Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Notas Almacenamiento y Recuperación de Información

Jaime Blanco.

RT 2007-08

Centro IOMMA

Caracas, Septiembre, 2007.

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Notas sobre Almacenamiento y Recuperación de Información

Jaime Blanco

RT 2007-08

Centro IOMMA
Caracas, Septiembre, 2007

Notas sobre Almacenamiento y Recuperación de Información

Jaime Blanco
jblanco@ciens.ucv.ve

Universidad Central de Venezuela. Facultad de Ciencias. Escuela de Computación.
Centro de Investigación de Operaciones y Modelos Matemáticos (CIOMMA)
Venezuela. Caracas
Apdo. 47002, 1041-A.

RT 2007-08

ABSTRACT

La generación de información es un proceso mucho más acelerado que nuestra capacidad de recolectarla y asimilarla, de ahí que se dedica mucho tiempo a buscar alternativas de cómo organizar y hacer accesibles rápidamente estas cantidades de información. Una clasificación de los datos de acuerdo a su estructura es: estructurados, no estructurados y semi-estructurados; y de acuerdo al tipo de dato que tengamos, vamos a manejarlos de diferente manera, y con tiempos de respuesta variables; así mismo, con sistemas de representación diferentes, algunos en bases de datos y otros con lenguajes de marcado como XML. También vamos a tener herramientas disponibles de acuerdo al tipo de dato que manejemos, como SQL y sistemas de búsqueda como XPath.

Keywords: Almacenamiento de Información, Recuperación de Información. XML.

Septiembre, 2007

Tabla de contenido

Lista de tablas

Lista de figuras

Clasificación de la Información y Métodos de Recuperación

La vida actual requiere de mucha información y normalmente la información la podemos encontrar principalmente en:

- Bibliotecas (cada vez más es posible consultar sus catálogos a través de Internet y la información en si misma).
- Organismos de gobierno y no gubernamentales (también cada vez más en Internet)
- Expertos en el campo que estudiamos (muy útiles para aclarar la estructura y las relaciones del tema que investigamos)
- Librerías
- Sistemas comerciales de bases de datos como EBSCO o Dialog

Desafortunadamente la generación de información es un proceso mucho más acelerado que nuestra capacidad de recolectarla y asimilarla, de ahí que se dedica mucho tiempo a buscar alternativas de cómo organizar y hacer accesibles rápidamente estas cantidades de información.

En esta sección presentaremos, previa diferenciación entre dato e información, una clasificación de los datos de acuerdo a su estructura, así como distintos tratamientos para su almacenamiento y recuperación.

1. DATOS VS INFORMACIÓN

Es importante definir y distinguir algunos conceptos que nos ayudarán a entender qué es lo que se almacena/recupera en una biblioteca digital, para de esta manera entender las diferencias entre ellos y determinar las guías a seguir para su tratamiento.

Contrario a lo que muchas personas pudieran pensar, datos e información no son lo mismo al momento de su almacenamiento y/o recuperación. Cuando pensamos en datos estamos hablando de componentes tangibles y cuantificables, pongamos el ejemplo de un artículo de congreso. Si consideramos los "datos" del artículo, estaremos considerando cosas como el título, el autor, las palabras clave, etc., que de alguna forma componen los "*metadatos*" del artículo. Por otro lado, la información consiste en el contenido del artículo, los temas que trata, las fórmulas que emplea, etc.

La diferencia antes mencionada es muy importante, y se debe considerar al momento de almacenar y/o recuperar la información. En el caso de los datos, podemos definir atributos de los documentos que nos parecen sobresalientes, al momento de realizar búsquedas; mientras que con la información tendríamos primero que leer el material para

determinar qué es relevante y qué no lo es. De allí la necesidad de disponer de metodologías adecuadas que permitan almacenar y recuperar información y no sólo datos.

2. TIPOS DE DATOS DE ACUERDO A SU ESTRUCTURA.

Una clasificación de los datos de acuerdo a su estructura es: estructurados, no estructurados y semiestructurados. A continuación serán presentados brevemente.

Una biblioteca digital, dadas las necesidades actuales y el crecimiento excesivo de información, debe poseer la capacidad de manipular los 3 tipos de datos mencionados, por lo que también serán presentados brevemente algunas consideraciones para cada tipo de dato, en el contexto de bibliotecas digitales.

2.1. DATOS ESTRUCTURADOS

Este tipo de datos esta caracterizado por:

- Estar organizados de alguna manera
- Son atributos o variables fuertemente tipeados (int, float, string)
- Cada atributo en una relación está definido para todos los registros
- Ejemplos: registros, base de datos relacional (Tabla 1 Datos Estructurados)

Nombre [char(10)]	cumpleaños [date]	sueldo [int]
Carlos	13-08-1980	5000
Juan	23-02-1977	7500

Tabla 1 Datos Estructurados

Una Biblioteca Digital (BD) cuenta con infinidad de datos estructurados, desde los datos bibliográficos de los documentos (catálogo), los perfiles del usuario, la información necesaria para el funcionamiento de agentes, negociaciones de comercio electrónico y demás aspectos inherentes a la administración de la biblioteca digital.

2.2. DATOS NO ESTRUCTURADOS

Este tipo de datos esta caracterizado por:

- No están organizados de acuerdo a algún patrón
- No poseen definiciones de tipos

- No existe el concepto de variables o atributos
- Ejemplos: documentos de texto sin estructura, e-mails, páginas de html (Tabla 2 Datos No Estructurados)

"Carlos nació el 13 de Agosto de 1980. El tiene un sueldo de 5000. Alguien más nació el 23 de Febrero de 1977, su nombre es Juan y su salario es de 7500"

Tabla 2 Datos No Estructurados

Como se puede observar no existe una manera automática de poder analizar este dato para hacer cuestionamientos, a esto nos referíamos en la sección anterior como información.

En una Biblioteca Digital, los documentos, aunque asociados a datos estructurados (metadatos) que los describen, son considerados como no estructurados y es difícil, aunque no imposible, hacer preguntas acerca del contenido de materiales que forman los acervos.

2.3. DATOS SEMIESTRUCTURADOS

Este tipo de datos esta caracterizado por:

- Variables pobremente tipeadas (x=1 es válido y x="hola" también es válido)
- Un registro no necesariamente tiene que tener todos sus atributos definidos. Mientras por ejemplo en una base de datos relacional un campo debe establecerse como NULL cuando no se tiene, en un ambiente de datos semiestructurados basta con omitir dicho atributo.
- Un atributo de un registro puede ser otro registro
- No existe necesariamente una diferencia entre un identificador de un campo y el valor mismo de este.
- Ejemplos: documentos SGML y XML

Los datos semiestructurados pueden ser representados como:

- **Árbol** (Figura 1)

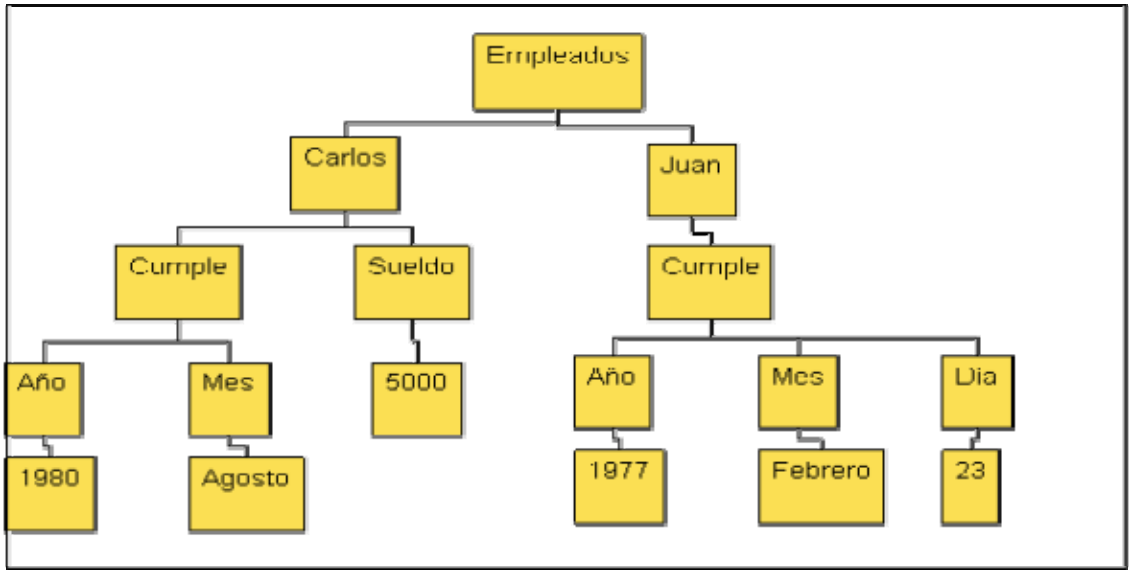


Figura 1 Representación datos semiestructurados en árbol

- **Texto indentado** (Tabla 3 Representación datos semiestructurados texto indentado)

```
Carlos
        Cumpleaños
                1980
                Agosto
                13
        Sueldo
                $5,000
Juan
        Cumpleaños
                1977
                Febrero
```

Tabla 3 Representación datos semiestructurados texto indentado

- **Lenguaje de Marcado** (Markup Language) (Tabla 4 Representación datos semiestructurados lenguaje marcado)

```
<compania>
<empleado id=”3”>
        <nombre>Carlos</nombre>
        <extension>5513</extension>
        <departamento>Ventas</departamento>
        <sueldo>5000</sueldo>
</empleado>
<empleado id=”1”>
        <nombre>Alfredo</nombre>
        <extension>2666</extension>
        <oficina>312</oficina>
        <departamento>Ejecutivo</departamento>
        <sueldo>Director</sueldo>
<empleado>
</compania>
```

Tabla 4 Representación datos semiestructurados lenguaje marcado

Otra característica de los datos semiestructurados es que no son creados necesariamente con la intención de ser analizados o más aún de ser interrogados, por ejemplo las páginas de HTML no son creadas con ese propósito mientras que archivos en XML (Tabla 5 Documento XML) si lo son.

```
<tesis>
    <indice>
        contenido indice
    </indice>
    <capitulo numero="1" titulo="introduccion">
        <seccion numero="1">
            contenido 1.1
        </seccion>
        <seccion numero="2">
            contenido 1.2
        </seccion>
    </capitulo>

    <capitulo numero="2" titulo="desarrollo">
        <seccion numero="1">
            contenido 2.1
        </seccion>
    </capitulo>

    <capitulo numero="3" titulo="conclusiones">
        <seccion numero="1">
            contenido 3.1
        </seccion>
        <seccion numero="2">
```

```
            contenido 3.2
        </seccion>
        <seccion numero="3">
            contenido 3.3
        </seccion>
    </capitulo>
</tesis>
```

Tabla 5 Documento XML

Aunque XML es un lenguaje que en un inicio fue mayormente utilizado para intercambio de datos, inclusive entre Bibliotecas Digitales, cada vez más se está utilizando para almacenar un sin fin de documentos y transacciones. Aún cuando hablamos de una tecnología bastante joven, estos datos y la forma en que son manipulados están en constante evolución y promete bastantes cosas para el futuro.

3. ALMACENAMIENTO Y RECUPERACIÓN DE INFORMACIÓN EN DATOS ESTRUCTURADOS

Hablar de datos estructurados es hablar de los primeros tipos de datos que existieron o mejor dicho, que fueron de interés para la comunidad computacional debido a su relación directa con los lenguajes de programación donde existe el concepto de estructura, registro, variable y campo, y las necesidades de las diversas compañías e instituciones de investigación. Nóminas, inventarios, estadísticas, cuentas y catálogos, son datos estructurados y era evidente la necesidad de crear algoritmos capaces de manipular este tipo de datos, o sea algoritmos de indexamiento.

3.1. ALGORITMOS

Recordemos que el principal problema que llevó a crear los índices fue y sigue siendo, la lentitud del disco duro, que durante muchos años ha sido el medio de almacenamiento por excelencia pero al ser un dispositivo mecánico implica que sea lento por naturaleza.

De los algoritmos de indexamiento podemos mencionar a 3 principales:

- **ISAM**: básicamente es la idea de tener un índice esparcido, de manera que el archivo de datos se agrupa por bloques y en cada bloque existe un "rango" ordenado de los datos.
- **BTree**: es la generalización de un árbol que posee varios índices por cada nodo o página que apuntan hacia niveles inferiores en cuyas hojas están los apuntadores hacia los datos.

- **Hash**: aunque no es propiamente un índice, esta técnica se basa en una fórmula que nos crea una dirección para un registro y una estructura "trie" que permite ubicar más fácilmente la dirección entre bloques de datos.

Las técnicas anteriores tienen sus ventajas y desventajas:

- **ISAM**:
  - Ventajas: es fácil de implementar
  - Desventajas: no es muy eficiente para grandes volúmenes de datos

- **BTree**:
  - Ventajas: fácil de implementar, extremadamente rápido
  - Desventajas: sacrifica un poco de espacio por la repetición de algunos elementos del árbol, pero actualmente esto es despreciable.
- **Hash**:
  - Ventajas: se puede tener un muy buen tiempo de respuesta en muchos casos
  - Desventajas: es difícil de implementar y no hay métricas precisas acerca del comportamiento del "trie"

Actualmente el BTree (junto con sus derivaciones y modificaciones) es la estructura más popular y eficiente; de manera que cuando se trata con datos estructurados en gran volumen siempre se considera dicha opción. Aunque los otros indexamientos también son utilizados, principalmente para aplicaciones que no requieran buen tiempo de respuesta o que manejen un conjunto reducido de datos.

3.2. HERRAMIENTAS

Podemos clasificarlas en 2 categorías:

- Sistemas manejadores de bases de datos (DBMS) que son sistemas complejos que garantizan la "acidez" de los datos (ACID Atomicidad Consistencia Aislamiento Durabilidad); aquí podemos mencionar entre los más populares: Sybase, Oracle, MySQL y PostgreSQL. Desafortunadamente la mayoría de estos manejadores son productos comerciales y son en extremo costosos; los únicos 2 gratuitos son MySQL y PostgreSQL (también Sybase en su versión Linux).
- Herramientas sencillas que cumplen con su función, pero no se preocupan por manejo de errores, concurrencia y otros factores. En la tabla a continuación mostramos algunas de las herramientas más usadas al momento de escribir este trabajo.

Siglas	Nombre	Dirección	Indexamientos
GDBM	GNU database	http://www.gnu.org/software/gdbm/gdbm.html	Hash
BDB	Berkeley Database	http://www.sleepycat.com/	Hash, BTree, Recno

JISP	Java Indexed Serialization Package	http://www.coyotegulch.com/products/jisp/	Hash, BTree

Tabla 6 Herramientas de Indexamiento de Datos

4. ALMACENAMIENTO Y RECUPERACIÓN DE INFORMACIÓN EN DATOS NO ESTRUCTURADOS

4.1. ALGORITMOS

Cuando un dato no tiene estructura por lo general nos referimos a algún tipo de documento. Si se desea hacer algún tipo de búsqueda habría que revisar si los términos que conforman la consulta se encuentran en el/los documentos. Esta tarea puede ser demasiado costosa que buscar un término se convierte en una búsqueda secuencial.

Sin embargo, se puede pensar en extraer todos los términos de cada documento e indexarlos (a manera de datos estructurados) de manera que tendremos una lista invertida que nos permite ir desde los términos hacia los documentos que los contienen. A esta actividad se le conoce como “indexamiento a texto completo” (fulltext indexing).

Para el tratamiento de este tipo de datos debemos recordar el procedimiento que sigue el manejo de documentos.

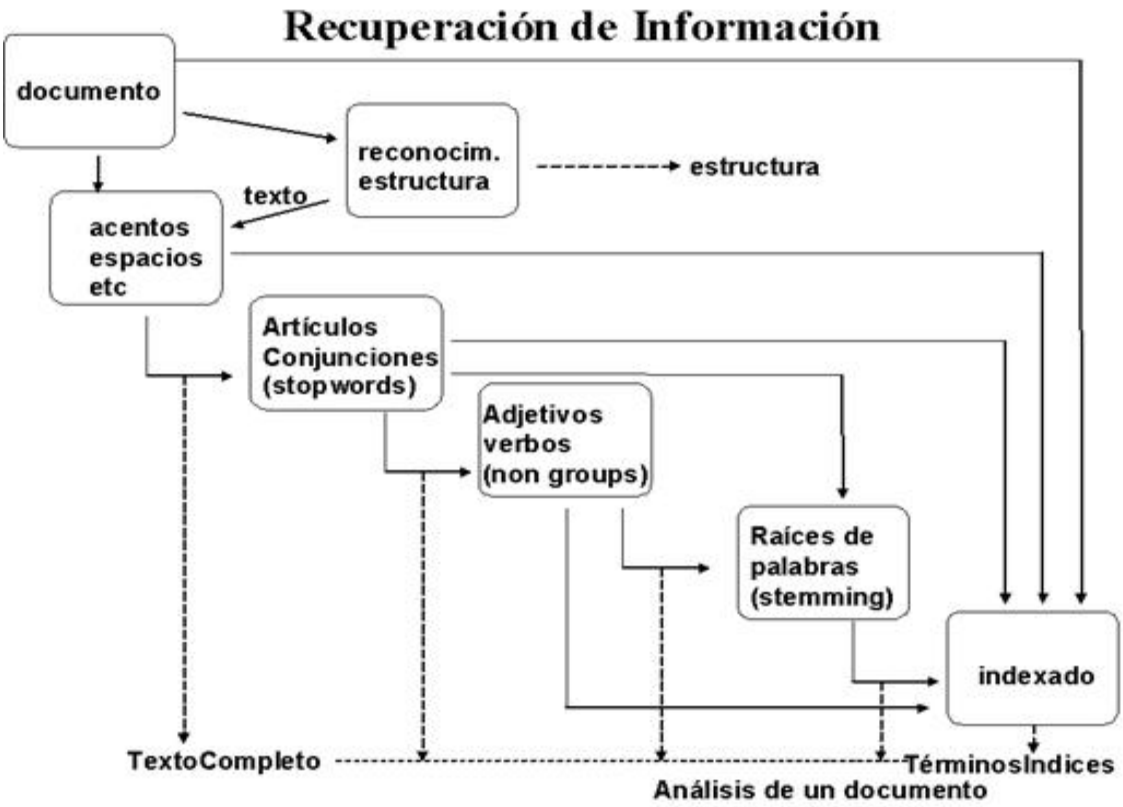


Figura 2 Procesamiento de un documento en un sistema de Recuperación de Información

Para no tener que almacenar e indexar todos los términos de un documento, la Figura 2 muestra algunas de las técnicas para eliminar aquellos que son innecesarios:

- Lematización: eliminación de palabras comunes en el lenguaje como artículos, conjunciones, etc.
- Truncamiento: dejar las palabras en sus raíces sintácticas, de manera que palabras como: computación, computacionalmente, computadora son reducidas a "comp" y se pueden hacer búsquedas por cualquiera de los términos indexados.

Es importante resaltar que actualmente muy pocos sistemas siguen aplicando la lematización, ya que en la práctica se ha demostrado que existen frases que poseen algunas de estas palabras y que al eliminarlas se impediría el obtener resultados para algunas búsquedas como "**La** Casa **de** Juárez".

Desafortunadamente lo anterior no es suficiente; cuando un usuario realice una búsqueda se encontrará con miles de documentos que satisfacen su consulta. Aquí surge la "relevancia", el poder ordenar los documentos resultantes de acuerdo a un criterio en el cual se considera la utilidad o similitud que tiene cada documento con la consulta del usuario.

Para poder determinar esta relevancia existen distintos modelos (Figura 3) de recuperación que se centran en características de los términos de los documentos, sus frecuencias dentro de cada documento y dentro de la colección.

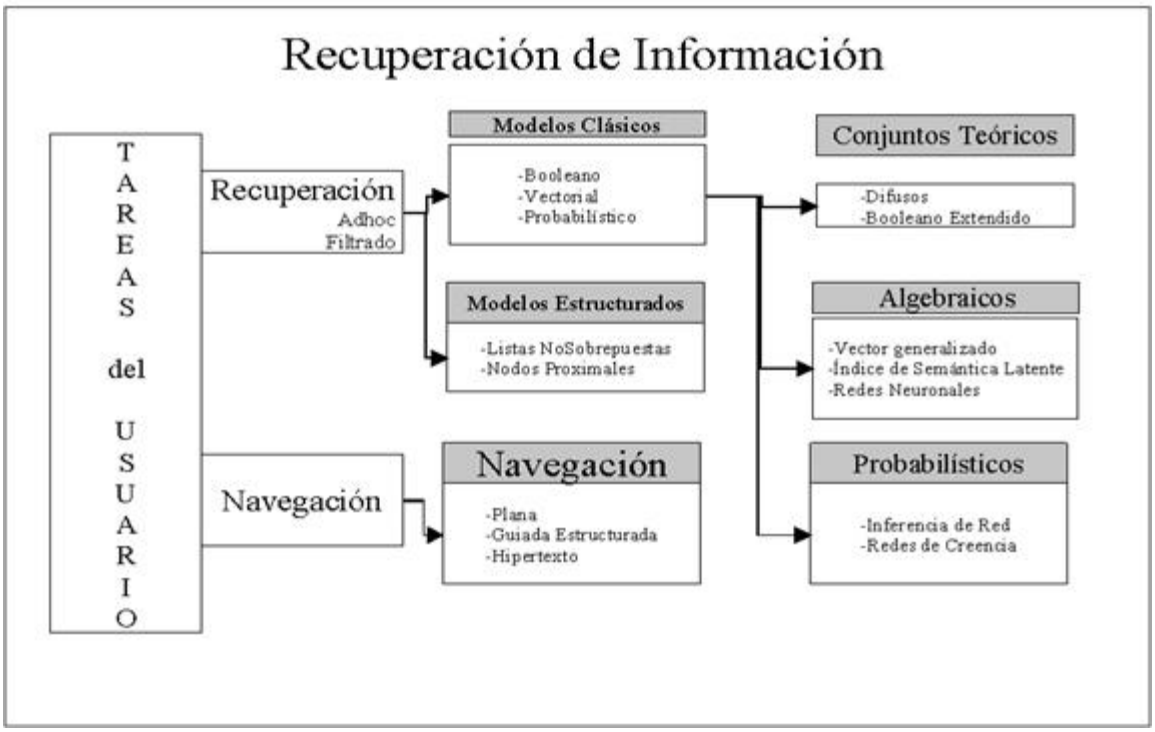


Figura 3 Modelos de Recuperación de Información

De manera que la arquitectura completa (Figura 4) de un sistema de recuperación de información, no solo se basa en el indexamiento de términos, sino también en el ordenamiento por relevancia de los resultados con base en los distintos modelos, existiendo retroalimentación del usuario que permite refinar las consultas y obtener documentos más apegados a las necesidades de cada persona.

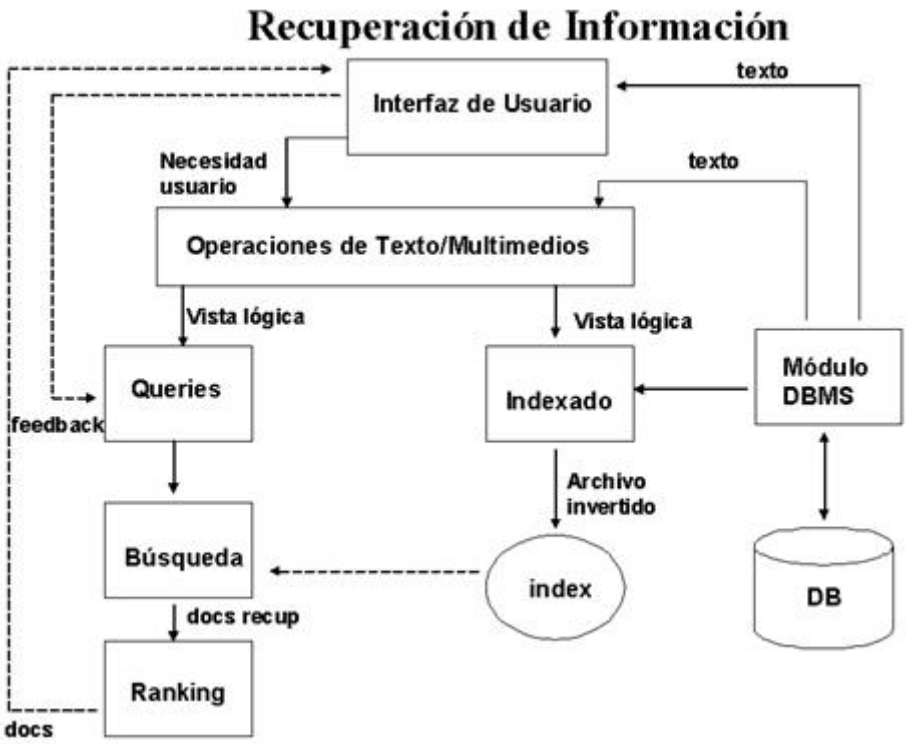


Figura 4 Arquitectura de un Sistema de Recuperación de Información

La complejidad de los modelos es muy distinta, al igual que la precisión con la que seleccionan los resultados; mientras el modelo booleano sólo verifica la existencia o no de algún término en el documento, el vectorial basa la similitud en el ángulo comprendido entre los vectores que forman cada documento y la consulta (dichos vectores se forman por el peso de cada término en el documento o consulta), recordando que dados 2 vectores u, v se tiene $u \cdot v = |u|\,|v| \cos\theta$

Los datos empleados por estos modelos pueden ser almacenados de 2 distintas maneras:

1) En un manejador de bases de datos
2) A través de un sistema de indexamiento a la medida utilizando listas invertidas

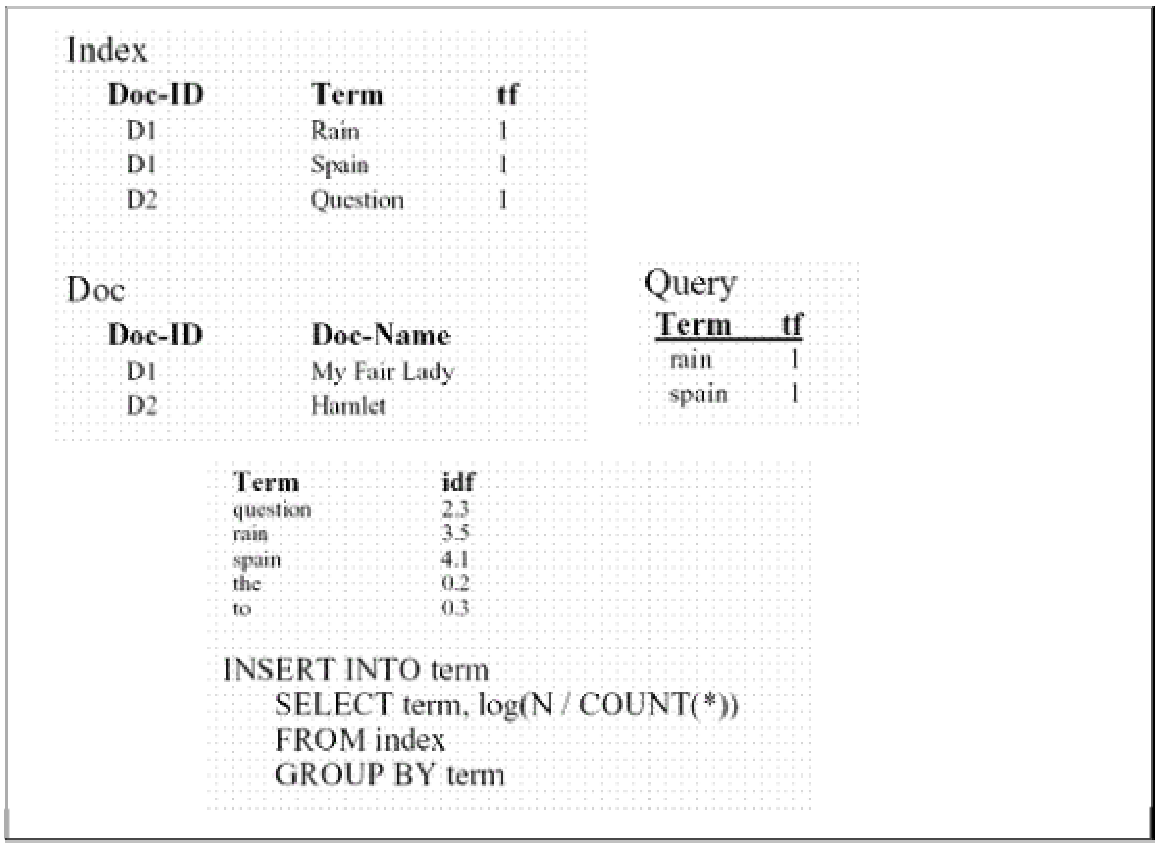


Figura 5 Esquema RI en una base de datos

La Figura 5 muestra el esquema que sigue una base de datos relacional para poder almacenar los datos necesarios por los modelos booleano y vectorial, los cuales son:

- tf (term frequency): la frecuencia de un término en cada documento
- idf = log (N/ni) Frecuencia inversa,
  N-número de documentos en la colección, ni-número de documentos donde aparece un término
- w = tf x idf, el peso de un término en un documento

De manera que la Figura 6 muestra la forma en que se realizaría una búsqueda aplicando el modelo booleano.

Mientras que la Figura 7 presenta la manera de realizar una búsqueda aplicando el modelo de espacios vectoriales.

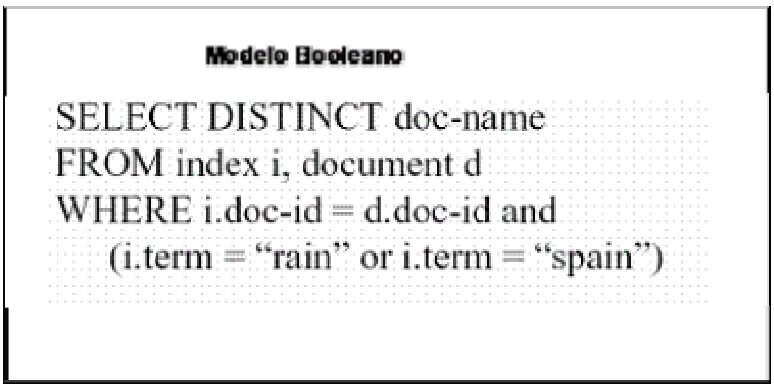


Figura 6 Búsqueda modelo booleano

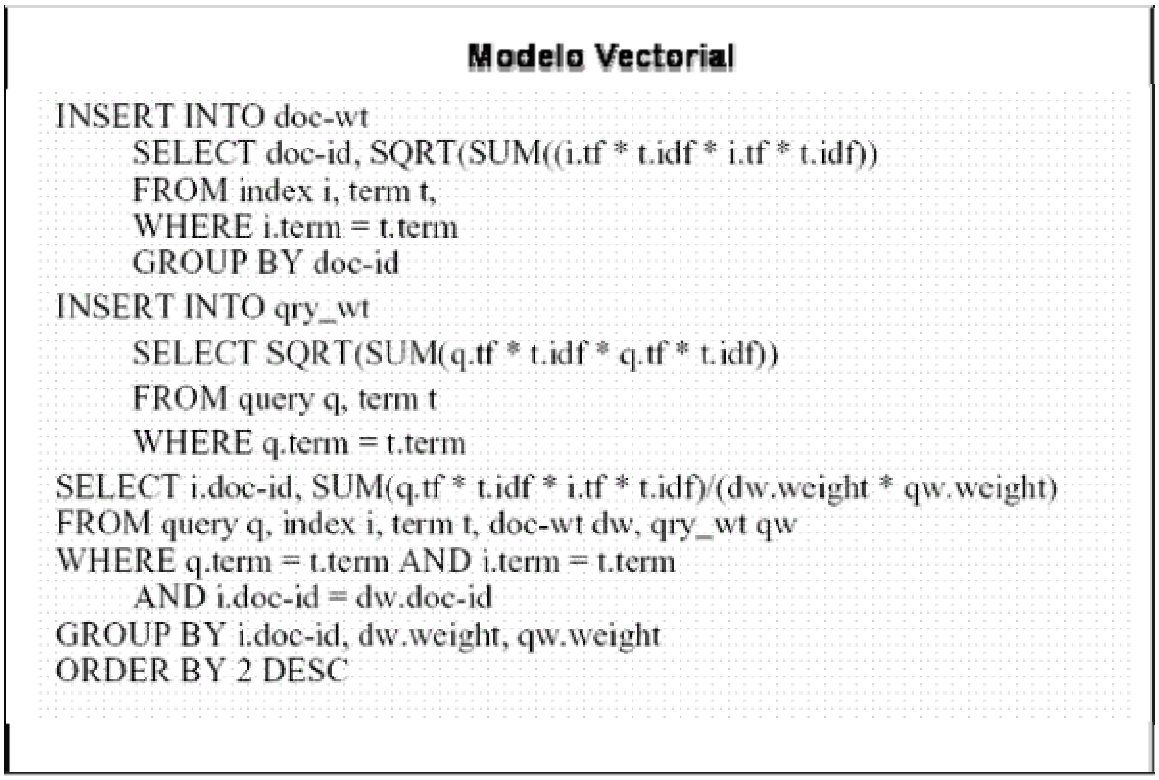


Figura 7 Búsqueda modelo vectorial

Se puede observar que el inconveniente de usar la base relacional radica en la penalización del tiempo de respuesta ya que se hacen varios cálculos para poder llegar a los resultados. Por otro lado el esquema de listas invertidas es más simple ya que de un término se desprende la lista de *posteo* de todos los documentos donde aparece dicho término, esto facilita las cosas debido a que precisamente esa es la secuencia de una consulta.

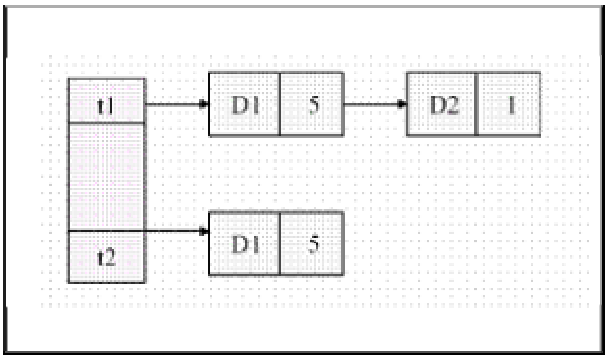


Figura 8 Esquema RI con listas invertidas (términos y documentos)

4.2. HERRAMIENTAS

Existen distintas implementaciones de los modelos de recuperación, las más populares se presentan en la Tabla 7 Herramienta RI.

Producto	Lenguaje	Modelos	Indexamiento	API desarrollo	Licencia
Managing Gigabytes	C/C++, envoltura Java	Vectorial	Propietario	Ninguno	Open Source
Xapian	C/C++	Probabilístico	Propietario	Limitado	Open Source
Lucene	Java	Vectorial	Propietario	Abierto y extendible	Open Source
MySQL FullText	C/C++ Java	Vectorial	Propietario ISAM	SQL	Open Source

Tabla 7 Herramienta RI

Muchas bibliotecas digitales han optado por usar Managing Gigabytes, debido a que es parte del sistema Greenstone; mientras que Xapian y Lucene fueron creados con el objetivo de indexar documentos de sitios de Internet. Aunque esto no excluye que cada herramienta pueda ser empleada en otro contexto.

La Figura 9 muestra la arquitectura de Hermes, específicamente de la instancia actual en U-DL-A. La debilidad de Hermes radica en que para poder aplicar los modelos se debe implementar un controlador (driver) que permita obtener información de las colecciones y de los términos que contienen (nuevamente haciendo referencia a frecuencias, pesos, etc) pero esto no es una tarea fácil ya que aunque se sabe de antemano el esquema de la base de datos, hay que desarrollar herramientas que llenen dichas tablas. Por otro lado no se había (hasta esta investigación) trabajado ninguna manera alterna de para almacenar dicha información.

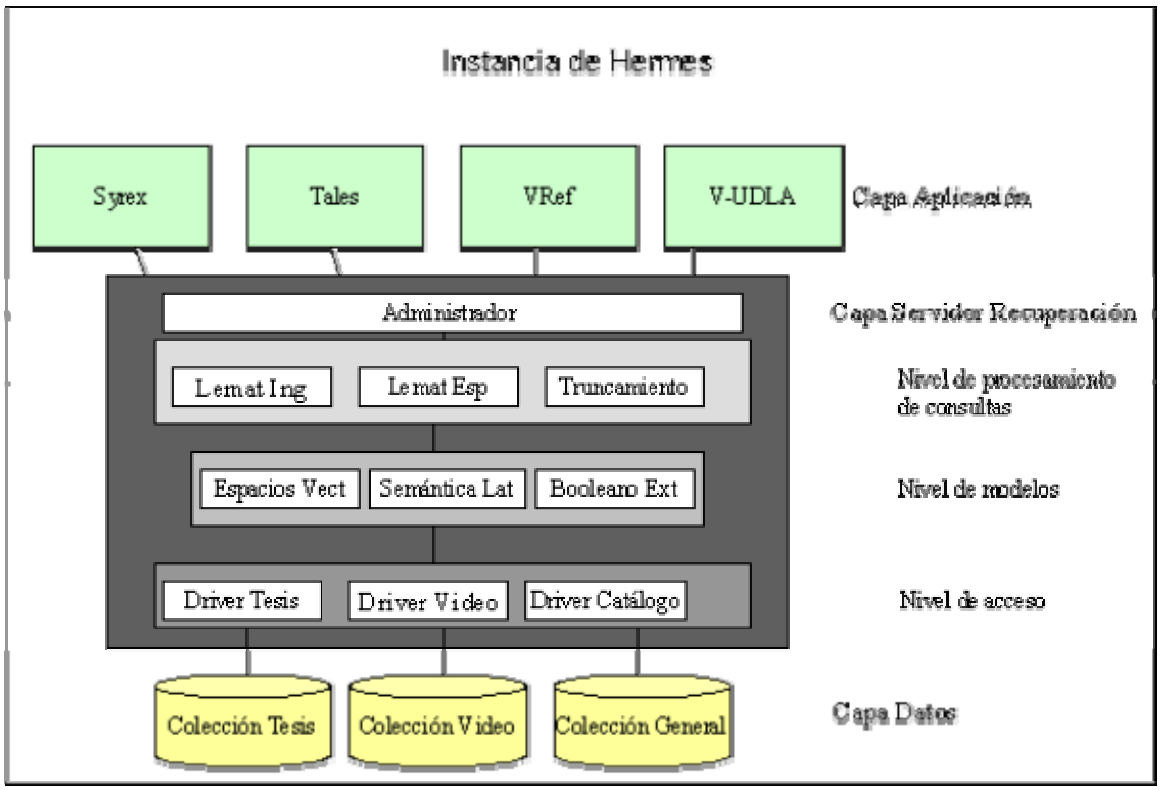


Figura 9 Arquitectura Hermes

5. ALMACENAMIENTO Y RECUPERACIÓN DE INFORMACIÓN EN DATOS SEMIESTRUCTURADOS

Es importante recordar que el máximo representante de los datos semiestructurados es XML, de manera que asociaremos estos datos con documentos XML.

5.1. PROCESAMIENTO DE DOCUMENTOS XML

Hablar de recuperación en XML es para muchos asociar un documento con sus respectivos metadatos. De manera que únicamente se mantienen algunos de los datos representativos del documento (ejemplo: la ficha bibliográfica) en alguno de los tipos de bases de datos mencionados anteriormente y el documento es almacenado en algún tipo de dispositivo de almacenamiento (Figura 10).

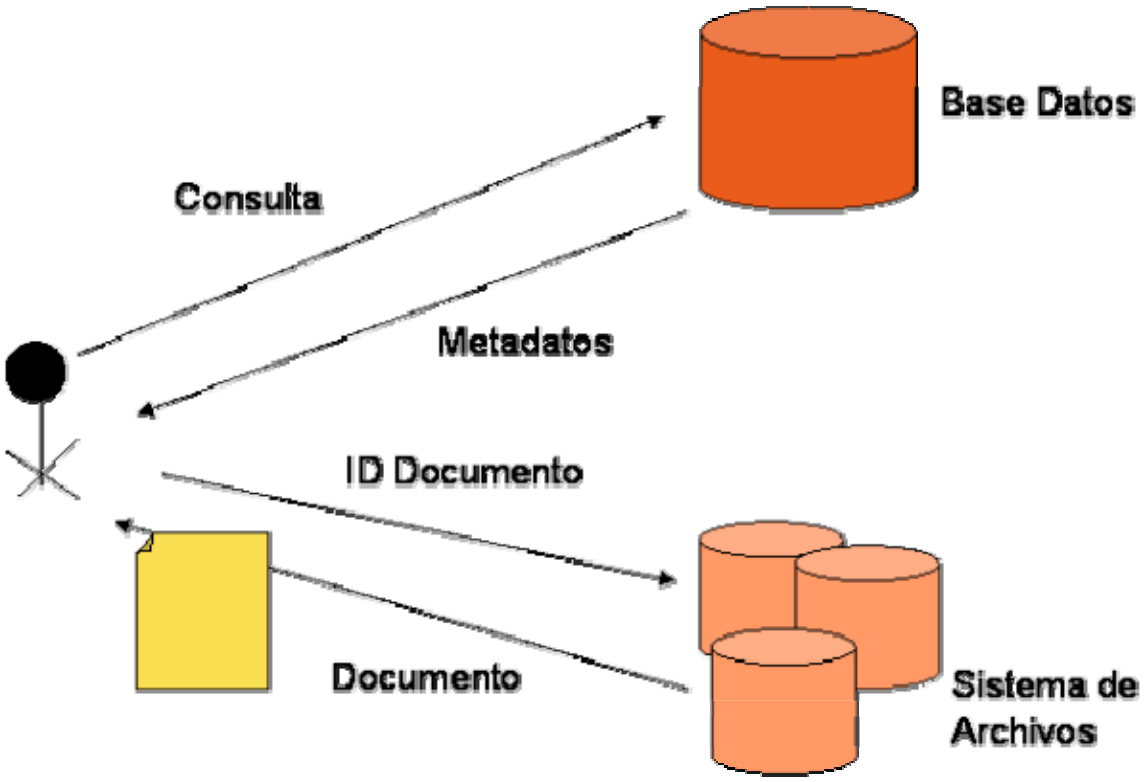


Figura 10 Sistema de almacenamiento XML tradicional

Sin embargo este esquema de almacenamiento y recuperación no aprovecha el contenido y mucho menos la estructura de los documentos.

Por otro lado las técnicas utilizadas para el análisis de documentos XML están basadas en 2 interfaces de aplicación SAX (Simple Api for XML) y DOM (Document Object Model) las cuales analizan el documento y son capaces de construir su árbol de objetos (elementos, atributos, etc) para poder realizar búsquedas o transformaciones en él. La Figura 11 muestra por ejemplo el procedimiento que realizan las hojas de estilo en XML para convertir un documento XML en uno HTML, analizando el documento y modificando partes específicas de un lenguaje al otro.

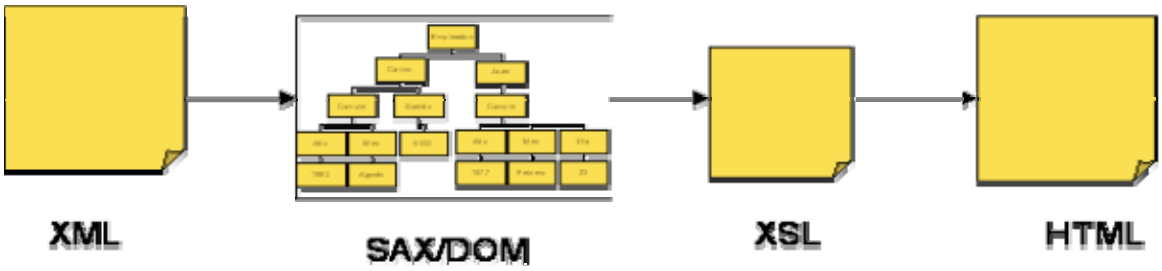


Figura 11 Formateado de un documento XML

Para realizar dichas búsquedas de los fragmentos a cambiar se emplean dentro de la hoja de estilo secciones de búsqueda, las cuales se basan en el lenguaje de consulta XPath, el cual como su nombre indica está vinculado con rutas de localización de partes del documento. El ejemplo de la Tabla 8 Búsqueda con XPath muestra una consulta en XPath que recupera el fragmento del documento (Tabla 4 Representación datos semiestructurados lenguaje marcado) que corresponde a toda la información del empleado con id=1.

/compania/empleado[@id='1']
<empleado id="1">   <nombre>Alfredo</nombre>   <extension>2666</extension>   <oficina>312</oficina>   <departamento>Ejecutivo</departamento>   <sueldo>Director</sueldo> <empleado>

Tabla 8 Búsqueda con XPath

Otro ejemplo se muestra en la Figura 10 donde se recuperan todos los nombres de los empleados de la compañía.

/compania/empleado/nombre
<nombre>Carlos</nombre> <nombre>Alfredo</nombre>

Tabla 9 Búsqueda con XPath

Esta búsqueda en XPath dentro del documento puede ser realizada por alguna herramienta como Xalan que es de grupo Apache o bien directamente alguno de los módulo integrados con la mayoría de los navegadores actuales. Dichas herramientas están basadas en SAX y DOM de manera que desde el punto de vista de recuperación de información este proceso resulta ineficiente.

Analicemos qué sucede cuando queremos realizar una búsqueda, por ejemplo la de obtener los datos de un empleado con un determinado ID. Se debe analizar y/o construir el árbol correspondiente y a partir de ahí buscar el nodo del ID deseado; si se desea hacer otra búsqueda en otro momento habría que volver a construir el árbol lo cual implica consumo de tiempo y probablemente de memoria si se trata de un documento extenso.

Por otro lado pensemos no en un documento sino en un conjunto de documentos (colección) se necesitarían recorrer todos los documentos, crear todos sus árboles y en cada uno de ellos realizar la búsqueda, el problema se complica aún más.

5.2. BASES DE DATOS XML

Es así como en los últimos meses ha surgido un nuevo concepto en bases de datos conocido como bases de datos XML (XMLDB), el cual define un modelo lógico de un documento XML y almacena y recupera documentos de acuerdo a ese modelo.

En otras palabras, las XMLDB son sistemas capaces de desglosar la información contenida en un documento XML y almacenarla de cierta forma que sea más fácil realizar búsquedas de dicha información.

Existen 2 tipos de XMLDB:

- XML Enabled Databases (Bases de datos habilitadas para XML): son aquellas que desglosan la información de un documento XML en su correspondiente esquema relacional o de objetos

- XML Native Databases (Bases de datos nativas de XML): son aquellas que respetan la estructura del documento, se pueden hacer consultas sobre dicha estructura y es posible recuperar el documento tal como fue insertado originalmente.

Hablando de datos semiestructurados donde si nos interesa la estructura del documento, entonces nos concentraremos con mayor interés en las XMLDB nativas.

Básicamente el concepto detrás de este tipo de bases de datos es la facilidad de representar la estructura de un documento XML en una modelo relacional. El esquema de la Figura 12, se basa en la especificación de un DTD (Document Type Definition) para documentos XML.

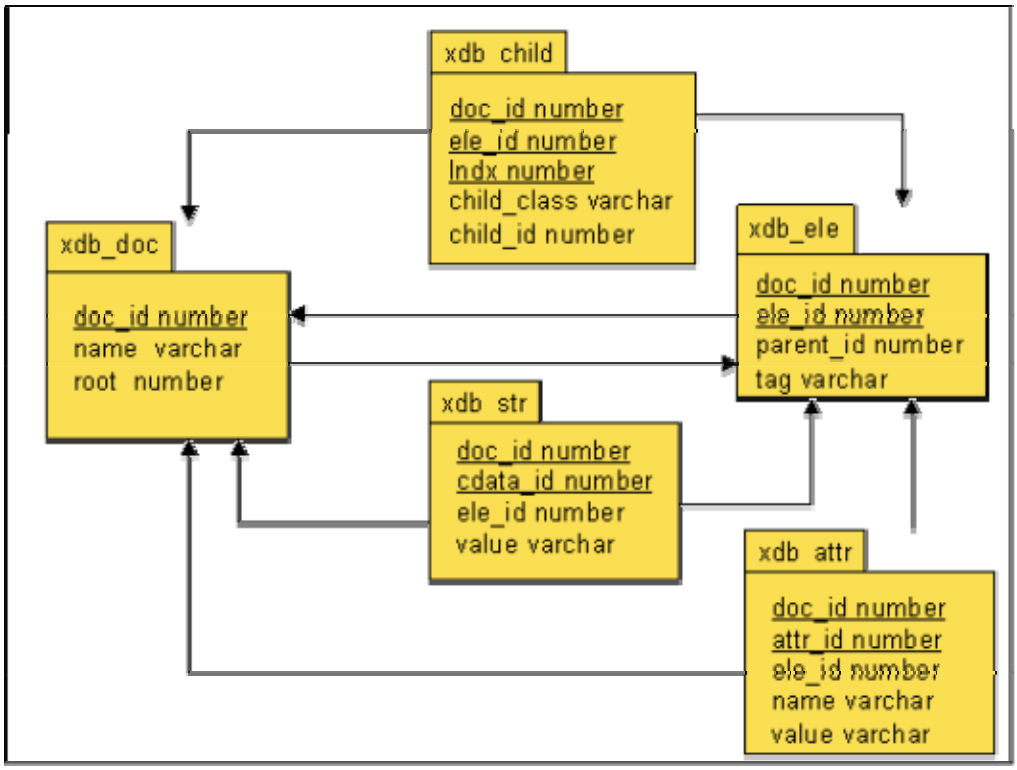
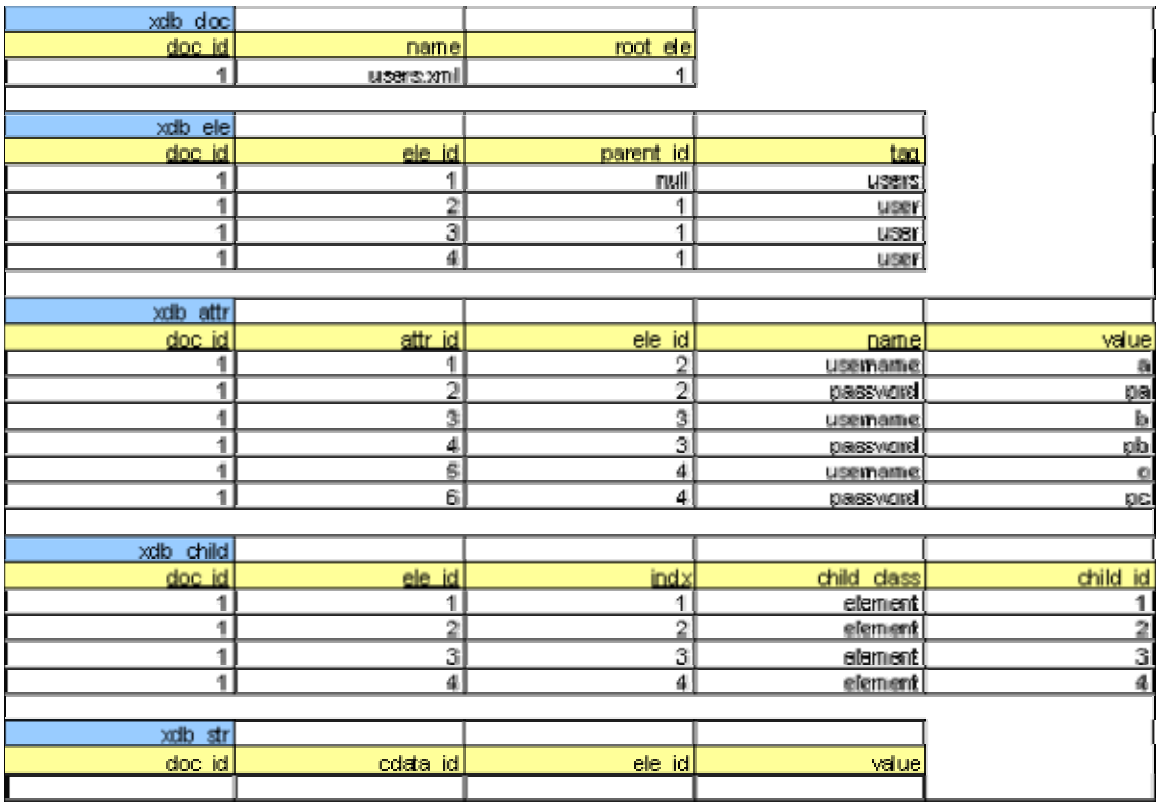


Figura 12 Esquema de almacenamiento en una XMLDB

En la Figura 13 se muestra el modelado que seguiría el código de la Tabla 10 Código XML de usuarios.

```
<users>
    <user username="a" password="pa"/>
    <user username="b" password="pb"/>
    <user username="c" password="pc"/>
</users>
```

Tabla 10 Código XML de usuarios



xdb_doc			
doc_id	name	root_ele	
1	users.xml	1	

xdb_ele			
doc_id	ele_id	parent_id	tag
1	1	null	users
1	2	1	user
1	3	1	user
1	4	1	user

xdb_attr				
doc_id	attr_id	ele_id	name	value
1	1	2	username	a
1	2	2	password	pa
1	3	3	username	b
1	4	3	password	pb
1	5	4	username	c
1	6	4	password	pc

xdb_child				
doc_id	ele_id	indx	child_class	child_id
1	1	1	element	1
1	2	2	element	2
1	3	3	element	3
1	4	4	element	4

xdb_str			
doc_id	cdata_id	ele_id	value

Figura 13 Ejemplo de datos en una XMLDB

Es importante hacer notar que en el ejemplo se nota claramente que el utilizar una XMLDB para datos que son en realidad estructurados no es muy conveniente ya que el modelado hace que se tenga una columna para todos los atributos, en este caso username y password.

La gran ventaja de las XMLDB radica en poder mantener la estructura de aquellos datos semiestructurados, de manera que utilizando XPath se pueden hacer consultas sobre determinados fragmentos no solo de 1 sino de muchos documentos sin tener que hacer el análisis de cada documento al momento de realizar las consultas.

6. REFERENCIAS

Birmingham, W. 1995. An Agent-Based Architecture for Digital Libraries. D-Lib Magazine Julio 1995, disponible en: http://www.dlib.org/dlib/July95/07birmingham.html

OAI. 2003. The Open Archives Initiative Protocol for Metadata Harvesting. Disponible en http://www.openarchives.org/

Proal. C. 2003. Almacenamiento y Recuperación de Información. Material del curso, disponible en: http://ict.udlap.mx/people/carlos

WebBase. The Stanford WebBase Project. Disponible en
http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/

Hull, D. 1996. Stemming Algorithms- A case study for detailed evaluation. Journal of the American Society of Information Science, disponible en:
http://citeseer.ist.psu.edu/509573.html