Capítulo 1

Estado del Arte

" El objetivo de todo sistema que tenga una relación con la búsqueda de la información, es el de ayudar a los usuarios a encontrar lo que buscan"

"G. Navarro and R.Baeza-Yates. Proc. ACM SIGIR' 95"

1.1 Introducción

La búsqueda de información se puede realizar a partir de una simple consulta o bien a través de la navegación, éste término en inglés es conocido como *"browsing"*. Significa efectuar una exploración libremente sobre una estructura dada, puede efectuarse de forma jerárquica o bajo la forma de una gráfica. Si la navegación tiene lugar sobre ligas de información, éstas serán denominadas ligas hipertexto. Estas pueden también ser explotadas no solo vía navegación si no también vía consulta directa. Pueden así mismo ser calificadas de hipermedia, cuando contienen diferentes fuentes de información de las que se encuentran en el texto (por ejemplo una gráfica, una imagen, un sonido, un video-clip, etc). Encontramos en la literatura de los sistemas hipertexto que la utilización de los términos *liga* o *documento* ha reemplazado al término

información. Estos términos conservan el mismo significado, un documento es una liga que puede representar ella misma un documento o una parte del documento.

Las técnicas de estructuración y de búsqueda de información son diversas, por consecuencia difícil de comparar y de clasificar. Ahora presentaremos de forma general los diversos tipos de búsqueda de información que podemos encontrar hoy en día, y de manera indirecta, los tipos de estructuración que éstos utilizan. (referirse al capítulo 2).

La gran cantidad de sistemas hipertexto nos permite buscar la información directamente en porciones de la red, tomando en cuenta solo su estructura. Una gran parte de los trabajos efectuados se apoya en SMBDR - Sistema Manejador de Bases de Datos de tipo Relacional, o su equivalente en inglés DBMS-R *"Relational Data Bases Manager System"* y recientemente orientados a objetos DBMSOO *"Data Bases Manager System Objet Oriented"*. Estos sistemas proporcionan lenguajes similares a los sistemas de lenguaje de consulta de tipo SQL "System Query Language" para la interrogación. Precisaremos en el capítulo 4 las características de los lenguajes de consulta y su aplicación, muy en particular para los documentos estructurados, MULTOS [Tha90], MAESTRO [Mac90], [Mac91], POQL [Chr96], LOREL [BRS+95] y Textriever [Bur91].

Con el fin de tener un panorama general de los modelos que han sido desarrollados sobre diferentes áreas en el campo de la búsqueda de información (bases de datos textuales, documentos estructurados, documentos semi-estructurados, multimedia e hipermedia), citamos los trabajos de Gonnet & Tompa [GT87] Gyssens, Paredaens & Van Gucht [GPV89], Tague, Salmien & McClellen [TSM91], Burkowski [Bur92], Navarro [Nav96] en lo que respecta al campo de las bases de datos textuales; Abiteboul, Cluet & Christophides [Acc+97], Christophides, Abiteboul Cluet & Scholl [Chr94], Güiting, Zicari, & Choy [Güiti84] en el campo de los documentos estructurados, Abiteboul, Quass, McHugh, Widom, Wiener [Aqmw+97], Bunaman, Davidson, Hillerbrand, Suciu [Bdhs96] en la interrogación de datos semi-estructurados, [Tha90] y

con la interrogación de documentos multimedia y finalmente [GS84] con los documentos hipermedia.

Otros modelos, como el modelo de Navarro [Nav96] permiten la definición de varias perspectivas, así como la consulta simultánea de éstas diferentes perspectivas. El lenguaje de consultas propuesto por Navarro está basado en la teoría de conjuntos, y proporciona la facilidad de utilizar operadores de tipo estructural de proximidad de una o varias jerarquías. Mas adelante, éste modelo utiliza un lenguaje de consultas que toma en cuenta los mecanismos de búsqueda sobre el contenido y la estructura.

1.2 Problemática

Los documentos electrónicos representan una clase importante de datos manipulables en diversos campos hoy en día: la edición electrónica, las bibliotecas automatizadas, la documentación, los sistemas médicos, etc. Sin embargo, ni los sistemas hipertexto ni los SBI - Sistemas de Búsqueda de Información, conocidos en inglés como IRS "*Information Retrieval Systems*"; proporcionan todas las funcionalidades necesarias para la manipulación de documentos, y su modelización, para definir lenguajes de consultas poderosos y flexibles, la concurrencia de acceso, etc.

Actualmente, no existe un método que permita construir un modelo de estructuración bastante flexible, es decir que integre consultas por contenido y por estructura. Las soluciones actuales se concentran en los problemas de expresividad o de eficacia. Las investigaciones en el campo de los SBI se abordan en dos áreas: la representación del contenido de los documentos (indexación) y el acceso a la información (interrogación) [Nav96].

Consideramos que la integración de técnicas de sistemas para la búsqueda de información: Sistemas Hipertextos (navegación) y los SMBD (estructura), Sistemas Manejadores de Bases de Datos, o mejor conocidos en inglés como DBMS *"Data Bases*

Manager System", es esencial. Es por ello, que la combinación de SMBD y de SBI asegurará una mejor administración de una base de documentos HTML, o de forma general, una mejor administración de una base de documentos estructurados.

Trabajos recientes [Brs92], [Chr94], [Bk94], [Bmn94], [Vab96] han mostrado que una integración de SMBDOO - Sistemas Manejadores de Base de Datos Orientados a Objetos, llamados también DBMSOO *"Data Bases Manager System Objet Oriented"* y de SBI dentro de un mismo sistema presentan ventajas remarcables con relación a los sistemas clásicos de SMBD.

1.3 Alcance

Nuestro estudio se enfoca particularmente en el modelo genérico propuesto por [Fou98]. Éste modelo para documentos estructurados se basa en las características generales de las relaciones de estructura sintáctica: (i) relación de composición, (ii) relación de secuencia y (iii) relación de referencia. Éste permite tratar datos textuales e imágenes fijas conforme a especificaciones de tipo estructural. También estamos interesados en el lenguaje de consultas declarativo de tipo OQL *"Objet Query Language"* [ODMG93] que propone interrogar documentos estructurados en relación a: a) su estructura y b) su contenido. Es decir, permite la utilización de operadores de tipo estructural y una navegación a través de la estructura del documento aún sin conocerla.

1.4 Objetivos de trabajo

El objetivo de nuestro estudio es el de explotar las propiedades del modelo genérico se basa en las características generales de las relaciones de estructura sintáctica (i) relación de composición, (ii) relación de secuencia y (iii) relación de referencia [Fou98], definidas para representar, almacenar y manipular documentos estructurados en un ambiente orientado a objetos [$O_2$97]. Uno de nuestros objetivos es el (i) de retomar

el concepto de alcance de los atributos (es decir, su cobertura y su dependencia), (ii) implementar un vector de tipo *"bitmap"* de dos niveles en conjunto con un archivo inverso (utilizando el concepto B-árbol, en inglés B-tree) con el fin de reducir el espacio de búsqueda. Buscamos de igual forma proporcionar nuevas facilidades de consulta para los documentos estructurados empleando la introducción de expresiones de caminos, término en inglés: *"Path Expressions"*, combinando de forma simple y homogénea las consultas sobre el contenido y sobre la estructura. Para cumplir con éste objetivo, utilizaremos la sintaxis del Lenguaje de consultas OQL [ODMG93], propuesto por el grupo ODMG (*Object Data Bases Management Group*) para responder a las exigencias de almacenamiento y de acceso a los documentos estructurados.

1.5 Organización de la tesis

El plan que seguiremos para la presentación de ésta tesis da inicio en el **Capítulo 2 titulado: – Sistemas de Búsqueda de Información y Modalidades de Búsqueda** - en el cual presentaremos los conceptos de base de un sistema de búsqueda de información, así como sus funcionalidades y en particular los diferentes modelos existentes para la búsqueda de información. (el modelo boleano, el modelo vectorial, el modelo probabilístico y por último el modelo lingüístico). También presentaremos el concepto de integración de un SMBD y de los SBI dentro de un mismo sistema, y para dar por terminado el capítulo presentaremos las diferentes modalidades de búsqueda (SMBD, SBI, Hipertexto e Hipermedia), haciendo énfasis en los SBI. Por otro lado, mostraremos las diferentes estructuras existentes utilizadas por los modelos, donde observaremos su uso en los SBI. Destacan entre éstas, la estructura de archivo plano, de archivo inverso, el concepto de uso de una firma. Y las arborescencias Patricia, término en inglés: *"Patricia PAT Trees"*.

A continuación, en el **Capítulo 3 titulado: – Normas de representación para los documentos estructurados**, hemos escogido dos normas que han sido definidas por la Organización Internacional de Estándares: ISO *"International Standard Organization"*

SGML [1] y ODA.[2]. Estas normas presentan la ventaja de ser fácilmente comprensibles para describir los tipos de elementos estructurales y las informaciones descriptivas de datos en forma de atributos hacia un nivel de tipo (por ejemplo: un tipo libro, un tipo periódico, tipo tesis, etc.) y por tipo de elemento (por ejemplo: tipo capítulo, sección, sub-sección, etc.). Aprovecharemos parte de ésta ventaja durante la fase de indexación estructural utilizada en la definición de nuestro modelo.

En seguida, presentamos el **Capítulo 4: – Sistemas para la consulta de documentos estructurados -.** Después de la presentación de las normas, describiremos los prototipos de acuerdo a dos enfoques diferentes: SBI - Sistemas de Búsqueda de Información y SMBD - Sistemas de Manipulación de Bases de Datos.

El Capítulo 5: – Modelización. Abordaremos la presentación de nuestro modelo que propone administrar documentos estructurados introduciendo los siguientes conceptos: i) la relación sintáctica, ii) el alcance de los atributos, que nos permite definir elementos que compartan las mismas propiedades, y iii) la utilización de un vector representado por ceros y unos (*bits)* en dos niveles, con el fin de reducir el espacio de búsqueda y finalmente iv) presentar la noción de expresión de camino *"Path Expression"* en nuestro modelo.

Capítulo 6 - Presentación del Prototipo myPDN *"my Personal Daily News"*, y describiremos las funcionalidades y componentes del prototipo myPDN, desarrollado para manipular documentos estructurados bajo una plataforma orientada a objetos: O_2.

Concluimos nuestro trabajo con la presentación de nuestras conclusiones y perspectivas ilustradas en el Capítulo 7.

---

1 SGML – Standard Generalized Markup Language

2 ODA Office Data Architecture

Capítulo 2

Sistemas de Búsqueda de Información

2.1 Introducción

Los SBI - Sistemas de Búsqueda de Información, también conocidos como IRS *"Information Retieval Systems"* surgieron de la necesidad de automatizar la manipulación y la búsqueda de grandes volúmenes de información, y hoy en día más utilizados en las empresas, en la administración pública, etc.

Los SBI son sistemas que almacenan, administran y manipulan un conjunto de documentos, de tal forma que permiten a los usuarios encontrar una respuesta a sus necesidades específicas de información [Nie90]. Son una herramienta que ayuda a los usuarios a encontrar los documentos que contengan la información sensata que responda a sus necesidades [Ouni96]. Finalmente, el usuario es el único que juzga la relevancia de los documentos encontrados por el sistema así como su utilidad, con relación a su necesidad personal.

2.2 Los diferentes tipos de búsqueda

Las técnicas de estructuración y de búsqueda de información son numerosas, es por ello que son difíciles de comparar y de clasificar: búsquedas secuenciales o indexadas, búsquedas sobre las ligas ya sea de tipo estático o virtual, búsquedas exactas o aproximadas.

a) búsqueda secuencial o indexada: para poder encontrar una información precisa en un conjunto de informaciones, la búsqueda puede efectuarse de una forma *secuencial* dentro de éste conjunto, y es el caso de las búsquedas de palabras o de expresiones regulares en el texto; la otra forma consiste en una indexación dentro de un conjunto de información, es decir, su descomposición en varios elementos y la indexación de cada uno de ellos y/o sus interelaciones. Un índex (o descriptor) de un elemento o de una relación es una representación de algunas de sus características bajo una forma explotable por el SBI. Los nombres, los tipos y los atributos son descriptores simples.

Las búsquedas secuenciales no efectúan un análisis léxico, sintáctico o semántico sobre cada información, imponen al usuario el conocimiento exacto de lo que busca, y no ofrecen ningún nivel de abstracción.

b) búsqueda sobre las ligas (estáticas o virtuales): una liga es estática si los elementos referenciados son fijos. Es virtual si la lista o la composición de los elementos referenciadiados es calculada en el momento de la activación de la liga de origen. Una liga virtual corresponde entonces a una consulta predefinida. Las ligas por palabras clave de [Conclin87] son un ejemplo de ligas virtuales: la activación de una liga por palabra clave nos llevará hacia todos los documentos que contengan esa palabra clave.

La búsqueda por consulta es adaptada si el usuario tiene una idea precisa de lo que el busca y conoce bien el lenguaje de indexación del sistema. En caso contrario, la búsqueda por navegación es más ventajosa por el hecho de reconocer mas fácilmente cualquier cosa de interés que la describa. Particularmente si las ligas y los elementos no son sistemáticamente del estilo de tipos, la navegación hipertexto conduce generalmente a fenómenos de desorientación y de sobrecarga cognitiva.

c) búsquedas exactas o aproximadas: el conjunto de elementos indexados susceptibles de búsqueda se llama cuerpo. Seguida del nombre y del tamaño de los elementos del cuerpo, y la precisión relativa de las indexaciones, la búsqueda de información puede tener dos objetivos sensiblemente diferentes i) encontrar los elementos de documentos que contengan la información lo mas cercana posible a la buscada. ii) encontrar si existen los elementos del documento que correspondan exactamente a un conocimiento exacto.

La búsqueda aproximada es típica de los sistemas de búsqueda documentaria, por el contrario la búsqueda exacta es mas bien utilizada para la adquisición de conocimientos, o de pequeños elementos de documentos indexados por un conocimiento.

2.3 Los SBIs y un SMBD

Los SBI y los SMBD son dos tipos de sistemas que manipulan conjuntos de datos. Estos consideran funciones para modelizar éstos datos: por una parte un ***Lenguaje de Descripción de Datos*** **(LDD), y por otra un** ***Lenguaje de Manipulación de Datos*** **(LMD)** para la manipulación y consulta. En cada uno de éstos sistemas, el usuario tiene acceso a los datos vía un lenguaje de consulta.

Sin embargo, el proceso puesto en juego durante una consulta es diferente. También los SMBD aplican un proceso de correspondencia estricto, en tanto que los SBI basan éste proceso sobre el concepto de relevancia de un documento en la consulta.

Investigaciones recientes [Vab96] han demostrado que una integración del SMBD y del SBI en un mismo sistema presenta ventajas remarcables. La figura 2.1 presenta un tipo de integración: el SMBD que administra los datos (de almacenamiento, de los métodos de acceso, ...) y las extensiones del SBI (métodos de indexación y de consulta) son integrados a nivel de la aplicación y construidos por encima del LMD y del SMBD.

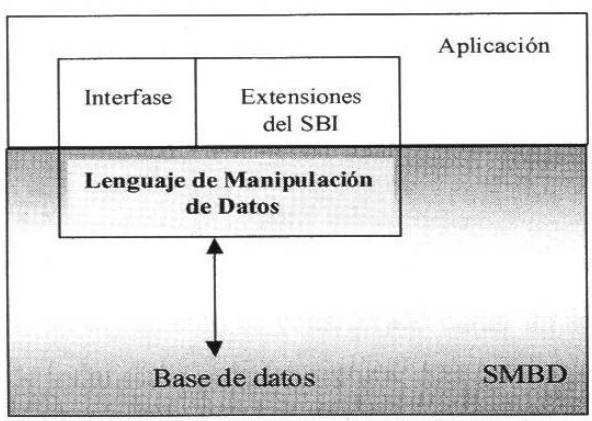

Figura 2.1 Integración del SBI - SMBD

Un ejemplo de integración es el prototipo PRIME [Ber88], [Mul93], [Mul95] y [Mec97]. En éste prototipo se administran un conjunto de expedientes médicos compuestos de datos de tipo texto y de imágenes radiográficas.

2.4 Funcionalidades de un SBI

Las funcionalidades de un SBI permiten extender el lenguaje de consultas para realizar consultas sobre el contenido semántico de los documentos.

Un SBI realiza dos funciones de manera clásica: **indexación** y **consulta.**

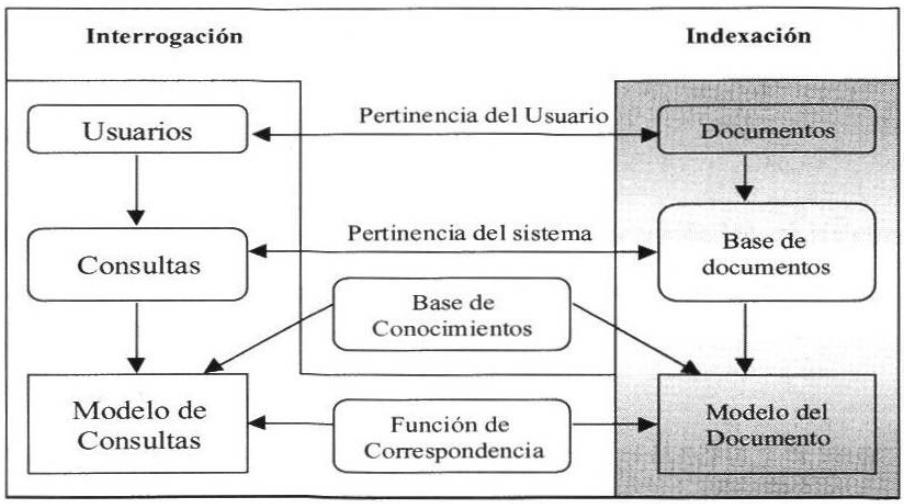

Figura 2.2 *Funciones de un Sistema de Búsqueda de Información*

2.4.1 Indexación

Esta fase es específica de los SBI, permite extraer los conceptos de los documentos de la base. En un SBI, un documento es considerado como un soporte que conduce la información. La fase de indexación permite capturar ésta información y representarla de acuerdo a un modelo: **El modelo de documento.** La información capturada corresponde al contenido semántico del documento, y la representación de éste contenido semántico es llamada un **índex de documento.**

La indexación consiste en expresar según el modelo de representación del contenido de los documentos (lenguaje de indexación) los elementos juzgados a priori como los más representativos de la información manejada por los documentos [Salton83].

Renunciar a ésta etapa de indexación implicaría probablemente una búsqueda **en la totalidad texto** o *"full text"*, que desgraciadamente exige demasiado tiempo para encontrar la información en un volumen considerable de documentos.

La elección de un lenguaje de consultas depende: (i) de la media manipulada, (ii) de la naturaleza de la indexación (manual o automática), y (iii) de la precisión.

2.4.2 Interrogación

La interrogación permite al usuario expresar sus necesidades, el sistema las pone en correspondencia con el cuerpo del texto con el fin de extraer lo más relevante [Salton83].

La fase de consulta es la fase de interacción entre el sistema y el usuario. En ésta fase, el usuario expresa su necesidad de información con la ayuda de un lenguaje de consultas (puede ser un lenguaje natural, o también un conjunto de palabras clave). Si bien el sistema va encargarse de traducir, ésta traducción se realizará de acuerdo al modelo de consultas. Este proceso proporciona una consulta interna. Después de ésta fase de comprensión de la consulta, el sistema calcula la correspondencia entre la consulta interna y cada índex de los documentos. Este cálculo, establecido vía la función de correspondencia, obtiene por resultado una lista ordenada de los documentos de la base. El primer documento de ésta lista es el que es considerado por el sistema como el más relevante, es decir el que responde mejor a la consulta, según el criterio del sistema. El último documento es el que es considerado por el sistema como el menos relevante. Este concepto de relevancia, es específico de los SBI y se basa en la proximidad entre las necesidades expresadas y los resultados proporcionados por el sistema.

Distinguimos dos tipos de relevancia:

La relevancia del usuario: cuando un documento propuesto por el sistema es juzgado relevante por el usuario.

La relevancia del sistema: cuando un documento es propuesto como relevante por el sistema dada una consulta (función de correspondencia).

2.5 Modelos de búsqueda de información

Un modelo de SBI define el modelo de documentos y de las consultas, así como la metodología de evaluación de la correspondencia entre la consulta y los documentos [Nie90] (c.f. figura 2.3).

En la siguiente sección presentamos algunos modelos clásicos existentes.

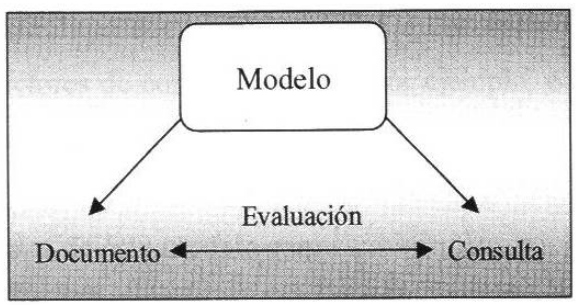

Figura 2.3 *Representación del modelo de un SBI*

2.5.1 El modelo boleano

El modelo boleano tradicional se basa en una indexación manual o automática binaria (un término indexado o no, un documento), los documentos son representados por conjunciones de términos (palabras claves) independientes. Las consultas son representadas por conjunciones, disyunciones y negaciones de términos. La función de correspondencia es la implicación lógica: un documento es juzgado relevante por una consulta si la representación del documento implica la representación de la consulta. La ventaja principal de éste modelo es su simplicidad, ésto permite en la práctica una implementación eficaz sobre el plano de desempeños cuantitativos. Por el contrario, en el plano cualitativo, éste modelo se encuentra limitado a causa de la evaluación binaria de la función de correspondecia (un documento es relevante, o no relevante), no existe entonces la posibilidad de clasificar las respuestas. El modelo se encuentra también limitado a gran escala por la pobreza del lenguaje de indexación (palabras clave).

2.5.2 El modelo vectorial

En el modelo vectorial [Salton83], los documentos y las consultas son representadas por vectores en un espacio a t dimensión (t dado el cardinal del conjunto de los términos de indexación) entonces los elementos son puntos que traducen la importancia del término en la consulta o en el documento.

Una medida de similitud documento-consulta o documento-documento puede ser definida a partir de vectores de correspondencia (por ejemplo, el coseno del ángulo de dos vectores en el espacio a t dimensiones).

Las ventajas principales de éste modelo son:

- ✓ Contrariamente al modelo boleano, éste no es binario, y el riesgo de encontrar una respuesta vacía es mínimo. El modelo no impone una estructuración lógica de la consulta.
- ✓ Toma en cuenta la importancia relativa de los términos de indexación en las expresiones del contenido de los documentos.
- ✓ Conlleva un método de evaluación de relevancia del sistema, y ofrece la posibilidad de ordenar la respuesta, por ejemplo siguiendo la relevancia en orden decreciente.
- ✓ Permite la reformulación automática de las consultas a partir de juicios de relevancia efectuados por el usuario, en inglés: "*relevance feedback* ".

El inconveniente principal de éste modelo

- ✗ Es la pobreza del lenguaje de indexación (palabras claves), donde la expresividad de los conceptos vía términos de indexación se mantiene débil.

2.5.3 El modelo probabilista

El modelo probabilista está basado en la idea de encontrar los documentos que tienen una fuerte probabilidad de ser relevantes en una consulta y que tienen al mismo tiempo una probabilidad mínima de ser no relevantes [Rijs79]. En la práctica, no existe un medio para obtener el valor exacto de éstas probabilidades. Para estimar entonces éstos valores, ya sea si calculamos el número de apariciones de los términos de indexación y suponiendo que éstos son descritos por una distribución de probabilidad [Salton83], o ya sea si efectuamos estadísticas sobre una muestra de cuerpos y repercutimos los resultados sobre todos los del cuerpo [Halin89].

La ventaja principal de éste modelo es que está adaptado para considerar las dependencias entre los términos [Rijs79]. Pero al contrario, es difícil poner en práctica un tal modelo y los valores estimados para las probabilidades de relevancia son a menudo poco fiables.

2.5.4 El modelo lingüístico

Los modelos lingüísticos están basados sobre un mismo tratamiento en lenguaje natural de documentos y de consultas para extraer y representar, de manera automática, su contenido semántico. La función de correspondencia tiene entonces por objetivo el comparar la semántica de consultas con la de los documentos.

2.6 Modos de Búsqueda

Los métodos para acceder a las informaciones contenidas en una base constituida de documentos estructurados son:

- Los SMBD
- Los SBI o Sistemas de Búsqueda de Información
- Los sistemas hipermedia e Hipertexto

2.6.1 Los SMBD

Estos sistemas permiten la manipulación de grandes cantidades de datos asegurando su persistencia, su seguridad y así mismo el compartirlos entre múltiples usuarios y/o múltiples computadoras. Además, éstos permiten administrar y manipular fácilmente los datos estructurados por lenguajes de consultas declarativos, como SQL [ISO89] para el modelo relacional y OQL [ODMG93] por el modelo orientado a objetos.

El principal inconveniente:

- ✗ Las consultas exigen un conocimiento exacto de la estructura de la base.

Las ventajas son:

- ✓ Estas son simples y rápidas de escribir.
- ✓ Existe una correspondencia exacta entre los valores de los atributos especificados en la consulta y los que están almacenados en la base. Por consecuente, siempre es posible de determinar, en una base dada, si una consulta admite o no respuestas.

2.6.2 Los SBI

La técnica comúnmente utilizada por los SBI tradicionales es la organización de fondos documentales como un conjunto de documentos a los que se les asocia palabras clave. Estas palabras claves representan el lenguaje de indexación para todo el contenido del texto. Estos son administrados automáticamente o manualmente por un análisis del contenido de los documentos [Loe94].

Una consulta realizada por el usuario es a su tiempo analizada por el sistema, transformada en palabras clave y enseguida comparada con el índice proveniente del contenido del texto. Esta comparación, llamada "*matching*", permite encontrar los documentos que van a la par de la consulta y eliminar los que son menos adecuados. (cf. figura 2.4). Además, el resultado es evaluado y ordenado, en relación a la relevancia de los documentos retenidos, llamada "*relevance ranking* ".

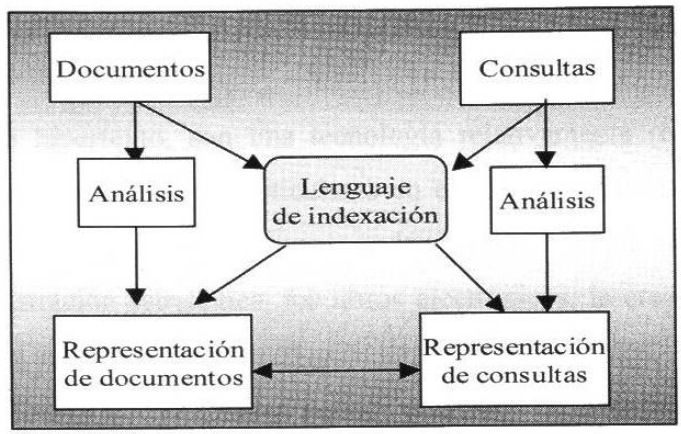

Fig. 2.4 *Principio de búsqueda textual en las SRI*

El concepto SBI ha sido introducido para explotar, en un primer plano, los documentos no estructurados. Una gran parte de las búsquedas en éste campo ha sido dedicada a la evaluación de diversos mecanismos de similitud entre una consulta y los fondos documentarios, en términos de la tasa de precisión, es decir el porcentaje de los documentos relevantes efectivamente encontrados, efectivamente relevantes).

2.6.3 Los Sistemas Hipertexto

La mayor parte de los sistemas informáticos tratan los textos y la información de forma lineal, es decir carácter por carácter. Este método no corresponde a la forma en la que el hombre trata la información. De hecho, la mayor parte de los individuos razonan de forma no lineal y efectúan asociaciones entre fragmentos de información aparentemente sin relación entre ellos [Nel80].

La invención del concepto hipertexto es generalmente atribuida a V. Bush, cuando publicó en 1945 su célebre artículo *"As We May Think"* [Bush45]. Más tarde, en 1960, T. Nelson definió la noción de hipertexto como la asociación del texto en lenguaje natural, y de las capacidades del computador para establecer las ligas interactivas y

visualizaciones dinámicas de un texto no lineal. Que no puede ser impreso convenientemente sobre una página tradicional [Nel67].

Los sistemas hipertexto, son una tecnología relativamente reciente, no existen todavía gran número aplicaciones particulares en el campo industrial [Niel90]. Por lo tanto, su aparición en campos diversos como en la ayuda en toma de decisiones [Beg88], [Ble91], la administración estratégica, los libros electrónicos, la creación de ambientes de programación, o aún todavía en la adquisición de conocimientos, justifican el interés que se les otorga y hacen que se esperen rápidos progresos en la materia.

2.6.4 Ejemplos de sistemas hipertexto

- **NoteCards**

  Un sistema clásico [Halaz88] es un ambiente hipertexto usado generalmente para estaciones de trabajo Xerox y ha sido desarrollado en Xerox PARC (Palo Alto Research Center) hacia 1985.

- **KMS**

  Otro sistema: KMS, "Knowledge Management System " es un sistema de visualización basado en menús, y fue desarrollado en 1972 en la Universidad de Carnegie Mellon.

- **Neptune**

  Neptune es un sistema destinado esencialmente a las aplicaciones llamadas HAM: " Hypertext Abstract Machine ", de hecho, éste concepto corresponde a un modelo genérico hipertexto que proporciona operaciones para crear, modificar y acceder a los nodos y ligas del documento, y ofrece un acceso rápido a cualquier versión del documento.

- **Hipercard**

  Hipercard es un sistema desarrollado por Bill Atkinson, disponible para la plataforma Macintosh desde 1987, éste es el sistema más comercializado en el mundo. Cuenta con numerosas facetas atractivas, entre éstas la integración

de una herramienta de diseño. Integra un lenguaje de programación HyperTalk, de fácil utilización que permite desarrollar rápidamente aplicaciones [Apple88].

2.6.5 Los Sistemas Hipermedia

Una tendencia importante en la evolución de las bases de datos es la del desarrollo de aplicaciones de almacenamiento y de recuperación de documentos multimedia más efectivas. Entre éstas aplicaciones encontramos los documentos hipermedia que resultan de la combinación de documentos multimedia e hipertexto.

El hipermedia y el hipertexto se van transformando en enfoques muy populares para la manipulación de un documento multimedia de manera no lineal [Conklin87]. En éstos sistemas, los datos se organizan en forma de redes de textos o datos multimedia.

Citemos como un ejemplo de aplicación del standard SGML: el lenguaje Hytime que provee medios para expresar la información hypermedia. Hytime es un lenguaje para describir y construir no solamente documentos estructurados pero también estructuras hiperliga. Permite la integración de los dos dominios de tratamiento de documentos: los documentos estructurados y los documentos hipermedia.

2.6.6 La diferencia entre SRI y SGBD

En lo que concierne a los SRI, la necesidad del usuario se expresa a través de una búsqueda que considere el contenido semántico de los documentos buscados. Los SRI se diferencian entonces de los SGBD puesto que éstos últimos analizan el contenido de los documentos que administran, y no se contentan con comparar de forma estricta los términos de las búsquedas con los elementos de la base.

2.7 Enfoques de modelos conceptuales

En la búsqueda de información, un modelo conceptual es un enfoque general de los sistemas de búsqueda de información. Muchas arquitecturas fueron propuestas para los modelos conceptuales de búsqueda de información como lo muestra el cuadro de la sección 2.8 [Frakes92]. Faloustos (1985) nos presenta tres enfoques fundamentales:

(i) búsqueda de una cadena de caracteres en el texto
(ii) búsqueda utilizando archivos en orden inverso
(iii) búsqueda utilizando una firma

Sin embargo, Belkin y Croft (1987) los clasifican de otra manera, dividiéndolos en técnicas de búsqueda: correspondecia exacta e inexacta.

En la primera categoría, encontramos técnicas de búsqueda de textos y de búsqueda boleana, mientras que en la segunda disponemos de los modelos probabilista, vectorial y de conjuntos, etc. El problema de éstos modelos consiste en que la categoría no es recíprocamente exclusiva, un sistema podría contener los aspectos de otros modelos.

La mayoría de los sistemas de búsqueda de información basados en el modelo boleano nos permite buscar dentro de grandes colecciones de documentos, puesto que los documentos están representados por un conjunto de palabras claves almacenadas en un archivo en orden inverso. Al contrario, en un sistema de búsqueda basado en el modelo de cadenas de caracteres *"string search"* a partir de un filtro o modelo de búsqueda *(pattern search)*, que se utiliza para buscar en pequeñas colecciones de documentos, éstos filtros son cadenas de caracteres o de palabras que se encuentran en cualquier parte del texto en cuestión, o bien al inicio de lugares particulares en el texto.

2.8 Clasificación de los sistemas de búsqueda de información según su modelo conceptual

Modelo Conceptual	Estructura de archivos	Tipo de Consulta	Operaciones	Operaciones con los documentos
Boleano	Archivo Plano	"Feed-Back"		Análisis
Boleano extendido	Archivo Inverso	Análisis	Ponderada	Visualización
Probabilista	Archivo de Firma	Boleano	Diccionario	Conjuntos
Búsqueda sobre una cadena	"Pat tree" (árboles patricia)	"cluster"	Stop list ó Listas de Parada †	"Rank"
Vectorial	"Hashing"	Troncatura	Ordenación	

Cuadro 2.1 *Clasificación de los SRI según el modelo conceptual.*

2.8.1 Estructuras de archivos planos

El aspecto fundamental de la concepción de sistemas de búsqueda de información es sin duda el tipo de estructura a ser usado para la estructura inferior de la base de datos del documento. Como lo vimos en el cuadro anterior, las estructuras de los archivos utilizadas por los sistemas de búsqueda de información son los archivos planos, los archivos en orden inverso, la firma, estrucuras en foma de árbol y los gráficos.

2.8.2 Enfoque de los archivos plano

Uno o más documentos se almacenan en un archivo, en general bajo la forma de un texto *"asci"* o EBCDIC *"Extended binary Coded Decimal Interchange"*. La búsqueda bajo éste enfoque se realiza en realidad a través de un filtro de correspondecia de la cadena *"pattern matching"*.

† Es un antidiccionario que contiene palabras "vacías" o sin significado (artículos, preposiciones, locuciones, adjetivos demostrativos, adjetivos posesivos, etc)

El problema principal al utilizar éste enfoque es el de encontrar todos los elementos o el primer elemento en una cadena de caracteres en el texto, donde ésta y el texto son cadenas de caracteres por encima del alfabeto. El interés es el de encontrar todas las correspondencias. Para buscar una cadena de caracteres de un largo *m* en un texto de largo *n* (dónde *n*>*m*), el tiempo de búsqueda es O(n).

Los algoritmos más significativos para la correspondencia de una cadena de caracteres dada son: el algoritmo Clásico de Knu th-Morris-Pratt (1977), con otras variantes del algoritmo de Boyer-Moore (1977), Shift o de Baeza-Yates, y de Gonnet (1989) el cual es probabilista.

2.8.2.1 El algoritmo básico.- es un método simplista que realiza una correspondencia en una cadena dada. Su objetivo, es el de tratar de encontrar una correspondencia de cualquier cadena de caracteres de un largo *m* en el texto.

```
   naivesearch( text, n, pat, m )/*search pat[1 .. m] in
text[1..n] */
       Char text[], pat [];
       Int n, m;
       {      int i, j, k, lim;
              lim = n-m+1;
              for (i =1; i<= lim; i++) /* cherche */
              { k=i;
                     for( j=1; j<=m && text[k] == pat[j], j++) k++;
                     if (j > m) Report_match_position (i-j+1);
       }}
```

2.8.3 Enfoque del archivo en sentido inverso

La mayor parte de los sistemas de búsqueda de información se apoya en los archivos en sentido inverso (o índex) obtenidos gracias a la indexación de los documentos ya sea de forma manual o automática.

Un archivo en sentido inverso es una lista de palabras clave y de señaladores asociados a los documentos en los cuales aparecen. Es decir una lista seleccionada (o en

sentido inverso) de atributos, donde cada atributo posee un vínculo que representa una referencia a los documentos que contienen la palabra. Ver el ejemplo en la figura 2.5.

Podemos recalcar que el inconveniente de éste enfoque es el tamaño del archivo en sentido inverso, entre 10% y 100% (aveces superior a la talla del texto) así como la actualización del archivo cada vez que se produce algún cambio.

Inconvenientes

- ✗ La colección (vocabulario) se limita a una colección de atributos, los cuales serán indexados. Las palabras que no se encuentren en la lista no serán indexadas, y por consecuente no podrán ser buscadas.
- ✗ Por motivos de tamaño, las palabras consideradas como palabras vacías (preposiciones, locuciones, etc) no serán indexadas.

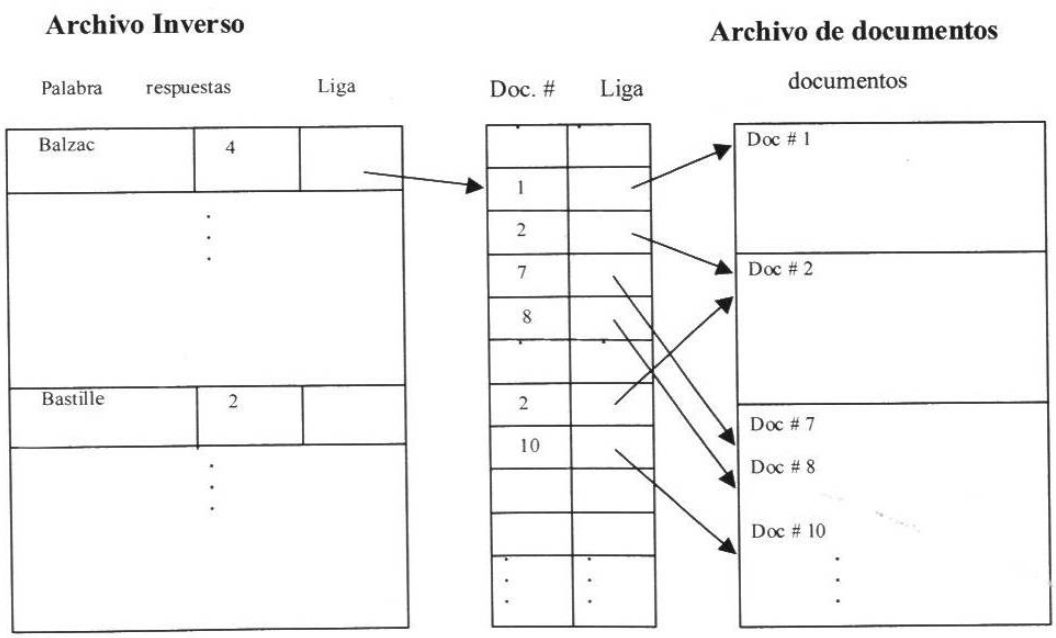

Figura 2.5 *Ejemplo de un archivo inverso*

2.8.4 Enfoque de la firma

Existen varias maneras de construir una firma utilizando el método clásico por ejemplo, los documentos se dividen en grupos lógicos, y luego cada uno obtiene un número fijo con un posicionamiento diferente: (es decir palabras desconocidas o *"non-stoplist".*) Cada palabra en un grupo es fusionada o*"hashed"* para obtener una firma. Un proceso de unión se aplica al grupo de firmas para obtener la firma del documento. La búsqueda se hace a través de una consulta que compare las firmas de una petición con las firmas de un documento.

El método de las firmas permite igualmente la aceleración de la búsqueda de datos textuales representados bajo la forma de un código binario, llamado firma.

2.9 Conclusión

Los modelos de búsqueda de información se basan en una correspondecia entre la consulta (o su representación) y las representaciones de los documentos. El mecanismo de búsqueda identifica los documentos susceptibles de responder a la consulta.

La respuesta es constituida por los documentos que tienen una representación que corresponde de forma exacta a la consulta (modelo boleano) o documentos que responden mas o menos a la consulta. En éste caso el sistema asocia un grado de supuesta relevancia a los documentos restituidos. Este grado corresponde a un grado de parecido entre la representación de la búsqueda y el documento (modelo vectorial) o a una probabilidad de relevancia (modelo probabilista).

Un inconveniente de éstos modelos de búsqueda de informaciones reside en el hecho de que éstos consideran los términos de indexación de manera independiente.

Capítulo 3

Normas de representación de los documentos estructurados

3.1 Introducción

La difusión de los sistemas de tratamiento de textos facilitó la producción de los documentos electrónicos pero no su intercambio. En efecto, la mayor parte de éstos sistemas introducen en un documento ciertos tipos de comandos bajo la forma de etiquetas, con el fin de determinar el aspecto físico del documento. Después de la concepción de la norma SGML el dominio de la edición debería enfrentar éste problema así como los problemas ocasionados por la compatibilidad del hardware.

Al comenzar los años ochenta, con el fin de responder a éstas múltiples preocupaciones, ciertos grupos de trabajo del ISO se interesaron en los trabajos realizados por algunos investigadores norte americanos, especialmente por WilliamTunncliffe de la GCA - "Grafic Communications association" y por Charles F. Goldfarb de la IBM. Tunnicliffe de la GCA trabajó en la definición de una codificación genérica, es decir una codificación independiente de todo sistema. Goldfarb es el creador de un lenguaje de etiqueteo generalizado GML "Generalized Markup Language", que describe la estructura de un documento de manera que ésta sea independiente de todo tipo de sistema y que permita el tratamiento del documento vía varios tipos de aplicaciones que utilicen editores de textos, formateadores o sistemas de búsqueda de información. El conjunto de éstos trabajos, reutilizados en un contexto de normalización,

concluyendo en la definición del SGML que fue adoptado por el ISO en 1986. [SGML86], [Bryan88] y [SGML90] ISO /IEC 8879.

La norma SGML fue rápidamente adoptada por diferentes asociaciones de editores de varios países, que vieron en ésta norma los medios necesarios para mejorar la calidad de los documentos, para racionalizar y automatizar sus tratamientos, para reducir el costo de publicación y abrir nuevas perspectivas. Sin embargo se debe comprender que el lenguaje SGML fue concebido con la intención de tratar únicamente una parte de los problemas ligados al mundo de la edición. SGML permite describir y manipular o intercambiar documentos bajo su forma revisable, es decir bajo una forma en la cual solo se representa la estructura lógica del documento. Esta norma se dedica a la etapa de creación de documentos.

Además, el SGML es un lenguaje de alto nivel, que asegura la sólida independencia entre los aspectos estructurales de un documento y los tratamientos que se le aplican. Puede entonces ser utilizado como un lenguaje de definición de bases de datos de textos.

3.1.1 Principio

El principio de SGML es el de describir la estructura lógica de un documento. Para esto, SGML utiliza un sistema de etiqueteo lógico que delimita los elementos de la estructura (los capítulos, las secciones, los párrafos, etc) y les asocia atributos. Ciertamente, existen dos tipos de etiquetas: una de inicio y una de fin para cada elemento. Estas contienen cada una el nombre del elemento al cual hacen referencia. Esta demarcación, que engloba simplemente la información agregada al texto, es esencialmente declarativa. En efecto, los tratamientos a aplicarse al documento no aparecen en el mismo: por consecuente varias aplicaciones pueden ser ejecutadas a partir de una misma descripción SGML de un documento. SGML utiliza una sintaxis de referencia concreta para definir la forma en la cual un documento puede ser codificado.

La sintaxis concreta precisa, por ejemplo, cual juego de caracteres puede ser utilizado, y de qué manera las etiquetas son representadas.

SGML utiliza también una sintaxis abstracta que permite definir tipos de documentos bajo una estructura de datos llamada DTD "Document Type Definition". El concepto de **tipo de documento** permite reagrupar documentos que posean una misma estructura genérica así como verificar si un documento dado es conforme a su tipo. Una DTD es de hecho una descripción genérica de demarcación que describe la estructura lógica de un documento. Indica los tipos de elementos que pueden ser encontrados en un documento (por ejemplo, los tipos de capítulos o de secciones) así como sus atributos y el orden en el cual pueden aparecer. Los nombres de éstos elementos ("capítulo" o "sección") se utilizan para generar etiquetas. Por ejemplo, un elemento capítulo será clasificado por una etiqueta de inicio **<capítulo>** y la etiqueta final **</capítulo>**. Llegamos a la conclusión de que SGML es un "meta-lenguaje" que permite generar lenguajes de "etiqueteo" para varios tipos de documentos.

Un documento SGML, para ser tratado por cualquier aplicación, debe en primera instancia, ser analizado por un analizador (o parser en inglés †). Un analizador SGML tiene un rol bien definido por la norma. A partir de la DTD del documento y del documento él mismo, el analizador realiza las acciones siguientes:

- Verificar la adecuación del documento a los principios generales del documento a los principios de la norma SGML. En particular, verificar la sintaxis de la DTD.
- Verificar la coherencia entre las declaraciones de "etiqueteo" al nivel de la DTD y el "etiqueteo" efectivo del documento.
- Completar el "etiqueteo" implícito. SGML abarca en efecto varias técnicas para reducir el número de etiquetas a ser introducido. El analizador se encarga entonces de reestablecer las etiquetas que fueron omitidas: todos los elementos

† Analizador gramatical.

son entonces abarcados por sus etiquetas de inicio y de fin. De la misma forma, se encarga de reemplazar en los documentos los llamados de entidades.

El analizador juega un rol esencial en el ciclo de producción SGML de un documento puesto que genera un documento apto para ser tratado automáticamente por otros programas.

3.2 El modelo SGML

3.2.1 Los documentos SGML: las estructuras específicas

Acabamos de señalar que el estándar SGML no nos da mas que un punto de vista lógico del documento. Otro aspecto importante de ésta norma es la naturaleza de la descripción SGML de un documento. Un documento SGML se presenta bajo una forma secuencial en la cual la estructura y el contenido están unidos en un mismo flujo de datos. De hecho, las etiquetas utilizadas para separar la estructura del contenido de un documento son cadenas de caracteres. La figura 3.1 nos muestra un ejemplo simple de un documento SGML.

Podemos observar que en el ejemplo de la figura 3.1, las etiquetas pueden ser utilizadas para hacer aparecer la arquitectura global del documento, es decir su división en capítulos, secciones y párrafos. Pero pueden así mismo ayudar a estructurar un elemento lógico de bajo nivel, como un párrafo. Si buscamos descomponer un párrafo, podemos llegar a definir elementos lógicos particulares que están mas ligados al aspecto físico de un documento que a su arquitectura. Sobre el mismo ejemplo, podemos citar el caso del elemento lógico *ital* que abarca la cadena de caracteres "un modelo conceptual". El elemento lógico *ital* indica que la cadena abarcada será puesta en itálica en el momento de su interpretación por el programa de compaginación.

```
<Libro>
 <Titulo>Los documentos electronicos</Titulo>
 <Autor>Abraham ALVAREZ</Autor>
 <Capitulo><Titulo_Capitulo>Introduccion</Titulo_Capitulo></Capitulo>
 <Seccion> ... </Section>
   ...
 <Capitulo><Titulo_Capitulo>modelos de búsqueda en el WWW</Titulo_Capitulo>
 <Parrafo>En la búsqueda de información <ital>el modelo conceptual</ital> es un
concepto general de sistemas de información. Diversas arquitecturas han sido
propuestas por los modelos conceptuales de búsqueda de información como lo
muestra la tabla ... </Parrafo>
 ...
</Libro>
```

Figura 3.1 *Ejemplo de un documento SGML*

Podemos recalcar que es posible acceder con mayor facilidad a la estructura lógica de un documento a partir de su representación arborescente. La figura 3.2 es un ejemplo de éste tipo de representación, fue tomado como ejemplo el documento presentado en la figura 3.1.

A través de ésta descripción, podemos distinguir fácilmente, por un lado, lo que forma parte de la arquitectura global de un documento y por otro lado lo que forma parte de la arquitectura fina del documento (es decir el extremo de la arborescencia ligada al contenido y a su presentación)

Un documento SGML posee una estructura lógica específica que debe ser conforme a la estructura genérica definida en su DTD. Una DTD define las entidades y los tipos de los elementos que pueden ser utilizados para describir un documento (cf. figura 3.3). Una entidad es un nombre dado a una parte de la DTD o de un documento. Este nombre puede ser utilizado como una referencia a otro documento SGML.

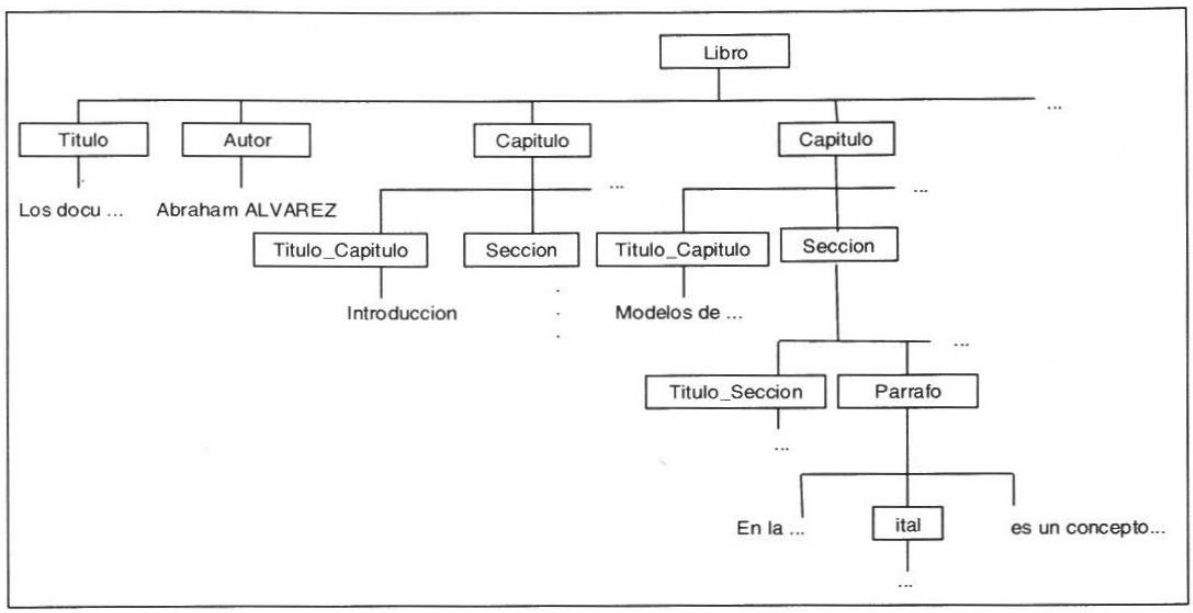

Figura 3.2 *Representación arborecente*

3.2.2 La Definición de Tipo de Documento: Las estructuras genéricas

SGML utiliza un formalismo de tipo gramatical para describir la estructura lógica genérica de cada elemento SGML. Una DTD está compuesta en una gran parte por una colección de reglas que describen los elementos SGML y sus atributos así como entidades. Los conceptos de elemento, atributo y entidad son aspectos importantes de la norma que presentaremos.

3.2.3 Los elementos SGML

Las reglas que describen los elementos SGML conllevan en la parte izquierda el nombre del elemento SGML, que designa en realidad el tipo del elemento, y en la parte derecha su *modelo de contenido*. Un modelo de contenido puede expresar la estructura de composición de un elemento SGML y al mismo tiempo la naturaleza de la información que le será asociada. La declaración de un elemento tiene la forma siguiente:

<!ELEMENT nom-elem (modelo de contenido)>.

El nombre del elemento servirá a la creación de las etiquetas que lo enmarcaran en el texto : <nom-elem> ... </nom-elem>.

En función del modelo de contenido, distinguimos :

- Los elementos SGML que no se pueden descomponer, contienen cadenas de caracteres que contienen eventualmente los llamados de entidades que hacen referencia, por ejemplo, a datos externos como elementos gráficos, imágenes, etc.
- Los elementos SGML que están compuestos por otros elementos SGML pueden a su vez, ser descompuestos eventualmente.
- Los elementos SGML que no se pueden descomponer son los elementos terminales de la gramática. Su modelo de contenido indica al analizador la naturaleza de las informaciones que pueden contener. Existen para ello tres nombres reservados:

"#PCDATA" (*Parsed Character DATA*) : el analizador o "parser" espera localizar caracteres.

"#RCDATA" (*Replaceable Character DATA*) : el analizador o "parser" espera localizar caracteres o un llamado a una entidad candidata a substituir.

"#CDATA" (*Character DATA*) : el analizador o "parser" no aplica ningún tratamiento a los caracteres leídos.

Los elementos SGML compuestos son elementos no terminales de la gramática. Su modelo de contenido define su estructura de composición genérica: precisa el tipo de sus componentes así como su orden de aparición, es decir que precisa los lazos legales autorizados entre ellos.

" ? " el elemento es opcional
" * " el elemento es opcional y puede repetirse
" + " el elemento puede repetirse

" & " entre elementos que pueden aparecer en cualquier orden. Simboliza la agregación.

" , " entre elementos ordenados. Simboliza la secuencia

" | " entre elementos presentados como alternativas. Simboliza la opción exclusiva.

Por ejemplo, la regla "un capítulo está compuesto por un título seguido por cero o varios párrafos seguidos de una o varias secciones" se expresa de la forma siguiente :

<!ELEMENT capítulo (titulocap, parrafo*, sección+)>

SGML autoriza la definición de elementos SGML con una estructura de composición vacía. Para el elemento *anote* que designa un llamado de nota, este caso da lugar a la declaración siguiente:

<ELEMENT nota (#EMPTY)>.

3.2.4 Los atributos SGML

Un atributo se utiliza para calificar, identificar o referenciar un elemento. Estos atributos aumentan, de hecho, la información que se puede ser eliminada de la estructura jerárquica del documento. La declaración de los atributos es de la forma:

<!ATTLIST nom-elem-atr (tipo o valores posibles) valor predefinido ...>.

Para cada atributo, ésta declaración incluye:

(a) el nombre del atributo

(b) el tipo del atributo

(c) el valor predefinido del atributo.

El tipo del atributo.- puede ser numérico, por ejemplo (NUMBER), alfabético (CDATA), alfanumérico (NAME, NUTOKEN, etc.) o especializado (por ejemplo, de

tipo ID sí su función es el de identificar un elemento o IDREF para referenciar un elemento).

El valor predefinido del atributo.- Permite saber si el valor del atributo es una constante (#FIXED), si debe ser obligatoriamente especificada por el utilizador (#REQUIRED) o por el sistema (#IMPLIED). Además la palabra clave #CURRENT indica que el valor por omisión utilizado es el último en haber sido declarado, en el caso en el que ningún valor haya sido especificado en el documento.

Damos aquí el ejemplo de la declaración del atributo *versión,* destinado al elemento *tesis* para indicar la evolución de la redacción:

<!ATTLIST Tesis, número de version #REQUIERED> En el texto, éste atributo aparece al nivel de la "etiqueta" de apertura del elemento tesis:

Los atributos tienen funciones variadas:

(a) Podrán contener informaciones y podrán ser ligadas ulteriormente a instrucciones de formateo. Luego de una operación de formateo, éstas instrucciones afectarán la presentación del elemento al cual se asociaron. Por ejemplo, el elemento figura puede poseer atributos que indiquen su tamaño.

(b) Podrán ser utilizados para controlar el tipo de datos importantes de un documento. Evitan de ésta manera los errores debidos a la introducción de datos. Un dato factual de un documento puede de ésta forma ser el valor de un atributo.

(c) Podrán servir a establecer ligas referenciales entre elementos.

Podemos ilustrar éste último caso por el llamado de nota en un documento que hace referencia a una nota ya mencionada. SGML permite expresar la liga "hace

referencia a " y "está referenciado por". Así todo elemento *nota* se verá asociado al atributo *id* tipo ID.

En el ejemplo <nota id=35>, la nota es asociada al valor 35 que la identifica independientemente de su posición en el texto. El atributo *id* puede ser declarado IMPLIED y será entonces generado por el sistema. Por otro lado, el atributo *refid* de tipo IDREF será adjuntado a todo elemento *anote*. Para el ejemplo anterior tendremos <anote refid=35> donde el atributo *refid* es REQUIRED. Este llamado de nota hace referencia a la nota cuyo usuario es el 35. Una nota podrá ser referenciada por uno o varios llamados de nota de ésta manera.

3.2.5 Las entidades

Pueden ser vistas como constantes nombradas o macro estructuras. Distinguimos dos tipos de entidades: las entidades generales y las entidades parámetrales.

Las entidades generales son designadas por símbolos y se declaran de la forma siguiente:

< !ENTITY nombre-entidad " texto de la entidad " >

Aparece en el texto de un documento bajo la forma " &nombre-entidad; ". Este llamado de entidad será reconocido por el analizador SGML que lo reemplazará por la cadena apropiada. Por ejemplo, si un documento contiene frecuentemente la cadena "base de datos orientada a objetos", podemos declarar la entidad siguiente:

< !ENTITY sgbdoo "bases de datos orientadas a objetos " >

ésta entidad será adoptada en el documento bajo la forma " sgbdoo".

El contenido de una entidad puede ser almacenado en un archivo, ésta entidad es llamada externa. Como lo mencionamos anteriormente ésto permite incluir los datos que no sean SGML como por ejemplo figuras. Esta entidad será declarada de la forma siguiente:

< !ENTITY nombre-entidad SYSTEM " camino absoluto " >

donde " camino absoluto " lleva al archivo en el cual esta almacenado el dato.

Las entidades parametrales son macros utilizados para aclarar la lectura de la DTD. Son declaradas de la forma siguiente:

< !ENTITY %nombre-entidad " texto-de-reemplazo " >

Esta entidad aparece en la DTD bajo la forma " %nombre-entidad ". Por ejemplo, si un párrafo está formado por cadenas de caracteres que deben ser puestas en valor (por ejemplo, en negrilla o en itálica), podemos hacer las afirmaciones siguientes :

```
<!ENTITY %val              "gras ital">
<!ELEMENT (gras |ital)     (#PCDATA)>
<!ELEMENT parrafo          (#PCDATA | %val)+>
```

La déclaración del elemento párrafo es de hecho equivalente a :

(#PCDATA | gras| ital)+>

3.2.6 Ejemplo simple de DTD

Como lo muestra el ejemplo de la figura 3.3, las descripciones de tipos son legibles incluso si conciernen documentos sólidamente estructurados.

```
<!DOCTYPE   Libro          [
<!ELEMENT   Libro          - - (Titulo_Libro,  Capitulo+)>
<!ELEMENT   Titulo_Libro   - - (#PCDATA)>
<!ELEMENT   Capitulo       - - (Titulo_Capitulo, Parrafo*,Seccion+)>
<!ATTLIST   Capitulo
            Autor NMTOKEN                #REQUIRED
            Fecha NMTOKEN                #REQUIRED
            Num_Capitulo NUMTOKEN        #REQUIRED
<!ELEMENT   Titulo_Capitulo - -  (#PCDATA)>
<!ELEMENT   Parrafo        - o  (#PCDATA | %val)+>
<!ELEMENT   Seccion        - o  (Titulo_Seccion, Parrafo+)>
. . . ] >
```

Figura 3.3 *Ejemplo de una DTD de un documento de tipo libro*

Según la definición a continuación, expresada en la sintaxis abstracta de SGML, un libro esta compuesto por un título, por un autor y por una secuencia de capítulos. El título y el autor se constituyen en base a una simple cadena de caracteres. Cada capítulo contiene un título, eventualmente seguido por una secuencia de párrafos. El símbolo " o " indica que las etiquetas pueden ser omitidas mientras que símbolo " - " hace que la presencia de ciertas etiquetas en el documento sean obligatorias. La primera columna concierne la etiqueta de apertura mientras que la segunda concierne la etiqueta de cierre. A partir de ésta DTD y de la sintaxis concreta de SGML, es posible etiquetar todo documento de tipo libro. Observamos un ejemplo de un tal etiqueteo en la figura 3.3.

Un documento SGML se divide en elementos que tienen la siguiente forma:

$$<n\ a_1 = v_1 \ldots a_k = v_k>\ c\ </n>$$

<n> y <n/>	Son etiquetas de principio y de final del elemento. Las etiquetas se utilizan para marcar la estructura global de un documento (capítulos, secciones, párrafos, notas de pie de página, etc).
n	Es el nombre de su tipo.
C	Es el contenido de n. El contenido de un elemento puede ser una cadena de caracteres, un elemento, o una secuencia de elementos o de cadenas de caracteres.
$a_1 = v_1$	Es un par atributo-valor. Los atributos son utilizados para calificar un elemento independientemente de su contenido (identificación, ligas referenciales, indicaciones de formateo, etc).

Durante la creación de un documento, se otorgan dos tipos de información: el contenido y la estructura. En los esquemas de etiqueteo SGML, la estructura de un documento se determina gracias a las etiquetas.

Las etiquetas se utilizan para marcar la estructura global de un documento (por ej. capítulos, secciones, sub-secciones, párrafos, notas de pie de página, etc.) lo que permite estructurar un documento de forma jerárquica, como un árbol ; el nivel superior es por ejemplo el libro, que esta dividido en capítulos, etc.

La estructura genérica de un documento se describe en una DTD. La DTD define las entidades y los tipos de elementos que pueden ser utilizados para describir un documento. Una entidad es un nombre dado a una parte de una DTD o de un documento. Este nombre puede ser utilizado como una referencia a otro documento SGML.

- - Las dos etiquetas son obligatorias
- o La etiqueta de inicio es obligatoria. La etiqueta de fin es opcional.
- o - La etiqueta de inicio es opcional. La etiqueta de fin es obligatoria.
- o o Las dos etiquetas son opcionales.

La sección siguiente define el contenido del elemento (PCDATA).

Un documento estructurado puede ser visto como una arborescencia de elementos de un tipo característico conllevando ligas transversales así como descriptores de elementos conforme al modelo [Fou98].

Varios lenguajes de representación de documentos han sido desarrollados entre los cuales ODA [Ass91] « *Office Document Architechture* » y SGML « *Standard Generalized Markup Language* ».

El modelo ODA hace la distinción entre la estructura lógica y la estructura física de un documento. La idea se define en el hecho de que el documento pueda ser considerado bajo dos puntos de vista diferentes : lógico y físico como lo podemos observar en la figura 3.4. Un ejemplo claro que se basa en éste punto de vista es el de MULTOS [Tha90].

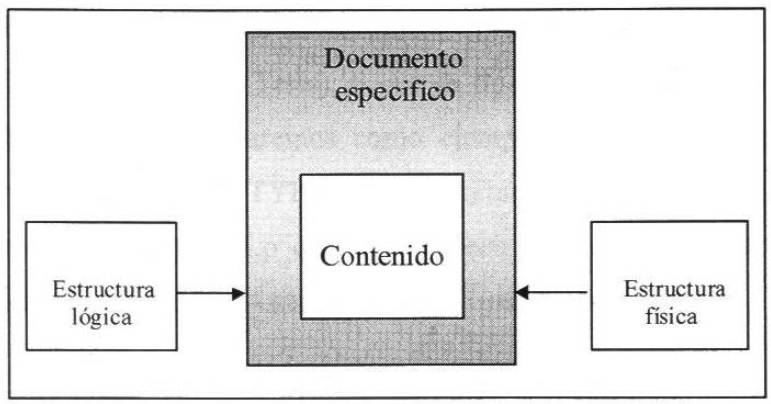

Figura 3.4 *Estructura lógica y física de un documento*

3.2.7 La estructura Lógica

Esta, describe la organización jerárquica de los datos del documento. Por ejemplo un documento dado se compone por un título, por una o varias secciones, y éstas mismas se componen por un título, y por una o varias sub-secciones.

3.2.8 La estructura física

Esta define una organización externa de los datos que componen el documento, es decir una presentación del documento. Esta estructura depende del su medio ambiente, por ejemplo del formato de la hoja (Letter, Legal, A4, etc.).

3.3 Conclusión

De la misma manera en la que fue mencionado al principio del capítulo, SGML es un lenguaje de alto nivel que nos permite definir la estructura lógica de los documentos. Esta técnica permite establecer un modo de acceso a documentos complejos utilizando las relaciones de estructura que unen los componentes de éstos documentos, es decir, la organización sintáctica. (cf. capítulo 6).

Además, el interés de las normas como SGML, HyTime y ODA son el permitir la **fácil descripción de los tipos de elementos estructurales y las informaciones**

descriptivas dadas bajo la forma de atributos al nivel de tipos (por ejemplo un tipo libro, un tipo Periódico, un tipo Tesis), o por su tipo de elemento (tipo Capítulo, tipo Sección,...). Para ilustrarlo tomaremos como ejemplo la figura 3.3 que describe un documento de **tipo Libro** [!DOCTYPE Libro] con un título (tipo **Titulo_libro**) y uno o varios capítulos (**Capítulo**) ... Lo que quiere decir que el tipo estructural **Capítulo** posee tres atributos [!ATTLIST] **Autor, Fecha y Num_Capítulo.**

A partir de lo cual, estos atributos permiten caracterizar los elementos de estructura de tipo capítulo. Para cada capítulo de un documento que se conforma a esta DTD, conoceremos los o el autor, la fecha de creación y el numero del capítulo en todo el documento.

Capítulo 4

Sistemas para la interrogación de documentos estructurados

4.1 Maestro

El sistema – *"Management Environment for Structured Text Retrieval and Organization"* desarrollado por la universidad de Queen's en Canadá [Mac90], [Mac91] es un medio ambiente integrado para la administración y la búsqueda de documentos estructurados. El interés de este sistema es la obtención de una representación interna de la estructura lógica de los documentos (es decir el árbol sintáctico), independientemente de su contenido con el fin de interrogar los documentos utilizando un lenguaje de consultas de alto nivel acopladas por un SRI. Además, el sistema permite efectuar actualizaciones de los documentos en línea. La carga de los documentos en la base y su representación interna están basadas en un traductor SGML. La estructura de los documentos es definida por la DTD con una anotación del tipo SGML.

4.1.1 El lenguaje de consultas MAESTRO

El lenguaje de consultas de Maestro posee un árbol sintáctico que se parece al SQL, siendo bastante influenciado por el lenguaje de manipulación de las matrices heterogéneas. La semántica del lenguaje esta orientada hacia la navegación en un árbol de un tipo característico que presenta la estructura lógica de un documento. Una o varias condiciones boleanas pueden ser aplicadas sobre cualquier elemento de la estructura lógica y /o al contenido de los documentos. El resultado de una consulta es siempre un conjunto de referencias a los fragmentos de documentos encontrados que el usuario puede denominar y usar.

MAESTRO autoriza dos categorías de consultas: una sobre el contenido y la otra sobre la estructura. En la primera categoría de consulta podemos utilizar tres tipos de operaciones algebraicas:

Operador	Función
Gets	Permite recuperar referencias a los documentos o a sus fragmentos satisfaciendo los criterios de una seleccción
Where	Expresa la selección
In	Verfica si el operador izquierdo es miembro de la lista representada par el operador derecha.
DocList	Es el conjunto de las referencias a los documentos encontrados
From	Precisa la búsqueda de un tipo de documentos
. .	Indica que la búsqueda de una expresión se encontrará en el contenido
Document	La palabra reservada "document " es utilisada para indicar al sistema que la búsqueda concierne a todos los documentos

Cuadro 4.1 Operadores sobre el contenido - Sistema MAESTRO

Ejemplo 1 : Obtener todos los documentos que contengan la palabra GRENOBLE y la palabra INFORMATICA .

DocList gets document where (GRENOBLE & INFORMATICA in document)

Ejemplo 2 : Obtener las tesis que contengan la expresión DOCUMENTO VIRTUAL o la palabra SGML

DocList gets document from Tesis where ((documento..virtual | SGML in document)

Podemos recalcar que el operador **from** restringe la búsqueda a un tipo preciso de documentos, como por ejemplo: Reportes o Tesis. En el ejemplo que damos utilizamos las Tesis. En la segunda categoría de consulta, ilustraremos como podemos acceder a los fragmentos de documentos en cualquier nivel de la estructura. Este tipo de consultas permite esencialmente, el navegar a través de la estructura jerárquica de los documentos.

Maestro ofrece un potente juego de operadores estructurales a excepción de los operadores para la negación de las relaciones de conjunción, y de los ancestros o descendientes directos de los elementos.

Operador	Función
Of	Permite el acceso a los decendientes de un elemento en la estructura lógica de un documento
First et Last	Obtiene la primiera o la última referencia de la estructura de un tipo de documento específico (tesis, reporte, documento, etc.) respectivamente.
.document	Busca la referencia inversa hacia el elemento padre

Cuadro 4.2 Operadores sobre la estructura - Sistema MAESTRO

Ejemplo 3: Obtener los títulos de la primera tesis en la base.

TitleList gets title of first document from Tesis

La declaración exacta de los caminos de acceso a los elementos buscados en una consulta es facultativa. Basta con declarar el nombre de la colección de los documentos.

Ejemplo 4 : Obtener el título general de las tesis de la base.

TitleList gets title of AbstractList.document from Tesis

En ésta consulta **`document`** es una referencia inversa hacia el elemento padre de un resúmen, es decir una tesis contenida en la lista de referencia **`AbstractList`**. Obtenemos el título general de la tesis o de las tesis.

4.1.2 Consultas sobre el contenido y sobre la estructura de los documentos

Operador	Función
Subsectn	Permite el acceso a las sub-secciones del documento
having ...	Restringe el contexto estructural de la evaluación de una consulta
Abstract	Permite el acceso al resúmen del document
Chapter	Para especificar el acceso a los capítulos del documento

Cuadro 4.3 Operadores sobre la estructura y sobre el contenido - MAESTRO

Ejemplo 5 : Obtener las sub-secciones de las tesis que contengan figuras cuya explicación (légende) contenga la palabra virtual

SubSectnList gets subsectn having figure where ("virtual" in caption)

En éste caso, podemos obtener sub-secciones con o sin figuras, el operador **`having`** permite seleccionar las sub-secciones que contengan figuras cuya explicación contenga la palabra virtual. Cuando varios caminos intervienen en una consulta, el operador **`having`** restringe el contexto estructural de evaluación de la consulta.

Ejemplo 6: **Obtener los capítulos de las tesis cuyo título de una sub-seccion contenga la palabra "Kosovo" y cuyo resúmen contenga la palabra "applet".**

ChapterList gets chapter having subsectn where ("Kosovo" in the title)
Tesis where ("applet" in abstract)

El operador **`having`** permite aquí la aplicación de la primera condición unicamente sobre los títulos de las secciones, **`(having subsectn)`** y no sobre los títulos de los capítulos.

4.1.3 Consultas sobre los atributos y los caminos de acceso

Este tipo de consultas permite interrogar los atributos (estilo SGML), e identificar un camino de acceso a la estructura lógica de los documentos de la base.

Ejemplo 7 : Dar las tesis en estado finalizado.

TesisList gets document from Tesis where status = "final"

Ejemplo 8 : Dar las tesis que contengan en todas sus secciones al menos tres sub-secciones

TesisList gets document having all section having any 3 subsectn.

4.2 Multos

El lenguaje de consultas MULTOS desarrollado en el cuadro de un proyecto ESPRIT [Tha90] permite la interrogación de los documentos multimedia, incluyendo las funcionalidades de un SBI. El lenguaje de consultas esta basado en un modelo conceptual de documentos que permite a los usuarios la búsqueda de informaciones del fondo documental por su estructura conceptual y por su contenido. MULTOS posee una arquitectura "cliente-servidor " bastante flexible y extensible par la administración de los documentos de tipo informático en un medio ambiente distribuido. El núcleo del sistema es un servidor de los documentos estructurados compuesto de cuatro módulos.

1. **El administrador de tipos** mantiene un catálogo con las estructuras conceptuales de los documentos y su organización en terminos de relaciones de inclusión (relación de tipo IS-A †).
2. **El administrador de clasificación** para los documentos importados a MULTOS del exterior (por ejemplo por un scanner).
3. **El administrador de colecciones** de documentos heterogéneos. Una colección es representada como un tipo particular de documentos (con un nombre, una fecha de creación, etc), el sistema provee los mecanismos necesarios de localisación de los documentos a través de una consulta o de la navegación.
4. **El intérprete de consultas** efectúa el analisis sintáxico asi como la descomposición y la optimización de las búsquedas.

† relación subtipo / supertipo, llamada también generalización.

Por encima de éstos módulos se encuentra el traductor de la estructura, que transforma un documento estructurado que posea una forma de intercambio (ODIF) conforme al standard ODA, en una forma interna reconocida por el sistema de almacenamiento. Este modulo provee todas las funcionalidades para el almacenamiento y la búsqueda de documentos estructurados bajo la forma ODIF así como diferentes métodos de acceso a los diferentes tipos de componentes de los documentos (por ejemplo, fecha cadena de caracteres, texto, etc.). Finalmente la persistencia de los documentos es garantizada por el sistema que integra técnicas de almacenamiento sobre un soporte magnético u óptico.

Multos utiliza la norma ODA como un standard de creación y de presentación de documentos, y ofrece todas las funcionalidades para el almacenamiento y la búsqueda de documentos estructurados. ODA - *"Office Document Architechture"* [ISO88], corresponde a la norma ISO 8613.

El lenguaje de consultas de MULTOS posee una sintaxis muy cercana a SQL [ISO89]. Dos clases de componentes de documentos son accesibles: los componentes activos y los componentes pasivos. Los primeros engloban los enteros, las cadenas de caracteres, los textos y las imágenes. Los segundos agrupan los gráficos y los sonidos. Un usuario puede expresar una consulta que tenga condiciones de selección basadas ya sea en el contenido de los documentos o en la existencia de sus componentes conceptuales. En el primer caso, las búsquedas se basan únicamente en los componentes activos. El resultado de una consulta es el conjunto de identificadores de documentos que satisfacen las condiciones de selección.

La característica principal de éste lenguaje es su capacidad para definir las condiciones de los componentes de los documentos buscados, sin precisar el tipo exacto de los documentos.

4.2.1 Sintaxis de las consultas en MULTOS

Una consulta tiene la forma siguiente:

FIND DOCUMENTS [VERSION <FIRST | LAST | ALL]

[SCOPE <colección de documentos>]

[TYPE <tipos conceptuales de documentos>]

[WHERE <condiciones sobre los atributos y los componentes conceptuales>

< conditiones sobre el contenido textual>

< conditiones sobre el contenido de las imágenes>]

WITH < composant | [type]

Operador	Función
Version	Esta cláusula permite declarar la versión de los documentos interrogados
Scope	Declara la coleción de documentos buscados
Type	Uno o varios tipos conceptuales pueden ser declarados
Condition	Define el contenido textual o las imágenes de los documentos
=, <, >, between	
Like	Para las cadenas de caracteres
Contains	Para el contenido textual de los documentos, éste operador permite también una búsqueda de palabras por proximidad
Is, in	Para formular las condiciones sobre atributos multivalores
With	Permite expresar las condiciones en cuanto a la estructura conceptual de los documentos

Cuadro 4.4 Lista de los operadores del lenguaje MULTOS

Ejemplo 9 : Dar los documentos que contengan la palabra "KOSOVO" y la palabra "SBI"

FIND DOCUMENTS

SCOPE Tesis

WHERE TEXT CONTAINS "KOSOVO" AND TEXT CONTAINS "SBI"

Ejemplo 10 : Dar las tesis cuyo estado está finalizado

```
FIND DOCUMENTS
SCOPE Tesis
WHERE DOCUMENT.status = "final"
```

4.3 Lorel

LOREL - *"Lightweight Object Repository Language"* [BRS+95] fue desarrollado por la Universidad Stanford en los EEUU, en el cuadro del proyecto TSIMMIS *"The Standford-IBM Manager of Multiple Information Sources"*. El objetivo TSMMIS era el de proveer un medio ambiente y las herramientas apropiados para la integración y el acceso a los datos que provengan de variadas fuentes heterogéneas.

El lenguaje LOREL está basado en un modelo objeto simple llamado OEM *"Object Exchange Model"*, particularmente adecuado para la representación de datos semi estructurados. Los objetos en OEM disponen de un *identificador* único, de una etiqueta, y de un valor. Una etiqueta es una cadena que sirve para dar un nombre (lógico) a los objetos, y puede ser utilizada como punto de ingreso a la base. El valor de los objetos atómicos es de tipo entero real, imagen; etc. y el de los objetos complejos es de tipo conjunto de identificadores de objetos.

4.3.1. El lenguaje de consultas LOREL

Lorel es un lenguaje con un nivel de tipos de bajo nivel, que ofrece las facilidades de búsqueda dentro de colecciones de datos heterogéneas, y un acceso navegacional a los objetos utilizando expresiones de caminos bastante poderosas. El lenguaje fue presentado de forma inicial en [QRS+95] (con una sintaxis y una semántica cercanas a SQL), posteriormente fue modificado en [AQM+96] (con una sintaxis y una semántica cercanas a OQL).

La última versión de LOREL se inspira básicamente en el lenguaje POQL [CACS94]. Describimos aquí abajo el lenguaje LOREL, basándonos en la versión que apareció en [AQM+96].

La semántica del lenguaje esta fuertemente inspirado en OQL y SQL.

Operador	Función
Select	Es utilisado para la construcción del resultado final relacionado a un nuevo objeto denominado Answer.
From	Esta cláusula declara una lista de « étiquettes d'arcs » para especificar los caminos de acceso a los objetos de una base, eventualmente las variables implicadas en la evaluación de una consulta
Where	La cláusula Where es utilisada para expresar las condiciones sobre los objetos de la base
Grep	Este operateur es utilisado para la búsqueda textual
Path-of	Est utilisado para transformar los caminos

Cuadro 4.5 Lista de los operadores de LOREL

Ejemplo 11 : Dar las tesis que contengan la palabra "KOSOVO" y la palabra "SBI". Nota: el nombre de la base ha sido llamado Thesis

```
select   Tesis.thesis
from     Tesis.thesis.#X
where    X grep "KOSOVO" and X grep "SBI"
```

Ejemplo 12 : Dar las tesis que contengan la expresión "documentos virtuales" o la palabra "SGML"

```
select   Tesis.thesis
where    Tesis.thesis.# grep "documentos virtuales"
           or Tesis.thesis.# grep "SGML"
```

Podemos señalar que en éste ejemplo, se omite la cláusula **`from`** puesto que la expresión Tesis.thesis, deriva de la expresión de camino de la cláusula **`select.`**

Ejemplo 13 : Dar los títulos de una tesis en la base

```
select   t
from     my_thesis.#.title t ;
```

Ejemplo 14 : Dar el título general de las tesis de la base

```
select  t.title
from     Tesis.thesis t ;
```

Ejemplo 15: Dar el título general y el resúmen de las tesis de la base

```
select t.title, t.abstract
from Tesis.thesis t ;
```

LOREL permite construir nuevos objetos como en el ejemplo anterior de los objetos tesis solamente con sus títulos y resúmenes.

Ejemplo 16: Dar las sub-secciones de las tesis que tengan figuras cuya explicación contenga la palabra "virtual"

```
select  s
from     Tesis.thesis.chapter.section.subsectn s
where s.figure.caption grep "virtual"
```

Aquí solamente las sub-secciones que contengan explicaciones serán consideradas en la expresión de camino " figure caption ".

Ejemplo 17 : Dar los capítulos de las tesis cuyo título de una sub-sección contenga la palabra "O2" y el resúmen contenga la palabra "SGBDO"

```
select  t.capítulo
from     Tesis.thesis t ;
where t.chapter.subsectn.title grep "O2" and  t.abstract grep "SGBDO"
```

Ejemplo 18 : Dar las tesis cuyas secciones contengan todas al menos tres sub-secciones.

```
select t
from   Tesis.thesis t ;
where for all s in t.chapter.section: 3 > count(select s.subsectn)
```

Ejemplo 19 : Dar los caminos de acceso a partir de las tesis hacia elementos bibliográficos que contengan "Lorel"

```
select   path-of(@P)
from     Tesis.thesis.# @P.bibelem b
where b grep "Lorel"
```

Donde:

La variable @P, representa los caminos que llevan hacia un elemento bibliográfico (bibelem). La función path-of fue utilizada para transformar los caminos, instancias de la variable @P, en cadenas de caracteres.

Podemos ver la potencia del lenguaje para interrogar, no solamente los datos, pero también su estructura. LOREL es un lenguaje de consultas bastante potente que permite interrogar documentos heterogéneos, estructurados o semi-estructurados, y que permite también expresar la casi totalidad de las consultas presentadas.

El aporte principal de éste enfoque reside en la flexibilidad en la interrogación de los datos semi-estructurados, incluso si no conocemos su estructura.

4.4 Textriever

Este sistema fue desarrollado en la Universidad de Waterloo en Canadá [Bur91], consiste en un motor de búsqueda de informaciones en un fondo documental orientado hacia los documentos estructurados. Al entrar, el usuario provee al sistema un documento con marcadores que declaran su estructura lógica, por ejemplo marcadores SGML. Después del cambio dentro de la base un documento es leído de forma secuencial y todos los elementos o las palabras del documento son identificados. Para

cada elemento o palabra diferente se crea una lista de concordancias. El conjunto de listas de concordancias define de forma uniforme la estructura lógica y el contenido de los documentos de la base.

4.4.1 Modelo formal

El modelo formal está basado en los conceptos matemáticos de conjunto y de secuencia. En éste contexto una **colección textual** T, correspondiente a un documento o a un conjunto de documentos del mismo tipo, es una secuencia acabada de **palabras** ω_0, $\omega_1,\ldots, \omega_{n-1}$ donde cada palabra W_i pertenece al diccionario V de la colección ($\omega_i \in V$).

Una lista de concordancias se define como un conjunto dado de límites que determinan extenciones contiguas « disjointes ». **G** es el nombre de la lista y ***m*** es su cardinalidad. Las extensiones contigüas pueden ser ya sea estáticas, es decir definidas a priori, ya sea dinámicas, es decir definidas durante una operación de búsqueda por extensiones contigüas estáticas adyacentes.

Finalmente, una **base documental** es un par o tupla $D = (T,C)$ donde T es una colección textual y C un conjunto de listas de concordancias para cada palabra $\omega_i \in V$ y cada elemento de la estructura lógica del documento. Este conjunto de listas de concordancias juega el rol de un archivo inversado para la estructura y el contenido de los documentos.

Ejemplo 20 : Dar las tesis que contengan la palabra "KOSOVO" y la palabra "APPLET".

<thesis> SW {"KOSOVO" , "APPLET"}

Ejemplo 21 : Dar las tesis que contengan la expresión "CLASE KOSOVO" o la palabra "SGML".

<thesis> SW {"CLASE KOSOVO"}{"SGML"}

Ejemplo 22 : Dar los títulos de una tesis enla base.

<title> SN {<thesis>}

Ejemplo 23 : Dar los títulos generales de las tesis de la base.

<title> SN {<thesis>} RN {<chapter>} RN {<section>} RN {<subsectn>}

Ejemplo 24 : Dar las sub-secciones de las tesis que tengan figuras cuya explicación contenga la palabra "estrella"

<subsectn> SW {<figure>} SWn{<caption> SW {"estrella")}}}

Ejemplo 25 : Dar los capítulos de las tesis cuyo titulo de una sub-seccion contenga la palabra "KOSOVO" y el resúmen contenga la palabra " applet"

<capitulo> SW {<subsectn>}SW{<title> SW {"KOSOVO")}}}

SW {<thesis>}SW{<abstract> SW {"applet")}}}

4.5 El Lenguaje de Consultas POQL

En ésta sección presentamos las funcionalidades del lengüaje POQL (Path Object Query Language) [Chr96] son una extensión del lengüaje de consultas OQL [ODMG93]. Una característica es la interrogación de los caminos (o de los atributos).

4.5.1 Los operadores

Operador	Función
and, or, et not	Operadores booleanos
Near	Operador de proximidad, que verifica si dos palabras existen en una cadena de caracteres, a una distancia inferior o igual a n palabras.
Contains	Este « predicado » permite comparar una cadena de caracteres
like	Permite expresar una forma limitada de expresiones regulares sobre la cadena de caracteres que utilice caracter " * "
from	Esta cláusula permite declarar el camino exacto de la estructura
.	Permite acceder a los valores de los atributos que utilisen la clausula select
Last	Operador de cercanía, que extrae le dernier valeur d'une liste.

Cuadro 4.6 Lista de los operadores del lenguaje POQL

4.5.2 Consultas sobre el contenido textual

Debemos utilizar la **clase Text** para representar el contenido textual de los documentos almacenados en O_2.

Ejemplo 26: Dar el titulo y la introducción (el primer capítulo) de las tesis que contenagan en los agradecimientos las palabras "acm" y "dupont" en una distancia de 20 palabras.

```
select   struct(title:t.title, intro:first(t.chapters))
                from t in Tesis
where t.acknowl contains ("acm near/20 Cluet")
```

El predicado ***contains*** permite comparar una cadena de caracteres con un modelo de palabras construidas con la ayuda de los operadores clásicos de concatenación, "de disjonction", "de fermeture de Kleene", etc.

La consulta siguiente ilustra como podemos combinar la navegación a través de la estructura de los documentos con ciertas condiciones sobre el contenido.

Ejemplo 27 : Dar los títulos de los capítulos de las tesis cuyo titulo de una sección contenga la palabra "HERENCIA" o "APPLET" y el título general contenga la palabra "KOSOVO"

```
select   c.title
from     Tesis{t}.chapters{c}.sections{s}
where t.title contains ("KOSOVO") and s.title contains ("HERENCIA or
APPLET")
```

La expresión de camino utilizada en la cláusula **from** permite acceder en primera instancia a las tesis de la base, posteriormente a los capítulos de cada tesis y finalmente a las secciones de cada capítulo.

Como para OQL [ODMG93], en POQL el "." indica un componente de n-uplet pero la representación de los elementos de una colección (el recorrido de un conjunto dentro de el camino) es diferente. En POQL el recorride de un conjunto se expresa en un camino por el nombre de la colección por ejemplo Tesis seguida de {t} donde t es una variable de datos cuya « portée » son los elementos de la colección.

En éste ejemplo podemos combinar la navegación a través de la estructura de los documentos con condiciones sobre el contenido.

La expresión de camino utilizada en la cláusula **from** permite acceder primero a las tesis de la base, luego a los capítulos de cada tesis y finalmente a las secciones de cada capitulo. La condition en la cláusula **where** restringe las tesis así como sus secciones y la cláusula **select** proyecta el resultado sobre los títulos de los capítulos seleccionados.

4.5.3 Consultas sobre las uniones de los tipos

La ultilización de las uniones de tipos para la representación de los documentos estructurados implica la introducción en el lengüaje de operadores nuevos ligados a éste constructor. En los ejemplos a continuación presentamos éstos operadores sobre datos de tipo union así como « raccourcis » que permiten administrar las uniones de forma transparente para el usuario. Tomemos por ejemplo la consulta siguiente que permite obtener el plan de tesis, plan que es siempre descrito siempre descrito en el último párrafo de la introducción de la tesis.

Ejemplo 28 : Dar el último párrafo en la introducción (es decir el primer capítulo) de las tesis.

```
Last ( select   b.parrafo
from    Tesis{t}.chapters{c}.sections{s}.bodies{b}
where tb ?parrafo
```

La expresión de camino en la cláusula **from** permite el acceso a los elementos **body** de las secciones del primer capítulo de una tesis

4.6 Conclusión.

El cuadro 4.7 puede resumir la conclusion de éste capitulo,éste presenta las diferentes caracteristicas de los sistemas y lenguajes asi como las de ls sistemas y lenguajes de consultas.

En la columna **operadores sobre la composición** se trata de determinar que tipo de operadores están disponibles para interrogar sobre la relación de composición de la estructura. Los sistemas como MULTOS [Tha90], MAESTRO [Mac90], [Mac91] y el lenguaje POQL [Chr96] proponen caminos de acceso para construir consultas que permitan navegar en la estructura de los documentos. En la columna **operadores sobre la secuencia** se trata de determinar que tipo de operadores están disponibles para interroga la relación de secuencia de la estructura. La interrogación de los composnentes repetidos (por ejemplo una secuencia de párrafos) se hace a través de operadores sobre las listas. MULTOS y los modelos basados sobre las expresiones PAT ignoran éste tipo de interrogación así que MAESTRO y el modelo de Navarro propone operadores de posicionamiento en una lista de componentes. En la columna **Aplicación de los attributs.** Consideramos dos tipos de atributos, los atributos locales y los atributos globales. Estos últimos se encuentran definidos a nivel del documento completo o de una colección de documentos mientras que los atributos locales se definen al nivel de los componentes del documento como lo propone la norma SGML. Por otro lado debemos saber que éstos tipos de atributos están disponibles para cada sistema y que los operadores están disponibles para ser manipulados.

En la columna **uso del contenido o tratamiento del contenido de los documentos** por un lado se debe saber que los tipos de documentos son administrados, es decir que los medios de comunicación tienen la autorización y por otro lado hemos

enumerado los operadores que permiten interrogar el contenido de los documentos. Las expresiones PAT y el modelo de Navarro se dedican particularmente a los documentos textuales. Un SGBD que soporta el lenguaje POQL puederecibir varios tipos de datos pero los operadores de interrogación sobre el contenido se limitan a los atributos y a las partes textuales de los documentos.

Sistema	Operadores basados en la composición	Operadores basados en la secuencia	Aplicación de los atributos	Aplicación sobre el contenido	Características
Maestro [Mac90], [Mac91]	Camino de acceso y uso de cuantificadores	Operador : "First", "last", y sobre las listas	Attributos puramente textuales sobre groupos de documents	Concordancia en el texto	
Multos [Tha90]	Camino de acceso		Operadores sobre los atributos alfanumericos	Concordancia en el texto	Autoriza la consulta de documentos multimédia sobre su estructura y su contenido. Permite definir las conditiciones sobre los componentes de los documentos búscados sin precisar los tipos exactos de los documentps
Lorel [BRS+95]	Camino de acceso con variables				Este lenguaje se basa en la estructura del texto (los árboles sintácticos)
POQL [Chr96]	Camino de acceso con variables	Operación sobre las listas	Atributos en la BD y variables de atributos	Concordancia en el texto y aproximación en el texto	Consulta en el camino de acceso
Navarro [Nav96]	Inclusión y aproximación sobre la estructura	Operación sobre las listas		Concordancia en el texto y aproximación en el texto	Combinación de vistas múltiples

Cuadro 4.7 Resúmen de las funcionalidades de los sitemas.

Finalmente, presentamos varios tipos de sistemas y lenguajes de consultas capaces de soportar diferentes tipos de dosumentos, y principalmente documentos estructurados. Creemos que el principal desafío al cual se enfrentan éstos sistemas de lenguajes de consulta tiene que ver con la flexibilidad de la inderrogación. Esta se traduce en la introducción de técnicas que permiten disfrazar la complejidad del documento después de la interrogación así como el acceso directo a los componentes estructurales pertinentes.