Capítulo III

Procesar documentos XML

APIS XML

Procesador XML

Parsers XML

Herramientas para trabajar con XML

Lenguajes que interactúan con XML

Búsquedas con XML

XML Query Language

Bases de datos XML (XDB)

Un documento puede ser llamado documento XML si se adhiere a las reglas de la especificación XML, una regla básica es que si un documento no esta bien construido, no debe seguir siendo analizado sintácticamente de manera normal, así los autores deben adherirse a los estándares XML y los escritores de software deben implementarlos estrictamente.

Analizar sintácticamente un documento XML es el primer paso para procesarlo. Un analizador toma un documento XML y comprueba si está bien construido o si es válido, luego construirá un árbol de análisis con todos los objetos del documento XML. El analizador pasará luego este árbol a un agente que mostrará algún tipo de vista del árbol o dará estilo al documento si a este se asocia una hoja de estilo.

Todos los analizadores XML deben comprobar que la construcción de un documento XML es correcta. Los analizadores XML son de dos tipos: Uno estándar, que comprueba que un documento está bien construido, y otro de validación, que coteja el documento con su DTD para que este sea válido. A continuación se explica en más detalle tanto API's, procesadores y analizadores de documentos XML.

3.1. API's para XML

Si se quiere realizar acciones con datos escritos en XML, el W3C y grupos independientes han especificado mecanismos (normas) para acceder a documentos XML y trabajar con ellos. Estas normas incluyen una jerarquía de objetos que tienen métodos y atributos con los que se tiene que trabajar y que simplificarán las tareas relativas al recorrido y acceso a las partes del documento.

Hay dos tipos importantes de APIs para XML:

- **APIs basadas en árbol**

  Estas mapean un documento XML a una estructura interna de árbol, permitiendo a una aplicación navegar por ese árbol. El Modelo de Objetos del Documento (DOM) del W3C es una recomendación API basada en árbol para documentos XML y HTML.

- **APIs basadas en eventos**

  Una API basada en eventos analiza eventos (inicio y fin de elementos) reportando directamente a la aplicación a través de llamadas de retorno (callbacks), y normalmente no construye un árbol. La aplicación implementa manejadores para tratar con los diferentes eventos. SAX es el mejor ejemplo conocido de esta API.

SAX se utiliza para hacer un recorrido secuencial de los elementos del documento XML y DOM implica la creación de un árbol en memoria que contiene el documento XML.

Se puede programar con algún lenguaje de programación para acceder a un documento XML y los creadores del lenguaje son los responsables de crear un API que cumplan las especificaciones de XML para que luego los desarrolladores puedan trabajar con ellas. Un lenguaje típico para trabajar con XML es Java y en este caso es SUN Microsystems la encargada de proveer el API que los desarrolladores en Java utilizarán para programar con XML.

3.1.1. API Simple para XML - SAX (Simple API for XML)

SAX es un API de secuencia de procesamiento de XML que permite procesar un documento XML sin almacenar mucho más que el contexto del nodo actual que se está procesando en memoria. Permite procesar archivos XML de gran tamaño sin que sea necesario ocupar un espacio en memoria demasiado grande.

SAX define una interfaz manejada por eventos y métodos para analizar XML. A medida que el analizador va leyendo el documento XML y encuentra los componentes del documento (elementos, atributos, valores, etc.) va detectando errores e invocando a las funciones que ha asociado el programador.

SAX es muy conveniente para propósitos dónde el usuario quiera leer un documento XML entero de inicio a fin, y realizar algún cómputo, como construir una estructura de datos que representa un documento, o resumir la información en un documento (calculando un valor promedio de un cierto elemento, por ejemplo). No es muy útil si el usuario desea modificar la estructura del documento de alguna manera complicada que involucre cambiar cómo se anidan los elementos.

Las ventajas de SAX son velocidad y simplicidad, además el documento no tiene que ser residente en memoria, no importa si están procesándose documentos muy grandes.

3.1.1.1. Interfaces de SAX

Una interfaz dice que métodos hay, y qué tipo de parámetros esperan. Es una especificación, no proporciona ningún código ejecutable cuando los métodos son llamados. SAX define cuatro interfaces básicas [WWW005]:

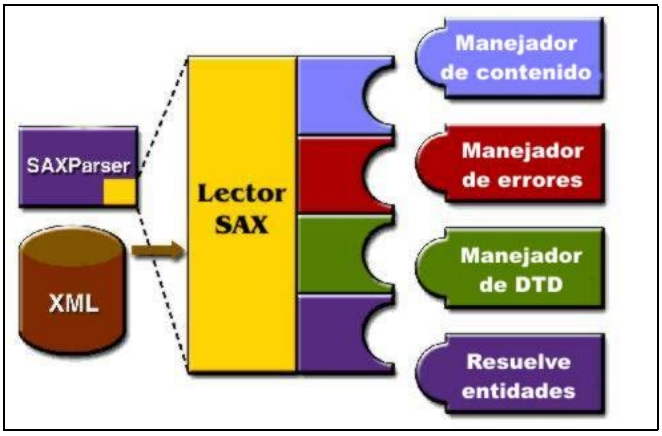

Figura 3.1. Interfaces de SAX.

- Manejador de contenido - ContentHandler

Requerido para eventos del documento general. Esta interfaz es el centro de todo proceso de XML con SAX. Es la que define todos los eventos que se producen a lo largo del proceso del documento; sus métodos se llaman para el inicio del documento, inicio y fin de elementos, y para los caracteres de datos contenidos dentro de los elementos.

- Manejador de DTD - DTDHandler

Permite manejar eventos relacionados con el análisis de una DTD. No atiende a eventos relacionados con la validación. Poco usada.

- Resuelve entidades - EntityResolver

Llamada para resolver las referencias a las entidades externas. Si los documentos no tienen ninguna referencia a entidades externas, no se necesitará implementar esta interfaz.

- Manejador de errores - ErrorHandler

Recibe información sobre errores en los datos del documento XML. Requerida para el manejo de errores. Los parsers (analizadores sintácticos) llamarán los métodos de esta interfaz para informar sobre todas las advertencias y errores.

3.1.2. Modelo de Objetos del Documento - DOM (Document Object Model)

El Modelo de Objetos del Documento, es una interfaz independiente de la plataforma o del lenguaje, que permite a los desarrolladores crear aplicaciones y guiones (scripts) para tener acceso y actualizar el contenido, estilo y estructura de los documentos XML. [LIB005]

El DOM es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento (elementos, atributos y estilo). [WWW006]

En el DOM, los documentos tienen una estructura lógica que es muy parecida a un árbol, más bien es como un "bosque", que puede contener más de un árbol. Sin embargo, el DOM no especifica que los documentos deban ser implementados como un árbol o un bosque, ni tampoco especifica cómo deben implementarse las relaciones entre objetos.

Los documentos sé modelizan usando objetos, y el modelo comprende la estructura de un documento, el comportamiento de este y de los objetos de los cuales se compone. En otras palabras, los nodos no representan una estructura de datos, sino que representan objetos, los cuales pueden tener funciones e identidades. Como modelo de objetos, el DOM identifica:

- las interfaces y objetos usados para representar y manipular un documento
- la semántica de estas interfaces y objetos, incluyendo comportamiento y atributos
- las relaciones y colaboraciones entre estas interfaces y objetos

El Modelo de Objetos del Documento consiste actualmente de dos partes, el Núcleo del DOM y el DOM HTML. El Núcleo del DOM representa la funcionalidad usada para los documentos XML, y también sirve de base para el DOM HTML. Una implementación conforme del DOM debe implementar el DOM HTML o bien las interfaces extendidas (XML), o ambas, con la semántica definida.

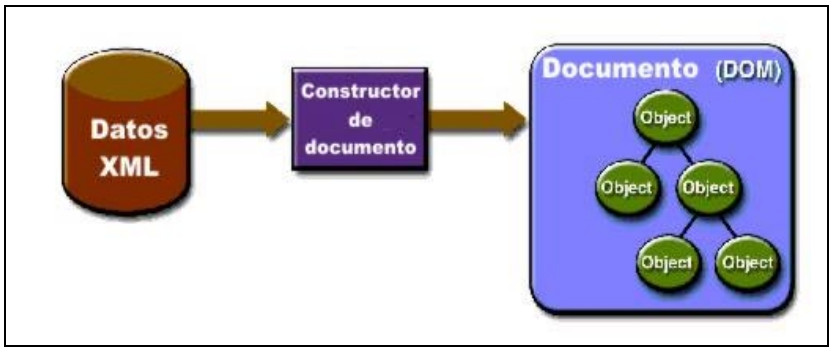

Figura 3.2. DOM

La especificación de nivel 1 del Modelo de Objeto de Documento (DOM) del W3C define lo que debería mostrar un DOM como propiedades, métodos y eventos. El W3C establece varios niveles de actuación:

- **Nivel 1:** Se refiere a la parte interna y modelos para HTML y XML. Contiene funcionalidades para la navegación y manipulación de documentos. Tiene 2

partes: la parte básica, referida a documentos XML, y la parte HTML referida precisamente a los HTML.

- **Nivel 2:** Incluye un modelo de objetos y una interfaz de acceso a las características de estilo del documento, definiendo funcionalidades para manipular la información sobre el estilo del documento. También incluirá un modelo de eventos para soportar los XML namespaces.
- Posteriores niveles especificarán interfaces a posibles sistemas de ventanas, manipulación de DTD y modelos de seguridad.

El DOM especifica interfaces que pueden utilizarse para manipular documentos XML o HTML. Es importante darse cuenta de que estas interfaces son una abstracción, constituyen un medio para especificar la forma de acceder y manipular la representación interna que una aplicación hace de un documento. Cada aplicación DOM es libre de mantener los documentos según una representación conveniente cualquiera, siempre y cuando soporte las interfaces mostradas en la especificación del DOM.

El objetivo es que cualquier script pueda ejecutarse de forma más o menos homogénea en cualquier navegador que soporte dicho DOM. Se podrá elegir el implementar modelos propietarios que son lo que ahora ofrecen Netscape y Microsoft, pero lo ideal sería tener una plataforma estándar potente y versátil en la que se pueda crear contenidos sin temor a no estar soportado por alguna marca o versión de navegador.

3.1.3. JDOM

JDOM es una API pensada específicamente para el procesado de documentos XML con Java. Al igual que DOM se basa en el análisis (parseado) de un documento XML y la construcción de un árbol de elementos, atributos, comentarios, instrucciones de procesamiento, etc. Una vez construido el árbol se puede acceder directamente a cualquier componente. A diferencia de DOM, que está pensada para ser una API genérica, implementable con cualquier lenguaje de programación, JDOM está expresamente pensada para Java, así pues los elementos, atributos, etc., se representarán mediante clases Java, sin que exista el concepto de Nodo propio de DOM.

Ventajas de JDOM:

- Pensada especialmente para Java y por tanto mejor integrada en este lenguaje.
- Facilidad de uso (para programadores que conozcan Java)

- Representación de documentos como árboles, lo que implica el acceso aleatorio a cualquier parte del documento (ventaja sobre SAX).
- API de libre distribución.

Desventajas de JDOM:

- Mayor consumo de memoria que SAX, ya que es necesario procesar y almacenar en memoria todo el documento
- Al ser una API diseñada para Java resulta poco portable a otros lenguajes de programación.
- El carecer del concepto de Nodo dificulta en cierta medida la navegación a través del árbol del documento. Por ejemplo, al obtener los hijos de un elemento, el resultado es una lista de objetos genéricos, y debemos comprobar qué representan esos objetos para saber si son elementos, atributos, etc.

3.2. Procesador XML

Un navegador Web es un procesador de HTML, este lee el documento, toma la información que necesita de sus elementos y contenidos y lo presenta en la ventana de navegación. Un procesador XML funciona de un modo muy parecido: lee documentos, interpreta su marca y lo muestra en un dispositivo de visualización, como una ventana de navegación o una impresora. Los procesadores hacen posible la presentación y distribución de documentos XML.

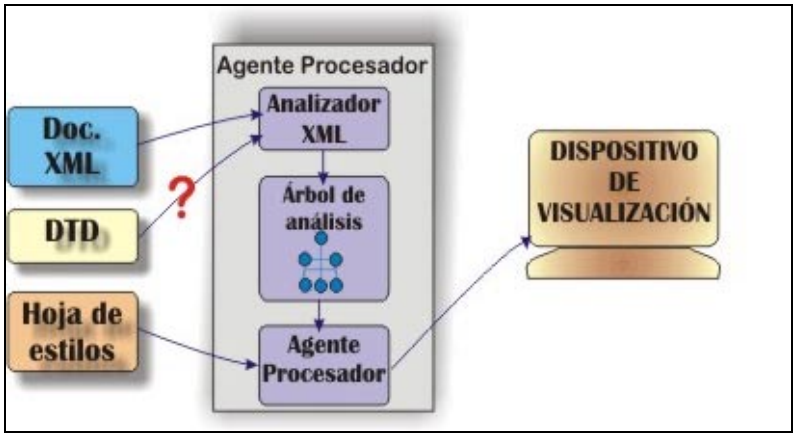

Figura 3.3. Procesar un documento XML [LIB006]

Aunque todos los procesadores XML esencialmente funcionan de la misma manera, los resultados finales pueden ser muy diferentes entre sí. Los procesadores que leen e

interpretan un documento de OSD(Open Software Description o Descripción abierta de software) instalarán uno o más paquetes de software basándose en los contenidos del documento, mientras que los que procesen un documento PGML(Precision Graphics Markup Language o Lenguaje de marca para gráficos de precisión) ofrecerán como resultado una imagen. El proceso y aplicación de un documento está determinado por sus contenidos. En el mundo de HTML, el único tipo de procesador que existe es el navegador Web.

Un procesador XML consta de dos partes bien diferenciadas:

- El analizador o parser
- El dispositivo de presentación

Un analizador es la herramienta que lee el documento y lo descompone en partes. Normalmente crea un árbol o catálogo con todos los elementos del documento junto con sus contenidos, y luego pasa esa información a un dispositivo que sabe cómo ha de presentar esos contenidos.

Todos los navegadores Web incorporan analizadores que examinan sus documentos HTML, la presentación que se puede ver en la ventana de navegación es la interpretación de los datos que le ha suministrado el analizador, todo navegador debe estar acompañado por un parser, pues necesita los datos que éste le ofrece para funcionar adecuadamente.

La mayoría de los procesadores que actualmente están en el mercado son simples analizadores que no se responsabilizan de la presentación de un documento. Muchos diseñadores de DTDs trabajan al mismo tiempo en procesadores que tratarán los documentos que se escriban para sus DTDs. Otros programadores escriben analizadores independientes orientados hacia una DTD u otra, pero capaces de analizar cualquier DTD que se le presente.

3.3. Parsers XML (Analizador, Intérprete)

XML es independiente del lenguaje de programación. Las APIs DOM (Document Object Model, Level 1, 2 & 3) y SAX (SAX 2.0 Simple API for XML) son abiertas e independientes del lenguaje y definen cómo acceder, validar y modificar los documentos XML. Sobre la base de estos APIs se definen los parsers de XML para analizar y validar documentos XML.

Un Parser es un módulo, biblioteca o programa que se ocupa de transformar un archivo de texto en una representación interna. En el caso de XML, como el formato siempre es el mismo, no necesitamos crear un parser cada vez que hacemos un programa, hay muchos parsers disponibles.

El parser XML es la herramienta principal de cualquier aplicación XML, mediante este parser no solamente podemos comprobar si nuestros documentos están bien formados o son válidos, sino que también podemos incorporarlos a nuestras aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML.

De acuerdo a su función, los parsers XML se dividen en: validadores y no-validadores:

- Los parsers validadores comprueban un documento comparándolo con su DTD para cerciorarse de que respeta las normas de su vocabulario. Si el documento viola alguna de las reglas de su DTD, el analizador advertirá del error y el proceso del documento se detendrá por completo. Un buen procesador validante no sólo informará de la presencia de un error, sino que elaborará un mensaje dependiendo de cuál haya sido la causa de este error. Así los procesadores validantes son una de las mejores herramientas de depuración disponibles.
- Un parser no validador sólo comprueba si un documento está bien formado. Estos parsers devuelven un error sólo en caso de que existan problemas en la construcción general de un documento. Estos parsers normalmente procesan los documentos con mayor velocidad, pues no han de comprobar tantos detalles como los parsers validantes.

Todos los parsers deben comprobar que la construcción del documento XML es correcta. Si un documento no está bien formado, el parser no deberá seguir analizándolo.

De acuerdo a la forma en que los parsers procesan los documentos se clasifican en: Los que forman un árbol de análisis *(DOM)* y los que analizan sintácticamente un documento como estructura plana (*SAX*).

3.3.1. Parsers DOM

DOM genera en memoria un árbol jerárquico del documento, este árbol jerárquico de información en memoria permite que a través del parser sea manipulada la información, permitiendo que en cualquier punto del árbol se pueda agregar o eliminar un nodo (Información).

3.3.1.1. Árbol XML generado por parsers DOM

Un parser DOM crea un árbol de análisis al analizar sintácticamente el documento XML. De esta forma un documento XML puede considerarse como una serie de objetos con ciertas propiedades. Los objetos pueden ser: el documento en sí, los elementos individuales, los comentarios, las cadenas de texto, etc. Un árbol de análisis es una estructura que actúa como marco para los objetos de documento, y muestra como están relacionados entre sí.

Un árbol es una estructura de datos compuesta de nodos conectados que empiezan con un solo nodo llamado raíz. La raíz conecta a sus nodos hijo cada uno de los cuales pueden conectar a cero o más hijos propios, y así sucesivamente. Los nodos que no tienen ningún hijo propio se llaman hojas. La propiedad más útil de un árbol es que cada nodo y sus hijos también forman un árbol. Así, un árbol es una estructura jerárquica de árboles en el que cada árbol se construye por árboles más pequeños.

Un árbol XML contiene siete tipos de nodos. Éstos son:

1. La raíz
2. Los elementos
3. El texto
4. Los atributos
5. Espacios de nombres - Namespaces
6. Instrucciones de procesamiento
7. Los comentarios

A continuación se muestra un documento XML y su representación de árbol:

```
<?xml versión=”1.0”?>
<!—comentario -- >
<elemento>
        <elemento></elemento>
        <elemento>
                texto<elemento></elemento>texto
        </elemento>
</elemento>
```

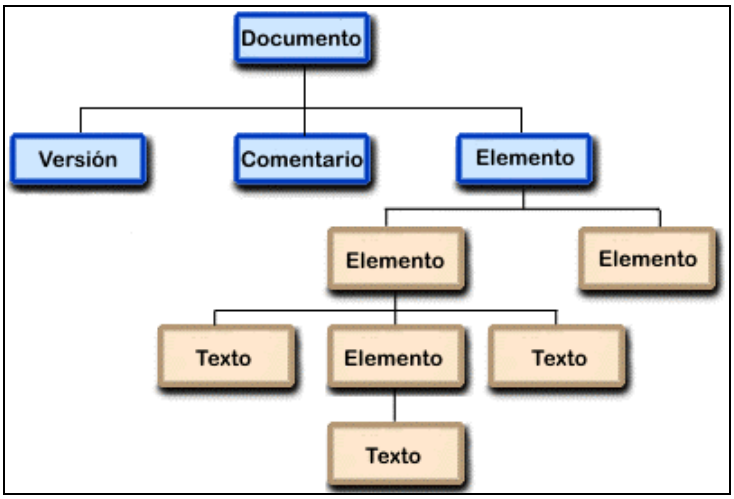

Figura 3.4. Árbol XML

3.3.2. Parsers SAX

El parser SAX genera eventos cada vez que se encuentra con una nueva marca XML o con un error (por documento mal formado o inválido). Es cuestión del manejador de eventos, implementar los métodos apropiados para actuar según los eventos. Este API está pensado para leer con rapidez documentos XML y reaccionar en función de su contenido pero no para representarlos en memoria para su presentación o modificación.

Un parser SAX procesa la información por eventos, conforme esta sea presentada (evento por evento), efectivamente manipulando cada elemento a un determinado tiempo, sin incurrir en uso excesivo de memoria.

Para comparar SAX con DOM podemos mencionar lo siguiente:

- SAX es un parser ideal para manipular archivos de gran tamaño, porque no hay necesidad de cargar el fichero entero en memoria, así la memoria consumida es mucho menor que la de DOM, ya que este genera un árbol en memoria.
- SAX es más rápido y sencillo que utilizar DOM, la sencillez tiene su precio, debido a que SAX funciona por eventos no es posible manipular información una vez procesada, en DOM no existe esta limitación ya que se genera el árbol jerárquico en memoria y es posible regresar a modificar nodos.
- SAX sirve cuando se quiere construir una estructura propia de datos. Una aplicación puede querer construir una estructura de datos usando objetos de alto nivel como libros, autores, y publicaciones antes que elementos de bajo nivel, atributos, e instrucciones de procesamiento.

- Si queremos una pequeña cantidad de información lo útil sería SAX. Es muy ineficiente y bastante innecesario leer todos los datos que están en la memoria, en lugar de leer sólo aquellos que nos interesan. Una de las mejores características del SAX es que hace muy fácil ignorar los datos en los que no se está interesado.

3.4. Herramientas para trabajar con XML

Existen en el mercado un sin número de herramientas que ayudarán a manejar y manipular la información en base al uso de la tecnología que esta revolucionando el mundo del Internet XML, se cuenta con parsers (analizadores), procesadores XML y XSL, editores XML, XSL y DTD, entre otras que van incorporándose con el avance de la tecnología.

3.4.1. Procesadores XML

Muchos de los procesadores han sido escritos en Java, permitiendo así su funcionamiento en los navegadores Web y en distintas plataformas, sin tener que modificarse. La técnica actual es utilizar Java para crear procesadores XML, y parece que seguirá siendo durante algún tiempo.

3.4.2. Parsers (Analizadores) XML

Los parsers nos sirven para comprobar que los documentos están bien formados, que están validados por el DTD/Schema correspondiente, para aplicar las hojas de estilo, etc. Hay parsers escritos en múltiples lenguajes, en función de la necesidad de la aplicación que se quiera desarrollar se utilizará uno u otro.

Muchos de los analizadores han sido escritos en Java, permitiendo así su funcionamiento en los navegadores Web y en distintas plataformas, sin tener que modificarse. A continuación mencionaremos algunos analizadores:

Expat [WWW007]

Sus características son:

- No validante
- Escrito en C
- Plataforma: Windows 95/ 98 y NT
- Su autor es James Clark

Expat se creó como parte del código fuente de Mozilla de Netscape, destinado a formar parte de la solución de soporte de XML de la versión 5 de Netscape Navigator. Es pequeño y veloz, y funcionará perfectamente una vez integrado en el entorno del navegador.

Lark [WWW008]

Sus características son:

- No validante
- Escrito en Java
- Se puede ejecutar en cualquier plataforma que soporte Java
- Su autor es Tim Bray

Fue uno de los primeros analizadores para XML que se creo, nació como un experimento que conjuntaba XML y Java para dar como resultado una herramienta de proceso bastante potente. Lark sólo puede leer e interpretar documentos XML, no presentarlos.

XML para Java [WWW009]

Sus características son:

- Validante
- Escrito en Java
- Se puede ejecutar en cualquier plataforma que soporte Java

Es uno de los más recientes analizadores, XML para Java está diseñado para ser algo más que un simple analizador, pues también puede usarse para generar XML. Podría ser la herramienta perfecta para todo aquél que vaya a trabajar con documentos XML más allá de los navegadores Web.

3.4.3. Editores XML

Un editor XML es una aplicación que nos ofrece facilidades para crear y editar documentos XML. Podemos diferenciar dos tipos de editores XML:

- Los que representan el fichero XML en forma de árbol. Estos presentan la estructura del documento XML en forma de árbol, y nos permiten construir nuestro documento trabajando sobre este árbol. Algunos de este tipo son:

  XML Notepad. Es un editor de XML desarrollado por Microsoft. Para su utilización es necesario tener instalado, como mínimo, la versión 5 del Explorer.

  Visual XML. Es un editor de XML escrito en Java.

- Los que presentan el documento XML en su formato original. Son editores normales de ficheros de texto, pero con facilidades de edición enfocadas al XML. Permiten garantizar que el autor no va a escribir documentos que no estén bien formados y puede leer el DTD para sugerir la introducción de elementos válidos. Algunos editores de este tipo son:

  XED, es un editor de XML que permite garantizar que el autor no va a escribir documentos que no estén bien formados y puede leer la DTD para sugerir la introducción de elementos validos.

  PSGML para Emacs. Es un modo superior de Emacs para trabajar con SGML que se ha modificado para soportar XML. Lee la DTD, puede utilizar un analizador externo para validar documentos, realiza coloración de sintaxis y otras muchas cosas.

En ambos tipos hay que diferenciar los que trabajan contra un DTD y, por lo tanto, validan el contenido del documento XML, y los que simplemente se aseguran que el documento XML este bien formado. La elección de uno u otro dependerá del tipo de documento que se esté escribiendo. Si se desea escribir un documento XML tipo, que permita introducir información en una base de datos se debe usar un editor del primer tipo. En cambio, si se quiere escribir un documento con gran cantidad de texto será más eficaz uno del segundo tipo.

3.4.4. Editores DTD

En un DTD se define cómo es la estructura de un documento XML, es decir, los elementos que formarán ese tipo de documento y cómo están relacionados. En XML no es obligatorio usar una DTD aunque sí es recomendable utilizarlas, al menos, durante el periodo de diseño y validación de los documentos.

Algunos de los más utilizados son:

- **EZDTD** Permite crear la DTD de forma visual y nos permite guardar la DTD diferenciando si va a ser para documentos XML o SGML. Nos permite guardar la DTD en formato HTML colocando enlaces de forma automática entre los elementos que define.
- **TDTD** para **Emacs,** que es un modo superior de Emacs para la edición de DTD. Realiza coloración de sintaxis e incluye algunas macros que facilitan la edición de construcciones comunes.

3.4.5. Editores XSL

Son editores que nos van a facilitar la creación de las hojas de estilo necesarias para poder visualizar el contenido de los documentos XML en el formato que se quiera. Existen muy pocos en el mercado, uno es el slide que es un editor para la plataforma emacs. Existen conversores de XML a PDF y viceversa.

3.4.6. Procesador XSLT

Los procesadores XSLT (*XSL Transformation*) son capaces de realizar transformaciones sobre nuestros documentos XML para obtener un fichero en otro formato resultante. Entre ellos destacan:

- **SABLOTRON**: Motor Open-Source que viene con la mayoría de las distribuciones de Linux.
- **XALAN**: Motor Open-Source implementado dentro del proyecto xml.apache.org. Existen dos implementaciones: Una en java y otra en C++.
- **XERCES**: Librería también de xml.apache.org que implementa un parser XML: También disponible en Java y C++.
- **Saxon**: Otro motor también muy conocido es implementado por Michael Kay. Según la documentación existente sobre XML, lo sitúan entre los más rápidos.

El IE5 es en si un procesador XSLT y, por tanto, es capaz de mostrar un documento XML utilizando un lenguaje de hojas de estilo.

3.5. Estudio de varios lenguajes que interactúan con XML

Se puede programar con el lenguaje de programación que se desee para acceder a un documento XML. Los creadores del lenguaje son los responsables de crear unas API que cumplan las especificaciones de XML para que luego los desarrolladores de cada lenguaje las encuentren y puedan trabajar con ellas. A continuación se indicara los lenguajes que poseen las herramientas necesarias para trabajar con XML.

3.5.1. XML & Java

Desde los primeros días de XML, los programadores de Java han estado en el centro de la comunidad de desarrolladores de XML. Muchos de ellos creen que Java es el lenguaje de referencia para el desarrollo de XML. Las dos tecnologías se complementan muy

bien: XML por la transportabilidad de los datos (datos independientes de la plataforma) y Java por la transportabilidad del software (procesamiento independiente de la plataforma). Es más, hay una serie de razones por las que Java es especialmente recomendable a la hora de trabajar con XML:

- Java siempre ha estado muy orientado a la web, tanto en clientes como en servidores, y XML se dirige a la próxima generación de la tecnología web.
- El fuerte soporte que tiene Java en Unicode cumple todos los requisitos del conjunto de caracteres en XML, por lo que los desarrolladores que no trabajan en inglés no tendrán ninguna desventaja.

Existen dos interfaces básicas para analizar XML con Java, estas son: SAX, que soporta una estructura muy pequeña, el procesamiento basado en eventos; y el DOM, que representa el resultado de analizar en una estructura de datos en forma de árbol. Las librerías que definen dichos APIs están incluidas en el fichero parser.jar.

El "API Simple para XML" (SAX) es el manejador de eventos, mecanismo de acceso que hace el proceso elemento-por-elemento. Este API lee y escribe XML al repositorio de datos o la Web, se usa en aplicaciones del lado servidor y de alto rendimiento.

El API DOM es generalmente fácil de usar, podemos emplear este API para manipular el árbol de objetos que encapsula la aplicación. El API DOM es ideal para aplicaciones interactivas porque el modelo de objetos se presenta en memoria, donde puede ser accedido y manipulado. Por otro lado, construir el DOM requiere leer toda la estructura XML y mantener el árbol de objetos en memoria, lo que significa un mayor consumo de CPU y de memoria. [*ver API's XML*].

3.5.2. JSP con XML

XML es un conjunto de reglas de sintaxis y guías para definir lenguajes de texto basados en marcas. La información que se expresa en un formato estructurado basado en texto puede ser fácilmente transmitida, trasformada e interpretada por entidades que entiendan la estructura. De esta forma XML nos trae los mismos beneficios multi-plataforma para el intercambio de información que el lenguaje de programación Java tiene para el procesamiento.

La tecnología JavaServer Pages (JSP) puede utilizar todo el poder de la plataforma Java para acceder a objetos del lenguaje de programación para analizar y transformar documentos XML.

La tecnología JSP proporciona servicios de documentos que combinan elementos de lenguaje de marcas estáticos y elementos creados dinámicamente mediante objetos del lenguaje Java. Las páginas JSP se ejecutan como Servlets Java, normalmente en el contexto de un servidor Web donde generan documentos de respuesta dependientes del contenido usando datos almacenados en bases de datos y en otros objetos de aplicación basados en servidor.

Los servidores de aplicaciones y las herramientas de desarrollo que soportan la tecnología JSP están disponibles a través de muchos vendedores y para un amplio rango de plataformas hardware y software.

La tecnología JSP proporciona un número de capacidades que son ideales para trabajar con XML, pueden contener cualquier tipo de datos basados en texto, por eso son correctas para generar documentos que contienen marcas XML. En particular, como parte del entorno Java, las páginas JSP pueden usar objetos que utilicen los nuevos APIs Java para procesar datos XML.

La tecnología JavaServer Pages y el XML son compañeros naturales que usan fuentes de datos heterogéneas y soportan clientes multi-lenguaje. Se puede usar la tecnología JSP para generar documentos XML, el principal requerimiento para generar XML es que la página JSP seleccione el tipo de contenido de la página de forma apropiada.

Para poder convertir XML en objetos del lado del servidor y extraer las propiedades del objeto. La conversión no es automática; tenemos que analizar manualmente un documento XML, y encapsularlo dentro de un componente JavaBeans. En el futuro sin embargo, la tecnología de unión XML/Java automatizará este proceso pues permitirá compilar un esquema XML en clases Java.

El entorno de software que se usara para analizar es el API para procesamiento de XML (*JAXP*) *versión 1.1* que soporta SAX, DOM level 2, y XSLT. JAXP puede ser descargado desde [WWW010] y para instalarlo, se descomprime en un directorio, se actualiza el classpath para incluir el árbol de ficheros *.jar. El API JAXP contiene:

- crimson.jar: el analizador XML por defecto.
- xalan.jar: El motor XSLT por defecto
- jaxp.jar: los APIs

La tecnología Java Server Pages hace fácil embeber trozos de código Java en documentos XML.

Como desarrollador Java, podemos ampliar las páginas JSP introduciendo etiquetas personalizadas que pueden ser desplegadas y usadas en una sintaxis al estilo HTML. Las etiquetas personalizadas también nos permiten proporcionar un mejor empaquetamiento, mejorando la separación entre la lógica del negocio y su representación.

3.5.3. XML & PHP

El lenguaje PHP es un lenguaje de programación de estilo clásico cercano a C o a JavaScript. Pero a diferencia de JavaScript que se ejecuta en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado es enviado al navegador.

El problema principal está en generar las páginas XML dinámicamente utilizando PHP y Cocoon (*estructura de publicación para XML*) para realizar las transformaciones. No puedes embeber PHP en una página XML porque Cocoon no lo reconoce.

Todavía existe una esperanza rodeando este problema y poder utilizar PHP para generar XML dinámicamente, se puede empezar con un script php que escriba cierto código XML válido en un archivo y luego redireccionar a esa página.xml.

PHP provee funciones que permiten parsear documentos XML mediante el parser Expat, que provee un conjunto de herramientas que interpretan pero no validan documentos XML. En PHP disponemos de funciones que facilitan el tratamiento de documentos XML, constituyen una extensión basada en el analizador EXPAT, para utilizarlas es necesario compilar PHP con las opciones adecuadas y que nuestro sistema tenga instalado EXPAT. EXPAT es un analizador de tipo SAX (devuelve los elementos XML de forma secuencial).

3.5.4. XML & Perl

Perl (Practical Extraction and Report Languaje) es un lenguaje de programación surgido a inicios de los noventa, que busca antes que nada el facilitar la elaboración de tareas comunes en sistemas tipo UNIX, donde tradicionalmente las tareas de administración y proceso de datos se realizan con herramientas muy rudimentarias y por demás hostiles al usuario o administrador, pero que se aplican sobre grandes cantidades de información por lo que se requiere que sean de alto rendimiento.

Perl surgió como una opción para una gran cantidad de herramientas de UNIX en las cuales se basa su propia sintaxis, buscando una máxima facilidad de programación e

integración. Sigue la filosofía de mantener un ambiente que sea capaz de detectar y corregir pequeñas omisiones del programador, y de proporcionarle una forma abreviada de realizar múltiples tareas. En una palabra, es una utilería que pretende facilitar el proceso de grandes volúmenes de información sin sacrificar el rendimiento.

Existen varios analizadores sintácticos XML para Perl, pero el más popular es el módulo XML: Parser. El módulo es una envoltura de Perl alrededor de Expat, un analizador sintáctico no validador.

XML::Parser es una fábrica de objetos que crea instancias de XML::Parser::Expat según sea necesario.

Existe un módulo DOM para Perl que se implementa como una sub-clase de XML::Parser. Éste cumple con la sintaxis DOM nivel 1.

3.6. El Futuro de XML

Es creciente la coincidencia de criterios sobre el potencial que aporta XML al Web. XML es un metalenguaje de marcas y su objetivo no es la presentación de los datos sino la descripción de los mismos extendiendo su posibilidad de transferencia a través del web.

Al ser XML un metalenguaje que permite el manejo de datos estructurados, es de esperarse que con el tiempo vayan apareciendo nuevos lenguajes derivados del mismo y cuyo objetivo sea la transferencia y tratamiento de datos específicos para diversas actividades.

Una de las iniciativas que esta siendo impulsada en la actualidad por el grupo XML Query Work Group del W3C es la especificación del lenguaje de consultas XML Query. El objetivo de XML Query es definir un lenguaje interactivo de consultas lo suficientemente flexible como para cubrir un amplio espectro de fuentes de información incluyendo Bases de Datos y documentos distribuidos a través del web.

Una de las aplicaciones que tendría XML es el desarrollo de buscadores, ya que es más fácil encontrar los datos requeridos en un documento XML debido a que la información en los documentos XML está etiquetada por su significado, con la utilización de DTDs estandarizados para distintas aplicaciones se podrían programar buscadores web que recuperen información de acuerdo a lo que realmente se necesita conocer.

3.6.1. Lenguaje de consulta XML - XML Query Language

3.6.1.1. XML como almacén de datos

Una base de datos almacena datos en una forma sencilla y definida. Cada grupo de hechos relacionados está contenido en un registro, cada registro tiene un número fijo de campos, y cada campo contiene información relacionada.

Un documento XML también puede construirse de esta forma, pero también puede contener niveles profundos de significado que se expresan en la anidación de los elementos. Un documento XML suele tener un número variable de elementos secundarios, en contraposición a un número fijo de campos de una base de datos.

En un recordset, el orden de los registros no tiene significado. Los registros pueden reordenarse para facilitar la búsqueda (*proceso que recibe el nombre de indexación*), pero el orden en el que ocurren no conlleva una significación semántica. Éste no es el caso de XML, el orden de un elemento de un registro suele conllevar una significación. Cuando estos factores se toman en cuenta, XQL aporta las siguientes posibilidades, que no son necesarias en SQL:

- Debe ser capaz de indexar el orden de elementos iguales.
- Debe ser capaz de trabajar con un número irregular de campos.
- Debe ser capaz de describir relaciones entre primario-secundario y ascendiente – descendiente.

En una base de datos, el número de campos tiene que ser constante, además cada campo suele tener un determinado espacio fijo reservado para el almacenamiento de datos, esto ocaciona que se pierda espacio al almacenar documentos narrativos. XML está perfectamente adaptado para almacenar información estructurada e irregular.

3.6.1.2. Escenarios de uso de XQL (XML Query Language) [WWW011]

En el W3C existe un grupo de trabajo encargado del desarrollo de la especificación XML Query, actualmente los requerimientos para dicha futura especificación están como borrador de trabajo, es decir, hasta que llegue a convertirse en una especificación puede pasar algún tiempo.

La meta del Grupo Activo “XML Query” es producir un modelo de datos para documentos XML, un juego de operadores de consulta sobre ese modelo de datos, y un lenguaje de consulta basado en esos operadores de consulta.

Las consultas operan en documentos solos o colecciones fijas de documentos, pueden seleccionar documentos enteros o subárboles de documentos que emparejan las condiciones definidas en el contenido y estructura del documento, y puede construir nuevos documentos basados en lo que se selecciona.

Dentro del borrador de los requerimientos para XML Query, se describen cómo pueden usarse las consultas de XML en varios ambientes. Con XML Query se pretende:

- Realizar consultas en documentos estructurados y colecciones de documentos, como los manuales técnicos, recuperar documentos individuales, generar tablas de contenidos, para buscar información en las estructuras encontradas dentro de un documento, o para generar nuevos documentos como resultado de una consulta.
- Realizar consultas en representaciones XML de datos procedentes de una base de datos u otras fuentes de datos tradicionales; para extraer los datos de estas fuentes, para transformar los datos en nuevas representaciones de XML, o para integrar múltiples datos de fuentes heterogéneas.
- Realizar consultas orientadas-a-documento y orientadas-a-datos en documentos como catálogos, registros de la salud de pacientes, registros de empleo, o documentos comerciales de análisis.
- Realizar consultas en archivos de configuración, perfiles de usuario, o logs administrativos representados en XML.
- Realizar consultas en colecciones de documentos manejadas por almacenes nativos de XML o servidores web.
- Realizar consultas para buscar catálogos que describan servidores de documento, tipos del documento, esquemas de XML, o documentos. Tales catálogos pueden combinarse para apoyar la búsqueda entre servidores múltiples. Un sistema de recuperación de documentos podría usar las consultas para permitir al usuario seleccionar el catálogo de servidor, representado en XML. Una vez que se selecciona un servidor, un sistema de recuperación podría consultar los tipos de documentos encontrados en el servidor y podría permitirle al usuario consultar esos documentos.
- Pueden usarse las consultas en muchos ambientes. Por ejemplo, una consulta podría embeberse en una URL, una página XML, o una página ASP o JSP; representada por una cadena en un programa escrito en un lenguaje de programación de uso general.

3.6.2. Bases de Datos XML – XML DataBases (XDB) [www012]

Debido a las aplicaciones que se pueden crear con XML están surgiendo nuevas herramientas, entre ellas se encuentran las Bases de Datos Nativas para XML, que almacenan, actualizan, y recuperan documentos XML.

La necesidad de procesar y almacenar XML ha liberado varios nuevos tipos de herramientas de software, uno de los cuales es la "base de datos nativa XML".

Las bases de datos también están evolucionando, los sistemas de administración de bases de datos relacionales, o RDBMS. Oracle, SQL Server, y Sybase usan SQL, para acceder a los datos almacenados en tablas normalizadas compuestas de filas y columnas. Ahora, con la aparición de XML, la necesidad de almacenar rápidamente, recuperar, y actualizar documentos XML en tiempo real es necesario. La próxima generación de bases de datos, XDBS, necesitará sobresalir en esta capacidad.

Una XDB es un sistema de gestión de base de datos que almacena, actualiza, y recupera documentos XML. Usa como su unidad fundamental de almacenamiento un documento XML. Así como los SGBDD usan una fila en una tabla como su unidad base de almacenamiento, en una XDB todas las operaciones funcionan a nivel de un documento XML.

3.6.2.1. Tipos de XDBS

Las XDBS están en sus etapas formativas, se construirían sobre una base de datos orientada a objetos, otras podrían usar archivos comprimidos con un esquema de posicionamiento (indexación), y otras podrían construirse sobre una base de datos relacional.

Las XDBS pueden clasificarse en dos tipos básicos: nativas y habilitadas para XML. Una base de datos nativa para XML (NXDB) se diseñaría para guardar documentos XML. Podría hacer uso de una tecnología preexistente tal como las técnicas de almacenamiento de datos orientadas a objetos, pero su misión es almacenar, recuperar, y actualizar documentos XML.

Una base de datos habilitada para XML (XEDB), puede construirse en lo alto de un sistema de administración de base de datos relacional u orientada a objetos. Una XEDB provee una capa de mapeo entre el documento XML y la estructura de base de datos así como también soporte para herramientas basadas en XML para recuperar y actualizar documentos XML.

En la mira se tiene un tercer tipo de XDB que es una convergencia de los dos tipos anteriores: Es un XDB que se diseña para manejar documentos XML pero que se construye sobre una tecnología preexistente de base de datos, combinándolos en un modelo y depósito unificado de datos.

En todos los casos estos productos están en sus inicios y las funciones de bases de datos relacionales, tal como procesamiento transaccional, integridad referencial, y la agrupación, es débil. Todos los productos encaran un desafío importante en el mercado y la comunidad de desarrolladores en que los conceptos, técnicas, y las arquitecturas son nuevas a la industria.

3.6.2.2. Arquitectura de XML DB

En una XDB, los documentos XML pueden agruparse lógicamente en lo qué se llama una colección. Así como los datos en una RDBMS se almacenan en un conjunto de tablas, la arquitectura de una XDB se construye sobre la noción de colecciones. Las XDBs permiten administrar un grupo determinado de documentos XML en una colección, proveyendo la capacidad para consultar y actualizar los documentos en su totalidad.

Conjuntamente con la capacidad para construir y administrar documentos XML en una colección, una XDB provee la capacidad de crear índices en porciones del documento XML, asegurando que las porciones claves de un documento estén disponibles para la búsqueda rápida. Además, puesto que una XDB almacena un documento XML en su formato nativo, las consultas que necesitan retornar un conjunto de documentos XML también serán rápidas.

Finalmente, una XDB incorpora la creación de DTDs, esquemas XML, o ambos. Esto permite validar un documento XML contra un esquema determinado y almacenar ese esquema con la base de datos.

3.6.2.3. ¿Cuándo usar una XDB?

Esta pregunta surge con cierta frecuencia en desarrolladores XML, ya que considerarían la posibilidad de almacenar un documento XML en un RDBMS como un BLOB. Sin embargo, una distinción importante necesita ser hecha con respecto a cómo una aplicación hace uso de un documento XML.

La función primaria que un documento XML provee a una aplicación, los términos datos-céntricos y documento-céntrico.

- **Documento XML dato-céntrico**

  Un documento XML que se usa como el vehículo para transportar datos desde una de aplicación a otra, si es una aplicación B2B o simplemente entre el browser y la base de datos, es probable que sea dato-céntrico. Un documento XML dato-céntrico se caracteriza por su uso de contenido altamente estructurado, y su enfoque en datos de transferencia a otra aplicación para procesamiento. Es una de las fortalezas primarias de XML, y la que hace el impacto más grande sobre el mundo de negocios de hoy.

  Algunos ejemplos de documentos XML dato-céntrico podrían ser una orden de compra, cotización de acción, permiso de conducir, o libro de direcciones. Estos documentos mayormente contienen contenido estructurado y una cantidad mínima de texto descriptivo o de contenido mixto. Además, el receptor destinado del documento XML es otra aplicación, tal como un browser de Internet, un sistema de facturación, o una aplicación de tarjeta de crédito.

- **Documento XML documento-céntrico**

  Los documentos XML documento-céntrico son todo lo contrario: contienen una mezcla de contenido estructurado y semiestructurado, y más bien están destinados a que una persona los lea o a que una aplicación los procese. Algunos ejemplos de documentos XML documento-céntrico serían libros, artículos, informes, etc. Los documentos XML documento-céntrico se caracterizan por una estructura más informal y un mayor contenido de texto descriptivo o contenido mezclado

Puede siempre no ser fácil hacer la distinción entre documentos XML dato-céntrico y documento-céntrico. De hecho, la mayoría de documentos contienen una mezcla igual de elementos estructurados y contenido semiestructurado.

Aunque podría ser difícil, muchas veces se necesitará determinar el uso de un documento XML en una aplicación. Si el uso de un documento XML es como medio de intercambio de datos, entonces una base de datos relacional sería más apropiada. Pero si la aplicación usa un documento XML como medio para capturar contenido, especialmente documentos legibles para humanos, entonces una XDB sería más apropiada.

3.6.2.4. Ventajas y desventajas del uso de XDBS

Aquí están algunas ventajas claves de usar una XDB para administrar los documentos XML:

- **Velocidad:** Una XDB puede recuperar documentos XML y sus elementos mucho más rápido que un RDBMS o una base de datos habilitada para XML(puesto que almacena el documento XML nativamente). No necesita reensamblar el documento o desempeñar ninguna unión compleja, como una base de datos habilitada para XML.
- **Herramientas de soporte de XML:** Un RDBMS no tiene incorporado ninguna herramienta de soporte para XML o herramientas que son subordinadas al producto, como lo hace una base de datos habilitada para XML.
- **Administra contenido semiestructurado:** Una XDB es flexible en su manipulación de contenido semiestructurado y su capacidad para administrar cambios.
- **Independencia de Esquema:** Una XDB no requiere ningún esquema para almacenar un documento XML, considerando que una base de datos habilitada para XML lo hace.

En contraste, aquí están algunas de las deficiencias de usar un RDBMS o RDB habilitado para XML para soportar documentos XML:

- El mapeo entre un documento XML y las estructuras de base de datos puede ser bastante complejo. El mantenimiento de este mapeo puede ser difícil mientras que el sistema evoluciona con el tiempo.
- El mapeo entre un documento XML y la base de datos perderá inevitablemente algo de su metadata.

3.6.2.5. El futuro de XML y las bases de datos

Los datos no son más la unidad fundamental de cómputo; más bien, el nuevo paradigma es el documento XML. Las aplicaciones se están moviendo lenta pero constantemente desde un punto de vista dato-céntrico a una estructura orientada a documento-céntrico.

La próxima generación de bases de datos es un nuevo conjunto de aplicaciones o bases de datos XML nativas, estos son los aspectos que cualquier sistema basado en XML necesitará para trabajar efectivamente:

- **Un lenguaje robusto de consultas:** Conjuntamente con la capacidad para consultar a través de un conjunto de documentos XML en una colección, esta herramienta proveería también la capacidad para actualizar documentos y porciones de documentos (sin tener que actualizar el documento entero).

Idealmente, el lenguaje de consultas sería similar en estilo a SQL para que los desarrolladores no tengan aún otro lenguaje de consultas que aprender. Y este lenguaje sería capaz de consultar contra un conjunto de tablas de base de datos relacional, permitiendo a los desarrolladores escribir una consulta que puede reunir los datos desde ambos almacenes de datos.

- **Repositorio para colecciones de documentos XML:** Con esto un desarrollador de base de datos podría crear una base de datos en una XDB y poblarla con cualquier número de colecciones de documentos.
- **Característica del browser:** Esto permitiría a los usuarios examinar el conjunto de documentos que residen en una colección, como el Explorador de Windows que permite a un usuario examinar mediante un conjunto de archivos en una carpeta.
- **Característica de Índice:** Esto permitiría a los desarrolladores crear y mantener índices sobre un conjunto de documentos XML.
- **Soporte de Transacción:** Esto proveería soporte para actualizaciones sobre un documento XML o cualquier porción de este, inclusive soporte a rollback.
- **Búsqueda de Texto:** Esto proveería todas las avanzadas búsquedas que estamos acostumbramos a usar.

Lo que realmente se necesita mirar es lo que XML y los servicios web abren para los desarrolladores de aplicaciones, es decir la nueva línea de aplicaciones que permitirán construir. Se habla mucho de lo que es XML y sus características pero no sobre lo que esta tecnología permite construir con relación a lo que se construía antes. Lo qué es nuevo y diferente sobre XML y los servicios Web es la generalidad de los datos, los datos son ahora auto-descriptivos, y las aplicaciones ahora tienen una manera universal de cambiarlos. Con XML y los servicios Web se puede centrar en la aplicación actual; en vez de traducir información de una aplicación en datos que tienen que adaptarse exactamente en filas y columnas de una base de datos relacional, se puede enfocar en los documentos que una aplicación procesa y comenzar a pensar en los términos que más estrechamente se alinean con el mundo real.