenzar con una letra o un guión bajo, seguido por cero o más letras, dígitos, puntos, guiones o guiones bajos.

- La especificación XML indica que los nombres de los tipos de elementos que comienzan con el prefijo xml (en cualquier combinación de mayúsculas y minúsculas) están reservados para estandarización. Aunque las versiones de algunos exploradores de Internet, no obligan esta restricción, es preferible no utilizar este prefijo para evitar futuros problemas.

Los elementos pueden contener información adicional en su propia definición, a parte del nombre, esta información se conoce como especificación de atributo. La especificación de atributo es una pareja nombre-valor relacionada con el elemento, por ejemplo:

<Empleado clave=”124” nombre=”Martha” apellidoPaterno=”García” apellidoMaterno=”López”></Empleado>

En este caso, observamos que se incluyen cuatro atributos: nombre, clave, apellidoPaterno y apellidoMaterno. En muchas ocasiones los elementos anidados pueden convertirse en atributos del elemento padre y viceversa.

El contenido de un elemento se compone por el texto que se encuentra entre los marcadores de inicio y de fin. Este texto puede corresponder, ya sea, a otro elemento, o a datos de caracteres los cuales componen el contenido informativo del elemento. Por ejemplo, regresando a la figura 2, el elemento Libro contiene otros elementos: Titulo, Autor, Edición, Precio; y estos a su vez contienen datos de caracteres. Los datos de caracteres pueden ser cualquier carácter, excepto el símbolo &, el símbolo < o la cadena]]>

En la definición de un elemento es posible combinar tanto atributos, elementos anidados y datos de caracteres, quedando a criterio del responsable de la definición del documento, tal y como se muestra a continuación:

```
<Empleado clave=”124”>
        Martha
        <apellidoPaterno>García</apellidoPaterno>
        <apellidoMaterno>López</apellidoMaterno>
</Empleado>
```

Las reglas para nombrar a los atributos son las mismas que aplican para nombrar a los elementos, además de esto, hay que considerar que solo es posible que un atributo debe aparecer una vez en el mismo elemento.

Las reglas para los valores de los atributos son las siguientes:

- Los caracteres para delimitar los valores pueden ser las comillas dobles o sencillas
- No se puede incluir el carácter de menor que (<) en el valor
- No se puede incluir el carácter de & en el valor
- No se puede incluir el carácter de delimitación en el valor

Instrucciones de procesamiento, comentarios y secciones CDATA

Instrucciones de procesamiento

Generalmente las instrucciones de procesamiento se utilizan para incrustar información específica de una aplicación en un documento XML, esta información es utilizada por el procesador de XML (aplicación que lee e interpreta el contenido del documento para que pueda ser manipulado y/o visualizado). Por ejemplo, cuando abrimos un documento XML en un navegador de Internet (éste último proporciona el procesador de XML), es posible que dentro del documento XML visualizado se haya hecho referencia hacia un archivo de hoja de estilos en cascada (CSS) con la finalidad de que se aplique un formato en el momento de visualizar los datos del documento XML, en este caso, el procesador de XML recuperará la referencia hacia el archivo de hoja de estilos en cascada (mismo que obtendrá de una instrucción de procesamiento correspondiente) y lo aplicará en la visualización del documento XML.

El formato de una instrucción de procesamiento es el siguiente:

<? *destino instrucción*?>

Donde *destino* es el nombre de la aplicación a la que se dirige la instrucción e *instrucción* es la información que se pasa a la aplicación que funciona como procesador de XML.

Para el nombre de destino se permite cualquiera, siempre y cuando cumpla las reglas que se enlistan a continuación:

- El nombre deberá comenzar con una letra o un guión bajo, seguido por cero o más letras, dígitos, puntos, guiones o guiones bajos
- El nombre xml, en cualquier combinación de mayúsculas o minúsculas, está reservado.

La instrucción puede estar formada por cualquier secuencia de caracteres, menos la pareja de caracteres ¿>, la cual se encuentra reservada para terminar la instrucción de procesamiento.

Las instrucciones de procesamiento dependen del procesador de XML, por ejemplo, existen instrucciones de procesamiento estándar reservadas para ser utilizadas por el navegador de Internet de Microsoft o se pueden definir nuevas instrucciones que sirvan para proporcionar información adicional a un procesador XML en particular.

Las instrucciones de procesamiento se pueden situar en cualquier lugar de un documento XML, siempre y cuando se encuentren fuera de otras marcas, es decir, se pueden encontrar en el prólogo, antes del elemento documento, o dentro del contenido de un elemento, todo depende de las necesidades de la misma aplicación.

Comentarios

Los comentarios se incluyen en un documento XML con la finalidad de hacer al documento más claro y aumentar su legibilidad para los seres humanos. Estos son ignorados por el procesador de XML.

Los comentarios comienzan con los caracteres <!—y finalizan con los caracteres -->, y pueden ser aplicados a una o más líneas de código dentro del documento. Dentro de las marcas que delimitan a los comentarios se puede insertar cualquier carácter, o secuencia de estos, que se desee, a excepción del guión doble, por ejemplo:

<!-- este es un comentario que sirve para aclarar la interpretación del documento a los seres humanos, es válido incluir el carácter > como el & (pero no el guión doble) -->

Los comentarios, al igual que las instrucciones de procesamiento, pueden insertarse en cualquier parte del documento XML, incluyendo al prólogo, menos dentro de cualquier marca, por ejemplo, los siguientes comentarios no son correctos:

<libro <!--este comentario se encuentra en un lugar prohibido --> >
<? Xml version=”1.0” <!--este comentario se encuentra en un lugar prohibido --> ?>

Secciones CDATA

Las secciones CDATA se usan para transformar en secuencias de escape bloques de texto que contienen caracteres que de otro modo serían considerados como código, por ejemplo, los caracteres & y <, los cuales no pueden insertarse directamente.

Las secciones CDATA comienzan con los caracteres <![CDATA][y finalizan con los caracteres]]>. Dentro de estos grupos de caracteres es posible escribir cualquier carácter, menos los

caracteres]], que son los que finalizan la sección CDATA. Todos los caracteres que se encuentran dentro de la sección serán tratados como una parte de los datos de caracteres del elemento en donde se encuentren, y no como marcas de XML. Las secciones CDATA no pueden anidarse.

Las secciones CDATA solo pueden aparecer dentro del contenido de un elemento. Al igual que los comentarios y las instrucciones de procesamiento, no pueden aparecer dentro de ninguna marca XML, pero a diferencia de estas, no pueden estar en el prólogo ni antes o después de la definición del elemento documento.

Capítulo II: Documentos XML válidos

Una vez que el documento XML está bien formado, la mayoría de las veces es necesario que también se encuentre constituido según ciertas normas especificadas en el contexto de la aplicación en donde se va a utilizar. Un documento que cumple con estas normas, es llamado documento válido.

Existen requisitos que siempre deben de cumplirse en un documento válido, los cuales se mencionan a continuación:

- Se debe incluir una declaración de tipo de documento adecuada dentro del prólogo del documento. Esta declaración de tipo de documento, contiene a su vez una definición de tipo de documento (DTD) y esta es la que indica la estructura del mismo documento XML, la declaración de tipo de documento es de la siguiente forma: <!DOCTYPE *elemento_raiz DTD*>, donde *elemento_raiz* corresponde al nombre del elemento documento y *DTD* es la definición de tipo de documento (la cual se explica más adelante)

- El documento XML debe de adaptarse a la estructura definida en la DTD de la declaración de tipo de documento.

El hecho de que el documento XML se apegue a la estructura definida en una declaración de tipo de documento en particular, permite que el documento XML se adapte a una estructura específica o a un conjunto de estándares. En el caso de que el documento este adaptado a una estructura específica, no será necesario escribir diferentes aplicaciones de procesamiento de documentos XML por el hecho de que existan dos documentos diferentes en su estructura, ya que esta situación nunca se dará.

Supongamos que se desea tener la posibilidad de intercambiar órdenes de compra entre tiendas de equipo de cómputo y sus proveedores mayoristas. Sin no existiera una definición de tipo de documento para los documentos XML que manejan la información de las órdenes de compra, sería muy probable que para cada cliente y proveedor utilizara una estructura arbitraria e incluyera los elementos que creyera conveniente manejar dentro de su documento XML, si se da el caso, tanto los clientes como los proveedores tendrían que contar con un procesador XML por cada proveedor (en el caso de los clientes) y por cada cliente (en el caso de los proveedores), lo que deriva en una inversión mayor. Sin embargo, si se incluye una definición de tipo de documento para los documentos XML de órdenes de compra, todos los documentos generados tendrían que apegarse a tal definición de documento y un solo procesador XML se utilizaría para todos los documentos, sin importar de qué cliente o proveedor provengan.

DTD (Definición de Tipo de Documento)

Una definición de tipo de documento es un conjunto de especificaciones que describen la estructura del documento XML, declarando todos los tipos de los elementos del documento, el orden de cada tipo del elemento y cualquier atributo, entidad, anotación, comentario o referencia incluida en el mismo.

Cuando un documento XML no contiene una DTD, el procesador XML solo se cerciora de que éste sea un documento bien formado, por el contrario, cuando se asocia un DTD al documento

XML, el procesador XML también revisa que dicho documento se apegue a la DTD, si es el caso, entonces se tratará de un documento válido.

Existen dos tipos de DTD que pueden ser aplicados a un documento XML: DTD interno, o subconjunto DTD interno y DTD externo, o subconjunto DTD externo. Ambos definen la estructura que deberá presentar un documento o conjunto de documentos XML, pero se diferencian por el lugar en donde se define esta especificación de documento. Ambos comparten un elemento llamado declaración DOCTYPE. Esta declaración se utiliza dentro de los documentos XML para informar al procesador de XML que utilizará dicho documento XML. El uso de un subconjunto DTD externo es necesario cuando existe más de un documento que usan un mismo DTD, de esta forma, cada documento está capacitado para hacer referencia a un único archivo DTD en su declaración de tipo de documento.

Es posible utilizar un subconjunto DTD interno, un subconjunto DTD externo o uno interno y externo simultáneamente, la siguiente tabla muestra el formato de la declaración de tipo de documento que aplica para cada caso:

Declaración de tipo de documento	Formato
Interno	<!DOCTYPE *elemento_raiz DTD*>
Externo	<!DOCTYPE *elemento_raiz* SYSTEM *URI_Documento*>
Interno y externo	<!DOCTYPE *elemento_raiz* SYSTEM *URI_Documento DTD*>

Cuando se utiliza un subconjunto DTD externo y un subconjunto DTD interno simultáneamente, el procesador XML combina sus contenidos mezclándolos, de esta forma, se generará una DTD completa a partir de ambos. Cuando se declara un atributo con el mismo nombre y tipo de elemento más de una vez, el procesador XML utiliza la primera declaración e ignora las siguientes y siempre el subconjunto DTD interno tiene preferencia sobre el conjunto DTD externo. También es posible ignorar una parte del subconjunto DTD externo, esto se hace empleando la sección IGNORE, la cual comienza por los caracteres <!IGNORE[y finaliza con los caracteres]]>

Una DTD está formada por un paréntesis cuadrado izquierdo, o corchete izquierdo, seguido de una serie de declaraciones de marcas y finalizado por un paréntesis cuadrado derecho, o corchete derecho. Las declaraciones de marcas son las siguientes:

- Declaraciones de los tipos de elementos
- Declaraciones de lista de atributos
- Declaraciones de entidad
- Declaraciones de notación
- Instrucciones de procesamiento.
- Comentarios
- Referencias de entidad de parámetro.

Declaraciones de los tipos de elementos

Las declaraciones de los tipos de elementos especifican los elementos que puede contener el documento XML, también indican el contenido y ordenamiento de los mismos. En conjunto,

todas las declaraciones de tipos de elemento, constituyen la estructura lógica del documento XML. Su formato es el siguiente:

<!ELEMENT *Nombre Especificación_contenido*>

Donde *Nombre* corresponde al nombre del tipo de elemento que se declara y *Especificación_contenido* define lo que el elemento puede contener y puede ser cualquiera de las siguientes formas:

Contenido cualquiera: Usando la palabra reservada ANY, se indica que el contenido del elemento en cuestión puede incluir cualquier tipo de contenido válido.

Contenido Vacío: Cuando se usa la palabra reservada EMPTY se indica que el elemento debe estar vacío y no podrá tener ningún tipo de contenido

Contenido de elemento o contenido hijo: Al indicar este tipo de contenido, el elemento puede contener elementos hijos sin importar las repeticiones o el orden, pero a diferencia de ANY, no puede incluir datos de caracteres entremezclados.

Para cada especificación de contenido del tipo contenido de elemento existe un modelo de contenido. El modelo de contenido indica los tipos de elementos hijos permitidos y también su orden.

La forma más básica del modelo de contenido es el llamado formato de secuencia, el cual indica que el elemento deberá tener una secuencia específica de elementos hijos. Por ejemplo, podemos indicar que el elemento Libro debe de tener un elemento titulo, autor y editorial en ese orden, de la siguiente manera:

<!ELEMENT Libro (titulo,autor,editorial)>

Para que un documento XML se apegue a este tipo de definición no debe omitir ningún elemento hijo, debe respetar el orden y no puede repetir un elemento hijo más de una vez.

Otra forma de modelo de contenido para el contenido de elemento es el formato de elección, el cual indica que el elemento puede contener cualquiera de un conjunto de posibles elementos hijos que son separados por caracteres |. Por ejemplo, para indicar que el elemento Libro puede tener un elemento titulo o autor o editorial (pero solo uno de ellos), el modelo de contenido del elemento Libro queda especificado de la siguiente manera:

<!ELEMENT Libro(titulo|autor|editorial)>

Los formatos de modelo anteriores pueden modificarse con el uso de los caracteres ?, +, y *. Su significado es el siguiente:

- Carácter de cierre de interrogación: significa que puede incluirse uno o ningún elemento
- Carácter de suma: significa que puede incluirse uno o más elementos
- Carácter asterisco: significa que puede incluirse ninguno o más elementos.

Por ejemplo, si se desea indicar que el elemento titulo puede o no aparecer; y si aparece que sea solo una vez, que el elemento autor puede aparecer una o más veces y que el elemento editorial puede o no aparecer; y que si aparece lo puede hacer más de una vez; el modelo de contenido del elemento Libro quedaría especificado de la siguiente manera:

<!ELEMENT Libro(titulo?,autor+,editorial*)>

También es posible utilizar los caracteres ?,+ y * para modificar todo el modelo de contenido, solo con situar el carácter después del paréntesis final, esto aplicaría para todos los elementos hijos, de tal forma que si queremos que el documento XML pueda incluir uno o más de los elementos hijo de cualquier tipo, en cualquier orden, el modelo de contenido del elemento Libro queda de la siguiente manera:

<!ELEMENT Libro(titulo,autor,editorial)+>

Si un modelo de contenido de elección se anida dentro de un modelo de contenido de secuencia o viceversa se da mayor flexibilidad. Por ejemplo, si se desea que el elemento Libro tenga un elemento hijo titulo, alguno de los elementos tema o categoría, y el elemento hijo editorial, el modelo de contenido del elemento Libro queda de la siguiente manera:

<!ELEMENT Libro (titulo,(tema|categoria),editorial)>

Contenido de datos de caracteres: Cuando un elemento contiene exclusivamente datos de caracteres se utiliza el modelo de contenido (#PCDATA). Si deseamos que el elemento hijo titulo contenga datos de caracteres, su modelo de contenido queda de la siguiente forma:

<!ELEMENT titulo(#PCDATA)>

Los datos de caracteres también se pueden combinar dentro del modelo de contenido para especificar que un determinado elemento debe contener datos de caracteres junto con más elementos hijos, siguiendo la sintaxis, tanto de contenido de elemento como contenido de datos de caracteres. Por ejemplo, para que un elemento titulo contenga datos y caracteres y cero o más elementos hijos subtítulo, se tiene que especificar un modelo de contenido como se muestra a continuación:

<!ELEMENT titulo (#PCDATA | subtitulo) >

En la siguiente tabla se muestran modelos de contenido que corresponden a los ejemplos mencionados anteriormente y un elemento válido correspondiente en el documento XML.

Modelo de contenido	Elemento XML
<!ELEMENT Libro ANY>	<Libro>El elemento puede tener cualquier contenido ya que de esta forma se encuentra especificado en el modelo de contenido del elemento</Libro>
<!ELEMENT Libro EMPTY>	<Libro></Libro>
<!ELEMENT Libro EMPTY>	<Libro/>
<!ELEMENT Libro (titulo,autor,editorial)>	<Libro> <titulo/><autor/><editorial/> </Libro>
<!ELEMENT Libro(titulo|autor|editorial)>	<Libro><titulo/></Libro>
<!ELEMENT Libro(titulo?,autor+,editorial*)>	<Libro> <autor></autor> <editorial/> </Libro>
<!ELEMENT Libro(titulo,autor,editorial)+>	<Libro> <autor>Primer autor</autor> <autor>Segundo autor</autor> <autor>Tercer autor</autor>

	<editorial>Editorial</editorial> <titulo>Titulo del libro</titulo> </Libro>
<!ELEMENT Libro(titulo,autor,editorial)?>	<Libro> <titulo>Titulo del libro</titulo> <autor>Autor</autor> </Libro>
<!ELEMENT Libro(titulo,autor,editorial)*>	<Libro> <titulo>Titulo del libro</titulo> <autor>Primer autor</autor> <autor>Segundo autor</autor> <autor>Tercer autor</autor> </Libro>
<!ELEMENT titulo(#PCDATA)>	<titulo> El elemento titulo solo debe contener datos de caracteres </titulo>
<!ELEMENT titulo (#PCDATA | subtitulo) >	<titulo> En este caso se puede incluir un elemento hijo del elemento titulo, llamado subtitulo <subtitulo/> <titulo>

Declaraciones de atributos

Al igual que con los elementos, en los cuales se utiliza un modelo de contenido para especificar los elementos hijos y el orden, para especificar la lista de atributos de un determinado elemento y el tipo de cada uno de ellos se utiliza la declaración de lista de atributos. Esta declaración de lista de atributos es la encargada de definir los nombres de los atributos que pertenecen al elemento, el tipo de datos de cada atributo, y además, de indicar cuales atributos son obligatorios y cuales no lo son. Para los atributos que no son obligatorios se puede definir un valor predeterminado. El formato general para una declaración de lista de atributos es el siguiente:

<!ATTLIST Nombre_Elemento Nombre_atributo Tipo Valor_predeterminado>

Nombre_Elemento corresponde al nombre del elemento asociado con el o los atributos.

Nombre_atributo corresponde al nombre del atributo, Tipo es el tipo de valor que deberá ser asignado al atributo y Valor_predeterminado, se utiliza para especificar si el atributo es o no obligatorio e indicar el valor predeterminado para el atributo.

El tipo de atributo (Tipo) puede especificarse como:

1. Tipo enumerado. El valor del atributo debe de ser una cadena entre comillas que coincida con alguno de los nombres de la lista de nombres que se enumeren, si esto no sucede, se produce un error de validez. Los valores de la lista se escriben entre paréntesis, separados por el carácter |. Estos valores corresponden a un símbolo de nombre (un símbolo de nombre es un nombre formado por una o más letras, dígitos, puntos, guiones o guiones bajos y pueden contener un símbolo de dos puntos, a excepción de la primera posición). Otra forma de indicar la lista de valores enumerados es con el uso de notaciones (una notación se utiliza para describir un formato de datos, la forma de utilizar notaciones se describe más adelante), para esto, se escribe la

palabra reservada NOTATION, seguida por un espacio y una lista entre paréntesis de nombres de notación, separados por el carácter |.

2. Tipo cadena. Cuando un atributo es de tipo cadena, éste puede recibir cualquier valor que sea una cadena entrecomillada. Para declarar un atributo de este tipo se utiliza la palabra reservada CDATA.
3. Tipo simbólico. El atributo de tipo simbólico sirve para especificar un atributo que tiene un valor especial y cumple con una restricción particular. Para indicar que un atributo es de tipo simbólico se utiliza una palabra reservada, esta palabra depende del comportamiento que se desea presente el atributo, las palabras reservadas que se pueden utilizar se muestran en la siguiente tabla:

PALABRA RESERVADA	DESCRIPCION
ID	Significa que el valor del atributo por elemento deberá ser único. El elemento solo puede tener un solo atributo simbólico ID, cuando el atributo sea de este tipo, el valor_predeterminado deberá de ser #REQUIRED o #IMPLIED
IDREF	Significa que el atributo deberá tener un valor que coincida con el valor del atributo ID de otro elemento dentro del mismo documento.
IDREFS	Es igual que con IDREF, sin embargo, en este tipo de atributo se pueden incluir referencias a más de un ID, estas referencias deberán escribirse separadas por espacios en blanco.
ENTITY	El valor del atributo deberá coincidir con el nombre de una entidad no analizada declarada en la DTD. Una entidad no analizada hace referencia a un archivo externo, que por lo general almacenará datos que no representen contenido en formato XML. La declaración de entidades se mostrará más adelante
ENTITIES	Es igual que el tipo de atributo ENTITY, sin embargo, en este tipo de atributo se pueden incluir referencias a más de una entidad. Estos nombres de entidades deben de estar en una cadena entre comillas y separadas una de la otra por un espacio en blanco
NMTOKEN	Corresponde un símbolo de nombre, un símbolo de nombre es un valor formado por una o más letras, dígitos, puntos, guiones o guiones bajos y puede contener un carácter de dos puntos (menos en la primera posición)
NMTOKENS	Al igual que con NMTOKEN, sin embargo, en este tipo de atributo se pueden incluir varios símbolos de nombre que van separados por espacios en blanco y en una cadena entre comillas.

El valor predeterminado (Valor_predeterminado) especifica si el atributo es obligatorio y, en caso de que éste no lo sea, le indica al procesador lo que se deberá hacer cuando el atributo se omita, por ejemplo, dar el valor predeterminado que el procesador utilizará en caso de que no se indique otro valor. Los formatos del valor predeterminado se muestran en la siguiente tabla:

Formato	Descripción
#REQUIRED	Indica que es obligatorio proporcionar un valor del atributo
#IMPLIED	Indica que se puede incluir u omitir el atributo, si se omite, no se dará ningún valor predeterminado al procesador
#FIXED "valor_predeterminado"	Valor_predeterminado es el valor predeterminado del atributo, si se utiliza, el atributo es opcional y si se omite el atributo en el documento, el procesador utilizará este valor como predeterminado. Este valor deberá ser compatible con el tipo del atributo.

"valor_predeterminado"	Al igual que con #FIXED, es otra forma de indicar un atributo opcional con valor predeterminado.

En la siguiente tabla se muestran algunos ejemplos de declaración de lista de atributos y su posible lista de atributos correspondiente en el documento XML.

Declaración de lista de atributos	Lista de atributos en el documento XML.
<!ATTLIST Libro ISBN ID #REQUIRED >	<Libro ISBN="84-481-3341-2">
<!ATTLIST Libro AREA CDATA>	<Libro AREA="Ciencias Sociales">
<!ATTLIST Libro Edición (1| 2 |3) "1">	<Libro Edición ="1"> O <Libro Edición ="2"> O <Libro Edición ="3"> O <Libro>
<!ATTLIST Libro Edición (1 | 2 | 3) "1" ISBN ID #REQUIRED Area CDATA #FIXED "Matemáticas" NPaginas NMTOKEN #REQUIRED>	<Libro Edición="1" ISBN="84-481-3341-2" NPaginas="120">

Declaraciones de entidad

Una entidad es un bloque de texto XML que se maneja como una unidad, de esta forma se puede utilizar en cualquier parte del documento. En la especificación de XML el concepto de entidad se refiere a un documento XML completo, a un subconjunto DTD externo, a un archivo externo y a una cadena entrecomillada definida como una entidad interna en la DTD, sin embargo, para las declaraciones de entidad en una DTD, solo aplican los dos últimos casos.

Clasificación de entidades

Entidad general y de parámetro: Las entidades generales incluyen contenido de documento: texto XML y otros datos, textuales o no, se pueden utilizar dentro del elemento documento. Una entidad de parámetro contiene texto XML, que se puede insertar dentro de la DTD.

Entidad interna y entidad externa: Una entidad interna es aquélla contenida dentro de una cadena entrecomillada. Una entidad externa es aquélla que se encuentra en un archivo separado

Entidad analizada y no analizada. Una entidad analizada es la que está formada por texto XML (datos de caracteres, marcadores o ambos). Cuando se inserta en un documento una referencia a una entidad analizada, la referencia se sustituirá con el contenido de la entidad (también denominado texto de sustitución), que se convertirá en parte integral del documento. El analizador XML recorrerá el contenido de la entidad de la misma manera que analizará el texto que se haya introducido directamente en el documento. Una entidad no analizada es aquélla que puede contener cualquier tipo de datos (datos XML y, especialmente, datos ajenos a XML). Los datos ajenos a XML pueden ser datos textuales (como un título), o no textuales (como los datos gráficos de una imagen). Debido a que las entidades no analizadas generalmente no contienen datos XML, su contenido no se insertará directamente en el documento mediante una referencia de entidad. Sin embargo, se puede asignar el nombre de la

entidad a un atributo de tipo ENTITY o ENTITIES, de manera que la aplicación pueda acceder al nombre y descripción de la entidad y realizar cualquier acción con los datos.

Debido a que las entidades se clasifican de la forma en que se describió arriba, y que cada una de estas clasificaciones incluye dos categorías, las combinaciones posibles nos indican que se pueden tener ocho, tal como se muestra en la figura 3:

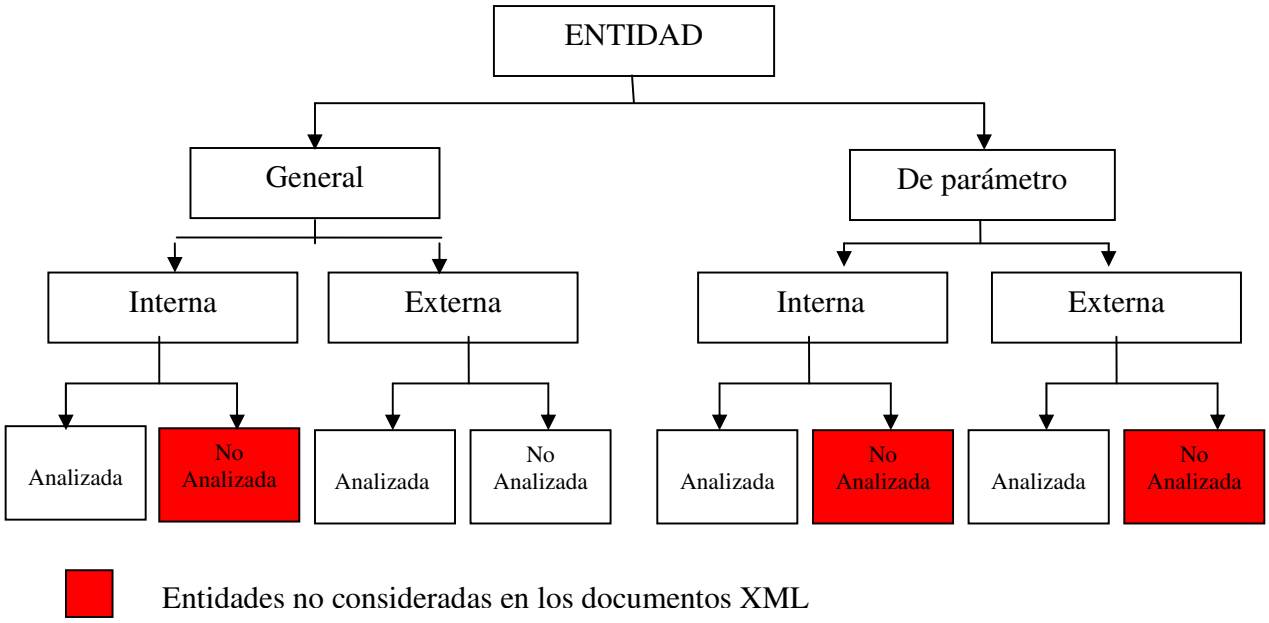

Figura 3. Tipos de entidades

Declaración de entidad general interna analizada

La declaración de una entidad general interna analizada tiene el siguiente formato:

<!ENTITY Nombre_entidad Valor_entidad>

Donde Nombre_entidad es el nombre de la entidad, las reglas para el nombre de la entidad son las siguientes:

- El nombre deberá comenzar por una letra o guión bajo, seguido por cero o más letras, números, puntos, guiones y guiones bajos
- La entidad puede tener en el documento el mismo nombre que una entidad parámetro, también puede tener el mismo nombre que un elemento o un atributo, sin que exista ningún conflicto con los nombres.
- Se distingue el uso de letras mayúsculas y minúsculas

Valor_entidad es el valor de la entidad. Este valor será una serie de caracteres, delimitados por comillas (llamada cadena entrecomillada o literal). Este valor deberá apegarse a las siguientes reglas:

- Los delimitadores de cadena pueden ser comillas dobles y comillas simples

- La cadena no puede contener el mismo tipo de comillas que se haya usado para delimitarla
- La cadena no puede incluir el carácter &, este es utilizado para comenzar una referencia de entidad general o de carácter.
- La cadena no puede incluir el signo de porcentaje
- El contenido de la cadena deberá ser válido para la ubicación en que se pretenda insertar la entidad.

En el siguiente ejemplo se muestra una definición de tipo de documento que define una entidad general interna analizada

```
<!DOCTYPE LIBRO
[
  <!ELEMENT LIBRO (TITULO,AUTOR,TEMA)>
  <!ELEMENT TITULO (#PCDATA)>
  <!ELEMENT AUTOR (#PCDATA)>
  <!ELEMENT TEMA (#PCDATA)>
  <!ENTITY TITULO    "Titulo:">
]
>
```

El documento XML puede estar formado de la siguiente manera:

```
<LIBRO>
  <TITULO>
     &TITULO; Lenguajes de programación
  </TITULO>
  <AUTOR> Pratt Zelkowits </AUTOR>
  <TEMA> Ingeniería de software </TEMA>
</LIBRO>
```

El procesador XML sustituye la referencia de entidad por su contenido, de esta forma, el ejemplo anterior es equivalente al siguiente código XML:

```
<LIBRO>
  <TITULO>
     Titulo: Lenguajes de programación
  </TITULO>
  <AUTOR> Pratt Zelkowits </AUTOR>
  <TEMA> Ingeniería de software </TEMA>
</LIBRO>
```

Como se observa en el ejemplo anterior, la referencia hacia la entidad general interna analizada es de la forma &Nombre_entidad; (en el ejemplo &TITULO;), y esta referencia puede estar en el contenido de los elementos, en los valores de los atributos y en el caso de la definición de tipo de documento, puede estar en el valor de una declaración de entidad interna.

Declaración de entidad general externa analizada

La declaración de una entidad general externa analizada tiene el siguiente formato:

```
<!ENTITY Nombre_entidad SYSTEM Literal_sistema>
```

Donde Nombre_entidad es el nombre de la entidad, las reglas que aplican para este valor son las mismas que con las entidades generales internas analizadas

Literal_sistema es una literal que describe la ubicación del archivo que contiene los datos de la entidad (URI), y su valor se delimita por comillas dobles o comillas simples, puede incluir cualquier carácter menos el tipo de comillas que se utilice como delimitador. Especifica el identificador uniforme de recursos (URI) del archivo que contiene los datos de la entidad. La URI puede ser totalmente calificada, es decir; que especifican la dirección completa del recurso (absoluta), o puede ser parcial, la cual indica la ubicación del recurso en forma relativa a la ubicación del documento XML. Cabe mencionar que el archivo de la entidad externa sólo debe contener componentes que se pueden incluir en un elemento, como por ejemplo elementos anidados o los datos de caracteres.

En el siguiente ejemplo se muestra una definición de tipo de documento que define una entidad general externa analizada

```
<!DOCTYPE Alumno
[
<!ELEMENT Alumno (Nombre,Grupo,Escuela)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Grupo (#PCDATA)>
<!ELEMENT Escuela ANY>
<!ENTITY NombreEscuela  SYSTEM "NombreEscuela.xml">
]
```

El contenido del archivo NombreEscuela.xml es el siguiente:

```
<Nombre>Escuela Secundaria Federal José Martí</Nombre>
```

El documento XML final puede contener un elemento llamado Escuela de la siguiente manera:

```
<Escuela>&NombreEscuela</Escuela>
```

El procesador XML lo interpretará como:

```
<Escuela>
        <Nombre>
        Escuela Secundaria Federal José Martí
        </Nombre>
</Escuela>
```

Al igual que con la entidad general interna analizada, la entidad externa analizada se declara de la forma &Nombre_entidad; y puede encontrarse en el contenido de un elemento y en el valor de una declaración de entidad interna (dentro de la definición de tipo de documento).

Declaración de una entidad general externa no analizada

Esta declaración tiene el siguiente formato

```
<!ENTITY Nombre_entidad SYSTEM Literal_sitema NDATA Nombre_notacion>
```

Nombre_entidad es el nombre de la entidad, las reglas que aplican para este valor son las mismas que con las entidades generales internas analizadas.

Literal_sistema es un literal de sistema, que describe la ubicación del archivo que contiene los datos de la entidad, su funcionamiento es idéntico que con los literales empleados en las ubicaciones de las entidades generales externas analizadas.
NDATA indica que el archivo de la entidad contiene datos no analizados

Nombre_notacion es el nombre de una notación declarada en la DTD. La notación describe el formato de los datos contenidos en el archivo del a entidad o puede proporcionar la ubicación de un programa que pueda procesar dichos datos. Para describir el formato de datos, es necesario que la notación proporcione la dirección de una descripción del formato, la dirección de un programa que pueda gestionar datos en ese formato o una simple descripción de formato, el formato general de una notación es el siguiente:

<!NOTATION Nombre SYSTEM Literal_sistema>

Donde Nombre es el nombre de la notación, por lo general se escoge un nombre descriptivo y que comience con una letra o guión bajo, seguido por cero o más letras, números, puntos, guiones o guiones bajos.

Literal_sistema es un literal de sistema donde se describe el formato de los datos, se puede delimitar por comillas dobles o simples y puede incluir cualquier carácter, a excepción del tipo de comillas que se utilice para delimitarlo. Esta información no será utilizada por el procesador XML, sino que se pasará a la aplicación que haga uso del documento.

El siguiente ejemplo muestra la DTD de un archivo XML que podrá utiliza una entidad general externa no analizada en el atributo de un elemento:

```
<!DOCTYPE Alumno
[
<!ELEMENT Alumno (Nombre,Grupo,Escuela)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Grupo (#PCDATA)>
<!ELEMENT Escuela (#PCDATA)>
<!ATTLIST Alumno Foto ENTITY #REQUIRED>
<!ENTITY Foto  SYSTEM "Foto.gif" NDATA imagen>
<!NOTATION imagen SYSTEM "Pbrush.exe">
]
```

El archivo XML que utiliza la definición de tipo de documento anterior puede estar conformado de la siguiente manera:

```
<Libro Foto="Foto">
  <Nombre>
       Pineda Castañeda Fernando
  </Nombre>
   <Grupo>
       Tercero C
   </Grupo>
   <Escuela>
       Escuela secundaria federal José Martí
   </Escuela>
</Libro>
```

En el caso de la entidad general externa no analizada, solo se puede utilizar en un atributo de tipo ENTITY o ENTITIES (como en el ejemplo anterior), de esta forma, el formato par utilizarse en el documento XML es de la siguiente manera:

<elemento Atributo_Entidad = "Nombre_entidad" >

Declaración de una entidad de parámetro interna analizada

La declaración de una entidad de parámetro interna analizada tiene el siguiente formato

<!ENTITY %Nombre_entidad Valor_entidad>

Donde Nombre_entidad es el nombre de la entidad, este puede comenzar por una letra o guión bajo, seguido por cero o más letras, números, puntos, guiones o guiones bajos. La entidad puede tener el mismo nombre que una entidad general sin que haya conflicto, además también puede tener el mismo nombre que un elemento o atributo. Al igual que con las entidades generales, el procesador de XML hace diferencia entre mayúsculas y minúsculas en los nombres de entidades de parámetros.

Valor_entidad es el valor de la entidad que debe estar delimitado, ya sea por comillas dobles o comillas sencillas y no puede contener en su valor el carácter que se haya utilizado para delimitarlo. Este valor no puede contener el símbolo de porcentaje y el símbolo &, solo cuando se comienza una referencia a entidad general o de carácter. El valor_entidad puede contener declaraciones de tipo de elemento, declaraciones de listas de atributos, instrucciones de procesamiento o comentarios pero no puede contener referencias ni declaraciones de entidades de parámetro.

En el siguiente ejemplo se muestra una DTD que contiene una entidad de parámetro interna analizada, el contenido de la entidad será insertado mediante la referencia a la entidad de parámetro (%ParamFoto)

```
<!DOCTYPE Alumno
[
<!ENTITY %ParamFoto
     '<!ENTITY Foto  SYSTEM "Foto.gif" NDATA imagen>
      <!NOTATION imagen SYSTEM "Pbrush.exe">'
>
<!ELEMENT Alumno (Nombre,Grupo,Escuela)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Grupo (#PCDATA)>
<!ELEMENT Escuela (#PCDATA)>
<!ATTLIST Alumno Foto ENTITY #REQUIRED>
%ParamFoto;
]
```

El archivo XML que utiliza la definición de tipo de documento anterior puede estar conformado de la siguiente manera:

```
<Libro Foto="Foto">
  <Nombre>
       Pineda Castañeda Fernando
  </Nombre>
   <Grupo>
```

```
        Tercero C
    </Grupo>
    <Escuela>
        Escuela secundaria federal José Martí
    </Escuela>
</Libro>
```

Como se observa, la inserción de una referencia de este tipo de entidad se hace sólo en el DTD, en un lugar donde puedan existir declaraciones de marcas.

Definición de entidad de parámetro externa analizada

Las entidades de parámetro externas analizadas se pueden utilizar para almacenar grupos de declaraciones relacionadas. Si se quieren utilizar archivos XML con datos sobre diferentes entidades, por ejemplo, libros, discos, revistas, etc., se pueden colocar las declaraciones de cada tipo de elemento en un archivo separado, de esta forma, se invertirá menos tiempo al utilizar las declaraciones, si se desea manejar más de una definición de tipo de documento. En este caso solo bastaría con incluir la declaración hacia el archivo de la entidad que contiene las declaraciones que se desean incluir en cada definición de tipo de documento.

El formato de la declaración de la entidad de parámetro externa analizada es de la siguiente manera:

```
<!ENTITY %Nombre_Entidad Literal_sistema>
```

Al igual que en las demás entidades, Nombre_entidad representa el nombre de la entidad y el valor deberá apegarse a las reglas para los nombres de entidades (descritas arriba)

Literal_sistema es una literal de sistema que describe la ubicación del archivo que contiene los datos de la entidad

El siguiente ejemplo muestra el uso de dos entidades de parámetro externas analizadas, en la primera parte del código se muestra el documento XML que contiene la definición de tipo de documento y los datos. Después se muestran los archivos que corresponden a las declaraciones de los elementos que se encuentran contenidos en las entidades:

```
<?xml version="1.0" standalone="no">
<!DOCTYPE Empleados
   [
     <!ELEMENT Empleados (Confianza | Sindicalizado) +>
     <!ENTITY %Confianza SYTEM "confianza.dtd">
     <!ENTITY %Sindicalizado SYSTEM "sindicalizado.dtd">
      %Confianza;
      %Sindicalizado;
   ]
>
<Empleados>

</Empleados>
```

El archivo confianza.dtd contiene las siguientes declaraciones:

```
<!ELEMENT confianza EMPTY>
<!ATTLIST confianza Número ID #REQUIRED>
```

<!ATTLIST confianza nombre CDATA>
<!ATTLIST confianza Fecha_ingreso CDATA>

El archivo sindicalizado.dtd contiene las siguientes declaraciones:

<!ELEMENT sindicalizado EMPTY>
<!ATTLIST sindicalizado Número ID #REQUIRED>
<!ATTLIST sindicalizado Número_sindicato ID #REQUIRED>
<!ATTLIST sindicalizado nombre CDATA>
<!ATTLIST sindicalizado Fecha_ingreso CDATA>

Como se observa, al igual que con las entidades de parámetros internas analizadas, la inserción de una referencia de este tipo de entidad se hace sólo en el DTD, en un lugar donde puedan existir declaraciones de marcas.

Esquemas XML

Los esquemas, al igual que la definición de tipo de documento, sirven para definir los elementos y estructura de un documento XML, de tal forma que éste pueda considerarse como un documento válido.

Los esquemas reemplazan a uso de DTD ya que estos se encuentran escritos en XML y por esta razón pueden extenderse fácilmente, son más completos, soportan espacios de nombres (los espacios de nombres se explican más adelante) y además soportan tipos de datos.

El esquema XML es una recomendación de W3C (Consorcio WWW), originalmente fue propuesto por Microsoft, pero en mayo del 2001 se convirtió en una recomendación de W3C.

Los esquemas XML deben de cumplir con las siguientes características:

- Ya que se trata de documentos XML, deben de comenzar con la declaración XML.
- Deben de tener un elemento raíz único, además el nombre de éste debe de ser schema
- Todas las etiquetas de inicio, deben tener su correspondiente etiqueta de finalización
- Las etiquetas son sensibles a mayúsculas y minúsculas
- Todos los elementos deben estar cerrados
- Todos los elementos deben encontrarse anidados correctamente
- Todos los valores de atributos deben estar acotados entre comillas
- Las entidades XML deben de usarse para caracteres especiales

La especificación de esquema XML se basa en una serie de elementos XML que la definen. Estos elementos se especifican en documentos DTD especiales los cuales se muestran en la siguiente tabla:

Elemento de esquema XML	Descripción
Schema	Es el elemento raíz de todo documento esquema de XML
datatype	Con este elemento se definen los tipos de datos utilizados en los elementos y atributos
ElementType	Define los elementos dentro del documento
element	Define un tipo de elemento que puede aparecer en otro tipo de elemento
group	Es utilizado para agrupar elementos

AttributeType	Define un tipo de atributo
attribute	Define un atributo que puede aparecer dentro de un elemento
Description	Define información adicional sobre un elemento o un atributo

Elemento Schema

Es el elemento documento o elemento raíz del documento esquema XML, sus atributos son los siguientes:

- name: Define el nombre del esquema que se crea
- xmins: Define el espacio de nombre del esquema y su valor es urn:schemas-microsoft-com:xml-data. (los espacios de nombre se explican en el siguiente capítulo)

Elemento Datatype

Es utilizado para definir los tipos de datos para el documento XML. De esta forma es posible definir distintos tipos de datos que se utilizarán en las definiciones de elementos de atributos XML. El formato del elemento datatype es la siguiente:

<datatype dt:type="tipo_de_dato"/>

Donde tipo_de_dato es la descripción del tipo de dato que se desea utilizar. El valor de este atributo se obtiene de la tabla que se muestra a continuación:

Tipo de datos	Descripción
bin.base64	Número en base 64
bin.hex	Número hexadecimal en base 16
Bolean	Tipo de dato booleano
Char	Tipo de dato carácter, de longitud 1
Date	Tipo de dato fecha
dateTime	Tipo de dato fecha y hora
dateTime.tz	Tipo de dato de fecha con hora y configuración regional
Entibies	Tipo de dato entidades
Entity	Tipo de dato de entidad
enumeration	Tipo de dato de enumeración
fixed.14.4	Número real con 14 dígitos enteros y 4 de fracción
Flota	Número de coma flotante
i1	Número entero de 1 byte
i2	Número entero de 2 bytes
i4	Número entero de 4 bytes
Id	Tipo de dato ID identificador (no repetible)
Idref	Tipo de dato IDREF, referencia a un identificador
Idrefs	Tipo de dato IDREFS, referencias a indentificadores
nmtoken	Tipo de dato NMTOKEN
nMTOKENS	Tipo de dato NMTOKENS
number	Número real
notation	Tipo de dato NOTATION
r4	Tipo de dato real de 4 bytes
r8	Tipo de dato real de 8 bytes

String	Tipo de dato de cadena
time	Tipo de dato de hora
time.tz	Tipo de dato de hora con configuración regional
ui1	Número entero sin signo de 1 byte
ui2	Número entero sin signo de 2 bytes
ui4	Número entero sin signo de 4 bytes
uri	Tipo de dato de identificador universal de recursos (URI)
uuid	Tipo de dato de identificador único globlal.

Un ejemplo es el siguiente:

<datatype dt:type="string" />

Elemento elementType

Este elemento es utilizado como definición de los tipos de elementos que deberá contener el documento XML que utilice el esquema. Es uno de los elementos más importantes y más utilizados ya que crea directamente los elementos que formarán la especificación del documento XML que se esté creando con la especificación del esquema. Los atributos de este elemento son los siguientes:

- name. Es el nombre del elemento
- model. Especifica el modelo de contenido del elemento. Este modelo puede ser un modelo abierto o cerrado. En un modelo abierto se permite tener dentro del elemento, más elementos hijos cuya definición no es necesaria en el esquema, este modelo es el predeterminado. En el modelo cerrado también se permiten elementos hijos pero, por el contrario, es necesaria la definición de cada uno de estos elementos en el esquema.
- content. En este atributo se define el tipo de contenido que presenta el elemento que se esta definiendo. Los valores que puede tomar este atributo se muestran en la siguiente tabla:

Valor	Descripción
empty	El elemento no debe tener contenido
textOnly	El contenido del elemento solo puede ser texto
eltOnly	El contenido del elemento deberá ser uno o más elementos hijos.
mixed	El contenido puede ser texto y uno o más elementos hijos.

- order. Especifica el número y orden de las repeticiones que representan los elementos agrupados dentro del elemento que se esta definiendo. Los valores que puede tomar este atributo se muestran en la siguiente tabla:

Valor	Descripción
one	Los elementos hijos solo deben de aparecer una vez
seq	Los elementos hijos deben de aparecer en el orden en que se están declarando
many	Los elementos hijos pueden aparecer las veces que sea necesarias y sin ningún orden en particular.

Elemento element

Este elemento es un hijo del elemento ElementType y se utiliza para declarar la aparición de elementos dentro de otros elementos o grupos de elementos. Los atributos del elemento element, son los siguentes:

Valor	Descripción
type	Atributo que sirve para indicar el tipo de elemento
minOccurs	En este atributo se indica el número de veces mínimo que puede aparecer el elemento
maxOccurs	En este atributo se indica el número de veces máximo que puede aparecer el elemento

Combinando los valores de los atributos minOccurs y maxOccurs, se puede controlar las veces que el elemento pude aparecer. Cuando minOccurs es igual a 0 y maxOccurs igual a 1, el elemento puede aparecer una vez o ninguna. Cuando minOccurs y maxOccrus son igual a 1, el elemento solo puede aparecer una vez. Cuando minOccurs es igual a 0 y maxOccurs igual a carácter *, el elemento pude no aparecer o aparecer una o más de una vez. Cuando minOccurs es igual a 1 y maxOccurs es es igual al carácter *, el elemento puede aparecer de 1 a n veces. Cuando minOccurs tiene un valor entero mayor que cero y maxOccurs igual al carácter *, el elemento puede aparecer cuando menos el valor que tiene minOccurs.

Elemento group

Este elemento sirve para organizar los elementos en un determinado orden, especifica que elementos aparecen, que tan seguido y en que orden. Los atributos de este elemento son:

Valor	Descripción
order	Es un atributo obligatorio, define el orden de los elementos dentro del grupo y toma uno de los valores one, seq, many. Cuando su valor es one, significa que solo uno de los elementos del grupo puede aparecer. El valor seq indica que todos los elementos del grupo pueden aparecer y en el orden especificado. El valor many indica que cualquiera de los elementos pude aparecer o no y si aparece no importa el orden.
minOccurs	En este atributo se indica el número de veces mínimo que puede aparecer el elemento
maxOccurs	En este atributo se indica el número de veces máximo que puede aparecer el elemento

Elemento AttributeType

Se utiliza para definir los atributos que pueden aparecer en los elementos del documento XML. Este elemento tiene los atributos que se muestran en la siguiente tabla:

Valor	Descripción
name	Es un atributo requerido, e indica el nombre del atributo que se está definiendo
dt.type	Atributo opcional, indica el tipo de dato del atributo
dt:values	Atributo opcional, se utiliza solo cuando el atributo dt.type es

	del tipo enumeración, y contiene la lista de valores permitidos
default	Atributo opcional, es el valor predeterminado del atributo. Si dt:type aparece, el valor de este atributo debe de apegarse al tipo de dato indicado
required	Atributo opcional, define si el atributo es requerido.

Elemento attribute

Este elemento sirve para indicar el lugar y el elemento que utilizará al atributo que se está definiendo

Valor	Descripción
type	Atributo que define el tipo de atributo
default	Define el valor predeterminado que tomará el atributo
required	Indica si el atributo será requerido

Capítulo III: Especificaciones W3C para el uso de XML

Espacios de Nombres en XML

Los espacios de nombres en XML sirven para evitar confusiones cuando se presentan nombres elementos y atributos que son idénticos a otros nombres de elementos o atributos pero que tienen un significado diferente (cuando se presenta esta situación se dice que hay una colisión). Los espacios de nombres en XML proveen un método simple para distinguir nombres de elementos y atributos por medio de su asociación con espacios de nombres identificados por referencias a URI (Identificador Universal de Recursos)

Un espacio de nombres XML se define como una colección de nombres, identificadas por una referencia a un identificador universal de recursos (URI), que son usados en documentos XML como tipos de elementos y nombres de atributos.

Para definir un espacio de nombres al que pertenece un elemento, es necesario añadir un atributo a la definición de elemento, donde el nombre del atributo sea xmlns (espacio de nombres XML) y el valor puede ser una cadena cualquiera, aunque por convención suelen ser URIs. Por ejemplo, podemos hacer esto añadiendo una entrada como la siguiente en la lista de definición de atributos de un elemento pedido:

```
<!ELEMENT pedido (%inline;)*>
<!ATTLIST pedido
   xmlns CDATA #FIXED "http://www.general.xml/pedido"
>
```

Declarar el atributo como #FIXED tiene varias características importantes:

- Evita que el documento especifique cualquier valor no correspondiente al atributo xmlns.
- Un elemento definido en este DTD se hace único, por eso no genera conflictos con un elemento que tenga el mismo nombre en otro DTD. Esto permite que múltiples DTDs usen el mismo nombre de elemento sin generar un error del analizador.
- Cuando un documento especifica el atributo xmlns para una etiqueta, el documento selecciona la definición de elemento con un atributo correspondiente.
- Cada nombre de elemento en nuestro DTD obtendría exactamente el mismo atributo con el mismo valor.

Cuando un documento usa un nombre de elemento que existe sólo en uno de los ficheros DTD que referencia, el nombre no necesita estar cualificado (un nombre cualificado es aquel que se encuentra compuesto por un nombre local opcionalmente precedido por un prefijo de espacio de nombre y el carácter dos puntos). Pero cuando se usa un nombre de elemento que tiene varias definiciones, se necesita algún tipo de cualificación. Cualificamos una referencia a un nombre de elemento especificando el atributo xmlns, como se ve a continuación:

```
< pedido xmlns="http://www.general.com/pedido" > … </pedido>
```

El espacio de nombres se aplica a ese elemento, y a cualquier elemento contenido dentro de él.

Usando el atributo xmlns, cuando necesitamos una sola referencia a un espacio de nombres, no es mucho trabajo, pero cuando necesitamos hacer la misma referencia varias veces, añadir dicho atributo se convierte en una tarea algo pesada. También hace difícil cambiar el nombre del espacio de nombres, en caso de que se quiera modificar posteriormente. La alternativa es definir un prefijo de espacio de nombres, que es tan sencillo como especificar xmlns, dos puntos y el nombre del prefijo antes del valor del atributo, como se ve a continuación:

```
<p:pedido xmlns:p='http://www.ejemplo/pedido'
      ...>
   ...
</p:pedido>
```

Esta definición configura p como un prefijo que puede usarse para cualificar el nombre del elemento actual y cualquier elemento dentro de él. Como el prefijo puede usarse en cualquier elemento contenido, tiene más sentido definirlo en el elemento raíz del documento XML. Cuando el prefijo se usa para cualificar un nombre de elemento, la etiqueta final también incluye el prefijo, como se muestra aquí:

```
<p:pedido xmlns:sl='http://www.example/pedido'
      ...>
   ...
   <producto>
      <p:cantidad> 3 </p:cantidad>
   </producto>
   ...
</p:pedido >
```

Finalmente podemos observar que se pueden definir varios prefijos en el mismo elemento, como se muestra a continuación:

```
<p:pedido xmlns:p='http://www.example/pedido'
            xmlns:pl="urn: ...">
       <pl:producto>
         <p:cantidad>3</p:cantidad>
        <pl:producto>
   ...
</p:pedido >
```

Veamos un ejemplo, donde se tiene información de libros en XML, pero, a su vez, se quiere usar HTML para mostrar la información:

```
 <h:html xmlns:xdc="http://www.xml.com/books"
       xmlns:h="http://www.w3.org/HTML/1998/html4">
 <h:head><h:title>Book Review</h:title></h:head>
 <h:body>
   <xdc:bookreview>
   <xdc:title>XML: A Primer</xdc:title>
    <h:table>
    <h:tr align="center">
       <h:td>Author</h:td><h:td>Price</h:td>
       <h:td>Pages</h:td><h:td>Date</h:td></h:tr>
    <h:tr align="left">
       <h:td><xdc:author>Simon St. Laurent</xdc:author></h:td>
       <h:td><xdc:price>31.98</xdc:price></h:td>
```

```
      <h:td><xdc:pages>352</xdc:pages></h:td>
      <h:td><xdc:date>1998/01</xdc:date></h:td>
    </h:tr>
    </h:table>
  </xdc:bookreview>
</h:body>
</h:html>
```

En el ejemplo los elementos que tiene el prefijo xdc están asociados a un espacio de nombres cuyo identificador es http://www.xml.com/books, mientras que los que tienen el prefijo h están asociados con un espacio de nombres cuyo identificador es http://www.w3.org/HTML/1998/html4.

Las declaraciones de espacios de nombres tienen ámbito. Esto significa que los espacios de nombres pueden aparecer en cualquier parte de un documento, pero que, como las variables programables, tienen ámbito y por tanto, sólo se aplican en su ámbito apropiado. Existen dos clases de ámbito, predeterminado y completo.

Un espacio de nombres predeterminado se declara en el elemento raíz, y se aplica a todos los elementos incompletos del documento. Un espacio de nombres completo se declara cuando un espacio de nombres más específico se reemplaza en alguna parte del documento.

Aunque para utilizar un espacio de nombres, éste se debe declarar, eso no significa que deba aparecer al principio del documento XML, puede aparecer en el elemento en donde se va a utilizar.

Conjunto de información XML (Infoset)

La especificación de Infoset define un conjunto de datos abstractos llamados conjunto de información XML (Infoset), su propósito es proveer una serie de definiciones consistentes para su uso en otras especificaciones que hacen referencia a un documento XML bien formado.

Un documento XML tiene un conjunto de información si se encuentra bien formado y satisface las ciertas restricciones de espacios de nombres que indica la especificación Infoset. No es un requerimiento que un documento XML sea válido para tener un conjunto de información.

Los conjuntos de información son creados por métodos diferentes a los que utiliza el analizador de un documento XML, el conjunto de información está compuesto por un determinado número de elementos de información, el conjunto de información de un documento bien formado debe de tener por lo menos un elemento de información llamado documento. Un elemento de información es una descripción abstracta de alguna parte del documento XML, cada elemento de información tiene propiedades asociadas.

Así que el propósito de Infoset es proveer un vocabulario común para describir los contenidos de un documento XML. Un procesador XML puede describir los contenidos del documento en términos de tipos de información estandarizada. Esto es importante ya que permite saber qué tipos son requeridos y qué tipos son opcionales, y de esta forma otras especificaciones como XLink, XPath, XSLT, etc., hacen uso de un estándar consistente.

XML canónico

Es posible que existan documentos XML que son equivalentes pero difieren en su representación física. Por ejemplo, dos documentos pueden diferir en su estructura con respecto al ordenamiento de sus atributos o la codificación que utilicen. El objetivo de la especificación del XML canónico es de establecer un método para determinar si dos documentos son idénticos, o si una aplicación no ha cambiado algún documento XML

La forma canónica de los documentos XML permite a ciertas aplicaciones conocer si la información de un documento o un subconjunto de este cambió. Esto se logra comparando la forma canónica del documento original antes de que se procese, con la forma canónica del documento resultante después de aplicar el procesamiento. Dicho de otra manera, la forma canónica de un documento XML es una manera de mostrar una representación física que asegura que dos documentos XML son equivalentes si sus canónicos son iguales, aunque estos sean aparentemente diferentes.

Asociación de Estilos a los documentos XML

Una hoja de estilo contiene instrucciones que son usadas para dar formato a un documento XML, con el uso de archivos de estilos se puede controlar la presentación de la información de los datos en un programa navegador de Internet.

Para que un documento XML pueda ser visualizado por medio de hojas de estilo, es necesario que éste se encuentre vinculado al archivo que contiene las hojas de estilo. Es recomendable que los archivos que contienen las hojas de estilo se encuentren separados de los documentos XML, ya que así se podrá aplicar un solo archivo de estilos a varios documentos XML.

Según la especificación, la asociación de un documento XML a un archivo que contiene las hojas de estilo que se van a utilizar, se hace mediante el uso de una instrucción de procesamiento llamada xml-stylesheet. Esta instrucción es permitida únicamente en el prólogo del documento, la siguiente instrucción asocia las hojas de estilos que se encuentran en el archivo estilos.css, al documento XML en el que está declarada:

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

Una hoja de estilo se encuentra compuesta por un conjunto de reglas y una regla contiene la información que se utiliza para presentar un determinado elemento del documento XML, y tiene el siguiente formato:

Selector {propiedad1:valor1;propiedad2:valor2;…propiedad3:valor3}

Donde Selector es el nombre del elemento al que se le aplica la información de la regla y dentro de los paréntesis se encuentran el bloque de declaración que contiene la lista de propiedad – valor separados por punto y coma.

Por ejemplo, suponiendo que se tiene el siguiente documento XML:

```
<?xml version=”1.0” ?>
<?xml-stylesheet type="text/css" href="estilos.css" ?>

<OrdenCompra Numero=”324”>
<Cliente Numero=”456”>
    <Nombre>González y Asociados</Nombre>
    <Calle>Benito Juárez</Calle>
    <Ciudad>Guadalajara</Ciudad>
```

```
      <Estado>Jalisco</Estado>
   </Cliente>
  <Fecha>21072005</Fecha>
  <Partida NumeroPartida="1">
        <Articulo Clave="123">
          <Descripcion>
            Mesa para computadora
          </Descripcion>
          <Precio>950</Precio>
        </Articulo>
        <Cantidad>2</Cantidad>
  </Partida>
  <Partida NumeroPartida="2">
        <Articulo Clave="198">
          <Descripcion>
            Silla para oficina
          </Descripcion>
          <Precio>560</Precio>
        </Articulo>
        <Cantidad>2</Cantidad>
  </Partida>
</OrdenCompra>
```

Al aplicarle el siguiente archivo de estilos (estilos.css), se puede visualizar el documento en un explorador de Internet como se muestra en la figura 4.

```
OrdenCompra
{
        display:block;
        margin-top:12pt;
        font-size:10pt;
}
Cliente
{
        display:block;
        font-style:italic;
        font-size:16pt;
}
Partida
{
        display:block;
        font-style:normal;
        font-size:medium;
}
```

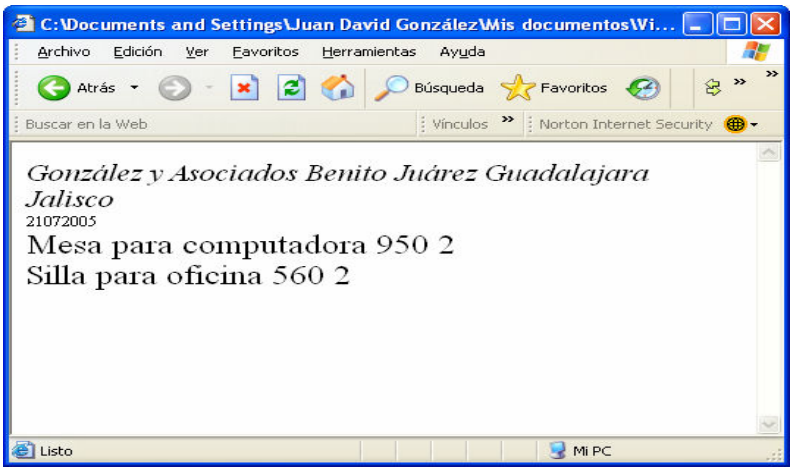

Figura 4: Documento XML con hoja de estilos asociada

Intercambio de fragmentos XML

Existen ocasiones en que se desea consultar o editar ciertas partes de documentos XML, ya sean entidades o partes de entidades, sin que sea necesario hacerlo sobre todo el documento. Esta especificación define una manera de enviar fragmentos de un documento XML, sin importar si estos fragmentos son entidades o no, eliminando la necesidad de enviar todo el contenido del documento.

El problema es que un elemento aislado de un documento XML puede que no contenga la información suficiente para ser procesado correctamente. El objetivo de esta especificación es ayudar a las aplicaciones que envían los fragmentos a mandar también la información adicional requerida, de esta forma los sistemas puedan intercambiar los fragmentos XML que sean necesarios.

Conceptualmente, la parte que posee el documento XML fuente completo considera un fragmento de ese documento, y usando la notación definida, construye una especificación de contexto de fragmento (a veces abreviada como fcs, es una cadena válida que describe un conjunto de información de contexto y su objetivo es proveer de información que no se pasa en el fragmento de código XML). El objeto que representa el fragmento extraído desde el documento fuente es llamado cuerpo del fragmento. El fcs y el cuerpo del fragmento son transmitidas al recipiente (la aplicación que analiza el fcs para darle al analizador XML la información necesaria contextual sobre el fragmento de código). El objeto de almacenamiento, en el cual el cuerpo del fragmento es transmitido, se llama entidad del fragmento. Finalmente, recipiente procesa el fcs para determinar el estado adecuado del analizador XML.

Enlace de recursos (XLink)

El lenguaje XLink define al lenguaje de marcado extensible (XML) elaborado para describir enlaces entre recursos. XLink provee un marco para crear tanto enlaces unidireccionales como estructuras más complejas como enlaces bidireccionales o multidireccionales. Permite que los documentos XML puedan mantener relaciones hacia más de un recurso y asociar meta-datos (información de los datos) con una liga.

No es necesario que los documentos enlazados sean documentos XML, las ligas pueden colocarse en un documento XML que lista las conexiones entre otros documentos, sin que estos sean también documentos XML. La mayoría de los navegadores Web no soportan XLink, sin

embargo, las aplicaciones que son creadas para fines particulares pueden utilizarlo y sacar mayor provecho de esta especificación.

Hay dos tipos de enlaces. Los primeros son los enlaces simples, que son básicamente como los que conocemos de HTML: un enlace desde un recurso local a uno remoto. Los otros son los enlaces extendidos, mucho más complejos pues permiten enlazar muchos recursos entre sí.

Un enlace simple define una conexión en un solo sentido entre dos recursos. La fuente o recurso de inicio de la conexión es el elemento de enlace mismo. El destino o recurso fin de la conexión está definido por un URI (Identificador Universal de Recursos). El enlace va desde el recurso de inicio hacia el recurso fin. El recurso de inicio es siempre un elemento XML. El recurso final puede ser un documento XML, un elemento particular de un documento XML, un grupo de elementos en un documento XML o cualquier otra cosa que incluso tal vez no sea parte de un documento XML.

Un enlace extendido describe una colección de recursos y una colección de rutas entre esos recursos. Cada ruta conecta exactamente dos recursos. Cualquier recurso individual puede ser conectado hacia uno de los otros recursos, dos de los otros recursos, ninguno de los otros recursos, todos los otros recursos, o cualquier subconjunto de los otros recursos en la colección. Este incluso también puede ser conectado hacia si mismo. En términos computacionales, un enlace extendido es un grafo en el cual las rutas son arcos, los documentos son vértices y las URIs, son etiquetas.

Los enlaces simples son muy fáciles de entender haciendo la analogía con los enlaces que se presentan en HTML. Sin embargo, no hay una analogía obvia para los enlaces extendidos, ya que aspectos como en qué forma se presentan, cómo las aplicaciones los tratan, que interfaces de usuario presentan hacia las personas, no están definidos.

Lenguaje de Rutas XML (XPath)

El objetivo principal de XPath es el de direccionar hacia partes de un documento XML. Como soporte para este objetivo principal, también proporciona facilidades básicas para manipulación de cadenas, números y valores booleanos. XPath utiliza una sintaxis compacta y de XML (no-XML) para facilitar el uso de XPath dentro de URIs y de valores de atributos XML. XPath opera sobre la estructura lógica abstracta de un documento XML, más que en su sintaxis superficial. XPath obtiene su denominación por el uso que hace de una notación de caminos, como en los URLs, para navegar a través de la estructura jerárquica de un documento XML.

Además de su uso para direccionar, XPath esta diseñado también de modo que tiene un subconjunto natural que puede usarse para cotejar (comprobar si un nodo encaja con un patrón o no).

XPath modela un documento XML como un árbol de nodos. Hay diferentes tipos de nodos, incluyendo nodos elemento, nodos atributo y nodos texto. XPath define un modo de calcular un valor de cadena para cada tipo de nodo. Algunos tipos de nodo también tienen nombres. XPath es totalmente compatible con los espacios de nombres (XMLNamespaces). Así, el nombre de un nodo se modela como un par consistente en una parte local y un (quizá nulo) URI de espacio de nombres; esto se llama un nombre expandido.

La construcción sintáctica básica en XPath es la expresión. Una expresión se ajusta a la regla de producción. Las expresiones son evaluadas para producir un objeto, que tendrá uno de los siguientes cuatro tipos básicos:

- Conjunto de nodos (una colección desordenada de nodos sin duplicados)
- boleano (verdadero o falso)
- número (un número en punto flotante)
- cadena (una secuencia de caracteres)

Por ejemplo, si tenemos un documento como el siguiente:

```
<Libros>
    <Libro>
      <titulo>Shogun</titulo>
       <genero>Novela</genero>
    </Libro>
    <Libro>
      <titulo>Cálculo diferencial e integral</titulo>
       <genero>Ciencia</genero>
    </Libro>
</Libros>
```

La siguiente tabla muestra algunos ejemplos de sintaxis y la descripción de los elementos que se obtienen:

Sintaxis XPath	Descripción
/Libros/Libro	Selecciona todos los elementos Libro que son hijos del elemento Libros
Libros	Selecciona el elemento documento Libros
Libro	Selecciona todos los elementos Libro
//Libro/titulo	Selecciona todos los elementos titulo que son hijos del elemento Libro
//Libros/Libro/*	Selecciona todos los elementos que están por debajo de la ruta Libro (todos los elementos titulo y genero en este caso)
//*	Selecciona todos los elementos

Lenguaje de punteros XML (XPointer)

XPointer se encuentra basado en una sintaxis no XML para identificar las localidades dentro de documentos XML.

XPointer tiene una sintaxis que no se encuentra basada en XML para identificar las ubicaciones dentro de los documentos. Un XPointer está concatenado al final de un URI como un fragmento de él, para indicar una parte específica de un documento XML en lugar del documento completo. La sintaxis de XPointer se basa en la sintaxis de XPath.
Como ya he indicado, XPointer es una extensión de XPath. Es imprescindible saber XPath para poder usarlo.
Una expresión XPointer se añade a un URI (Uniform Resource Identifier), como puede ser un URL (Uniform Resource Locator) o un URN (Uniform Resource Name).
La idea es añadir al final del URI lo siguiente:

```
#xpointer( expresion )
```

Un detalle muy importante a tener en cuenta es que se pueden concatenar expresiones XPointer que se evalúan de izquierda a derecha mientras devuelvan un conjunto vacío de nodos. Así, el siguiente ejemplo:

documento.xml#xpointer(/libro/capitulo[@public])xpointer(/libro/capitulo[@num="2"])

Se buscaría por en primer lugar el conjunto de nodos delimitado por /libro/capitulo, y solo en el caso de que no existiese ninguno, se buscaría a continuación por /libro/capitulo[@num="2"].

XML Base

El lenguaje XML Linking define al Lenguaje de Marcado Extensible elaborado para describir enlaces entre recursos. Uno de los requerimientos de base sobre XLink es el soportar HTML enlazando conceptos de una forma genérica. El elemento HTML BASE es uno de los conceptos que ha sido tomado en cuenta por el Grupo de Trabajo XLink. BASE permite a los autores especificar claramente el URI base de un documento con el fin de permitir la resolución de URIs relativos en conexiones a imágenes externas, applets, programas de tratamiento de formularios, hojas de estilo, y otros.

XML Base es un mecanismo para proveer servicio de URI base a XLink, con el fin de que sea igualmente posible de ser usado dentro de otras aplicaciones XML beneficiándose así sobre el control adicional sobre URIs relativos sin tener que implementar completamente XLink y que puede hacer uso del mismo. La sintaxis consiste en un único atributo XML denominado xml:base.

El despligue de XML Base está realizado dentro de las referencias reglamentadas por las nuevas especificaciones, por ejemplo XLink y el XML Infoset. Las aplicaciones y las especificaciones constuidas sobre estas nuevas tecnologías deberían nativamente soportar XML Base. El comportamiento del atributo xml:base dentro de las aplicaciones basadas sobre las especificaciones que no tienen referencias normativas directas o indirectas dentro de XML Base no está definido

Inclusiones de documentos XML (Xinclude)

Xinclude es una especificación que introduce un mecanismo general para incluir documentos XML (representados por sus conjuntos de información infosets), para su uso por aplicaciones que necesitan tal característica. XInclude permite al autor escoger cómo incluir otro documento XML en una nueva composición de contenido — bien como etiquetado o bien como texto. Adicionalmente, en XInclude no es precisa la declaración de entidades XML, las cuales eran necesarias usando DTDs con el método antiguo.

Xinclude substituye a las entidades externas de las definiciones de tipo de documento y resuelve algunos problemas que estas presentan, como lo son:

- Los archivos individuales no pueden ser utilizados independientemente del documento en el que se están incluyendo, las entidades externas no son documentos completos y por lo tanto no se consideran como documentos XML bien formados.
- Si alguna de las entidades externas se pierde, entonces todo el documento se considera como un documento mal formado
- La referencia a una entidad no puede apuntar a un archivo de texto plano o a un archivo HTML, solo documentos XML pueden ser incluidos

Xinclude puede ser utilizado haciendo uso del elemento include del espacio de nombres http://www.w3.org/2001/XInclude. Como se muestra a continuación:

<?xml version="1.0"?>

```
<Libro xmlns:xi="http://www.w3.org/2001/XInclude">
  <titulo>The Wit and Wisdom of George W. Bush</titulo>
  <xi:include href="elementos.xml"/>
</Libro>
```

También se pueden utilizar URLs absolutas:

```
<?xml version="1.0"?>
<Libro xmlns:xi="http://www.w3.org/2001/XInclude">
  <titulo>The Wit and Wisdom of George W. Bush</titulo>
  <xi:include href="http://www.archivosxml.com/elementos.xml"/>
</Libro>
```

Transformaciones XSL (XSLT)

Una transformación expresada en XSLT describe reglas para transformar un árbol fuente en un árbol resultante. La transformación es archivada asociando patrones con plantillas. Un patrón es comparado con los elementos en el árbol fuente. Una plantilla se instancia para crear parte del árbol resultante. El árbol resultante se encuentra separado del árbol fuente. Al construir el árbol resultante, los elementos del árbol fuente pueden ser filtrados y reordenados.

Una transformación expresada en XSLT es llamada estilo. Esto es porque cuando XSLT se está transformando en el vocabulario de formato XSL, la transformación funciona como un estilo.

Un estilo contiene un conjunto de reglas plantilla. Una regla plantilla tiene dos partes: un patrón el cuál es comparado contra los nodos en el árbol fuente y una plantilla que puede ser instanciada para formar parte del árbol resultante. Esto permite que el estilo pueda aplicar a una amplia clase de documentos que tienen estructuras similares en sus árboles fuente.

Un procesador de estilo XSL acepta un documento XML o datos XML y un estilo XSL y produce la presentación deseada por el diseñador del estilo de esa fuente de datos XML. Existen dos aspectos de este proceso de presentación: primero, la construcción del árbol resultante del árbol XML fuente y segundo, interpretar el árbol resultante para producir el resultado en el formato que se desea presentar. El primer aspecto es llamado transformación de árbol y el segundo es llamando formato. En la figura 5 se muestran estos dos procesos de transformación:

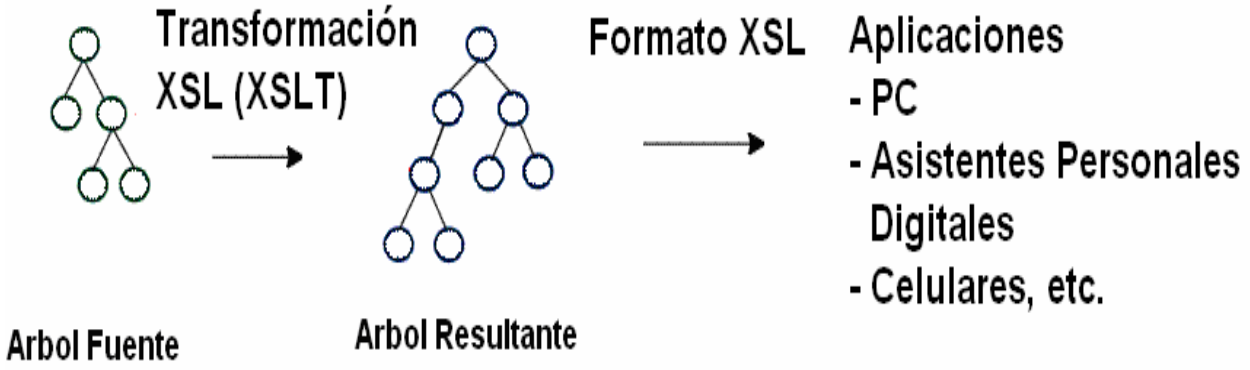

Figura 5: Transformación y formato XSLT

Consultas XML (XQuery)

XQuery esta hecho con la finalidad de poder realizar consultas sencillas para extraer información de documentos XML que se encuentran tanto localmente, como publicados en el Web. El lenguaje de consultas XML esta siendo usado muy ampliamente, a pesar de que la especificación todavía no esta terminada. El objetivo original de XQuery es el de hacer consultas para extraer datos que se encuentran en archivos XML, pero su uso se extendió para hacer manipulaciones de documentos XML de una forma libre, por ejemplo, para extraer información de documentos XML hacia archivos HTML para la presentación de los datos.

Modelo de objeto de documento (DOM)

El modelo de objeto de documento es un conjunto de interfaces para la programación de aplicaciones, neutrales a cualquier plataforma y lenguaje, que permiten a los programas y programas basados en lenguajes de guiones, manipular de forma dinámica el contenido, estructura y estilo de los documentos XML.

Esta interfaz adopta la forma de un modelo de objetos compuesto de objetos, propiedades, métodos y eventos que pueden representar y manipular los componentes de un documento XML. El DOM almacena los datos de un documento XML en una estructura jerárquica de árbol que imita la del propio documento.

El DOM se puede utilizar para obtener acceso a cualquiera de los componentes de un documento XML, incluidos sus elementos, atributos, instrucciones de procesamiento, comentarios y declaraciones de entidad. En el DOM se puede cargar cualquier documento XML. Cuando se efectúa la carga de un documento XML en el DOM, éste se lee desde el comienzo hasta el fin y se almacena en el DOM como un modelo lógico de nodos.

La interfaz de programación del DOM permite que las aplicaciones recorran el árbol y manipulen sus nodos. Cada nodo está definido como un tipo de nodo concreto según las constantes enumeradas del DOM XML, que además definen nodos primarios y secundarios válidos para cada tipo de nodo. En la mayoría de los documentos XML, los tipos de nodos más comunes son elementos, atributos y texto. Los atributos ocupan una posición especial en el modelo de objetos, puesto que no se consideran nodos secundarios de uno principal, sino propiedades de elementos.

En la siguiente tabla se enumeran algunos de los objetos DOM que se pueden utilizar para trabajar con documentos XML, junto con los tipos de nodos de XML a los que representan.

Objeto DOM	Tipo de nodo de XML
XMLDOMDocument	Representa el documento XML en su totalidad. Este objeto expone propiedades y métodos que permiten desplazarse, realizar consultas y modificar el contenido y la estructura de un documento XML.
XMLDOMNode	Representa un solo nodo del árbol del documento. Este objeto es el objeto base para obtener acceso a los datos del DOM XML e incluye compatibilidad con los tipos de datos, los espacios de nombres y los esquemas XML.
XMLDOMNodeList	Representa una colección de nodos. Este objeto permite la repetición y las operaciones de acceso por índices en la colección actual de IXMLDOMNode.
XMLDOMElement	Representa un elemento del documento XML.

XMLDOMAttribute	Representa un atributo del documento XML.

Capítulo IV: Algunas Aplicaciones XML

Servicios Web

Los servicios web se pueden definir como componentes de software que, mediante interfaces públicas, brindan funcionalidad a otras aplicaciones, haciendo uso de una conexión http y mensajes basados en XML.

Los servicios web se comunican usando el protocolo simple de acceso a objetos (SOAP). Este es un protocolo basado en XML que resuelve el problema de la incompatibilidad que se presenta cuando varías aplicaciones en diferentes plataformas desean intercambiar datos y servicios. SOAP usa una sintaxis que permite construir aplicaciones que en forma remota invocan métodos de objetos, elimina la necesidad de que dos sistemas tienen que ejecutarse en la misma plataforma o tengan que estar escritos en el mismo lenguaje de programación. En lugar de llamar los métodos por medio de un protocolo binario propietario, un paquete SOAP usa una sintaxis abierta y estándar para hacer el llamado a los métodos, esta sintaxis es XML.

Toda la información entre la aplicación cliente y el objeto que proporciona el servicio es enviada como datos entre etiquetas en una cadena XML. Esta cadena esta compuesta por texto plano que puede ser enviado por medio del protocolo http y cruzando la mayoría de los contrafuegos. Esta característica es importante, ya que no es necesario que los administradores de la red tengan que abrir más puertos que el puerto 80 (la mayoría de las redes lo tienen abierto) y de esta forma no comprometen la seguridad de la red.

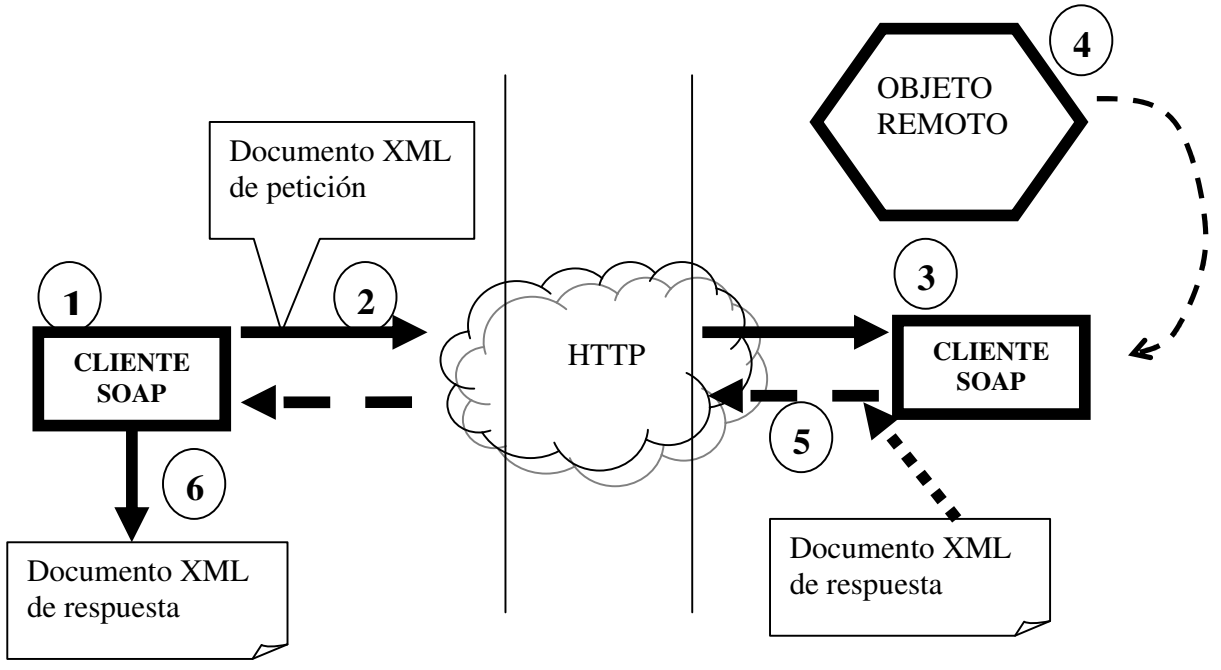

Figura 6: Petición y respuesta por medio de servicios web

Como se observa en la figura 6, un cliente SOAP envía la petición sobre http, a un servidor que se encuentra escuchando peticiones del otro lado. Este servidor captura el mensaje, interpreta

la petición e invoca al método que se encuentra en un objeto. Este objeto regresa algo que es de utilidad para la aplicación que hizo la petición y responde por medio del servidor SOAP.

Las etapas que se muestran en la figura 6 son las siguientes:

1. Un cliente SOAP (que puede ser cualquier programa escrito sobre cualquier lenguaje de programación que tenga soporte para XML y que además esté en cualquier plataforma), crea un documento XML que contiene la petición. Este documento debe de estar contenido en un envoltorio SOAP.
2. El paquete (petición) es enviada por medio de una conexión http.
3. Una servidor SOAP (aplicación que se encuentra escuchando peticiones SOAP) recibe el mensaje, lo analiza y llama al objeto apropiado, pasando los parámetros necesarios que fueron incluidos en el paquete.
4. El objeto ejecuta la función solicitada y regresa el resultado al servidor SOAP. El servidor SOAP empaqueta la respuesta en un envoltorio SOAP.
5. La respuesta es enviada de regreso hacia la computadora que hizo la petición.
6. El cliente SOAP espera por la respuesta. Cuando la recibe quita el envoltorio SOAP y envía respuesta a la aplicación que la necesita.

Los servicios Web son ampliamente utilizados, ya que presentan las siguientes ventajas:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad y contrafuegos sin necesidad de cambiar las reglas de filtrado.

XHTML

XHTML (Lenguaje Extensible de Marcado de Hipertexto) es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, que se encuentra diseñado para mostrar datos, con la de XML, que se encuentra diseñado para describir los datos.

XHTML acaba con vicios que se cometen al escribir páginas utilizando el lenguaje HTML, los más comunes son:

- Etiquetas HTML no cerradas.
- Etiquetas HTML en orden incorrecto.
- Omisión de elementos importantes.
- No distinguir entre mayúsculas y minúsculas.
- Uso de etiquetas no necesarias al escribir los documentos.

XHTML, al estar orientado al uso de un etiquetado correcto, exige una serie de requisitos básicos a cumplir en lo que a código se refiere. Entre estos requisitos básicos se puede mencionar una estructuración coherente dentro del documento donde se incluirían elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados, etc. Esto se logra aplicando las siguientes reglas:

- Cada etiqueta de apertura tendrá una etiqueta de cierre, al menos de que se trate de la forma abreviada del elemento vacío.
- Todas las etiquetas se deberán de escribir en minúsculas

- Las etiquetas se deberán abrir y cerrar en el orden correcto
- Las etiquetas deberán llevar todos los atributos definidos para cada una de ellas
- El primer elemento XML de los documentos XHTML deberá ser <html>, y deberán presentar sendos elementos <head> y <body> para construir un documento XHTML completo y escrito correctamente, así mismo, el elemento <title> debe de ser el primero dentro del elemento <head>
- Los atributos deberán presentar sus valores entre comillas
- Los atributos no podrán aparecer sin valores
- Los distintos elementos dentro del documento se distinguirán en forma única en el ámbito de su documento mediante el atributo ID
- Los elementos <script> y <style> deberán incluirse dentro de una sección CDATA.

Se han creado tres definiciones de tipo de documento para XHTML, la razón de que haya tres se basa en que esta variedad permite distintas implementaciones de XHTML, variando la exactitud con la que se va a exigir un código XHTML válido. Estas definiciones corresponden al nivel de validación de los documentos XHTML y son las siguientes:

Validación estricta, es el nivel más estricto, no permite elementos como <font> o <table>, para dar formato a los documentos será necesario la aplicación de estilos. Para utilizar esta definición se debe incluir la siguiente línea justo después de la declaración de xml:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
Validación transitoria, es un nivel menos estricto, permite los elementos <font> y <table> y también acepta el uso de estilos. Para utilizar esta definición se debe incluir la siguiente línea justo después de la declaración de xml:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">

Validación de conjunto de marcos, esta validación incluye los elementos para creación de aplicaciones que utilizan marcos. Para utilizar esta definición se debe incluir la siguiente línea justo después de la declaración de xml:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/frameset.dtd">

XML y bases de datos

Un documento XML es una base de datos en el sentido estricto del término. Esto no hace diferencia de otro archivo con que tiene un formato distinto (por ejemplo, archivos de texto plano) ya que una base de datos es un conjunto de información que guarda cierta relación. Pero XML tiene algunas ventajas, como lo es el hecho de que un documento XML se encuentra auto descrito e independiente de cualquier plataforma, sin embargo, con respecto al tiempo de acceso a la información, en comparación con un manejador de bases de datos relacionales, es mucho más lento. Así que en el desarrollo de una aplicación que pretende utilizar documentos XML como base de datos se deben de considerar los aspectos mencionados anteriormente.

XML provee muchas de las funcionalidades encontradas en una base de datos: almacenamiento, esquemas, consultas e interfaces de programación, pero carece de características como son índices, seguridad, transacciones, integridad de de datos, acceso multiusuario, desencadenadores y consultas a través de múltiples documentos, sin embargo, es posible utilizar XML como base de datos si la aplicación que se esta desarrollando va a ser

utilizada por pocos usuarios, tendrá poca información y no se necesita que el desempeño sea alto.

Algunos de los aspectos que hay que tomar en cuenta para considerar a XML como una herramienta para guardar información, además de los puntos mencionados anteriormente, son:

a) La información se va a transportar utilizando XML, entonces tal vez sea conveniente que los datos almacenados guarden una estructura similar en el transporte como en el almacenamiento
b) La información va a ser publicada en el Web y se le da mantenimiento periódicamente pero no se cuenta con los recursos necesarios como para hacer uso de un manejador de bases de datos relacional comercial.
c) La información se manipula y consulta por varias aplicaciones las cuales se ejecutan en diferentes plataformas.

También es importante distinguir si la base de datos almacenará información de entidades (centrado en datos) o documentos (centrado en documentos), ya que ambos guardan características particulares.

Los documentos XML centrados en datos generalmente hacen uso de XML como transporte de datos, estos documentos son caracterizados porque guardan una estructura regular, representan entidades que están compuestas por datos granulares (por ejemplo una silla), los elementos no están compuestos por contenido mixto, en la mayoría de los casos se especifica un tipo de dato tanto para los elementos como para los atributos. Los documentos que se utilizan para almacenar órdenes de compra, datos científicos, etc., son ejemplos de documentos centrados en datos.

Los documentos XML centrados en documentos están diseñados para el uso de los seres humanos, ejemplos de estos son los correos electrónicos, la publicidad, revistas, libros. Estos se encuentran caracterizados por tener una estructura irregular, gran cantidad de datos no granulares y mucho contenido mixto.

No siempre la información que se desea almacenar es representada totalmente por alguno de los tipos de documentos mencionados anteriormente. Por ejemplo, es posible que un documento XML centrado en datos que almacena la información de los productos además de contener datos granulares, contenga datos que no son granulares dentro de las descripciones, o por el contrario, que un documento XML centrado en documento que almacena la información de artículos, contenga datos granulares, como lo es el nombre de autor o la fecha de edición. A pesar de estas situaciones, es importante distinguir si los datos serán representados por documentos XML centrados en datos o documentos XML centrados den documentos, ya que esto ayudará a decidir qué tipo de base de datos se va a utilizar.

Para transferir datos entre los documentos XML y la base de datos, es necesario hacer una correspondencia entre el esquema XML y el esquema de la base de datos (llamado mapeo). Las aplicaciones para transferir los datos están construidas con base a esta correspondencia. Estas aplicaciones pueden utilizar un lenguaje de consultas XML como XQuery o simplemente transferir los datos de acuerdo a la correspondencia.

Existen dos tipos de mapeos, aquellos que se encuentran basados en tablas y los que se encuentran basados en objetos relacionales.

El mapeo basado en tablas es usado por muchas aplicaciones comerciales que se encargan de transferir los datos entre los documentos XML y las bases de datos relacionales. Este mapeo hace un modelo de documentos XML que se encuentra constituido por una tabla sencilla o un

conjunto de tablas, de esta forma, la estructura de un documento XML puede ser como se muestra a continuación:

```
<BaseDeDatos>
  <tabla>
  <registro>
      <columna>…</columna>
       …
      <columna>…</columna>
  </registro>
   …
 </tabla>
…
  <tabla>
    <registro>
      <columna>…</columna>
       …
      <columna>…</columna>
   </registro>
    …
</tabla>
</BaseDeDatos>
```

Según el diseño y la aplicación que va a utilizar el modelo, se puede especificar si los datos de una columna son almacenados como elementos hijos o como atributos, así como qué nombres se usarán para cada elemento o atributo.

El mapeo basado en objetos relacionales crea un modelo de documentos XML que utiliza árboles de objetos. En este modelo, los elementos con atributos, los elementos con contenido, los elementos con contenido mixto, son modelados generalmente como clases (llamados tipos de elementos complejos). Los elementos simples y atributos son modelados como propiedades escalares (llamados tipos de elementos simples). Se hace una correspondencia del modelo con la base de datos relacional relacionando las clases con las tablas, las propiedades escalares con columnas, y las propiedades que identifican a los objetos con llaves primarias. Por ejemplo, suponiendo que se tiene el siguiente documento XML que representa órdenes de compra:

```
<OrdenCompra Numero=12345”>
  <Cliente Numero=”456”>
     <Nombre>González y Asociados</Nombre>
     <Calle>Benito Juárez</Calle>
     <Ciudad>Guadalajara</Ciudad>
     <Estado>Jalisco</Estado>
  </Cliente>
 <Fecha>21072005</Fecha>
 <Partida NumeroPartida="1">
       <Articulo Clave="123">
         <Descripcion>
           Mesa para computadora
         </Descripcion>
         <Precio>950</Precio>
       </Articulo>
       <Cantidad>2</Cantidad>
 </Partida>
 <Partida NumeroPartida="2">
```

```
      <Articulo Clave="198">
        <Descripcion>
          Silla para oficina
        </Descripcion>
        <Precio>560</Precio>
      </Articulo>
      <Cantidad>2</Cantidad>
</Partida>
</OrdenCompra>
```

Este documento puede modelarse como un árbol de objetos de cuatro clases y seis objetos (OrdenCompra, Cliente, Partida y Producto), como se muestra en la figura 7:

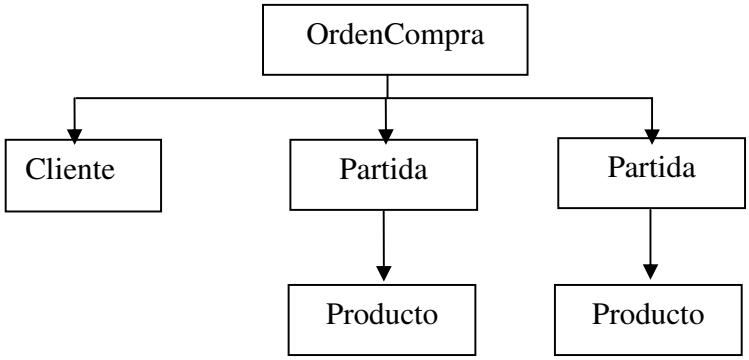

Figura 7: Arbol de objetos generado a partir de un documento XML

Formularios XML (Xforms)

Los formularios XML fueron creados por el W3C para proporcionar una herramienta que pueda servir en la captura de información hacia documentos XML, reemplazando las formas que se utilizan en los archivos XHTML. Las reglas para describir y validar los datos en un formulario XML se expresan todas en XML.

Los formularios XML se dividen en dos partes: el modelo de formularios XML (modelo XForm) y las interfaces de usuario de formularios XML (interfaces de usuario XForm), el primero está orientado a la validación y lógica de la información que se captura, y el segundo al despliegue y captura de datos.

El modelo de formularios de XML

El modelo de XForm define los siguientes elementos:

Elemento	Descripción
<model>	Es la sección que describe los datos, contiene a los elementos <instance> y <submission>
<instance>	Define la sección en donde los datos serán colocados, dentro de este elemento se encuentra la estructura del documento XML que se captura.
<submission>	En este elemento se indica la forma en que los datos serán enviados hacia el servidor, tiene dos atributos que son: id, que sirve para especificar el nombre que identifica a la forma y method, que sirve para

	especificar la manera de envío de los datos por el protocolo HTTP.

Ejemplo:

```
<model>
  <instance>
    <Cliente>
      <Nombre/>
      <Numero/>
    </Cliente>
  </instance>
   <submission id="form1" method="get"
  action="submit.asp"/>
</model>
```

Interfaces de usuario de formularios XML

En la sección de interfaz del usuario se pueden utilizar los siguientes controles:

Control	Ejemplo
Control para introducir información	<input ref=”Cliente/nombre”> <label>Nombre:</label> </input> O <input bind=”Cliente/nombre”> <label>Nombre:</label> </input> En ref o bind se especifica a qué elemento hijo del elemento <instance> se hace referencia, aquí se usa la sintaxis de XPath
Etiqueta	<label>Nombre:</label> Este elemento se utiliza dentro del los elementos <input>, <secret>, <textarea>, <submit>, <select>, <select1>,<range>
Control tipo contraseña	<secret ref="Cliente/contraseña"> <label>Contraseña:</label> </secret> En ref se indica a qué elemento hijo del elemento <instance> se hace referencia utilizando la sintaxis de XPath
Control de entrada de múltiples líneas	<textarea ref="comentarios"> <label>Comentarios</label> </textarea> En el atributo ref se especifica a qué elemento hijo del elemento <instance> se hace referencia, utilizando la sintaxis de XPath
Control para envío de datos	<submit submission="forma1"> <label>Enviar Datos</label> </submit> En el atributo submission se indica el nombre de la forma que fue indicado en el atributo id del elemento <submission>
Control para desplegado de	<output ref=”Cliente/nombre”/>

datos	En ref la cadena XPath que hace referencia al elemento del cual se desea obtener la información
Control para la carga de archivos al servidor	<upload bind="name"> <label>Archivo:</label> <filename bind="file"/> <mediatype bind="media"/> </upload>
Control de selección de un único elemento en una lista	<select1 ref="sexo"> <label>Sexo:</label> <item> <label>Hombre</label> <value>H</value> </item> <item> <label>Mujer</label> <value>M</value> </item> </select1>
Control de selección de uno o más elementos en una lista	<select ref="idioma"> <label>Idioma:</label> <item> <label>Inglés</label> <value>Ing</value> </item> <item> <label>Español</label> <value>Esp</value> </item> </select1>
Control para seleccionar un valor de un rango de valores	<range ref="longitud" start="0" end="100" step="5"> <label>Longitud:</label> </range>

Espacios de nombres de formularios XML

Para referenciar al espacio de nombres de xForms se tiene que escribir la siguiente línea:

```
<html xmlns:xf="http://www.w3.org/2002/xforms">
```

Tipos de datos

En los formularios XML se pueden utilizar los tipos de datos definidos en los esquemas, para utilizarlos hay que indicar los siguientes espacios de nombres, quedando la declaración anterior de la siguiente forma:

```
<html
xmlns:xf="http://www.w3.org/2002/xforms"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Después de haber declarado los espacios de nombres que se van a utilizar, se puede indicar el tipo de dato en cada elemento del documento XML:

```
<xf:instance>
<Cliente xmlns="">
  <nombre xsi:type="xsd:string"/>
  <numero xsi:type="xsd:i4"/>

</Cliente>
</xf:instance>
```

También se puede utilizar el elemento bind para asociar tipos de datos con elementos del documento XML de la siguiente forma:

```
<xf:bind nodeset="/person/size" type="xsd:integer"/>
```

Propiedades

Es posible usar propiedades que definen restricciones u otros atributos que afectan el comportamiento de los controles en los formularios, como se muestra en la siguiente tabla:

Propiedad	Ejemplo	Descripción
required	<bind nodeset="Orden/Precio " required="true()"/>	Establece al elemento Precio como requerido, el valor predeterminado de esta propiedad es falso
type	<bind nodeset="Orden/Precio " required="decimal"/>	Establece al elemento Precio como del tipo decimal
readonly	<bind nodeset="Orden/Precio " readonly="true()"/>	Establece al elemento Precio como de solo lectura, el valor predeterminado de esta propiedad es falso
relevant	<bind nodeset="Orden/Precio " relevant ="true()"/>	Establece al elemento Precio como relevante. Indica si el elemento es relevante, los elementos no relevantes por lo general se ocultan y no se envían de regreso al servidor, el valor predeterminado de esta propiedad es verdadero.
calculate	<bind nodeset="Orden/Total " calculate="Orden/Precio * 1.15"/>	Calcula el valor del elemento Total multiplicando el valor del elemento Precio por 1.15
constraint	<bind nodeset="Orden/Total " constraint=". > Orden/Precio”	Indica que el valor del elemento total debe de ser mayor al valor del elemento precio.
p3ptype	<bind nodeset="Nombre" p3ptype="user.personname.given"/>	Se indica información del tipo P3P en el elemento Nombre.

Acciones y eventos

Cada evento tiene un elemento de destino, el cual representa el punto donde la acción principal se lleva a cabo, desde el punto de vista de un evento en particular. También cada evento puede tener una acción predeterminada que es ejecutada cuando el evento sucede. Sin embargo, en algunos casos un elemento que observe puede desear tener alguna acción sobre los eventos, en este caso el observador es padre del elemento destino. El nivel 2 del modelo de objeto de documentos define cómo los eventos se propagan.

La propagación de eventos significa que un evento puede afectar a un objeto y a un conjunto de objetos relacionados. Cualquiera de los objetos potencialmente afectados pueden bloquear el evento o sustituirlo por uno diferente (propagación de eventos hacia arriba). El evento se transmite desde el nodo en que se origina hacia todos los nodos padre.

La propagación de eventos es dividida en dos etapas: captura y burbujeo. La captura ocurre primero, cuando el nodo raíz tiene la opción de observar el evento, nodos adicionales siguen la ruta desde la raíz hasta el elemento objetivo. Cualquiera que escucha tiene la habilidad de terminar la propagación, o evitar la acción predeterminada del evento. Después de haber alcanzado al nodo objetivo, el evento revierte su curso y hace un regreso hacia el elemento raíz, en el proceso llamado burbujeo. Si el evento termina sin que se haya cancelado, la acción predeterminada suceda mientras los que escuchan no hayan bloqueado dicha acción.

En el diseño de formas HTML, el lenguaje de guiones es usado para indicar si alguna acción específica es necesaria. Por ejemplo, una forma puede tener un botón que copia los valores desde una sección donde se leen los datos, a otra sección donde se escriben, pasando la información de una serie de elementos HTML a otra. Suponiendo que existe la función, en JavaScript por ejemplo, que ejecuta la acción anterior, el código HTML del botón es parecido al siguiente:

```
<input type=”button” id=”cp” value=”Copia valores” onclick=”CopiaValores();”>
```

En términos de eventos del modelo de objeto de documento, esto representa el registro de un observador del elemento input, viendo el evento click sobre el objetivo, y cachando el evento al llamar a un código escrito en un lenguaje de guiones. De esta forma el código en el atributo será llamado cuando el usuario presione el botón.

El nivel 2 del modelo objetos de documento define un elemento que escucha, que aunque no es usado directamente en los formularios de XML, es necesario examinarlo. Este elemento tiene ocho atributos definidos, los cuales se muestran el la siguiente tabla:

Elemento	Descripción
target	Atributo que especifica mediante el uso de un IDREF, el elemento objetivo para el cual observar los eventos.
handler	Mediante el uso de una referencia URI especifica el elemento que se utilizará para tratar (cachar) el evento. En los formularios XML estos elementos son llamados acciones.
event	En este atributo se especifica el nombre del atributo que será observado
observer	Por medio de un IDREF especifica el elemento al cual el observador será relacionado.
phase	Indica la fase del modelo de objeto de documento nivel 2, en cual el objeto será activado (captura, burbujeo). Un observador no puede ver en la fase de captura y burbujeo al

	mismo tiempo.
propagate	Determina si el evento continuará propagándose.
defaultAction	Determina si la acción predeterminada será activada para el evento

Usando las acciones de los formularios XML, el ejemplo del código HTML quedaría de la siguiente forma:

```
<trigger>
  <label>Copia Valores</label>
  <action ev:event="DOMActivate">
    <setvalue ref="Envio/Direccion" value="../Factura/Direccion"/>
    <setvalue ref="Envio/CP" value="../Factura/CP"/>
  </action>
</trigger>
```

A continuación se describen algunas de las acciones de los formularios de XML.

Nombre	**Descripción**
message	Envía un mensaje al usuario
setvalue	Establece un valor para determinado elemento. El valor puede ser obtenido de otro elemento utilizando XPath o puede ser indicado literalmente
setfocus	Redirecciona el foco hacia un determinado elemento. Es necesario utilizar un atributo llamado control que es del tipo IDREF
send	Provoca que el formulario envíe de vuelta los valores al servidor
reset	Limpia el elemento model indicado
load	Provoca que el navegador cargue otro recurso, puede ser en la misma ventana o en una nueva
insert	Inserta nuevos nodos en el modelo, utilizando una plantilla
delete	Borra nodos del modelo

Conclusiones

El hecho de que XML se encuentre enfocado a crear documentos que se encargan de describir la estructura y significado de los datos, junto con la característica de que está concebido para ser utilizado sobre Internet, ha provocado que sus aplicaciones se encuentren en una gran diversidad de sistemas. Basta con revisar la extensa lista de las iniciativas en diferentes industrias, las cuales van desde estándares para crear documentos musicales (como lo son el lenguaje estándar de descripción musical o la iniciativa de codificación musical), hasta estándares para el intercambio de documentos de negocio a negocio (lenguaje de marcado electrónico para negocios ebXML).

En este momento existen plataformas de desarrollo de aplicaciones que utilizan a XML como un elemento central, tal es el ejemplo de la plataforma .Net de Microsoft, en la cual las piezas de código que se ejecutan tienen la característica de describirse a si mismas y con esto proporcionan la capacidad de que las aplicaciones que se ejecutan en estas plataformas, puedan compartir componentes de código reutilizable sin que haya ningún riesgo de hacer referencia a versiones del mismo componente que no san las correctas. Existe también un enorme soporte con librerías de clases y de funciones para varios lenguajes de programación con respecto al manejo de XML.

Otro de los aspectos importantes de este lenguaje y que es necesario destacar, es su uso para desarrollar aplicaciones distribuidas sobre Internet, aplicaciones que no se encuentran limitadas a ejecutarse en una plataforma en particular, reutilizando servicios y compartiendo información. Imaginemos sistemas de información que por ejemplo, en el área de la medicina, puedan compartir expedientes médicos de una manera confiable, o sistemas que se encuentren interconectados para la toma de decisiones en el caso de presentarse un sismo o cualquier catástrofe natural.

En el ámbito comercial, el uso de XML prevé ahorros considerables con respecto a la compra y venta de productos y servicios entre clientes y proveedores al generar documentos electrónicos que se encuentran apegados a un estándar basado en XML, las empresas que desean ser competitivas en estos tiempos de intercambio global están obligadas a adoptar este tipo de tecnología.

Las grandes compañías de tecnología de información como IBM, Oracle, Microsoft, Sun Microsystems, y otras han efectuado grandes inversiones en XML y están contribuyendo activamente en el proceso de estandarización. XML se fortalece con especificaciones nuevas y con la revisión constante de las especificaciones existentes que le permiten utilizarlo en aplicaciones efectivas y cada vez más variadas, por ejemplo, en un futuro los formatos de los archivos generados por aplicaciones de oficinas, como lo son los documentos de texto u hojas de cálculo, podrán ser especificados en documentos XML, lo cual provocará que se compartan fácilmente por distintas aplicaciones sin importar el propietario de dichas aplicaciones y la plataforma en que se ejecuten.

Con respecto a las aplicaciones distribuidas, estas contarán con mayor seguridad al compartir información, ya que podrán utilizar documentos XML que contengan elementos con información codificada, y también se podrá asegurar la autenticidad de la información, gracias a la inclusión de firmas electrónicas.

En lo que se refiere a los manejadores de bases de datos relacionales, XML se perfila como el estándar que se utilizará para intercambiar información entre diferentes bases de datos, sin importar de qué manejador de base de datos se trate, ya que cada vez son más los

manejadores de bases de datos relacionales que incorporan instrucciones especiales en sus lenguajes transaccionales para extraer información que se encuentre en un formato XML.

En ese sentido es importante que desarrolladores de sistemas e ingenieros de software conozcan las características, las especificaciones, las iniciativas en diferentes industrias y campos de aplicación de este lenguaje, para que cada vez sean más los sistemas de información que aprovechen esta tecnología.

Glosario

W3C
Es un consorcio internacional donde el personal, el público y miembros de diferentes organizaciones trabajan juntos para desarrollar estándares para el WWW.

Analizador XML
Es la aplicación que se encarga de verificar que la sintaxis del documento sea la correcta y en ocasiones que también se trate de un documento válido al aplicar una definición de tipo de documento o esquema.

ASCII
El Código Estándar Americano para el Intercambio de Información es un conjunto de caracteres y códigos de caracteres basados en el alfabeto romano y usados en el Inglés moderno. Los códigos ASCII representan texto en las computadoras, diferentes equipos de comunicación y dispositivos de control que trabajan con texto.

API
La interfaz de programación de aplicaciones es un conjunto de especificaciones que permiten que una pieza de código pueda comunicarse con otra. Uno de los propósitos fundamentales es proveer a las aplicaciones un conjunto común de funciones básicas.

Elemento
Corresponde a una etiqueta dentro de un documento XML

GML
Es un conjunto de etiquetas, el cual fue creado por IBM y tiene el propósito de simplificar la descripción de un documento en términos de su formato, estructura y partes de contenido y sus relaciones entre estas partes.

HTML
El lenguaje de marcado de hipertexto es un lenguaje de etiquetas que se encuentra diseñado para la creación de páginas en el WWW y otra información que puede ser vista por medio de un cliente llamado navegador. Este lenguaje es usado para estructurar el formato de la información que se presenta.

HTTP
El protocolo de transferencia de hipertexto es un protocolo de petición-respuesta entre clientes y servidores usado en el WWW.

Internet
Es un sistema de redes de computadora interconectadas que comparten información. Esta información es transmitida por medio de enrutamiento de paquetes utilizando un protocolo estandarizado, llamado Protocolo de Internet

ISO

La Organización Internacional de Normalización es una organización internacional compuesta por representantes de cuerpos internacionales de estándares. Se encarga de producir estándares para la industria y comercio mundial.

Lenguaje de Marcado

Es un lenguaje que combina texto e información extra sobre su contenido. La información extra, por ejemplo puede ser sobre la presentación del contenido y esta es expresada utilizando etiquetas, las cuales engloban al texto principal.

Metalenguaje

Es un lenguaje que es usado para crear sentencias que serán utilizadas en otro lenguaje.

SGML

Es un metalenguaje que sirve para definir lenguajes de marcado para documentos, se encuentra estandarizado y desciende del lenguaje GML

URI

Un identificador uniforme de recursos es un elemento del protocolo de Internet (IP) que consiste en una cadena de caracteres que hace referencia a un recurso. Esta cadena esta compuesta conforme a cierta sintaxis.

URL

Un localizador uniforme de recursos es una dirección estandarizada para recursos que existen en Internet, la cual permite conocer en dónde se encuentran dichos recursos. Un URL es también un tipo de URI ya que además de indicar la localización, identifica al recurso.

Web o WWW

Es un servicio que opera sobre Internet en el cual los elementos que se comparten son llamados recursos y se identifican mediante identificadores globales, estos últimos llamados identificadores uniformes de recursos.

Bibliografía

Elliotte Rusty Harold, W. Scott Jeans, "**XML in a Nutshell**", Segunda Edición, Estados Unidos, Editorial O'Reilly, 2002

Fitzgeral Michael, "**XML Hacks**", Estados Unidos, Editorial O'Reilly, 2004

González Oscar, "**Guía práctica para usuarios XML**", España, Editorial Anaya Multimedia, 2001

Travis Brian E. "**XML and SOAP programming for BizTalk Servers**", Estados Unidos, Editorial Microsoft Press, 2000

Williamson Heather, "**XML Manual de referencia**", España, Editorial McGraw-Hill, 2001

Young Michael J. "**XML Step-by-Step**", Estados Unidos, Editorial McGraw-Hill, 2000