UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

LENGUAJE XML COMO SOLUCIÓN A LAS BASES DE DATOS Y SU REPLICACIÓN

RENÉ AMILCAR MONROY HERNÁNDEZ
ASESORADO POR: ING. JORGE G. GÓMEZ MÉNDEZ

GUATEMALA, NOVIEMBRE DE 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LENGUAJE XML COMO SOLUCIÓN A LAS BASES DE DATOS Y SU REPLICACIÓN

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

RENÉ AMILCAR MONROY HERNÁNDEZ
ASESORADO POR: ING. JORGE G. GÓMEZ MÉNDEZ

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	
VOCAL II	Lic. Amahán Sánchez Alvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIO	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. José Ricardo Morales Prado
EXAMINADOR	inga. Elizabeth Domínguez Alvarado
EXAMINADOR	Ing. Juan Alvaro Díaz Ardavin
SECRETARIO	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

LENGUAJE XML COMO SOLUCIÓN A LAS BASES DE DATOS Y SU REPLICACIÓN

Tema que me fuera asignado por la Coordinación de la Carrera de Ingeniería en Ciencias y Sistemas en septiembre de 2003.

René Amilcar Monroy Hernández

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería

Guatemala, Octubre de 2005.

Por este medio informo que he revisado el trabajo de Graduación titulado: **"LENGUAJE XML COMO SOLUCIÓN A LAS BASES DE DATOS Y SU REPLICACIÓN"**, desarrollado por el estudiante **RENÉ AMILCAR MONROY HERNÁNDEZ**.

El mencionado trabajo llena los requisitos para dar mi aprobación, e indicarle que el autor y mi persona somos responsables por el contenido y conclusiones de la misma.

Atentamente,

Ing. Jorge G. Gómez Méndez
ASESOR

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Carrera de Ciencias y Sistemas

Guatemala, Noviembre de 2004.

Ingeniero
Jorge Armin Mazariegos
Coordinador de la Carrera de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Vettorazzi:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **RENE AMILCAR MONROY HERNÁNDEZ**, titulado: **"LENGUAJE XML COMO SOLUCIÓN A LAS BASES DE DATOS Y SU REPLICACIÓN"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad paa suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
Y Revisión de Trabajos de Graduación

ACTO QUE DEDICO

A Dios, por ser mi guía en mis momentos de alegría y superación; y mi guardián en mis momentos de tristeza y derrota. Por ser mi fuente de inspiración emocional inagotable, todo eso y más, es mi Dios.

A mis padres, porque soy fruto del esfuerzo que ustedes sembraron, porque dejaron a un lado los lujos de la vida cotidiana por darnos a mis hermanos y a mí lo más valioso: Educación y Moral.

Pero en especial a mi Papá, porque a pesar de todas las adversidades que hemos vivido como familia aún esperamos que te liberes de las cadenas que te separa de nosotros, “Papá ya es tiempo”.

A mis hermanos y familiares, por su comprensión y cariño brindado en todo momento.

A mis amigos, con ustedes pasamos buenos y malos momentos que nos ayudaron a ser mejores como personas. “**Porque cuando no quieres ver atrás y tienes incertidumbre de ver adelante, ve al lado y verás a un amigo**”. Gracias amigos.

Y muchas gracias a **las personas que me quisieron y apoyaron cuando menos lo merecía, porque era cuando más lo necesitaba**.

AGRADECIMIENTOS

A la Universidad de San Carlos de Guatemala, Facultad de Ingeniería, en especial a la Escuela de Ingeniería en Ciencias y Sistemas. A mi asesor de tesis, por el tiempo dedicado en esta etapa final de mi carrera. Y a todas las personas que de alguna forma han influido en mi formación académica.

ÍNDICE GENERAL

3. XML Y BASES DE DATOS.

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

ADO Objetos de datos *ActiveX - (ActiveX Data Objects)*. Mecanismo de acceso a datos de algo nivel propuesto por *Microsoft*. Puede utilizarse en distintas herramientas, como *Visual Basic, Delphi* e incluso *Scripts*.

API Interfaz para programas de aplicación – (*Application Program Interface*). Conjunto de convenciones de programación que definen cómo se invoca un servicio desde un programa.

ASCII Código estándar americano para el Intercambio de información - (*American Standard Code for Information Interchange*). Es un conjunto de caracteres considerados como estándar. Se compone de 128 códigos, del 0 al 127, en el que tienen cabida los caracteres utilizados habitualmente en el lenguaje inglés. No forman parte de *ASCII*, las letras acentuadas ni la Ñ, así como tampoco los caracteres de muchos lenguajes del éste.

ASP Páginas de servidor activas - (*Active Server Pages*). Se conoce como ASP al motor que permite procesar en el servidor *WEB* documentos compuestos de HTML y código ejecutable, generando documentos dinámicos ante la solicitud de los clientes.

Base de datos Colección de archivos almacenados en un sistema de computación. Son almacenados para un uso posterior.

DBMS *Database Management System*. Apelativo con el que se conocen las aplicaciones para la gestión de datos, se conoce como base de datos.

Bien-formado Un documento XML bien formado sigue todas las reglas de la especificación de XML, pero no es necesariamente válido, según una declaración de tipo de documento asociada.

CSS Hoja de estilo en cascada – (*Cascade Style Sheet*). Conjunto de instrucciones HTML que definen la apariencia de uno o más elementos de un conjunto de páginas *WEB* con el objetivo de uniformizar su diseño.

DOM *Document Object Model*. Es una plataforma que permite a los programas acceder dinámicamente y poner al día el contenido, estructura y estilo del documento.

DTD Definición del tipo de documento - (*Document Type Definition*). Encierra la definición formal de un tipo de documento y especifica la estructura lógica de cada documento.

Hardware Neologismo de uso universal característico del mundo de la informática. Son todas aquellas partes físicas que componen una computadora.

HTML Lenguaje de etiquetado de hipertexto – (*HyperText Markup Language*). Estándar mundial para el diseño de páginas o documentos que componen el *World Wide Web* o *WEB*. No es propiamente un lenguaje de programación sino de descripción de documentos, compuesto de un conjunto de etiquetas en el cual se especifican los atributos del contenido.

HTTP Protocolo de transferencia de hipertexto - (*HyperText Transfer Protocol*). Protocolo mundial usado para facilitar la comunicación entre los clientes y los servidores que conforman la *WEB*. Cada vez que se abre su navegador y accede a una sede web está utilizando el protocolo *HTTP*, para recuperar el documento, y el lenguaje HTML, que describe el documento.

Internet Un sistema mundial de redes interconectadas. Fue concebida en 1969 por un organismo del gobierno de Estados Unidos, *Advanced Research Projects Agency* (ARPA), en aquel momento se le llamó *ARPAnet*. De *ARPAnet* se dio origen a lo que conocemos ahora como Internet, un sistema público y cooperativo que ofrece gran variedad de servicios a todos sus usuarios.

JAVA Es un lenguaje de programación diseñado por *Sun Microsystems* en 1995 especialmente para usarse en el entorno de computación distribuida de la Internet.

JDBC Forma abreviada para *Java Database connectivity*. Es utilizado para ejecutar sentencias SQL en múltiples tipos de base de datos.

JDK *Java Development Kit* - Paquete de desarrollo para *Java*. Es un paquete de herramientas, creado y ofrecido por la empresa *Sun*, para permitir el desarrollo de aplicaciones para la plataforma *Java*.

OLAP Procesamiento analítico en línea - (*On Line Analytical Processing*). Se trata de procesos de análisis de información. Estos sistemas están orientados al acceso en modo consulta.

OLTP Procesamiento transaccional en línea - (*On Line Transactional Processing*). Se trata de los procesos clásicos de tratamiento automático de información, que incluyen altas, bajas, modificaciones y consultas.

OOP Programación orientada a objetos - (*Object Oriented Programming*). Metodología de programación basada en la modelación de objetos mediante código, definiendo sus estructuras de datos y las acciones que pueden realizarse sobre ellas. Para poder utilizar las técnicas de orientación a objetos se necesita un lenguaje de programación orientado a objetos.

***Parser* XML** Es un procesador que lee un documento de XML y determina la estructura y propiedades de los datos.

RAM Memoria de acceso aleatorio - (*Random Access Memory*). Memoria del ordenador que puede ser leída y escrita de manera aleatoria, sin necesidad de ir secuencialmente de un *byte* al siguiente.

RDBMS Sistema de gestión de bases de datos relacionales - (*Relational Database Management System*). Con este término se conoce a los servidores que gestionan bases de datos en las cuales la información se estructura en filas y columnas, creando tablas.

SAX API para XML basado en eventos, reporta los eventos como el inicio o final de los elementos de la aplicación y mientras pasan a través del documento.

Servidor Es un programa de computación que da servicios a otros programas de computación, ya sea en la misma computadora o en una diferente.

Software Programa o conjunto de programas con que se alimenta una computadora para que funcione. Se refiere a todo aquello que no se puede tocar físicamente en una computadora.

SQL Lenguaje de consultas estructurado - (*Structured Query Language*). Lenguaje orientado a la creación de consultas de bases de datos relacionales.

Web Server	Máquina conectada a la red en la que están almacenadas físicamente las páginas que componen un sitio *WEB*.
Web Service	Servicio público en internet el cual e generado por un servidor o por un programa instalado en el servidor.
XLL	Lenguaje extensible entre *Links* – (*eXtensible Linking Language*). Define el modo de enlace entre diferentes enlaces.
XML	Lenguaje extensible de marcas – (*eXtensible Markup Language*). Provee de un formato para la descripción de datos estructurados. Lo que pretende es crear un lenguaje estandarizado para el envío y recepción de información, así como búsquedas *WEB* sobre varias plataformas. Lenguaje de marcas que guarda similitudes con HTML al derivar de una misma raíz, tiene por finalidad describir la estructura de la información, en lugar de su aspecto.
XML *Server*	Servidores de internet especialmente diseñados para satisfacer los requerimientos del estándar XML.
XSL	Lenguaje extensible del estilo del documento – (*eXtensible StyleSheet Language*). Define e implementa el estilo de los documentos XML.

XSLT	Lenguaje extensible de transformación de estilos de documentos – (*eXtensive StyleSheet Language Tranformation*). Transforma un archivo de XML en un archivo HTML u otro formato basado texto para permitir que los browsers lo exhiban en una página *WEB*.

RESUMEN

El presente trabajo de graduación contiene un conjunto de conocimientos y explicaciones de la aceptación de la tecnología XML por su capacidad de integración y separación entre contenido y presentación existente en bases de datos y su facilidad de implementación.

El primer capítulo, describe porque XML se ha convertido en una de las más importantes tecnologías para la gestión de infraestructuras basadas en la *WEB*.

El segundo, hace referencia al concepto más esencial de sistema de información, como lo son las bases de datos, cuál ha sido la evolución y estado actual de la tecnología de bases de datos.

Para que en el tercer capítulo sea vista la necesidad de la existencia de estándares de intercambio de información respecto a bases de datos, con el objetivo de que las organizaciones puedan compartir su información de una manera más cómoda, más automática y eficiente.

Se finaliza dando una explicación de la forma como las empresas están resguardando su información, como haciendo más eficientes sus aplicaciones al replicar información para evitar congestionar sus servidores, asegurando un tiempo de respuesta mayor.

OBJETIVOS

- ## General

Demostrar la forma como los diferentes proveedores de bases de datos aplican y utilizan soporte XML en bases de datos y su replicación.

- ## Específicos

1. Proporcionar el conocimiento necesario a estudiantes y profesionales de la tendencia actual de integración en cuanto a contenido y presentación de la información que contiene una base de datos a través de la tecnología XML.

2. Dar una explicación y demostración real de las ventajas que conlleva la implementación de bases de datos con soporte XML.

3. Aportar el conocimiento de la diferencia de bases de datos con soporte XML y bases de datos XML nativas.

4. Detallar las características más importantes de los 3 proveedores de bases de datos con soporte XML más grandes en el mercado y un análisis comparativo entre ellos.

5. Dar una explicación y demostración de cómo los proveedores de bases de datos implementan el servicio de replicación de sus bases de datos con soporte XML.

INTRODUCCIÓN

Con el lenguaje XML, finalmente se tienen los medios para conectar los sistemas de información con prácticamente cualquier sistema. Los datos de XML bien-formado, constituye el desarrollo de aplicaciones más rápidas y poderosas, y la naturaleza extensible de sus medios normales puede adaptarse a una gran variedad de usos.

El volumen del trabajo y la carga de información involucrado no ayuda a decidir cómo debe ser la estructura de la base de datos. Cualquier base de datos debe tener robustez en copias (*backup*), opciones de restauración, replicación, un optimizador de consultas, integridad del referencial y un modelo de la transacción sólido. Mirando a los proveedores de bases de datos se duda que ellos realmente puedan construir y sostener esas capacidades más rápidamente que los vendedores de RDBMS que incorpora el almacenamiento de XML nativo.

Hasta hace unos años, la replicación de bases de datos era una medida extrema que las pocas compañías consideraban, y mucho menos llegar a implementarlo. Ahora, más negocios están interesados en la replicación de bases de datos por dos razones principales: Los negocios están más dispersos geográficamente y sus empleados necesitan el acceso a sus datos, y muchas compañías trabajan con gran cantidad de Información.

1. INTRODUCCIÓN A XML

1.1 ¿Qué es XML?

El lenguaje extensible de marcas (XML, *eXtensible Markup Language*) es un meta-lenguaje de marcas que provee de un formato para la descripción de datos estructurados. Esto facilita las declaraciones, haciéndolas más precisas y obteniendo unos resultados de búsquedas más significativos en varias plataformas.

El lenguaje extensible de marcas XML, es un subconjunto de SGML (*Standar Generalized Markup Language*) optimizado para el *WEB*. Fueron creados bajo la supervisión del *Word Wide Web Consortium* (*W3C*) organismo que vela por el desarrollo de internet y sus estándares.

XML y SGML, son metalenguajes, es decir, lenguajes que ofrecen la posibilidad de definir otros lenguajes, con la peculiaridad de ser más simple al momento de describir documentos. El lenguaje extensible de marcas XML, es un lenguaje que fue creado para dotar a la *WEB* o a una red corporativa de más funcionalidad, para poder acceder a la información de una forma más flexible y adaptable.

Esto sucede, porque solo almacena información y además lo hace de forma estructurada, ya que la propia información tiene estructura interna. En cambio HTML almacena la información junto con su formato, esto hace que se cree una gran dependencia entre la información, el formato y el programa con el que fue creado. Ya que si sólo se almacena la información y su estructura, fácilmente se pueden dar múltiples formatos al documento.

1.2 Características de XML

- XML es un lenguaje más pequeño que SGML. Los diseñadores de XML recortaron las partes de SGML que no fueran necesarias para el manejo de datos en *WEB*. El resultado un lenguaje más simple y adaptado a internet.

- XML estructura los datos.

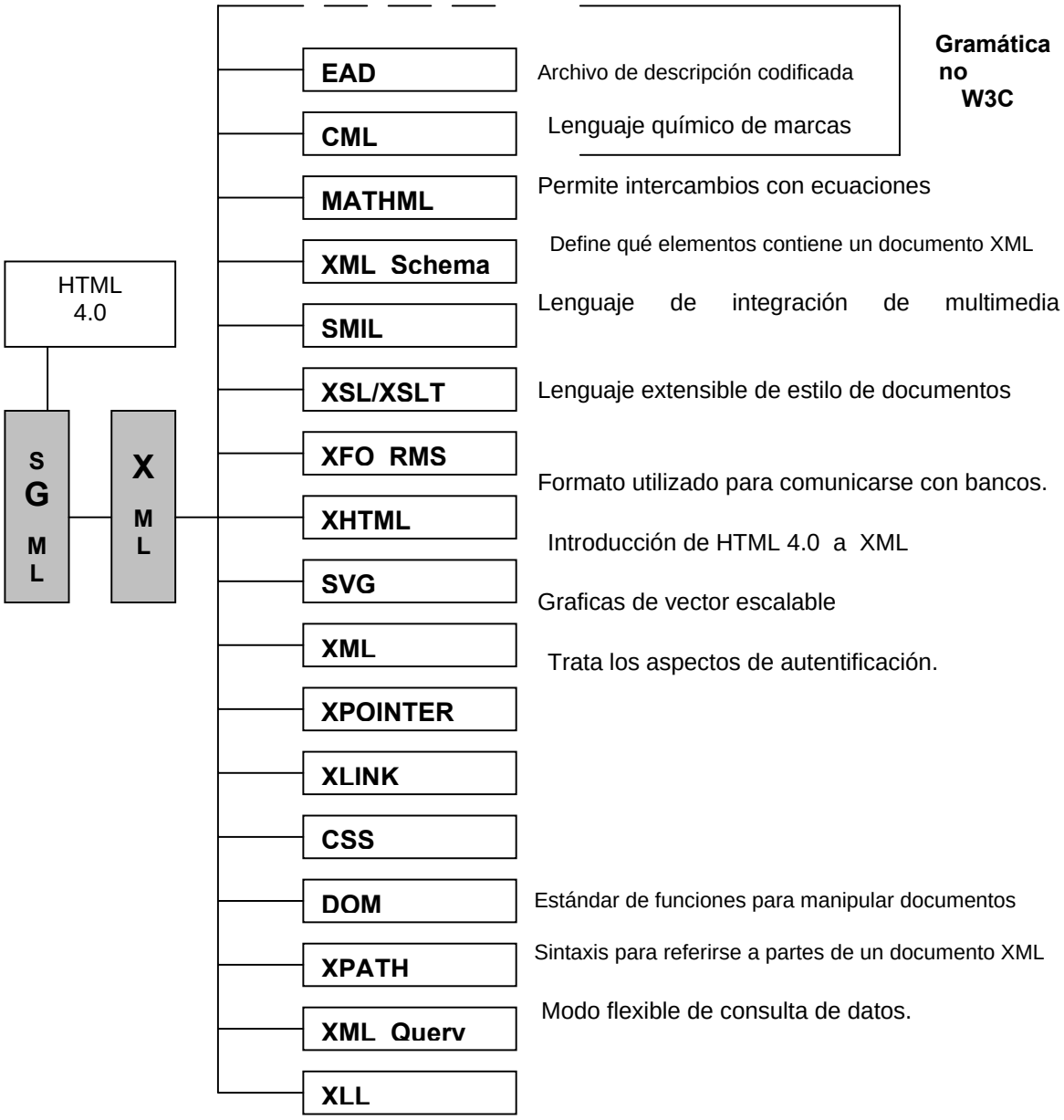
- XML es texto. Lo que permite a los usuarios interpretar el contenido de los archivos y modificar las distintas etiquetas.

- XML es una familia de tecnologías. Dicha familia, XSL (*eXtensive StyleSheet Language*), DTD (*Document Type Definition*), XLL(*eXtensible Linking Language*), entre las más importantes (Figura 1); está formada por un conjunto de módulos que ofrecen servicios útiles para cubrir distintas tareas requeridas.

- Se parece a HTML. Ambos hacen uso de etiquetas y pertenecen a la familia SGML. Mientras HTML especifica qué es cada etiqueta y cómo se visualiza en el navegador, XML usa las etiquetas para delimitar piezas de datos y deja la interpretación para la aplicación que los lee.

- XML goza de licencia libre y es multiplataforma.

Figura 1. Familia de XML



En ningún caso XML pretende sustituir a HTML, sino impulsar su evolución hacia XHTML (*eXtensible HyperText Markup Language*), el nuevo estándar de HTML que integra las reglas de XML y facilita su comunicación.

Algunos de los objetivos planteados por el grupo de trabajo XML y el *W3C* son:

- XML debe ser directamente utilizable sobre internet.
- XML debe soportar una amplia variedad de aplicaciones.
- XML debe ser compatible con SGML.
- Debe ser fácil la escritura de programas que procesen documentos XML.
- El número de características opcionales en XML debe ser absolutamente mínimo, idealmente cero.
- El diseño de XML debe ser formal, conciso y preparado rápidamente.
- Los documentos XML deben ser fácilmente creables.
- La brevedad en las marcas XML es de mínima importancia.

1.3 Especificaciones de XML

1.3.1 XLL

XLL, *eXtensible Linking Language*. Es el estándar encargado de definir el modo de enlace entre diferentes enlaces. Este lenguaje extensible tiene dos importantes componentes: *Xlink* y el *Xpointer*. *Xlink* define la forma en la que los documentos XML deben conectarse entre sí. *Xpointer* describe cómo se puede apuntar a un lugar específico de un determinado documento XML.

1.3.2 CSS

La especificación CSS, *Cascading Style Sheets*. Se utiliza para agregar el estilo del formato a un archivo de HTML. Las hojas de estilo son un lenguaje suplementario que fue especialmente desarrollado para HTML.

Puede ser utilizada para dar forma a XML, pero no puede transformar y generar estructuras nuevas. Se expresa mediante reglas en un fichero de texto plano. Cada regla contiene el nombre del elemento al que se aplica y el estilo definido de estilos a un elemento XML.

1.3.3 XSL

XSL, *eXtensible StyleSheet Language*, antes llamado "*xml-style*". Su función es la de definir e implementar el estilo de los documentos XML. Permite modificar el aspecto de un documento. Este estándar está basado en el lenguaje de semántica y especificación de estilo de documento. Guarda cierta relación con las hojas de estilo en cascada (CSS), pero según la especificación del *W3C* se espera que el CSS sea usado para visualizar simples estructuras de documentos XML y XSL puede ser utilizado donde se requiera más potencia de diseño como documentos XML que encierran datos estructurados (tablas, organigramas, etc.). Se considera más potente que las hojas de estilo en cascada.

1.3.4 XSLT

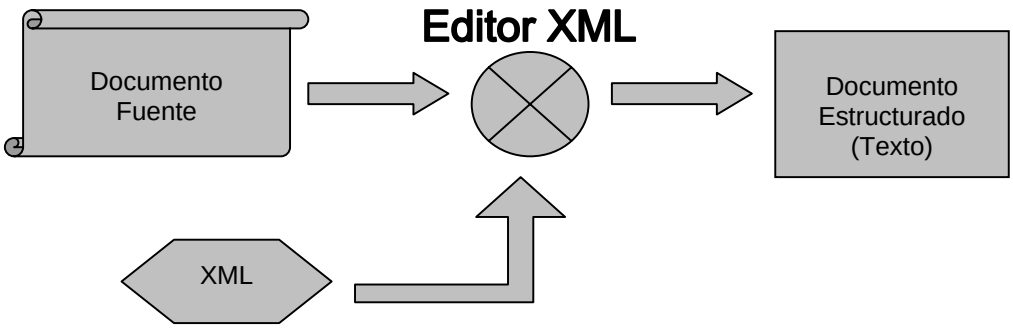
XSLT, *eXtensive StyleSheet Language Tranformation*. Permite transformar un archivo de XML en un archivo de HTML u otro formato basado texto. Puesto que los browsers no pueden exhibir documentos de XML directamente, es necesario transformar XML en el HTML para permitir que los browsers lo exhiban en una página *WEB*, como resultado la transformación XSLT.

1.4 Arquitectura y sintaxis de XML

1.4.1 Características de la arquitectura

- Un metalenguaje y una metodología de diseño
- Sistema de capas envolventes.
- Cimientos.
- Estructura.
- Fachada.
- Subsistemas especializados.
- Técnicas normalizadas de ensamblado de componentes.
- Paralelismo con la construcción.

Figura 2. Arquitectura básica



1.4.2 Sintaxis de XML

Un ejemplo de la sintaxis que utiliza XML es:

```
<!ELEMENT tabla (fila*)>
<!ELEMENT fila (Id, Titulo, Autor)>
<!ELEMENT Id (#PCDATA)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Autor (#PCDATA)>
```

```
<?xml version="1.0" ?>
<!DOCTYPE tabla SYSTEM "tabla.dtd">
<tabla>
   <fila>
      <Id>
         1
      </Id>
      <Titulo>
        Don Quijote de la Mancha
      </Titulo>
      <Autor>
        Miguel de Cervantes
      </Autor>
    </fila>
   <fila>
      <Id>
         2
      </Id>
      <Titulo>
         Ivanhoe
      </Titulo>
      <Autor>
         Walter Scott
      </Autor>
    </fila>
   <fila>
      <Id>
         3
      </Id>
      <Titulo>
         El Nombre de la Rosa
      </Titulo>
      <Autor>
         Umberto Eco
      </Autor>
    </fila>
</tabla>
```

Tabla I. Resultado del uso de la sintaxis de XML

El resultado sería una tabla con las siguientes filas y columnas:

Id	Titulo	Autor
1	Don Quijote de la Mancha	Miguel de Cervantes
2	Ivanhoe	Walter Scott
3	El Nombre de la Rosa	Umberto Eco

1.5 DTD Y XML esquema

1.5.1 DTD

(DTD, *Document Type Definition*), definición del tipo de documento. Es un archivo que encierra una definición formal de un tipo de documento y especifica la estructura lógica de cada documento. Define tanto los elementos de una página como sus atributos. Se define mediante:
<!DOCTYPE etiquetaraiz SYSTEM "fichero.dtd" [declaración interna]>

1.5.2 XML esquema

Se implementa como vocabulario XML. Define qué elementos puede contener un documento XML, cómo están organizados, y que atributos y de qué tipo pueden tener sus elementos.

1.6 XML como arquitectura de sistemas de información

Una aplicación XML representa un modelo de datos:

- Codifica datos de acuerdo a un esquema "semántico".
- Analiza y extrae información de un documento (repositorio) XML.
- Obtiene información de otras fuentes y puede combinarse con el documento XML para un posterior proceso.
- Codificar la información en XML y genera un fichero de texto donde se combina la información fuente con metainformación estructural.
- Un documento XML es válido si cumple un conjunto de restricciones estructurales denominado gramática.

- XML como lenguaje para representar Meta-modelos.
- UML representado en XML.

1.7 Uso de XML

1.7.1 Gestión documental

- Edición y publicación electrónica. Ofrecer mecanismos para mostrar datos. Bajo el nombre de DOM (*Document Object Model*) se está desarrollando una API que sea soportada por todos los procesadores de XML y HTML. La idea detrás de esta API es que se pueda representar documentos XML en los navegadores *WEB*.

- Gestión de grandes volúmenes de documentos.

- Buscadores inteligentes. Debido a que la información en los documentos XML está etiquetada por su significado de forma precisa.

- Procesamiento del lenguaje. Diccionarios, herramientas filológicas, etc.

- Gestión de conocimientos. Redes semánticas, tesauros, etc.

- Intercambio de información entre sistemas heterogéneos.

- Documentación técnica de manuales. Especificación de *software*, configuración, otros.

1.7.2 Sistemas transaccionales

- Información más accesible y reutilizable, porque la flexibilidad de las etiquetas de XML pueden utilizarse sin tener que amoldarse a reglas específicas de un fabricante, como es el caso de HTML.

- Transacciones proveedor-consumidor: *Suply-chain*.

- Acceso a transacciones internas de la empresa. Acceso en tiempo real al catálogo.

- Transacciones entre empresas. Interoperabilidad entre sistemas de información heterogéneos.

1.7.3 Desarrollo de aplicaciones con XML

Existen cinco tipos de aplicaciones que impulsarán el desarrollo del XML:

- Se irá pareciendo cada vez más a una arquitectura cliente-servidor.

- Aplicaciones que intentan transferir una parte significativa de la carga del proceso del servidor al cliente *WEB*.

- Aplicaciones que precisen que el cliente *WEB* presente diferentes versiones de los mismos datos a diferentes usuarios.

- Aplicaciones en las que agentes *WEB* inteligentes intentan adaptar la búsqueda de información a las necesidades de los usuarios individuales.

- Operaciones para comercio electrónico.

Es importante destacar que XML aporta mucha potencia y flexibilidad a las aplicaciones basadas en la Web, proporcionando numerosas ventajas a los programadores y usuarios:

- Búsquedas con más significado.

- Programación de aplicaciones *WEB* flexibles.

- Integración de datos procedentes de fuentes dispares, la capacidad de buscar en varias bases de datos no compatibles entre sí.

- Varias vistas de los datos, XML sirve de complemento para el HTML.

- Actualizaciones granulares, no es necesario volver a enviar un conjunto completo de datos estructurados cada vez que cambia parte de dichos datos, sólo es preciso enviar el elemento modificado del servidor al cliente.

2. PANORAMA GENERAL DE LA ADMINISTRACIÓN DE BASES DE DATOS

2.1 ¿Qué es un sistema de bases de datos?

Un sistema de bases de datos es un sistema computarizado para llevar registros; también se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada; o un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos cuya finalidad es almacenar información con base en peticiones, por ejemplo:

- Agregar nuevos archivos vacíos a la base de datos.
- Insertar datos dentro de los archivos existentes.
- Recuperar datos en archivos existentes.
- Modificar datos en archivos existentes.
- Eliminar datos de los archivos existentes.
- Eliminar archivos existentes.

Cualquier base de datos actual requiere un mantenimiento, basado en una buena administración y gestión de los datos que contiene, de los procesos asociados y de los usuarios que pueden acceder. Debe contener entidades de información relacionadas vía organización y asociación. La arquitectura lógica de una base de datos se define mediante un esquema que representa las definiciones de las relaciones entre las entidades de información.

La arquitectura física de una base de datos depende de la configuración del *hardware* residente.

2.2 ¿Por qué una base de datos?

Algunos casos donde se aprecia las ventajas de un sistema de bases de datos sobre los métodos tradicionales basados en papel:

- Compactación. No hay necesidad de archivos en papel.
- Velocidad. Las máquinas pueden manipular datos más rápidamente.
- Menos trabajo laborioso. Se puede eliminar gran parte del trabajo de llevar los archivos a mano.
- Actualidad. Cuando se necesite, la información estará a nuestra disposición precisa y actualizada.
- Globalización de la información.
- Permite mantener la integridad en la información. Tiene por objetivo que sólo se almacena la información correcta.
- Independencia de datos. Entre programas y datos.

2.2.1 Beneficio del enfoque de base de datos

Algunas ventajas especificas que surgen del control centralizado:

- Los datos pueden compartirse. Desarrollar nuevas aplicaciones para operar sobre los mismos datos.

- Es posible reducir la redundancia. Cada aplicación tiene sus propios archivos exclusivos, produciendo desperdicio de espacio de almacenamiento. No toda la redundancia puede o debe ser eliminada, en ocasiones hay razones sólidas, tácticas o del negocio. Cualquier tipo de redundancia debe ser controlada cuidadosamente.

- Es posible evitar la inconsistencia. Es inconsistente cuando no toda la información ha sido actualizada, proporcionando información incorrecta o contradictoria. Por lo tanto se asegura que todo cambio realizado a cualquiera de las dos entidades será aplicado también a la otra en forma automática.

- Es posible brindar un mejor manejo de transacciones.

- Es posible mantener la integridad.

- Es posible hacer cumplir la seguridad.

2.3 Sistemas de bases de datos relacionales

Edgar F. (Ted) Codd, un matemático formado en *Oxford*, que había entrado en *IBM* en 1949, empezó a trabajar en una serie de informes técnicos acerca de una manera “nueva” de organizar y acceder a los datos.

Codd propuso que los sistemas de bases de datos deberían presentarse a los usuarios con una visión de los datos organizados en estructuras llamadas relaciones.

Las relaciones se definen como conjuntos de tuplas, donde el orden no es importante, detrás de una relación puede haber cualquier estructura de datos compleja que permita una respuesta rápida a una variedad de consultas. Concibió un sistema donde el usuario sería capaz de acceder a la información con comandos parecidos al lenguaje natural y donde la información estuviera almacenada en "tablas".

Durante su desarrollo se produjo la publicación de la separación en tres niveles de los SGBD (sistema de gestión de base de datos) descrita por el informe del comité *ANSI/SPARC* de 1975. Los niveles eran **externo**, **conceptual** e **interno**.

Respecto a los lenguajes de consulta, Codd introdujo el álgebra relacional, consideró uso de la lógica para realizar consultas. Y es el SQL, una mezcla de sintaxis inspirada en el lenguaje natural (inglés), que se convierte en el lenguaje estándar de consultas.

2.4 Sistemas de bases de datos orientadas a objetos

Con la entrada de los lenguajes orientados a objetos, los investigadores se plantearon transportar estas ideas a las bases de datos y permitir que el tipo de datos marcara cómo se representaba y se manipulaba dependiendo de los métodos que se definían para dicho tipo o clase.

En este modelo la información sobre una entidad se almacena como un objeto persistente y no como una fila en una tabla.

Esto, lo hace más eficiente en términos de requerimientos de espacio y asegura que los usuarios puedan manipular los datos sólo de las maneras en las que el programador haya especificado. También es más eficiente en el uso de espacio de disco requerido para las consultas, ya que en vez de almacenar la consulta, simplemente se construye una serie de índices (punteros) a los objetos seleccionados.

Se podría pensar, que las bases de datos orientadas a objetos deberían de haber superado en la práctica a las relacionales. A veces se denominan **postrelacionales**.

2.5 Sistemas de bases de datos distribuidas

Una base de datos distribuida (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas en diferentes máquinas interconectadas a través de una red de comunicaciones, las cuales tienen la capacidad de un procesamiento autónomo, pero pueden realizar operaciones locales o distribuidas.

Los principales factores que distinguen un SBDD de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben de estar comunicados por medio de algún tipo de red.

Las características de las bases de datos son las siguientes:

- Autonomía local. Todas las operaciones de sitio se controlan en ese sitio.
- No dependencia de un sitio central.

- Operación continua. Nunca debería apagarse para que realice alguna función.
- Independencia con respecto a la localización.
- Independencia con respecto a la fragmentación. Los datos pueden almacenarse en la localidad donde se utilizan con mayor frecuencia.
- Procesamiento distribuido de consultas. Convertir transacciones de usuario en instrucciones para manipulación de datos.
- Manejo distribuido de transacciones. Control de recuperación y concurrencia.
- Independencia de réplica. Copias almacenadas o réplicas.
- Independencia con respecto al equipo.
- Independencia con respecto al sistema operativo.
- Independencia con respecto a la red.

2.5.1 Ventajas y desventajas de las bases de datos distribuidas.

2.5.1.1 Ventajas

- Mediante la replicación, las bases de datos distribuidas pueden presentar cierto grado de tolerancia a fallas.
- Mayor fiabilidad y disponibilidad.
- Mejor rendimiento.
- Los datos se pueden colocar en donde se accedan más frecuentemente.

2.5.1.2 Desventajas

- El control y manejo de los datos, debido a violaciones de seguridad.
- La habilidad para asegurar la integridad de la información en presencia de fallas.
- El control de concurrencia y los mecanismos de recuperación son mucho más complejos.

2.6 Arquitectura de los sistemas de bases de datos relacionales

2.6.1 Niveles de la arquitectura

La arquitectura se divide en tres niveles, conocidos como **interno**, **conceptual** y **externo**. (Figura 3).

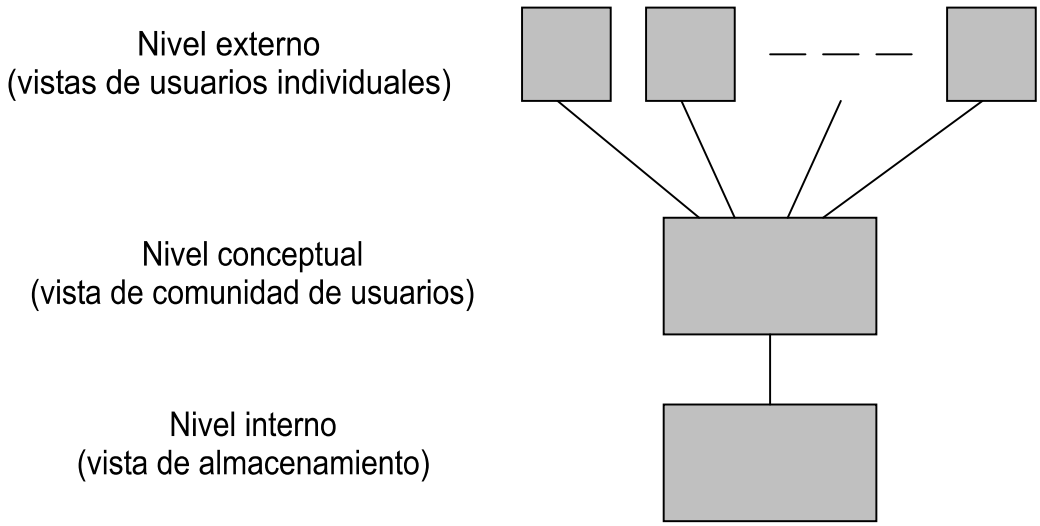
- El nivel interno, también conocido como nivel físico, es aquel que esta más cerca del almacenamiento, es el que tiene que ver con la forma en que los datos se encuentran almacenados físicamente.

- El nivel conceptual, también conocido como nivel lógico, es un nivel de indirección entre el nivel interno y el externo.

- El nivel externo, también conocido como nivel lógico de usuario, es el más próximo a los usuarios, es el que tiene que ver con la forma en que los usuarios individuales ven los datos.

El nivel externo tiene que ver con las percepciones de usuarios individuales, mientras que el nivel conceptual tiene que ver con la percepción de una comunidad de usuarios.

Figura 3. Niveles de la arquitectura



2.7 El sistema de administración de bases de datos

Entre la base de datos física y los usuarios del sistema, hay una capa de *software* conocida como sistema de administración de base de datos (DBMS). Todas las solicitudes de acceso a la base de datos son manejadas por el DBMS.

El DBMS es el componente de *software* más importante del sistema en general, otros comprenden las utilerías, herramientas de desarrollo de aplicaciones, ayudas de diseño, generadores de informes y el administrador de transacciones o monitor.

Al hacer la petición al DBMS sucede lo siguiente:

1. Un usuario emite una petición de acceso, utilizando algún sublenguaje de datos especifico (por ejemplo: SQL).

2. El DBMS intercepta esa petición y la analiza.

3. El DBMS inspecciona el esquema externo para ese usuario, la transformación externa/conceptual, el esquema conceptual, la transformación conceptual/interna y la definición de la estructura de almacenamiento.

4. El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada.

2.7.1 Administrador de datos

Los datos son uno de los activos más valiosos de una empresa, lo que implica que habrá alguna persona identificable que tendrá la responsabilidad central sobre el manejo de los datos. Es labor del administrador de datos o **DA** decidir en primer lugar qué datos deben ser almacenados en la base de datos y establecer políticas para mantener y manejar esos datos una vez almacenados.

2.7.2 Administrador de base de datos

El técnico responsable de implementar las decisiones del administrador de datos es el administrador de base de datos o *DBA*. Por lo tanto, el *DBA*, a diferencia del administrador de datos, es un profesional.

El trabajo del *DBA*, consiste en crear la base de datos real e implementar los controles técnicos necesarios para hacer cumplir las diversas decisiones de las políticas hechas por el administrador de datos.

El *DBA* también es responsable de asegurar que el sistema opere con el rendimiento adecuado y de proporcionar una variedad de otros servicios técnicos. Por lo regular, el *DBA* tendrá un equipo de programadores de sistemas y otros asistentes técnicos.

2.8 Creación de una base de datos

Para crear una base de datos se deben realizar dos etapas de diseño: **un diseño lógico** y **uno físico**.

El diseño lógico de una base de datos es un modelo abstracto de la base de datos desde una perspectiva de negocios, mientras que el diseño físico muestra como la base de datos se ordena en realidad en los dispositivos de almacenamiento de acceso directo. El diseño físico de la base de datos es llevado a cabo por los especialistas en bases de datos, mientras que el diseño lógico requiere de una descripción detallada de las necesidades de información del negocio de los negocios actuales usuarios finales de la base de datos.

2.8.1 Tipos de base de datos según su diseño lógico

- Bases de datos documentales. Se derivan de la necesidad de disponer de toda la información en el puesto de trabajo y de minimizar los tiempos del acceso a aquellas informaciones que no están estructuradas convenientemente. Esto se debe a que a la procedencia de la información es muy variada (informes, notas diversas, periódicos, revistas, otros).

- Bases de datos distribuidas. Es aquella que se almacena en más de un lugar físico. Partes de la base de datos se almacena físicamente en un lugar y otras partes se almacenan y mantienen en otros lugares.

- Bases de datos orientadas a objetos e hipermedia. Estas son capaces de almacenar tanto procesos como datos. Las bases orientadas al objeto deben almacenar información no convencional (como imágenes estáticas o en movimiento, colecciones de sonidos, entre otros).

2.9 Sistemas de gestión de bases de datos

Sistema desarrollado que hace posible acceder a datos integrados que atraviesan los límites operacionales, funcionales u organizacionales de una empresa.

Objetivos en el uso de un sistema de gestión de base de datos:

- Oportunidad, asociado a la eficiencia y eficacia.
- Disponibilidad, permitiendo la accesibilidad de datos.
- Consistencias (oportunidad + disponibilidad), como calidad de datos.
- Evolución, para adaptarse al entorno.
- Integridad, en el nivel de los datos así como el sistema.

Objetivos del sistema de gestión de base de datos que podemos identificar son:

- Independencia de datos.
- Accesibilidad limitada.
- Datos al día y sin redundancias.
- Consistencia.
- Interfaz única.
- Entrada directa a los datos.
- Recuperación por diferentes accesos.
- Función completa de interrogantes.
- Estandarización.
- Seguridad.

Las funciones básicas de un sistema de gestión de base de datos son:

- Permitir a los usuarios crear nuevas bases de datos y especificar su estructura, utilizando un lenguaje o interfaz especializado, llamado lenguaje o interfaz de definición de datos.

- Dar a los usuarios la posibilidad de consultar los datos y modificarlos, utilizando un lenguaje o interfaz apropiado, llamado lenguaje de consulta o lenguaje de manipulación de datos.

- Permitir el almacenamiento de grandes cantidades de datos durante un largo periodo de tiempo, manteniéndolos seguros de accidentes o uso no autorizado y permitiendo un acceso eficiente a los datos para consultas y modificaciones.

- Controlar el acceso a los datos de muchos usuarios, impidiendo que las acciones de un usuario puedan afectar a las acciones de otro sobre datos diferentes y que el acceso simultáneo no corrompa los datos.

2.10 Bases de datos y la *WEB*

Uno de los aspectos que se han notado más recientemente en el campo de las bases de datos es el crecimiento vertiginoso de internet. La conexión de las bases de datos con la *WEB* ha ido progresando de una interrelación realizada a través de herramientas, en la que prácticamente todo SGBD proporciona un módulo o toda una serie de herramientas para publicar la información de la base de datos en la red, siendo accesible desde cualquier punto, utilizando un navegador.

Hay que destacar que la tecnología *WEB* ha hecho evolucionar la tecnología cliente/servidor de dos capas a una tecnología estructura en tres capas (1. cliente / 2. servidor de aplicaciones / 3. servidor de datos), aunque la mayoría de los aspectos del paradigma cliente/servidor son aplicables a la *WEB*.

La mayoría de accesos *WEB* de hoy en día disparan alguna forma de generación de contenido de una base de datos, mientras que el comercio electrónico está destinado a hacer un uso intensivo de las aplicaciones basadas en un SGBD.

El interés de la comunidad científica y de las empresas creadoras de sistemas de bases de datos se centra en la revisión o extensión de modelos de datos y de lenguajes de consulta, la integración de datos tan diversos, la reconcepción de los índices, las transacciones y el procesamiento de consultas, con el objetivo de adaptarse a las características y la escala de los datos en la *WEB*.

Se han identificado nuevos problemas, como saber tratar con el solapamiento de información y la detección de copia, así como cuestiones específicas de la *WEB* como herramienta de publicación.

3. XML Y BASES DE DATOS

3.1 Definición de bases de datos XML

El e*Xtended Markup Language*, XML, es un estándar potente y de amplia aceptación para guardar y comunicar información acerca de objetos; esto es documentos, imágenes, fotos o incluso objetos del mundo real. Permite la codificación de información separada de la forma en la que se debería presentar al usuario.

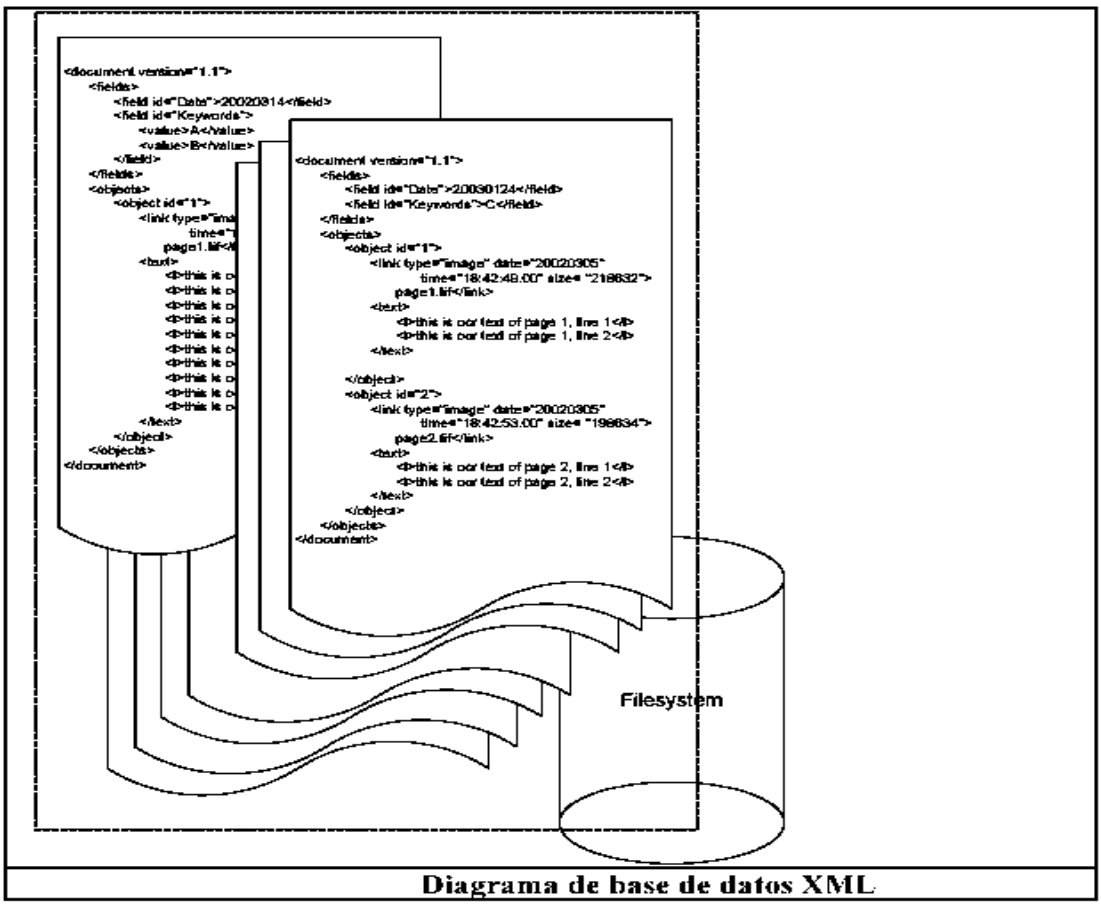
En su esencia los datos XML, son simplemente datos *Unicode* o *ASCII* que se pueden ver como un documento de texto estructurado lógicamente. Los archivos XML se almacenan en un archivo, o sobre todo como registros en múltiples archivos. Estos archivos se pueden cargar en un analizador XML. Para encontrar un fragmento específico de información en los contenidos de un nodo o atributo XML, habrá que procesar completamente el archivo XML.

Una base de datos XML se puede ver como una colección de documentos XML en la que cada documento representa un registro (Figura 4). Cada documento XML es un archivo en el sistema de archivos y contiene una cadena válida XML.

La estructura de un documento XML sigue un mismo esquema XML, aunque no es necesario que sea así. Este es uno de los beneficios de las bases de datos XML, ya que cada archivo se puede configurar de forma estructurada por lo que es independiente, pero fácilmente accesible.

El tener colecciones de documentos con un esquema independiente le proporciona a la base de datos flexibilidad y hace que el desarrollo de la aplicación sea más fácil.

Figura 4. Diagrama de base de datos XML



En los últimos años se ha visto necesaria la existencia de estándares de intercambio de información, con el objetivo de que las organizaciones puedan compartir su información de una manera más cómoda, automática y eficiente.

En lo que concierne a las bases de datos, el XML permite integrar sistemas de información hasta ahora separados:

- Sistemas de información basados en documentos (ficheros ó archivos), tienen estructura irregular, anidados profundamente, utilizan tipos de datos relativamente simples y dan gran importancia al orden.

- Sistemas de información estructurados (bases de datos relacionales) tienen una estructura muy regular, son relativamente planos, utilizan tipos de datos relativamente complejos y dan poca importancia al orden.

El hecho de que el XML permita expresar información de características y estructuras muy diversas **no quiere decir que XML venga a sustituir a las bases de datos tradicionales**.

- XML no es una base de datos.
- No es tampoco un modelo de datos.
- XML es simplemente un lenguaje de marcas, y un documento XML, no es nada más que un documento de texto.

Cuando entendemos un documento XML y su DTD asociada, la interpretación del mismo respecto a una semántica; podemos llegar a comparar la tecnología XML con las bases de datos. Aparecen muchas cosas en común, sin que éste sea una base de datos:

- La tecnología XML usa uno o más documentos (ficheros ó archivos) para almacenar la información.

- Define esquemas sobre la información (DTD's y otros lenguajes de esquema XML).

- Tiene lenguajes de consulta específicos para recuperar la información requerida (*XQL, XML-QL, QUILT*, etc.).

- Dispone de interfaces de programación (SAX, DOM).

Pero aparecen muchas más cosas que lo diferencian. La tecnología XML carece, de algunas de las siguientes características comunes en las bases de datos, debido a que no es en si una base de datos:

- Almacenamiento y actualización eficientes.
- Índices, seguridad, transacciones, integridad de datos, acceso concurrente, disparadores, etc.

Por tanto es imposible pensar que XML se vaya a utilizar para las tareas transaccionales de una organización para las cuales sigue estando sobradamente más justificado utilizar una base de datos.

A pesar a ser un lenguaje de marcas, XML permite representar la mayoría de modelos de datos existentes: de red, jerárquico, relacional, objetual, otros. **Se puede representar la información contenida en una base de datos relacional en uno o más documentos XML**.

La interrelación entre XML y el modelo objetual es cada vez más fuerte, especialmente en lenguajes de consulta.

Éstos suelen inspirarse en el SQL y en otros lenguajes de consultas, aunque también utilizan ciertas nociones de la programación funcional y de las técnicas de búsqueda de rutas de árboles de directorios, ya que un documento XML se puede ver como un árbol de términos, recorrible a través de caminos.

Existen numerosas propuestas de lenguajes de consultas: *XML-QL* (1998-1999), *XQL* (1998), *Quilt* (2000). Al igual que en SQL, la fuente y el resultado XML de las consultas son compatibles, permitiendo el encadenamiento (subconsultas).

Recientemente, el *W3C XML Query Working Group* ha publicado la primera versión del borrador del lenguaje *XQuery*, destinado a ser el lenguaje de consulta estándar sobre XML y que recoge muchas de las características de sus predecesores. Existen numerosas propuestas relacionadas con el XML y las bases de datos:

- El *ISO/IEC JTC1/WG4 N1946* desarrolla estándares para representar metainformación, accederla y modificarla en sistemas de información basados en documentos SGML.

- El *ISO/IEC JTC1/SC32* están definiendo maneras de almacenar documentos estructurados en bases de datos relacionales, como parte del *SQL/MM*.

- *SQL/XML* contiene las especificaciones para representar datos relacionales SQL (específicamente filas y tablas de filas, vistas y resultado de consultas) en formato XML, y viceversa. También se están desarrollando especificaciones para representar esquemas SQL en XML, como acciones (*Insert, Update, Delete*) y la especificación de protocolos relacionados con el transporte de XML cuando se utiliza con SQL.

3.1.1 Demandas de almacenamiento XML

XML se está convirtiendo en el formato de datos de elección para una amplia variedad de soluciones de sistemas de información. Los beneficios que a menudo se asocian con la utilización de XML incluyen independencia de la plataforma, bajo costo de entrada, y la habilidad de compartir datos de manera transparente. XML puede además acomodar una variedad de datos incluyendo texto, imágenes y sonido. Sin embargo, nuevos requerimientos de habilidades y problemas administrativos causados por la creciente cantidad de datos XML deben también ser administrados.

La mayoría de las aplicaciones tradicionales de negocio y de las aplicaciones basadas en internet dependen de bases de datos. Existen diversos modelos diferentes de bases de datos, sin embargo, la mayoría son relacionales. Para mantener datos en una base de datos, éstos deben ser obtenidos y almacenados de una manera consistente, confiable y eficiente.

3.1.1.1 Requerimientos de almacenamiento XML

Los documentos y los requerimientos de almacenamiento de datos XML pueden ser pensados a menudo en dos categorías generales: **centrados en los datos** y **centrados en los documentos**.

- Las demandas centradas en los datos suelen incluir documentos de facturación típicos, órdenes de compra, y documentos menos estructurados, y es apropiada para ítems como contenidos de periódicos, artículos y publicidad.

Si el documento XML tiene una estructura bien definida y contiene datos actualizables usados en maneras diversas, el documento es típicamente centrado en los datos.

- Los datos centrados en el documento tienden a ser más impredecibles en tamaño y contenido que los centrados en los datos los cuales son altamente estructurados, con tipos de datos de tamaño limitado y reglas menos flexibles para campos opcionales y contenido.

Los sistemas de almacenamiento XML deben acomodarse eficientemente con ambos tipos de requerimientos de datos, dado que XML está siendo ampliamente usado en sistemas que administran ambos tipos de datos. La mayoría de los productos se enfocan en servir uno de esos formatos de datos mejor que el otro.

Las bases de datos relacionales tradicionales son típicamente mejores al tratar con requerimientos centrados en los datos, mientras que los sistemas de administración de contenido y de documentos son típicamente mejores para almacenar datos centrados en el documento.

Los sistemas de bases de datos deben ser capaces de exponer los datos relacionales como XML y almacenar el XML recibido como datos relacionales para transferir, obtener y almacenar transparentemente los datos requeridos por la aplicación.

3.1.1.2 Estrategias para el almacenamiento de datos XML

Con el objeto de mover datos entre un documento XML y una base de datos y viceversa, los datos en la base de datos deben ser mapeados a la estructura del documento XML. El documento XML completo es mapeado a una columna única en la base de datos.

Estrategias más complejas incluyen el mapeo de cada elemento a una columna correspondiente en la base de datos o el mapeo de la estructura del documento XML a la base de datos. Las estrategias existentes pueden ser descompuestas en tres métodos básicos:

- **Almacenar el documento completo en forma de texto, como un gran objeto binario (*BLOB*) en una base de datos relacional.** Esta es una buena estrategia si el documento XML contiene contenido estático que sólo será modificado cuando el documento completo es reemplazado. Esta estrategia es común para datos centrados en el documento conforme estos datos son obtenidos y almacenados típicamente como una unidad. Almacenar el documento completo en formato de texto es fácil de implementar dado que no se necesita mapeo o traducción, pero puede limitar también la búsqueda, indexamiento y granularidad de la obtención de documentos XML. El documento XML se mantiene como era antes de ser almacenado, lo que minimiza el trabajo de rearmarlo luego de su obtención.

- **Almacenar una forma modificada del documento completo en el sistema de archivos.** Este método es popular cuando el número de documentos es pequeño y los documentos XML son infrecuentemente actualizados y transferidos entre sistemas de archivos.

Almacenar una forma modificada del documento completo en el sistema de archivos es bastante limitado dado que el sistema de archivos no hace una base de datos muy buena. Este método funciona para un número menor de documentos XML y típicamente puede ser incluido solamente en el diseño dado que el documento XML se está moviendo por el sistema de archivos.

- **Mapear la estructura de los datos en el documento en la base de datos.** Mapear un documento XML que contiene tablas u objetos. Si la estructura del documento XML no es compatible con la estructura de la base de datos, el documento debe ser transformado para ajustarlo a la estructura de la base de datos antes de almacenarlo. Mapear la estructura de los datos en el documento a la base de datos es una opción muy popular y es el foco de muchas características habilitadoras de XML.

3.1.1.3 El proceso de mapeo y traducción

Para modelar una estructura de base de datos como XML, la estructura XML debe ser capaz de acomodar algunos conceptos básicos incluyendo claves primarias, claves secundarias, tablas, y columnas. Este proceso puede irse a los extremos en dos direcciones:

- En una dirección se puede hacer una tentativa para mapear la base de datos relacional al esquema estándar de la industria, lo que implica típicamente una cantidad significativa de programación personalizada para traducir el esquema relacional al esquema XML estándar de la industria.

- La base de datos relacional existente se podría descargar en una estructura XML por defecto con los elementos que representan las columnas y los atributos que representan valores de columna.

El proceso de mapeo y traducción puede ser descompuesto en pasos básicos:

1. Crear el esquema XML con un elemento para cada tabla y los atributos correspondientes para cada columna no clave que será mapeada. Las columnas que no permiten valores nulos pueden ser marcadas como requeridas mientras aquellas que permiten valores nulos pueden ser marcadas como opcionales en el esquema XML. Las columnas pueden ser también anidadas como elementos, pero pueden surgir problemas cuando el mismo nombre de columna es usado en más de una tabla. De esta manera el enfoque más simple es mapear las columnas como atributos XML donde las colisiones de nombre en el esquema XML no son un problema.

2. Crear las claves primarias en el esquema XML. Un enfoque podría ser agregar un atributo para la columna clave primaria con un *ID* agregado al nombre de la columna. Este atributo podría necesitar ser definido en el esquema XML como de tipo *ID*. Pueden surgir problemas de colisión al crear claves primarias en el esquema XML. A diferencia de las bases de datos relacionales donde las claves primarias necesitan ser únicas sólo dentro de una tabla, un atributo *ID* dentro de un documento XML debe ser único a través de todo el documento.

Una manera es agregar el nombre del elemento (nombre de la tabla) al valor de la clave primaria (valor del atributo). Esto asegura que el valor es único a través del documento XML.

3. Modelar las relaciones de clave secundaria. Esto se puede lograr mediante el anidamiento de elementos bajo el elemento padre. un *ID* de esquema XML puede ser usado para apuntar a una estructura XML correspondiente conteniendo un *IDREF*.

Tabla II. Resumen de relación esquema XML y esquema relacional

Documento XML (esquema)	Esquema relacional
Elemento	Tabla
Atributo o elemento anidado	Columna
Atributo *ID*	Clave Primaria
IDREF o elemento anidado	Clave Secundaria
#REQUIRED, #IMPLIED	*NULL, NOT NULL*

Se pueden usar muchas variaciones de esquemas XML para representar la misma base de datos relacional. Al entender las capacidades y técnicas disponibles cuando se construye el esquema XML, tanto el tamaño del documento como el tiempo de procesamiento pueden ser minimizados.

Este entendimiento requiere conocimiento detallado de las capacidades de los esquemas XML y además un entendimiento de cómo las API's XML procesarán realmente el documento XML. En la sección 3.4 (API's XML para bases de datos), se explicará más a detalle las API's XML para bases de datos.

3.2 Definición de bases de datos XML nativas

Durante los últimos años, XML se ha convertido en una de las más importantes tecnologías para la gestión de infraestructuras basadas en el *WEB*. La aceptación de esta tecnología se debe principalmente a dos de sus características: **su capacidad como mecanismo de integración** y **la separación existente entre contenidos y presentación**.

La solución para el almacenamiento de datos en formato XML es la de disponer de sistemas de gestión de información que sean capaces de gestionar datos en su formato nativo. El almacenamiento de datos en formato XML nativo tiene diversas ventajas sobre la gestión realizada por las bases de datos relacionales convencionales.

- No es necesaria ninguna conversión de formatos y la estructura de los documentos se mantiene intacta.

- Los sistemas relacionales típicos no están dotados para la gestión jerárquica de datos que tiene XML, por lo que necesitan mecanismos de conversión a medida para el almacenamiento de información en este formato.

- Tampoco disponen de mecanismos efectivos para contemplar el dinamismo de los documentos XML cuyo formato puede variar con facilidad.

Los lenguajes de consulta típicos como SQL tampoco están preparados para la búsqueda de información en sistemas XML. Su diseño inicial era para la realización de consultas en tablas y estaban sujetos a un esquema prefijado. Son necesarios nuevos lenguajes de consulta que permitan la identificación de información en repositorios XML. Estos lenguajes pueden aprovechar la potencia de los recursos *XLink* y *XPath* pertenecientes al estándar XML.

Estos recursos deberán complementarse con sistemas de indexado eficaces que optimicen el acceso a los recursos.

Una de las soluciones más eficaces para almacenar datos en formato XML es la de bases de datos XML nativas. Estas bases de datos proporcionan la funcionalidad necesaria para almacenar datos en formato XML y realizar consultas en la información gestionada. Están especializadas en almacenar datos en formato XML manteniendo el modelo XML asociado intacto.

Una base de datos XML nativa define un modelo lógico para cada documento XML y almacena y recupera información ajustada a ese modelo lógico. Estos modelos lógicos deben incluir elementos, atributos, y deben contemplar la ordenación de documentos. Las bases de datos XML nativas tienen un documento XML como unidad fundamental de almacenamiento, al igual que una base de datos relacional tiene un registro que pertenece a una tabla determinada.

Las bases de datos XML nativas no son bases de datos verdaderamente XML, almacenan información en formato XML. Son sólo un modelo lógico. Una base de datos XML nativa cumple con definir el modelo (lógico) de un documento XML y almacena y recupera los documentos según ese modelo.

El modelo debe incluir elementos, atributos, *PCDATA*, y un orden para el documento. No se requiere que tenga una estructura asociada para el almacenamiento físico de los datos. Puede ser construida sobre una base de datos relacional, jerárquica u orientada a objetos, o bien utilizar formatos de almacenamiento propietarios como archivos indexados o comprimidos.

Las bases de datos XML nativas no representan un nuevo modelo de base de datos de bajo nivel ni pretenden reemplazar a los sistemas de almacenamiento actuales. Se ha planteado como una herramienta de soporte al desarrollo que permitan a los desarrolladores disponer de una solución robusta para el almacenamiento y manipulación de datos en formato XML.

Las bases de datos XML nativas almacenan los documentos creando modelos lógicos. Estos modelos incluyen diferentes niveles de agregación y complejidad, así como un soporte completo para la gestión de datos semi-estructurados.

Los modelos son automáticamente mapeados por las bases de datos XML nativas al mecanismo de almacenamiento correspondiente. El mapeo utilizado por estas bases de datos garantizará la correcta utilización del modelo asociado al documento original. Una vez que se ha almacenado la información, y con el objeto de aprovechar al máximo la capacidad de estos sistemas, es necesario seguir utilizando herramientas XML que permitan realizar otros tipos de operaciones.

La principal **ventaja de las bases de datos XML nativas es la abstracción**. El programador no tiene por qué conocer los detalles de cómo se almacena la información, pudiendo construir sus aplicaciones con total libertad.

3.2.1 Almacenamiento XML nativo

Las bases de datos XML nativas se especializan en almacenar datos XML y almacena intactos todos los componentes del modelo XML. No viene a ser un medio nuevo de almacenaje de información a nivel físico ni viene a remplazar las bases de datos actuales. Solo es una herramienta para asistir al desarrollador proveyéndole de un almacenaje robusto y manipulación adecuada de documentos de XML.

Es necesario resaltar que todas las bases de datos XML nativas difieren en sus características dependiendo de la implementación y del proveedor del producto.

Las bases de datos XML nativas gestionan los documentos como si fueran colecciones de datos, permitiendo realizar consultas y manipulaciones sobre esos documentos en forma de conjuntos de información. Es un concepto similar al de las tablas en las bases de datos relacionales. Se puede almacenar cualquier documento XML en la colección independientemente de su estructura, permitiendo la realización de consultas en todo el conjunto de datos con una gran flexibilidad. Las bases de datos XML nativas que soportan esta característica se denominan **independientes del esquema**.

Disponer de colecciones de documentos independientes del esquema proporciona a la base de datos una alta flexibilidad y facilita el desarrollo de aplicaciones. Pero esta característica dificulta la tarea de los administradores de base de datos debido a posibles problemas de integridad de datos.

3.2.2 Aplicaciones de XML nativo

3.2.2.1 *Queries* en bases de datos XML nativas

XPath es el lenguaje de consulta estándar para bases de datos XML nativas. Para facilitar la realización de consultas en bases de datos XML nativas, se ha modificado el estándar inicial para dotar a *XPath* de capacidades de consulta en colecciones de documentos. Lamentablemente, *XPath* no se diseñó inicialmente para la realización de consultas y las modificaciones realizadas hasta ahora aportan posibilidades muy limitadas.

Algunas de las limitaciones de consulta más evidentes de *XPath*:

- La falta de mecanismos de agrupación, ordenación y enlazado de información entre documentos (*join*).
- El soporte para la gestión de diferentes tipos de datos.

Debido a estas limitaciones, *XPath* necesita ser complementado para disponer de capacidades realmente interesantes de consulta. Algunas de las limitaciones pueden ser resueltas mediante la utilización de plantillas XSL y XSLT, pero de una forma realmente artificiosa. Por tal motivo, se está desarrollando un lenguaje de consulta más orientado a bases de datos XML llamado *XQuery*.

Para mejorar el funcionamiento de consultas, las bases de datos XML nativas soportan la creación de índices para los datos almacenados en colecciones. Estos índices se pueden utilizar para mejorar la velocidad de ejecución de la consulta. Los detalles de qué poner en el índice y cómo se crean variarán entre los productos.

3.2.2.2 Modificaciones en bases de datos XML nativas

Las actualizaciones (*Update*) son un área de verdadera debilidad para la mayoría actual de los productos para bases de datos XML nativas. Requieren recuperar un documento, cambiarlo con su *API* para XML, y devolverlo a la base de datos.

Algunos productos tienen lenguajes propietarios para actualización que permiten que se realice actualizaciones dentro del servidor, y un par de bases de datos XML nativas de fuente abierta apoyan el uso de *XML:DB XUpdate* para el mismo propósito. Es probable que estas sean áreas problemáticas hasta que *XQuery* proporcione un lenguaje de actualización.

3.2.2.3 Áreas de aplicación

Solamente hay un requisito indispensable para una aplicación que desee utilizar una base de datos XML nativa, la aplicación debe utilizar XML. Más allá de esto, no hay reglas terminantes para qué tipo de usos se debe o no construir una base de datos XML nativa.

Algunas áreas de aplicación potenciales incluyen:

- Portales de información corporativa.
- Datos de catálogo.
- Bases de datos de manufactura de partes.
- Almacenaje de información médica.
- Registros de transacción *B2B*.
- Bases de datos de personalización.

Más allá de esto, las bases de datos XML nativas son simplemente una herramienta nueva y su utilidad será determinada por la creatividad de los desarrolladores que las implementen.

3.3 Generación de documentos XML desde bases de datos relacionales

Como ejemplo se definirá una forma de generar documentos XML desde bases de datos relacionales. Usando la definición DTD de *Microsoft* para fuentes relacionales se forma el componente *ADO*, esta forma es una solución factible para intercambiar información XML entre aplicaciones.

Para esta solución se usará *Microsoft Access, ADO 2.5*, y el uso *Parser Microsoft.XMLDOM* del *Internet Explorer* en el servidor, en esencia se procederá a hacer una consulta a una base de datos.

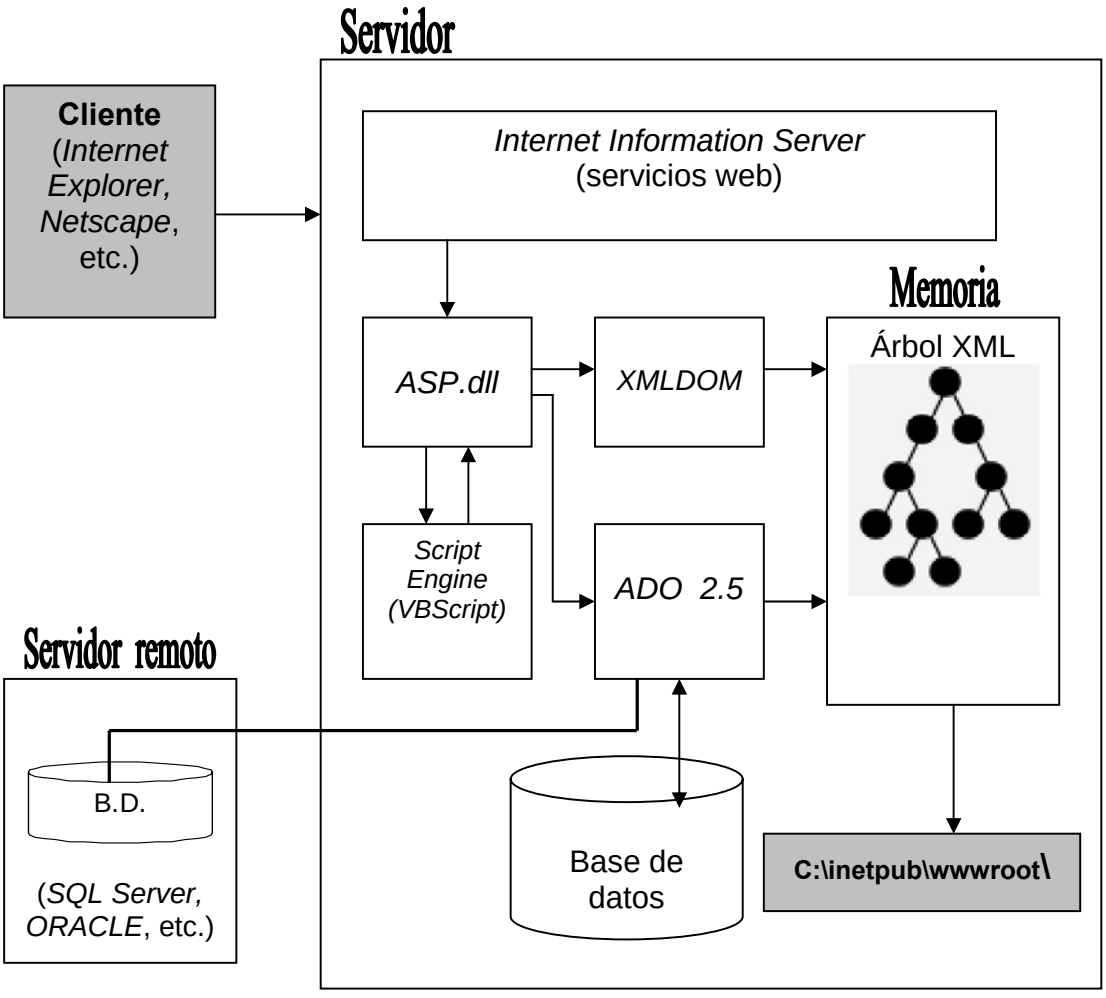
Esta solución se da para ambientes *Windows*, que corren *Internet Information Server 2.0* (*IIS 2.0* ó más), que tengan instalado *ADO 2.5*, el *Parser XMLDOM* del *Internet Explorer 5* instalado en el servidor, y para este ejemplo se utiliza la base de datos *Access 2000*.

3.3.1 Arquitectura de la aplicación

Para hacer la consulta a la base de datos se extraerá un conjunto de registros usando un *Recordset*. Posteriormente se construirá basándose en el *Recordset*, los nodos de un árbol, a manera de crear la estructura XML en Memoria. Cuando se tenga la estructura del documento XML, se procederá a guardar desde memoria a disco duro de la computadora, a manera de crear un archivo con la extensión *xml*.

De esta forma se puede actualizar dinámicamente información XML que se utilice en aplicaciones *WEB* que son accesibles desde cualquier dispositivo conectado a internet, o que interactué con la aplicación que se posea.

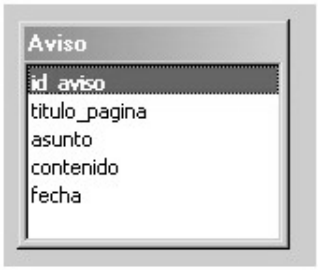
Figura 5. Arquitectura de la aplicación



3.3.2 Estructura de la base de datos

Para este ejemplo de base de datos emplearé una estructura simple, sólo se utilizará la tabla **Aviso**, cuya estructura es la siguiente:

Figura 6. Estructura de tabla de base de datos



Donde los campos:

El campo “id_aviso” es llave primaria, que es de tipo numérico, “titulo_pagina”, “asunto”, “fecha” son de tipo texto, y el campo “contenido” es de tipo memo.

3.3.3 Ejemplificación de la programación de la aplicación

Se mostrará y explicará fragmentos del código que permite desarrollar la solución.

Se define el lenguaje *VBScript* que se utilizará en la codificación.

<%@ Language=VBScript %>

Se declaran todas las variables a utilizar en la aplicación.

```
<%
'-------------------------------------------------------------------------------------------------
Response.ContentType = "text/xml"
'------------------------------ DECLARACIÓN DE VARIABLES ---------------------
Dim dbConeccion          //para crear la conexión.
Dim TMPdir               //obtiene la dirección temporalmente del servidor.
Dim DSNtemporal          //guarda el Driver de la base de datos.
Dim objFaq               //objeto etiqueta.
Dim objPregunta          //pregunta si crea un nuevo elemento .
Dim objRespuesta         //respuesta a la creación de un elemento.
Dim oParser              //crea el documento Microsoft.XMLDOM
Dim rsAyudas             //para acceder al Recordset.
Dim sConsulta            //guarda la consulta a la base de datos.
Dim objFaqs              // guarda la etiqueta raíz del documento XML.
Dim otitulo_pagina       //guarda el titulo de la pagina.
Dim objPI                //objeto que guarda instrucciones a procesar.
```

Se realiza la conexión con la base de datos, usando el objeto *Server* y el método *CreateObject* de ASP para crear la instancia al componente *ADO*.

```
'----------------------CONEXION A LA BASE DE DATOS-------------------------------
Set dbConeccion = Server.CreateObject("ADODB.Connection")
TMPdir = Server.MapPath("db.mdb")
DSNtemporal = "DRIVER={Microsoft Access Driver(*.mdb)};"
DSNtemporal = DSNtemporal & "DBQ=" & tmpdir
dbConeccion.Open DSNtemporal
```

Se crea la instancia al componente XMLDOM de Microsoft.

```
'----------------------- SE CREA LA INSTANCIA A  XMLDOM----------------------------
Set oParser = Server.CreateObject("Microsoft.XMLDOM")
```

Se crea la instancia al objeto *Recordset* y se obtiene el *Recordset* de la base de datos, utilizando para ello la conexión a la base de datos y una instrucción SQL.

```
'------------ SE CREA LA INSTANCIA AL RECORDSET Y SE ACCESA---------------
Set rsAyudas = Server.CreateObject("ADODB.Recordset")
sConsulta = "SELECT * FROM Aviso"
rsAyudas.Open sConsulta, dbconeccion, 0, 3
```

Se crea la etiqueta raíz del documento XML. (faqs)

```
'----------------- SE CREAN LAS ETIQUETAS DEL XML - ETIQUETA-----------------
Set objFaqs = oParser.createElement("faqs")
```

Se crean otras etiquetas de nuestro árbol XML.

```
'------SE CREA LA ETIQUETA: titulo_pagina Y SE ANEXA AL NODO: faqs---
Set otitulo_pagina = oParser.createElement("titulo_pagina")
otitulo_pagina.Text = rsAyudas("titulo_pagina")
objFaqs.appendChild(otitulo_pagina)
```

Se recorre el Recordset para ir llenando los nodos del árbol XML.

```
'------------------- CICLO REPETITIVO EN EL RECORDSET: rsAyudas---------------
Do While Not rsAyudas.EOF
'----------------- SE CREA LA ETIQUETA: faq Y SU ATRIBUTO: id_aviso------------
        Set objFaq = oParser.createElement("faq")
        objFaq.setAttribute "id_aviso", rsAyudas("id_aviso")
```

```
'--------- SE CREA LA ETIQUETA: pregunta Y SE ANEXA  AL NODO: faq---------
        Set objPregunta = oParser.createElement("pregunta")
        objPregunta.Text = rsAyudas("pregunta")
        objFaq.appendChild objPregunta
'-------- SE CREA LA ETIQUETA: respuesta Y SE ANEXA  AL NODO: faq---------
        Set objRespuesta = oParser.createElement("respuesta")
        objRespuesta.Text = rsAyudas("respuesta")
        objFaq.appendChild objRespuesta
'-------------------- SE ANEXA LA ETIQUETA: faq AL NODO: faqs--------------------
        objFaqs.appendChild(objFaq)
'----------------------- SE MUEVE AL PROXIMO REGISTRO-----------------------------
‘--------------------- SE ELIMINA VALOR A LA VARIABLE:  objAyuda------------------
        rsAyudas.MoveNext
        Set objFaq = Nothing
Loop

Se cierra la etiqueta raíz de manera de que el árbol XML sea valido.
'-------------------------------SE CIERRA LA ETIQUETA: faqs-----------------------------
Set oParser.documentElement = objFaqs

Se agrega la instrucción de preprocesamiento al árbol XML
'------------- SE AGREGA LA INSTRUCCION DE PROCESAMIENTO--------------
Set objPI = oParser.createProcessingInstruction("xml", "versión""1.0"" encoding=""iso-8859-1""")
oParser.insertBefore objPI, oParser.childNodes(0)
Se estructura el árbol en memoria, y se lo despliega en el monitor.
```

'---------- SE COLOCA EL ARBOL COMPLETO EN MEMORIA----------
Response.Write oParser.xml

Se utiliza entonces el método save (guardar) del XMLDOM para guardar en el disco duro.
'---------- SE GUARDA EL ARBOL DEL XMLDOM AL DISCO LOCAL----------
oParser.save(Server.MapPath("ayudas.xml"))

Se cierran todos los objetos instanciados.
'---------- SE CIERRAN LOS OBJETOS INSTANCIADOS----------
rsAyudas.Close
dbconeccion.Close
oParse.Close

Se destruyen todos los objetos creados en memoria.
'---------- SE ELIMINAN DE MEMORIA LOS OBJETOS----------
Set dbConeccion = Nothing
Set objFaq = Nothing
Set objPregunta = Nothing
Set objRespuesta = Nothing
Set oParser = Nothing
Set rsFaq = Nothing
Set objFaqs = Nothing
Set otitulo_pagina = Nothing
Set objPI = Nothing
%>

3.4 API's XML para bases de datos

Un API (*Application Programming Interface*), se utiliza para la conexión con bases de datos relacionales en el cual emplea lenguaje SQL para la manipulación de la información.

Para procesar documentos XML, la mayoría de las herramientas XML trabajan con los API's SAX o DOM, permitiendo a las herramientas de XML tratar con bases de datos como si fueran documentos XML, de esta forma, se obvia la necesidad de convertir una base de datos.

3.4.1 *Driver JDBC*

Un *JDBC* está escrito y compilado de acuerdo a las especificaciones de *JAVA* y puede ejecutarse en cualquier plataforma sin necesidad de recompilar. Esta arquitectura está basada en *ODBC*. *JDBC* permite conectarse a distintas bases de datos simplemente cambiando los parámetros iniciales.

3.4.2 API SAX

SAX es un API para XML basado en eventos. El analizador SAX reporta los eventos como el inicio o final de los elementos de la aplicación y mientras pasan a través del documento. Como el analizador reporta los eventos mientras visita las diferentes partes del documento, no tiene que construir una estructura interna. Esto reduce los recursos de sistema, lo que hace a este analizador atractivo para grandes documentos.

Para implementar el API SAX para bases de datos, se necesita implementar un analizador que opere sobre una fuente de datos *JDBC*, iterar sobre cada fila y columna, y generar los eventos apropiados mientras iteramos.

3.4.3 API DOM

El API DOM sigue una construcción estilo árbol. Los elementos tienen relaciones padre-hijo con otros elementos. Con este API, el analizador construye una estructura interna por la que una aplicación puede navegar. DOM permite que una aplicación tenga acceso aleatorio al documento estructurado como un árbol, y el coste es el incremento de la memoria utilizada.

Para construir un árbol DOM para una tabla de base de datos, se puede iterar sobre las filas y columnas y construir nodos para un árbol mientras se visita.

3.5 Descripción de rendimiento de manejadores de bases de datos XML

3.5.1 *Oracle9i*

Oracle9i permite la creación de documentos XML, tanto en programas cliente como en programas servidor, incluye un nuevo tipo de datos llamado *XMLType*. Este tipo posee unas funciones que se pueden utilizar para acceder, extraer y consultar datos XML usando expresiones *XPath*. El *XPath* es una recomendación del *W3C* que especifica el modelo de datos y la gramática para navegar por un documento XML. Es posible generar documentos XML a partir de los resultados de una consulta, o si se usa *XSU Java* o *PL/SQL API*, como parte de una misma consulta.

Algunos servicios muy interesantes que ofrece la nueva versión de *Oracle9i* son:

- Soporte para *JavaBeans, servlets* y *JavaServer*.
- Soporte de conexiones seguras SSL sobre una conexión de *WEB* (*HTTP*).
- Capacidad de encriptamiento de la información.
- Utilería XML SQL mejorada.
- Conjunto de herramientas para desarrollo de XML.

Oracle9i incluye otra serie de componentes para permitir manipular documentos XML. Algunos de estos componentes son:

- Componentes para el soporte de bases de datos:
  - *XMLType*: Es un nuevo tipo de datos para almacenar, consultar y recuperar documentos XML.
  - *SYS_XMLGEN*: Es una función SQL para crear documentos XML.

- *SYS_XMLAGG*: Es una función SQL para agregar múltiples documentos XML.
- *DBMS_XMLGEN*: Es un paquete para crear documentos XML a partir de consultas SQL.
- *URI support*: Almacena y recupera referencias globales y internas.
- *Text support*: Soporta el *XPath* en el *XMLType* y columnas de texto.

- XML *Developer's Kit* (*XDK*) para *Java*
- *XDK* para *Java Beans*
- *XDK* para C
- *XDK* para C++
- *XDK* para *PL/SQL*

Oracle9i esta perfectamente preparado, para almacenar buscar y recuperar documentos XML, de las siguientes formas:

- **Como documentos XML descompuestos:** Cuando los documentos se guardan en fragmentos. Los datos XML se guardan en forma de objetos relacionales, luego se utiliza el *XML SQL Utility (XSU)*, u otra de las funciones SQL o paquetes disponibles, para generar el documento XML a partir de las instancias a dichos objetos relacionales.

- **Como documentos XML compuestos o completos:** Cuando se guardan los datos XML en formato *XMLType*, se usan funciones como *Extract()* y *ExistsNode()*, o indexación *Oracle Text*, para buscar los documentos.

3.5.2 *SQL Server 2000* de *Microsoft*

SQL Server 2000 es la primera versión de *SQL Server* que soporta XML, pero las primeras entregas de tecnología XML desarrolladas por *Microsoft* ya podían ejecutarse bajo las versiones 7.0 y 6.5, aunque estos últimos podrían mermar considerablemente el rendimiento si se están manipulando grandes tablas de resultados con estructuras complejas.

SQL Server 2000 lo que desea conseguir con XML es transformar datos relacionales estándares a formatos XML.

SQL Server 2000 soporta un nuevo tipo de datos que permite almacenar XML en las bases de datos relacionales, de forma nativa. Disponer de un tipo de datos nativo XML es uno de los criterios que hace de *SQL Server 2000* un producto compatible con XML.

Además, *SQL Server 2000* también introduce:

- URL (*Universal Resource Locator* o localizador de recursos universal) que permiten acceder a *SQL Server* a través del protocolo *HTTP*.

- La cláusula *"FOR XML"* dentro de la sentencia *"SELECT"*, pudiendo así recuperar resultados en formato XML.

- Procedimientos almacenados del sistema para la manipulación de datos XML.

- Operaciones por lotes basadas en XML, también llamadas *"Grams* de actualización*"*, para operaciones por lotes y extensiones proveedoras de *SQL Server OLE DB* para XML.

3.5.2.1 Acceso a través de *HTTP*

SQL Server 2000 permite introducir una sentencia SQL en el campo dirección de IE 5.0 (*Internet Explorer 5.0* de *Microsoft*) y recuperar registros en formato XML:

http://<SERVIDOR WEB>/
<DirectorioVirtual>
?sql=<CONSULTA SQL CODIFICADA>+FOR+XML+RAW

La consulta utiliza el protocolo *HTTP*, lo cual permite utilizar cualquier tipo de software cliente existente o que se puede crear. La consulta dirige la petición al servidor *WEB*, IIS (*Internet Information Server*, de *Microsoft*), a través de una raíz virtual ubicada en el servidor *IIS* y que debe configurarse previamente para que pueda utilizar las extensiones XML de *SQL Server*.

3.5.2.2 Cláusula *"FOR XML"*

En *SQL Server 2000*, el empleo de la nueva cláusula *"FOR XML"* dentro de una sentencia *"SELECT"* genera documentos XML en lugar de un conjunto de filas. Esta nueva cláusula puede utilizarse tanto en consultas como en procedimientos almacenados. Los argumentos admitidos por *"FOR XML"* son *"XML mode, SchemaOptions y ELEMENTS"*.

FOR | [XML mode [, SchemaOption] [, ELEMENTS]]

3.5.2.3 Procedimientos almacenados del sistema

SQL Server 2000 introduce seis nuevos procedimientos almacenados del sistema para manipular los datos XML: *sp_xml_preparedocument, sp_xml_fetchdocument, sp_xml_removedocument, sp_xml_insertfromxml, xp_xml_removexml y sp_xml_fetchintoxml*.

Los procedimientos almacenados para la manipulación de datos XML sólo están disponibles por cada conexión que se establece. Esto no los hace especialmente útiles en aplicaciones *n-tier* (multicapas), que reservan conexiones precisamente para asegurarse la escalabilidad. Además, los procedimientos ofrecen otro tipo de funcionalidad, ya que permiten almacenar (*preparedocument*) y recuperar documentos mediante un puntero (*fetchdocument*).

3.5.2.4 *Grams* de actualización

Los *Grams* de actualización son operaciones por lotes, basadas en XML, destinadas a insertar, eliminar y actualizar con el siguiente formato general:

```
<sql:sync xmlns:sql="urn:schemas-microsoft-com:xml-sql">
    <sql:before>
        <NOMBRE_TABLA [sql:id="valor"] col="valor" col="valor" ../>
    </sql:before>
     <sql:after>
        <NOMBRE_TABLA [sql:id="valor"] [sql:at-identity="valor"]
col="valor col="valor" ../>
    </sql:after>
</sql:sync>
```

3.5.3 *DB2* de *IBM*

DB2 de *IBM* ofrece soporte XML mediante su producto *DB2 XML Extender*. *XML Extender* provee nuevas funciones que permiten el almacenamiento y manipulación de documentos XML. Documentos XML enteros pueden ser almacenados en bases de datos *DB2* como datos carácter, como una única columna o como archivos externos, y aún pueden ser administrados por *DB2*.

IBM anuncia la disponibilidad en español de la versión 7 de su gestor de bases de datos *DB2*, *Universal Database (UDB)*, donde incorpora nuevos módulos y herramientas que complementan las prestaciones *Business Intelligence* y *e-business*.

DB2 Extender define un nuevo tipo de datos y funciones para textos que utilicen el soporte integrado en *DB2*. También se beneficia del soporte *DB2* para objetos de hasta un tamaño de 2 *gigabytes*, y utiliza elementos que comprueban la integridad de los datos de texto en las tablas.

3.5.3.1 *DB2 XML Extender*

DB2 XML Extender especialmente diseñado para ser utilizada en entornos de comercio electrónico empresa-consumidor (*B2C*), esta nueva capacidad incluida en la versión 7 de *DB2 UDB*, posibilita el almacenamiento de datos en formato XML como una nueva columna de tipos de datos y generar posteriormente documentos XML para ser enviados al cliente final a través de un buscador de internet.

Se proveen funciones para obtener el documento XML completo o elementos individuales o atributos del documento XML. El producto XML Extender sirve además como repositorio para la administración de DTD. *DB2* almacena un documento XML como una única columna o mapea el documento XML a múltiples tablas y columnas.

El producto *XML Extender* está diseñado para proveer capacidades de búsqueda rápida con páginas (archivos) XML. Esto es útil para aplicaciones que comparten archivos XML o aplicaciones que buscan contra archivos XML como motores de búsqueda. El indexamiento es realizado usando definición de acceso a datos *(Data Access Definition, DAD)* para definir los elementos y atributos XML que deben ser indexados. Funciones definidas por el usuario *(User-defined functions, UDFs)* pueden ser usadas para insertar, seleccionar o actualizar un documento completo o elementos y atributos dentro de un documento.

3.6 Acceder y generar código XML con datos en *SQL Server 2000*

Se empieza por abrir una conexión a la base de datos *"*empresa*"* de *SQL Server*. Se crea un objeto *SqlCommand* que seleccionará todas las filas de la tabla *"*cliente*"* de la base de datos. En este comando, se utiliza la cláusula *FOR XML* para indicar que se solicita que *SQL Server* devuelva los resultados de la consulta en forma de documento XML. El indicador *XMLDATA* de la cláusula *FOR XML* especifica que se debe devolver un esquema de datos XML. El esquema se agrega al principio del documento como un esquema en línea.

‘----------CONEXIÓN A LA BASE DE DATOS “EMPRESA”----------
Dim sConnection as String = "server=(local)\\VSdotNET;
Trusted_Connection=yes; database=empresa"

‘----------OBJETO SQLCOMMAND QUE SELECCIONARÁ TODAS ----------
‘----------LAS FILAS DE LA TABLA “CLIENTE”----------
Dim mySqlConnection as SqlConnection = new SqlConnection(sConnection)
*Dim mySqlCommand as SqlCommand = new SqlCommand("select * from*
cliente FOR XML AUTO, XMLDATA", mySqlConnection)
mySqlCommand.CommandTimeout = 15

mySqlConnection.Open()

3.7 Análisis comparativo

Los tres vendedores líderes (*Orale RDBMS, SQL Server 2000* de *Microsoft* y *DB2* de *IBM*) ofrecen similares capacidades para el manejo de XML. *Oracle9i* ofrece una rica interfaz de programación para datos XML usando las *clases, servlets* y *utilidades Java*. *SQL Server 2000 de Microsoft* ofrece flexibilidades para la obtención de estructuras de datos XML a través de varios métodos. Los tres soportan las estrategias de almacenamiento más comunes aunque las capacidades objeto-relacionales de *Oracle9i* pueden proveer mayor flexibilidad en esta área y prevenir parte de la traducción XML. El producto *DB2 de IBM* parece estar un paso atrás de la competencia con su soporte XML; aunque es posible soportar las tres estrategias comunes de almacenamiento, este soporte de la herramienta para las estrategias no parece ser tan rico.

Tabla III. Esquema comparativo de capacidad

Capabilidad	***SQL***	***XML***	***Java***	***Oracle***
Procesamiento transaccional en línea (OLTP)	X			X
Análisis	X			X
Datos relacionales (estructura)	X			X
Documentación sin estructura		X		X
Mensajes semi-estructurados		X		X
Integración y transformación		X		X
Programación orientada a objetos de la lógica del negocio			X	X

Se podría decir que en cierta medida *SQL Server 2000* de *Microsoft* pretende ser el servidor de bases de datos genérico para *Windows*, porque promete integración con todos sus productos. La escalabilidad es total. Puede funcionar en un Servidor *NT* multiprocesador de elevados recursos, como también puede funcionar en una portátil con *Windows 95/98*. Las características de *SQL Server 2000* de *Microsoft* son impresionantes. Tiene soporte de transacciones OLPT, datos relacionales con soporte XML, posibilidad de ejecutar consultas en paralelo, así como homogéneas y distribuidas, bloqueos dinámicos a nivel de filas, optimizador de consultas, estadísticas automáticas, *Unicode* nativo, replicación avanzada, replicación dinámica de datos y otros. Pero lo mejor de todo es la sencillez de comprensión y navegación entre procesos así como lo intuitivo de la herramienta.

Orale9i atendiendo a las características de manejabilidad escalabilidad rendimiento y soporte entre plataformas y XML tiene una arquitectura de servidor con ejecución multihilo y rendimiento de multiprocesadores simétricos (SMP). Las bases de datos pueden crecer hasta límites que en la práctica pueden considerarse inalcanzables y gracias a un rediseño respecto a versiones anteriores se han eliminado ciertos cuellos de botella. En lo referente a la escalabilidad residen en las tablas particionadas que son aquellas que pueden dividirse en múltiples dispositivos de almacenamiento de acuerdo con valores preestablecidos, el multiplexado y *pooling* de conexiones. La gestión de conexiones se realiza mediante cierto *software* específico, el de red, que antes se llamaba *SQL NET*, ahora se denomina *Net*, que aparte de cambiar de nombre ha avanzado tanto en las características (interfase de programación, navegador de Internet con soporte *Java*, XML y soporte de red adicionales), rendimiento y en facilidad. Pero lo que incrementa drásticamente el rendimiento es el soporte de referencia a objetos que son rápidos punteros de tablas los cuales sustituyen a las letras y tradicionales unidades relacionales.

La ventaja de *Oracle9i* y *DB2 de IBM* es que es compatible con *SQL92,* que es el lenguaje SQL estandarizado por *ANSI* e *ISO*, donde estandariza muchos de los puntos que anteriormente se dejaba al criterio de los fabricantes.

Una razón muy importante que pone a *Oracle* superior al soporte en su base de datos es que se puede usar en múltiples plataformas, mientras que *SQL Server* de *Microsoft* fue creado para ser usado únicamente sobre plataformas *Microsoft*. *Oracle* esta disponible en múltiples plataformas, incluidas *Windows, UNIX, MVS, VM, HPMPEXL, Siemens ICL, Novell Netware, OS/2* y *Linux*.

Se critica el uso del modelo relacional para el almacenamiento XML. La crítica más común es la sobrecarga y la programación extra asociada con el mapeo y traducción de datos estructurados XML en datos relacionales. La estructura jerárquica de un documento XML es también una forma que no se acopla bien a las bases de datos relacionales. La estructura jerárquica de un documento XML es representada típicamente por *joins* en la base de datos relacional.

Numerosos niveles de anidamiento en los documentos XML pueden causar problemas significativos de rendimiento y diseño en bases de datos relacionales cuando los documentos XML anidados resultan en tablas y operaciones *join* adicionales que pueden rápidamente degradar el rendimiento.

Como se resalta, en la sección de "almacenamiento XML nativo", difieren las características de implementación del proveedor de producto. Existen bases de datos XML nativas que fueron diseñadas específicamente para almacenar documentos en su estructura original en la base de datos XML, dando como resultado tiempos de respuesta más rápidos. Esto en contraste con manejadores de bases de datos relacionales que están obligados a instalar capas de conversión para permitir el almacenamiento de documentos XML.

Antes de almacenar XML, los manejadores de bases de datos relacionales necesitan un modelo de datos definido de forma apropiada, adaptar datos XML a estos modelos es costoso y lento, y en algunos casos no son tan flexibles para expandir las estructuras de los documentos almacenados. Además estas bases de datos XML pueden escalar desde plataformas *Windows NT / Windows 2000* hasta plataformas *mainframe* en *Unix*.

Hay razones para usar los tipos de bases de datos relacionales y los productos de bases de datos existentes para almacenar XML, aún cuando no sea de forma nativa.

- Las bases de datos relacionales y orientadas a objetos son bien conocidas, mientras que las bases de datos XML nativas son nuevas.

- Como resultado de la familiaridad con las bases de datos relacionales y orientadas a objetos, los usuarios se inclinan a ellas especialmente por el rendimiento.

4. REPLICACIÓN EN BASES DE DATOS

4.1 ¿Qué es replicación de una base de datos?

La replicación de datos es utilizada para el mantenimiento automático de copias de datos en múltiples computadoras. Consiste en tener varias copias de algún tipo de dato, garantizando la consistencia entre dichas copias. Además ha de ser totalmente transparente al programador de aplicaciones y usuarios finales.

Debe proveer las siguientes funcionalidades:

- Escalabilidad. Replicar tanto volúmenes grandes de información como pequeños.
- Mapeo y fragmentación de datos. Las copias pueden ser idénticas o semánticamente equivalentes.
- Replicación variada. Replicación no solamente de datos, también de otros tipos de objetos.
- Replicación sincrónica y asincrónica.
- Mecanismo de descripción de datos y objetos replicados.
- Mecanismo de inicialización.
- Mecanismo de seguridad.

- Mecanismo de log.

- Mecanismo de recuperación.

4.1.1 Tipos de replicación

4.1.1.1 Según el sentido del viaje de los datos

- Unidireccional. Se tiene un nodo actualizable y otro que contiene una copia del principal pero es sólo de lectura. Los datos se modifican en el primero y se replican en el segundo.

- Bidireccional. Se tienen nodos actualizando los datos, de tal forma que la replicación se produce en ambas direcciones. La lectura y la escritura se realiza en cualquiera de los nodos.

4.1.1.2 Según la oportunidad en la que ocurre

- Replicación Sincrónica. Llamada también de consistencia fuerte. El grado de desactualización de los datos replicados respecto a los datos originales es cero. Latencia cero. Se utilizan transacciones llamadas "todo o nada". Una transacción no debe ser aceptada si todos los sitios intervinientes no están de acuerdo o disponibles.

- Replicación Asincrónica. Este tipo de tecnologías se basan en la captura de transacciones completadas en el sitio principal, para ser aplicadas luego en los sitios secundarios. Llamada también de consistencia débil, existe una desactualización entre las copias de los datos.

4.1.1.3 Realización en línea o fuera de línea

- Replicación de datos *On-line.* La replicación de datos *On-line* consiste en copiar datos de un nodo a otro mediante una conexión. Se emplea en casos en que el tiempo de recuperación debe ser corto. Los datos en el nodo alterno deben estar replicados en tiempo real en todos los dispositivos de almacenamiento.

- Replicación de datos *Off-line.* Involucra dos o más sitios de que almacenan su información en algún medio o lo envían de un nodo a otro vía mensajería. Se utiliza principalmente con aquellas aplicaciones en las cuales el tiempo de recuperación no es una prioridad para el negocio.

4.2 Razones para hacer una replicación

4.2.1 Conflictos

Estos suceden cuando una misma información es modificada en varios nodos dentro de un intervalo de tiempo que contiene al intervalo de replicación, y cuando la replicación es asíncrona.

Los tipos de conflictos que pueden ocurrir en un sistema de replicación son:

- Conflicto de pérdida de unicidad. Ocurre cuando la replicación de un registro viola una integridad (de llave primaria o unicidad).

- Conflicto de actualización. Ocurre cuando una fila está siendo actualizada por dos transacciones diferentes simultáneamente.

- Conflicto de eliminación. Ocurre cuando con una transacción elimina un registro mientras que otra transacción lo está actualizando o eliminando.

Existen métodos de resolución de conflictos como:

- Ignorar. Ignora el error aunque no se haya actualizado.
- Aplicar. Aplica los cambios basándose en la llave de la tabla involucrada.
- Suma y promedio. Utiliza una fórmula para obtener el valor a actualizar.
- Mayor. Inserta al final el valor que sea mayor.
- Menor. Inserta al final el valor que sea menor.
- Prioridad. Inserta al final los valores de las columnas de mayor prioridad.
- Función. Define la lógica de resolución de conflictos con un programa complejo.

4.2.2 Beneficios de una replicación

- El tiempo para escribir la réplica es demasiado pequeño, los datos son vigentes en cualquier momento.

- La replicación no consume recursos del *CPU* (Unidad de Procesamiento Central).

- El *hardware* es el encargado de sincronizar los datos si la conexión o el disco fallan.

- El modo de escritura es completamente parametrizable.

- La solución es independiente de la tecnología con la que están construidos los discos.

- Debido a que existen múltiples dispositivos de lectura pueden existir mejoras en el desempeño de lectura.

- La distancia entre los nodos se encuentra únicamente limitada por la tecnología de red.

- Cuando se replica la base de datos se pueden hacer y deshacer transacciones para alcanzar el nivel de actualidad de datos deseado en la réplica.

4.2.3 Dificultades de una replicación

- La distancia entre los sitios está limitada por el tipo de cable que conecta los arreglos de discos.

- No hay manera de mantener el orden lógico de la escritura de datos.

- Los datos se escriben en un bloque físico de datos, cualquier disrupción en la base de datos o error humano también se reproducen en el sitio remoto.

- Se necesita *hardware* adicional para armar el *cluster*.

- El desempeño del sistema está influenciado por diversos factores: sobrecarga de la unidad central de procesamiento, el número de transacciones, y el tiempo de utilización de la unidad de procesamiento.

- El ancho de banda que emplea en una replicación es considerable o la utilización de un canal de comunicaciones dedicado.

4.3 Replicación de bases de datos XML

Cuando replicamos la definición de un servicio a un sitio para ser ejecutado localmente, se puede replicar tanto los datos como las llamadas a los servicios, para que el servicio se invoque en los datos locales, con menor costo de transferencia de datos.

En los estándares del modelo de datos de XML, un documento dinámico XML dinámico puede interpretarse como un árbol etiquetado. **Los nodos del árbol representan elementos y atributos** de XML y l**as terminales representando componentes de la relación** entre los elementos del documento. Algunos elementos, llamados elementos de función, tienen un significado especial y representan llamadas a las funciones de Servicios *WEB*.

4.3.1 Replicación de información y servicios

Los nodos de un documento tienen identificadores (*ID*'s). Todos los hijos del mismo nodo con el mismo *ID* son consideradas réplicas de un único nodo. También, un nodo puede registrar más que un nodo padre, en estos casos, estos nodos son considerados réplicas de un mismo único nodo padre.

Una forma de replicación puede lograrse reproduciendo no sólo los fragmentos del documento, sino también los Servicios *WEB* (*WEB Services*).

Esto puede ser usado en declaraciones de Servicios *WEB* cuya definición sea dada en términos de consultas de *XQuery*. Cuando la información es usada por los Servicios *WEB* es replicada hacia otro sitio, el servicio también es replicado.

Fragmento de código colocado en el *Portal Sky*

```
<documento nombre = "SkyPortal">
  <estado><estado_nombre>Colorado</estado_nombre>
   <vacacional>
    <turístico ID = "AspResort"><nombre>Aspen</nombre>
     <cond_nieve ID = "AspSC"> good
       <clima = "UnisysWeather" clima_nombre = "CondicionNieve"
           frecuencia ="cada hora"
           validación = "ultima">
           <parámetros><turístico>Aspen</turístico></parámetros>
       </clima>
     </cond_nieve>
     <hoteles ID = "AspenHotel">
     </hoteles>
    </turístico>
   </vacacional>
  </estado>
</documento>
```

Los Servicios *WEB* de el *Portal Sky*

function OperaSkyturístico($estado)
implementation:XQuery
for $x in documento(”SkyPortal”)/estado[estado nombre=$estado]
/vacacional/turístico[cond_nieve/value()=”good”]
return $x

function HotelesInfo($estado, $turístico)
implementation:XQuery
for $x in documento(”SkyPortal”)/estado[estado nombre=$estado]
/vacacional/turístico[nombre=$turístico]/hoteles
return $x

Esto permite a los servicios ser ejecutado localmente y reducir el costo de comunicación, la decisión puede tomarse basado en el costo descrito como "**la replicación vrs. el acceso remoto**".

4.4 Principales manejadores con soporte de replicación a bases de datos XML

4.4.1 *Oracle9i*

La herramienta *Replication Management Tool* es parte de *Oracle Enterprise Manager* y proporciona una interfase gráfica al usuario (GUI, *Graphic User Interface*) para preparar, manejar, y supervisar el ambiente de replicación. La herramienta *Replication Management Tool* incluye *Wizards* que guían a través de muchas operaciones y configuraciones importantes.

Figura 7. Herramienta *Enterprise Manager* de *Oracle9i*

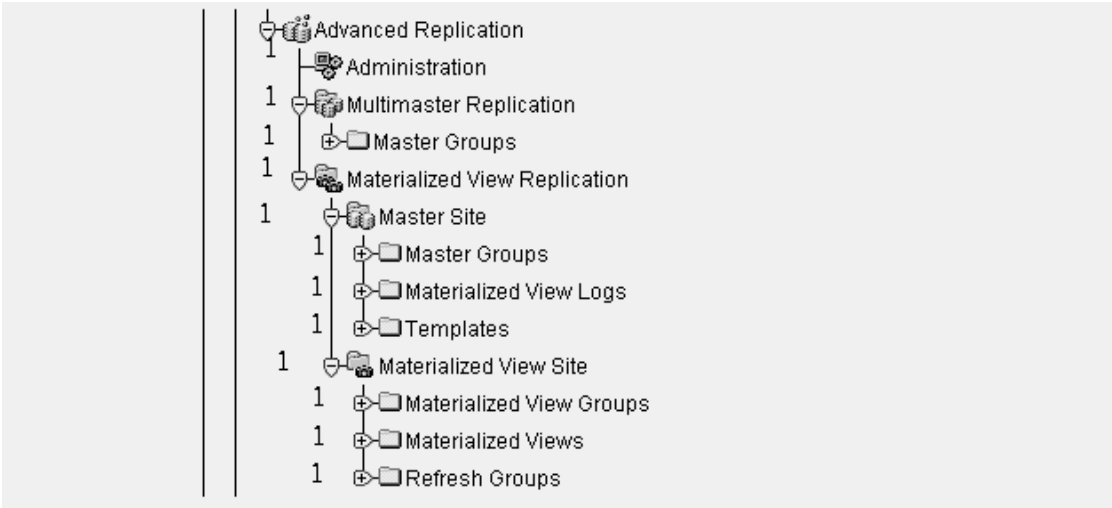
El navegador en *Oracle Enterprise Manager* permite:

- Acceder a todos los nodos en el ambiente de replicación.

- Desplegar y ocultar objetos y carpetas para que usted pueda navegar hacia lo que se desea supervisar o manejar.

- Al pulsar el botón derecho en un objeto o carpeta permite crear una nueva ejecución de una operación de un objeto existente.

Figura 8. Herramienta *Replication Management Tool* de *Oracle9i*



4.4.2 *SQL Server 2000* de *Microsoft*

El modelo utilizado en *SQL Server* se basa en la metáfora de "publicadores/distribuidores/suscriptores". Un SQL Server que participa de un ambiente de replicación, cumple al menos uno de los roles de: servidor publicador (emite datos replicados), servidor distribuidor (distribuye datos replicados) y servidor suscriptor (recibe datos replicados). Datos de solo lectura deben manejarse en *cache* para mejorar la ejecución de la aplicación (XML).

4.4.2.1 Replicación *Snapshot* y *Transactional*

La replicación *Snapshot* es el tipo más simple de todos, donde el publicador replica datos tal como están en la base de datos en un momento dado. La publicación se puede realizar en forma cronogramada o por demanda.

La replicación transaccional realiza un monitoreo de los cambios a los datos que son realizados en el publicador, son propagados a los suscriptores en forma cronogramada o en forma continua, de forma tal que se aproxima a una transacción en tiempo real.

La replicación *Snapshot* y *Transactional* se basan en un modelo de replicación en una sola dirección, desde un único publicador hacia los suscriptores. En los casos en que se desee flujo de datos desde los suscriptores hacia el publicador, se dispone de la opción *Immediate Updating Subscribers (IUS)*.

Se dispone adicionalmente de la opción *Queued Updating Subscriber (QUS)*, donde los datos modificados en una réplica son aplicados usando 2 servidores, a los datos locales y también al servicio de cola, denominado *Microsoft Message Queue*.

4.4.2.2 Replicación *Merge*

En este tipo de replicación, *SQL Server* realiza 3 cambios en el esquema de la base de datos.

- Establece una columna de identidad especial que permite identificar a cada tupla a través de sus múltiples copias existentes en el ambiente de replicación.

- *SQL Server* instala *triggers* para monitorear los cambios en los datos y llevar un registro de los mismos en un conjunto especial de tablas del sistema.

- *SQL Server* agrega un conjunto de tablas del sistema, encargadas monitorear los datos alterados, sincronizarlos y detectar y resolver los conflictos.

4.4.3 *DB2* de *IBM*

4.4.3.1 DB2 DataPropagator Relational

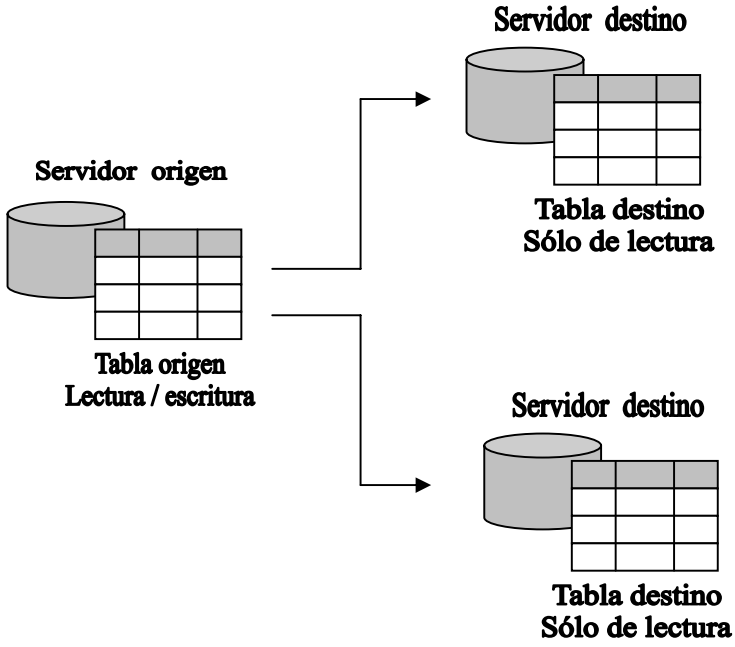
Puede combinar configuraciones para replicación y así satisfacer las necesidades de la empresa.

- Datos distribuidos.
- Actualización en donde sea.

4.4.3.1.1 Datos distribuidos

Una fuente de datos primarios reside en un servidor origen. Al producirse cambios a los datos orígenes, son replicados a uno o más tablas destino las que residen en cualquier parte en una red distribuida. Las tablas destino son únicamente de lectura; por lo que no necesita mecanismos de detección de conflictos porque no ocurren durante la actualización de la replicación. Las aplicaciones pueden usar las tablas destino, las que son copias locales para que éstas no carguen excesivamente la red o el servidor central. Esta configuración es útil si se necesita compartir los datos entre varios sitios y no se desea reducir el rendimiento de sus aplicaciones.

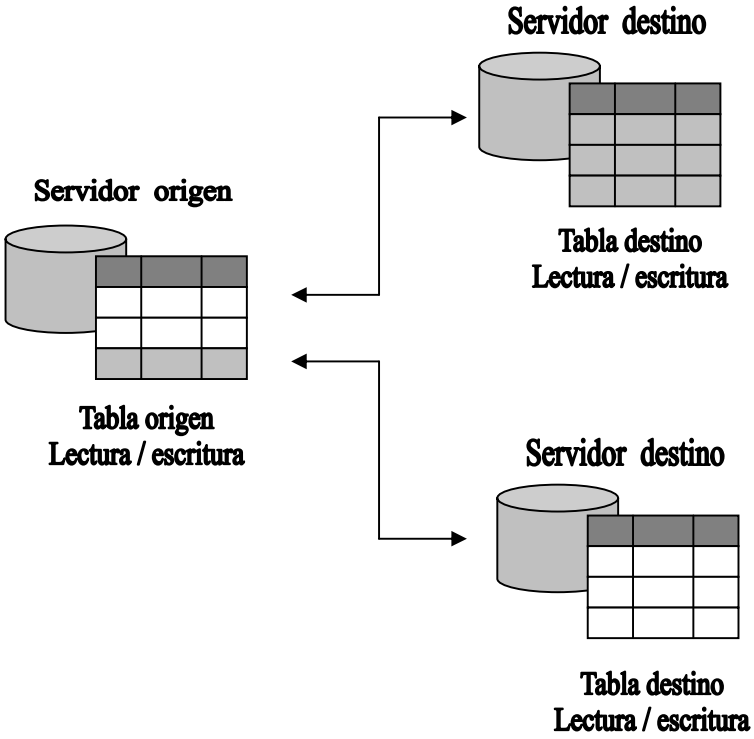
Figura 9. Esquema de "datos distribuidos" con *DB2 DataPropagator Relational*



4.4.3.1.2 Actualización en “donde sea”

Un servidor origen también tiene tablas destino que son las copias de lectura/escritura. Al producirse cambios en las tablas destinos son aplicados en la tabla origen, las que mantienen los datos más actuales. Al aplicarse los cambios a la tabla origen se propaga a todas sus tablas destino. Conflictos de actualización podrían ocurrir cuando los datos se replican. Si se desea usar este tipo de configuración de replicación, se debe decidir que método de resolución de conflictos emplear. Sus opciones van desde ignorar los conflictos y rechazar cualquier actualización, arriesgándose a perder un poco de información, hasta diseñar un sistema para que un conflicto nunca pueda ocurrir, previniendo todos los posibles conflictos.

Figura 10. Esquema de “actualización en donde sea” con *DB2 DataPropagator Relational*



4.4.4 Análisis de la solución

Existen en el mercado soluciones de replicación tanto de *hardware* como de *software*. Empresas ofrecen subsistemas de almacenamiento que permiten realizar replicación a través de *hardware*, entre un sitio principal y sitios de contingencia. Se pueden encontrar soluciones de replicación por *software* que abarcan ciertas aplicaciones en forma conjunta, es posible manejar la replicación nativa de cada aplicación como lo implementan *SQL Server 2000 de Microsoft*, *Oracle9i* y *DB2 de IBM*.

Cuando se trata de replicación de *hardware*, las tareas se realizan por controladoras de los subsistemas de almacenamiento. Pero las controladoras de un proveedor de *hardware* generalmente no permiten replicar hacia controladoras de otro proveedor, por lo que las soluciones de replicación por *hardware* implican la implementación de subsistemas de almacenamiento de un único proveedor, cuando se trata de la replicación por *software* no existen limitaciones de este tipo.

Cuando se habla de replicación por *software*, podemos emplear la replicación por sesiones *HTTP*, utilizando el protocolo de replicación *DRP*. El objetivo del protocolo *DRP* es mejorar significativamente la eficiencia y fiabilidad de los datos a través de sesiones *HTTP*. El Protocolo de distribución y replicación por sesiones *HTTP* fue diseñado para replicar eficientemente archivos con una estructura jerárquica a un gran volumen de clientes.

Usando el protocolo *DRP*, el cliente puede descargar y actualizar solo la información que ha cambiado desde la última vez que fue revisada.

Este protocolo proporciona disponibilidad y versiones de los datos que se distribuyen, define un índice con un conjunto de información que describen las versiones, los tamaños, los tipos y la estructura jerárquica de la información. Finalmente diferencia las peticiones, para enviar únicamente los ficheros diferentes.

4.5 Caso Práctico

4.5.1 Replicación por sesión *HTTP*:

Usando sesiones *HTTP* se puede hacer una replicación hacia un servicio de resguardo de seguridad (*backup*) o hacia una base de datos.

- Para replicar usando una sesión *HTTP* hacia un servicio de resguardo de seguridad (*backup*), se puede usar la siguiente configuración:

```
<configuration>
 <web-app>
 <default-session-replication>true</default-session-replication>
 </web-app>
</configuration>
```

- Para replicar usando una sesión *HTTP* hacia una base de datos, se puede usar la siguiente configuración:

<configuration>
<web-app>
<session-config> <default-session-replication>false</default-session-replication> <session-replication-class>com.iona.j2ee.plugins.distsession.DistributedHttpSessionFactory</session-replication-class>
*<res-ref-link>**EjemploSession**</res-ref-link>*
<tx-isolation-level>request</tx-isolation-level>
<scrub-timeout>100000</scrub-timeout>
</session-config>
</web-app>
<resources>
<resource>
*<resource-name> **EjemploSession**</resource-name>*
<jndi-name>iona:cloudscape/demos</jndi-name>
</resource>
</resources>
</configuration>

La etiqueta *<distributed>* activa o desactiva las sesiones distribuidas, el valor por defecto es "falso". La etiqueta *<res-ref-eslabón>* especifica a la base de datos usar sesiones distribuidas.

Se crean las tablas necesarias automáticamente en la base de datos si no existen, usando las declaraciones de SQL siguientes para crear las tablas:

CREATE TABLE HttpSession (ID CHAR(12) PRIMARY KEY, CreationTime NUMERIC(20,0), LastAccessTime NUMERIC(20,0), MaxInactiveInt NUMERIC(11,0))

CREATE TABLE HttpSessionAttribute (ID CHAR(12), ValueCounter NUMERIC(5,0), ValuePart CHAR(40), PRIMARY KEY (ID, ValueCounter))

4.6 Análisis de servidores XML

XML, comienza una nueva era en internet, se perfila como la solución al dilema del intercambio universal de información, al aparecer servidores de internet especialmente diseñados para satisfacer los requerimientos de XML. Entre sus funciones destacan el almacenamiento y manipulación de datos XML, características mejoradas en la búsqueda de datos, integración de fuentes diversas de información en bases de datos XML unificadas.

4.6.1 Principales servidores XML

4.6.1.1 *BizTalk Server* de *Microsoft*

Especialmente diseñado para el comercio electrónico, el objetivo de *BizTalk* es el de desarrollar una red dentro de su portal *Microsoft Network*, que permita poner en contacto a empresas y consumidores. Todo ello implicará cambios y mejoras en herramientas de comercio electrónico ya existentes.

Requerimientos mínimos del sistema:

- Sistema operativo que soporta: *Windows 2000*

Hardware:

- *Intel Pentium III* procesador 800 MHZ.
- Mínimo: 128 *megabytes* (MB) de RAM; recomendado: 256 MB.
- 6 *Gigabyte* (GB) de disco duro.
- Unidad de *CD-ROM*.
- Adaptador para tarjeta de red.
- Monitor VGA o súper VGA.

Software:

- *Microsoft Windows 2000 Server* con *Service Pack 3*.
- *Microsoft SQL Server 2000* con *Service Pack 2*.
- *Internet Explorer 6.0* con *Service Pack 1*.
- *Microsoft BizTalk Server 2002*.
- *Microsoft .NET Framework* con *Service Pack 2*

4.6.1.2 *Tamino XML Server*

Tamino mejora las prestaciones del servidor XML, centrándose en los Servicios. Las herramientas de *Tamino XML Server* simplifican la implementación en todas las áreas de utilización del XML. Los nuevos servicios:

- *Tamino WebDAV Server*: protocolo *Web-based Distributed Authoring and Versioning (WebDAV)*, que permite que múltiples usuarios puedan colaborar en un mismo documento a través de internet.

Los usuarios pueden acceder a los documentos XML almacenados en *Tamino* utilizando aplicaciones que soporten este estándar.

- *Tamino X-Plorer*: interfase gráfica de usuario, similar en su configuración y utilización al explorador de *Windows*, reproduce la estructura y los contenidos de los documentos XML y ofrece unas eficaces funciones de búsqueda.

- *Tamino X-Application Framework*: basado en *Java*, vincula las páginas *WEB* creadas con herramientas estándar con *Tamino XML Server*, sin que se precise ningún tipo de programación adicional.

Requerimientos Mínimos del Sistema:

Plataformas:

- *Microsoft Windows NT/2000/XP, Unix - AIX, Unix - Linux, Unix - Solaris*.

Hardware:

- *Intel Pentium III* procesador 700 MHZ.
- 256 *megabytes* (MB) de RAM.
- 5 *Gigabyte* (GB) de disco duro con sistema de archivos *NTFS*.

Software:

- *Microsoft Windows 2000* ó *XP*.
- *Internet Explorer 5.0* con *Service Pack 1*.
- *Apache 1.3.10* ó *Microsoft IIS*, versión 5.

4.6.1.3 *Sonic Business Integration Suite*

Ofrece una plataforma escalable e integral para una integración empresarial distribuida, basada en estándares, es una herramienta de integración elaborada en base a una arquitectura de servicios para la empresa.

- *Sonic XML Server:* ofrece capacidad de almacenamiento XML escalable y optimizado, búsquedas, servicios de transformación y procesamiento, utilizando estándares *XSLT* y *XQuery* a fin de apoyar las labores de auditoria, registro, organización y compilación de datos, así como la captación de datos empresariales y las funciones de monitorización.

CONCLUSIONES

1. XML, es un lenguaje de marcado de documentos diseñado para trabajar en internet; es simple, flexible y fácil de utilizar. Consigue un intercambio automatizado de documentos en internet que permite el desarrollo de aplicaciones existentes y de otras nuevas.

2. XML permite integrar sistemas de información hasta ahora separados. Sistemas de información basados en documentos (o archivos), tienen estructura irregular, anidados profundamente, utilizan tipos de datos relativamente simples y dan gran importancia al orden. Y sistemas de información estructurados (bases de datos relacionales): tienen una estructura muy regular, son relativamente planos, utilizan tipos de datos relativamente complejos y dan poca importancia al orden.

3. Uno de los aspectos más que se ha marcado más recientemente en el campo de las bases de datos es el crecimiento vertiginoso de internet. La conexión de las bases de datos con la *WEB* debe seguir progresando, por lo que la inclusión de XML es casi necesaria para poder utilizar un único formato estándar para su comunicación sin importar a qué proveedor pertenezca.

4. Las bases de datos XML nativas no representan un nuevo modelo de base de datos ni pretenden reemplazar a los sistemas de almacenamiento actuales.

Se han planteado como una herramienta de soporte al desarrollo que permitan a los desarrolladores disponer de una solución robusta para el almacenamiento y manipulación de datos en formato XML.

5. En los estándares del modelo de datos de XML, una replicación de un documento dinámico XML dinámico puede interpretarse como un árbol etiquetado. Cuya definición sea dada en términos de consultas de *XQuery*, para la lectura de la información en los nodos.

RECOMENDACIONES

1. En su calidad de estándar, XML ofrece muchas ventajas a las organizaciones, desarrolladores de *software*, sitios *WEB* y usuarios finales. Las oportunidades aumentarán cuantos más formatos de datos se creen para mercados claves, como el mercado de búsqueda avanzada en bases de datos, banca en línea, médico, comercio electrónico, otros.

2. La mayoría de proveedores de soluciones de bases de datos con soporte XML prefieren desarrollar sus propias soluciones de almacenamiento en vez de utilizar tecnologías existentes y consolidadas. Puede ser un claro síntoma de que los sistemas más extendidos (sistemas relacionales y orientados a objetos) no se adaptan fácilmente al paradigma XML. Sería necesario realizar pruebas comparativas entre los productos que emplean soluciones a medida con el resto, para determinar las carencias y ventajas de cada una de las alternativas.

3. Las bases de datos XML nativas, no es la intención sustituir los sistemas existentes de bases de datos, son simplemente otra herramienta para los desarrolladores en XML, y cuando son aplicadas en circunstancias correctas es que pueden rendir ventajas significativas.

4. Es necesario se que siga creando nuevos diseños para replicación de bases de datos XML nativas, con el objeto de que su implementación sea más sencilla y eficaz, dado que la solución que proponen los proveedores de bases de datos con soporte XML es muy similar al ya existente con bases de datos relacionales.

BIBLIOGRAFÍA

1. Abiteboul, Serge y otros. *Dynamic XML Documents with Distribution and Replication*. http://wwwrocq.inria.fr/verso/Gemo/Projects/axml/. 2003.

2. Acosta, José Javier y Jara, Juan José. *XML (eXtensible MarkUp Language)*. http://www.ramon.org/xml/articulos/intro_xml-html.htm. Universidad Católica Nuestra Sra. de la Asunción. 2001.

3. Barckey, Chad y otros. *Schema-Independent XML Database System*. Universidad de California. http://www.rpbourret.com/xmldbms/index.htm 2002.

4. Bremer, Jan-Marco y Gertz, Michael. *On Distributing XML Repositories*. Universidad de California. 2002.

5. Bourret, Ronald. *XML Database Products*. ttp://www.rpbourret.com/xml/ XMLDatabaseProds.htm. 2003.

6. Buck, Lee. *Modeling Relational Data in XML*. http://www.rpbourret.com/ xml/XMLAndDatabases.htm. Enero 2001.

7. Burke, Paul. XML: El último gran invento. *SQL Server Magazine*. http://www.windowstimag.com/sqlmag/atrasados/01_sep00/articulos/inve nto.htm Septiembre 2000.

8. Campos, Manuel. Familia XML. http://www.inf.utfsm.cl/~rmonge/ sd/XML.pdf Universidad Técnica Federico Santa Maria. 2001.

9. Date, C.J. Introducción a los sistemas de bases de datos. Séptima Edición. Traducción: Ruiz Faudón, Sergio Luis María. Editorial Pearson Educación. 2001. 05pp.

10. De la Herrán Gascón, Manuel. Administración y optimización de bases de datos *Oracle*. http://www.redcientifica.com/oracle/c0001p0005.html 2003.

11. Emerick, Jerry. Administrando el almacenamiento XML. http://www.acm.org/crossroads/espanol/xrds8-4/XML_RDBMS.html. 2002.

12. Fernández Panadero, Carmen. XML (*eXtensible Markup Language*) un elemento de gran importancia para el progreso de la red. Universidad Carlos III de Madrid. Abril 2001.

13. García Arenas, María Isabel. Introducción a XML. http://geneura.ugr.es/~maribel/xml/introduccion/index.shtml. 2001.

14. García Cárdenas, Edgar Arturo. XML. http://www.lavariable.com/art/xml/axml003/axml003.asp#Intro. Agosto 2003.

15. González Joyanes, Eduardo y otros. Alcances y limites de XML en *WEB*. Universidad Católica de Ávila. 2002.

16. Grupo alternativo. Un vistazo a las bases de datos XML. http://www.mexicoextremo.com.mx/noticias/database-intro.php3. Septiembre 2003.

17. Grupo de investigación en ingeniería *WEB* y almacenes de datos. XML y bases de datos multidimensionales. http://gplsi.dlsi.ua.es/iwad/investigacion.html. Febrero 2003.

18. Dr. Henseler, Hans. Indexación de bases de datos XML. *ZyLAB Technologies*. http://www.zylab.com. 7 de febrero, 2003.

19. Hernández Orallo, José. La disciplina de los sistemas de bases de datos. Historia y situación actual. Universidad Politécnica de Valencia. Mayo 2002.

20. Holger, André. Bases de datos XML nativas (NXD). http://www.rpbourret.com/xml/XMLAndDatabases.htm. Marzo 2003.

21. Institucional Itec. *DataPropagator Relational 5.1*. http://www.software.ibm.com/data/dpropr. 2003.

22. Manríquez, Felipe. *Oracle9i* mejores prácticas de administración. http://neuronet.cl. Enero 2003.

23. Melendi Palacio, David. Bases de datos XML nativas. http://www.xmlglobal.com/prod/db/index.jsp 2003.

24. Mendoza Canales, César. Arquitectura del sistema de bases de datos. Cap. 5. Universidad Nacional Autónoma de México. Octubre 2003.

25. Muñoz, Álvaro. Documentos XML desde bases de datos relacionales. http://www.gixml.com. Grupo de Interés en XML. Agosto 2001.

26. *Oracle Corporation*. *Oracle9i Advanced Replication*. http://download-west.oracle.com/docs/. 2003.

27. Palos, Juan Antonio. API's XML para bases de datos. http://www.programacion.com/bbdd/articulo/xmlapisdata/. 2003.

28. Pisapati, Sarma. *Introduction to Native XML Databases*. http://www.xml.com/pub/a/2001/10/31/nativexmldb.html. Octubre 2001.

29. Robles, Jesús. *Oracle XML DB*. XML Technology Center. http://otn.oracle.com/tech/xml/content.html. 2002.

30. Rodríguez Aguilar, Rosa María. Modelado conceptual. Universidad de Burgos. 2000.

31. *SQL Server Magazine*. XML y *SQL Server 2000*. http://www.windowstimag.com/sqlmag/atrasados/01_sep00/articulos/XML_1.htm. Septiembre 2001.

32. Urudata, Monty. Replicación de datos a nivel corporativo. http://www.cp.com.uy/92/www.urudata.com.uy. 2000.

33. Vegas, Jesús. Una Introducción a XML. http://www.infor.uva.es/~jvegas/cursos/web/xml/ixml/ixml.html Dpto. de Informática, Universidad de Valladolid. Junio 2001.

34. Verdiell Cubedo, Jordi. Bases de datos para la *WEB*. http://icc2.act.uji.es/e16/2001/jordi_verdiell.pdf. 2001.

35. Wait, Brad. *Oracle Corporation*. *Using XML in Oracle Database Applications*. http://technet.oracle.com/tech/xml/info/htdocs/otnwp/about_oracle_xml_products.htm. Marzo 2001.

36. Zurita Rendón, Hafiz. SAX: API simple para XML. http://www.megginson.com/www.megginson.com/SAX/. Septiembre 2003.