Algoritmos especializados para almacenamiento y la recuperación de información bibliográfica

(Segunda de dos partes)

JUAN VOUTSSÁS MÁRQUEZ
Centro Universitario de Investigaciones Bibliotecológicas
de la UNAM, 04510, México D.F., Tel: 56-23-03-29
E-Mail: voutssas@servidor.unam.mx

MIGUEL AGUSTÍN RUIZ VELASCO Y ROMO
Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas
de la UNAM, 04510, México D.F., Tel: 56-22-36-76
E-Mail: mrvyr@servidor.unam.mx

Artículo

RESUMEN
Los bancos de datos bibliográficos desarrollados en computadores de tipo personal se han visto limitados en cuanto al número de fichas y su rendimiento en función del tamaño de la plataforma en la que se desarrollan e instalan. Los manejadores comerciales de bases de datos para estos equipos han sido construidos de acuerdo con necesidades de tipo general en el mercado y no contemplan las características propias de la información bibliográfica, por lo que decrecen en rendimiento rápidamente en función al tamaño del banco de datos. En la primera parte del documento se analizó esa problemática y las características propias de la información bibliográfica en lo tocante a su inclusión en bancos de datos electrónicos, y se presentó un modelo de compresión de datos bibliográficos en forma de algoritmo que permite entre un 40 y 70 % de compresión sin menoscabo de las características propias de la información bibliográfica. En esta segunda parte del documento se presentan las técnicas para crear y comprimir índices preconstruidos de recuperación y archivos de recuperación por palabras en búsqueda libre, así como las técnicas para acceder a ellos y serle presentados al usuario final. En esa parte se concluye que bancos de datos de cientos de miles de fichas y millones de palabras de recuperación pueden comprimirse en el espacio de un *cd-rom (650 Megabytes)*, e incluso extrapolar estos valores a cotas mucho mayores.

Palabras Clave: Bases de Datos, Automatización de Índices, Algoritmos.

Specialized Algorithms for the Storage and Retrieval of Bibliogrhaphic Information (Second of two parts)

Juan Voutssás-Márquez
Miguel Agustín Ruiz-Velasco y Romo

ABSTRACT:
Bibliographical data bases which have been developed in PC computers have been limited regarding total number of fiches and their performance due to the size of the platform in which those data bases are developed and installed. Commercial PC data base management softwares have been constructed with a general approach, thinking in standard applications, and do not consider the particular features of the bibliographic information. Thus, they decrease in performance exponentially in relation with the size of the data base. In the first part of this survey, this problems were discussed, as well as those typical features of the bibliographic information regarding to its inclusion in computerized data bases. A bibliographic data compression model was introduced as an algorithm, allowing between 40 and 70 % of compression rate without loosing information quality. In this second document, procedures for creation and compression of preconstructed indexes are presented, as well as retrieval files for word free-searching. Some techniques for creation and retrieval of both access paths are fully discussed. In this part the final conclusion shows that data bases with several hundreds of thousands of records owning several millions of retrieval words can be compressed to the available space of a cd-rom (650 Mb), and even expanded to greater figures.
Key Words: Database, Index Automation, Algorithms.

Introducción

La primera parte de esta obra (Voutssás, 1999) presentó la problemática del rendimiento del manejo de bancos electrónicos de información bibliográfica. En ella pudo observarse que existe una relación directa entre el tamaño del banco de datos y la plataforma de cómputo en donde éste se encuentre instalado. El patrón de esto ha seguido las siguientes implicaciones*: una plataforma pequeña limita a que en ella por fuerza sólo pueden montarse bancos de datos pequeños; la instalación de un banco de datos grande implica forzosamente el uso de una plataforma grande.* Dicho de otra forma: el límite del tamaño de un banco de datos es directamente proporcional al tamaño de la plataforma en la que se instala.

Por supuesto siempre se pueden almacenar muchos más registros con un disco duro de mayor capacidad, pero el rendimiento en cuanto a velocidad y precisión en la recuperación desciende rápidamente a valores fuera de lo razonable. El tamaño de un banco de datos no sólo está limitado por el tamaño del disco.

Se plantearon también las definiciones de un banco de datos pequeño o grande. *De forma totalmente arbitraria y sólo para fines de este trabajo* definimos como banco de

datos pequeño a aquel que se encuentra por debajo de 150,000 registros o fichas. Y definimos como plataforma pequeña a aquella constituida por un sólo computador personal (PC) con un solo procesador de tipo *Intel* operando bajo el sistema operativo DOS o Windows, y con velocidad *pequeña*. Como la velocidad de los procesadores ha estado cambiando muy rápidamente a lo largo de este proyecto, el hecho de fijar un número de Megahertzios como "pequeño" o "grande" es una noción muy inestable y que se vuelve obsoleta rápidamente. Definamos entonces velocidad "grande" a la máxima velocidad del procesador que esté a la venta en un momento dado, y "pequeña" a la velocidad menor o igual a la mitad de ese máximo. Es decir, si la máxima velocidad de procesador que está a la venta al momento de leerse este documento es, digamos 1000 Megahertzios, entonces de 500 a 1000 Megahertzios serán considerados como velocidad "grande"; y por debajo de 500 Megahertzios será una velocidad considerada como "pequeña". Extrapólense estos números al momento de la lectura. Los principios enumerados aquí se han comportado linealmente durante varios años operando con un sistema operativo *MS-DOS* o *Windows* personal. Se consideró plataforma grande también a aquellas de tipo *risc* y que se manejan bajo sistemas operativos *Unix* o semejantes.

A lo largo de la primera parte se presentaron en detalle las características propias de los bancos de datos bibliográficos, así como el algoritmo para comprimir y preparar los registros como paso previo a su almacenamiento.

Este artículo describe como continuación la forma conceptual que permite el almacenamiento y la recuperación tanto de índices como de palabras para bancos de datos con información bibliográfica.

Al iniciar este proyecto estudiamos las bases de datos comúnmente utilizadas que se ejecutaban en plataforma *Intel* tales como *dBase*, *Clipper* o *FoxPro*, cuyas características de funcionamiento fueron diseñadas para manejar datos de los tipos denominados jerárquicos, relacionales y de red. Algunas características de estos tipos de base de datos bibliográficos se pueden manejar directamente con el propio programa de aplicación o paquete, pero otras deberán simularse vía un programa adicionado. A este tipo de archivos (*.dbf) de almacenamiento de datos con fichas los denominamos genéricamente archivos *dBase*, independientemente de su fabricante, dado que su estructura interna es idéntica y son compatibles entre sí. Por el contrario los archivos de índices asociados a los de información (*.dbf) son totalmente incompatibles dadas sus estructuras internas diferentes, definidas por el fabricante, así como sus terminaciones: *.ndx, *.mdx para *dBase*; *.ntx para *Clipper* y *.idx, *.cdx para *Foxpro*.

Como se había planteado, las principales características de un registro típico en cualquier banco de datos, hablando en general son:

- Los campos tienen longitud fija.
- Cada campo ocurrirá exactamente una vez, se use o no.
- No puede existir información repetitiva en cada campo.
- No pueden existir subcampos en un campo.

Principales características de un registro típico *para una base de datos bibliográfica*:

- Los campos deben ser de longitud variable.
- Cada campo puede existir de 0 a *n* veces.
- En cada campo se permite tener información repetitiva.
- Cada campo, según sus características, puede opcionalmente tener subcampos.

Estas características fueron descritas ampliamente en el artículo anterior, así como la forma de compactación y almacenamiento de la información bibliográfica.

Debemos realizar procesos computacionales para transformar la información bibliográfica con el fin de poder accederla vía programas que manejen bases de datos genéricas. El objetivo final es llevar a una plataforma de un tamaño dado a cotas mucho mayores en el número de registros *SIN* causar una notable disminución en el rendimiento.

Se ha observado que entre los usuarios que consultan estos bancos de datos, existen dos tipos principales de búsqueda que los ayudan a encontrar la información deseada:

- Los índices o listas preparadas de antemano que ayudan al usuario a "barrer" la información en su búsqueda de una entrada deseada (*browsing)*. Se trata de índices previamente construidos, como por ejemplo los de autores, títulos o editoriales, que permiten revisar la información en orden alfabético ascendente o descendente con el fin de encontrar alguna entrada de la cual no se tiene la referencia exacta.
- La búsqueda "libre", es decir aquella que introduce alguna o algunas palabras deseadas que existen en el registro.

Los procedimientos de transformación para la generación de la información son entonces:

- El almacenamiento por índices.
- El almacenamiento por palabras.

Y por ende, los procedimientos de transformación para la consulta de la información son:

- La recuperación por índices.
- La recuperación por palabras.

En las características del diseño hemos tomado en cuenta que:

- Los procedimientos de almacenamiento deben generar en su totalidad todas las respuestas posibles. Estos procedimientos analizarán la información bibliográfica y generarán los registros de consulta en forma exhaustiva.
- Los procedimientos de recuperación trabajarán sobre los registros de consulta previamente generados por los procedimientos de almacenamiento. Una vez localizada la información correspondiente, se accederá a la información bibliográfica para proceder a desplegar íntegramente su contenido.

- El usuario deberá de tener una respuesta lo más rápida posible, siempre dentro de lapsos razonables.
- Múltiples usuarios podrán estar consultando simultáneamente el mismo banco de datos.
- No importa mucho que los procedimientos de almacenamiento sean "lentos" por ser exhaustivos puesto que se realizan una sola vez antes de que el usuario conozca el banco, pero el proceso debe hacer siempre posible que los usuarios consulten más rápidamente la información solicitada.
- Con estas ideas en mente logramos que equipos de cómputo que incluían serias limitaciones en su configuración (procesador, memoria, disco, etcétera), hubieran funcionado correctamente con enormes bancos de información bibliográfica (por ejemplo: LibrUNAM, TesiUNAM, SeriUNAM, ColMex, Biblat, etc.), lo cual nos permite suponer que se puede escalar esta tecnología a niveles todavía superiores y obtener rendimientos sorprendentes.

Ejemplo de información bibliográfica (dos registros de un banco de tesis):

CAMPO	DESCRIPCIÓN
FIC	000001
ESC	Facultad de Ingeniería
TIT	Sistema para la planeación y control del mantenimiento en grandes instalaciones.
NOM	Argüelles Romo Julio
NOM	Voutssás Márquez Juan
@@	
FIC	000002
ESC	Facultad de Ingeniería
TIT	Análisis de los índices de Productividad En La Industria
NOM	Elizalde Topete Jaime
NOM	Ruiz Velasco Y Romo Miguel Agustín
NOM	Veytia Fernández Mario
@@	
@@ significa "fin de la ficha"	

ALMACENAMIENTO POR ÍNDICES

Es el procedimiento que nos permite generar las bases de datos de consulta de índices, tomando como información origen la base de datos bibliográfica.

Definimos como índice a una relación ordenada de campos del mismo tipo. Dichos campos existen de 0 a *n* veces dentro de cada registro; en el ejemplo, el campo nombre (NOM) existe dos veces en el primer registro y tres veces en el segundo.

Cuando a un campo se le quieran generar índices, éstos podrán crearse en forma independiente uno por uno o combinando dos o más campos en un sólo índice, según las necesidades. Pueden generarse índices de todos los campos que se considere que lo ameritan.

Para el campo de DESCRIPCIÓN en el archivo de índices, deberá fijarse su longitud a un tamaño apropiado, dado que los campos de este tipo de archivos son para este caso de longitud fija; es decir, no se requiere de absolutamente toda la longitud del campo para ser éste ordenado aceptablemente.

La información que será puesta en el campo DESCRIPCIÓN deberá quedar normalizada, lo que quiere decir que los caracteres de toda la línea deberán estar en mayúsculas y sin diacríticos (acentos, diéresis, cedillas, circunflejos, tildes, etcétera.) aunque para el idioma español no ha ocasionado problemas dejar la "Ñ". También habrá que eliminar los espacios múltiples entre palabras y todos los espacios en blanco al principio del campo, etcétera. Si con todos estos procedimientos aún quedan caracteres, éstos se meterán al índice hasta los primeros *n* caracteres que quepan en el tamaño del campo de DESCRIPCIÓN. Si por el contrario encontramos un campo nulo, esa información no será tomada en cuenta y no se generará el registro para ese campo.

Índices independientes

Utilizando el ejemplo anterior:

El índice de NOMbre quedará como:

RENGLÓN	DESCRIPCIÓN	FICHA
0001	ARGUELLES ROMO JULIO	00001
0002	ELIZALDE TOPETE JAIME	00002
0003	RUIZ VELASCO Y ROMO MIGUEL AGUSTIN	00002
0004	VEYTIA FERNANDEZ MARIO	00002
0005	VOUTSSAS MARQUEZ JUAN	00001

El índice de ESCuela quedará como:

RENGLÓN	DESCRIPCIÓN	FICHA
0001	FACULTAD DE INGENIERIA	00001
0002	FACULTAD DE INGENIERIA	00002

Como vemos en este campo hay repetición de descripciones, lo que en un banco de numerosos registros nos generará sin duda unos archivos enormes y en los cuales habrá mucha información ociosa, ya que la frase "FACULTAD DE INGENIERIA" puede aparecer miles y miles de veces. Para evitar eso procederemos a utilizar un archivo auxiliar que nos evite los registros repetidos del índice.

El índice de ESCuela único y su tabla auxiliar quedarán como:

RENGLÓN	DESCRIPCIÓN	OCURRE
0001	FACULTAD DE INGENIERIA	00002

RENGLÓN	FICHA
0001	00001
0001	00002

El archivo quedará entonces como un archivo en donde cada renglón contiene la única ocurrencia de cada asiento encontrado en el archivo original, sin repeticiones, y una tabla auxiliar. Por ejemplo, este archivo en el caso del banco de tesis de la UNAM sólo contiene una treintena de renglones que corresponden a las escuelas que producen tesis. Si no se quitaran las repeticiones serían alrededor de 200,000 renglones o registros, con un enorme número de repeticiones de cada una. Este archivo queda mejor sustituido por la versión sin repeticiones, que en el lado derecho nos indica cuantas veces se repite dentro de todo el banco (campo OCURRE).

La tabla auxiliar nos sirve para tener la información de todas y cada una de las repeticiones de los registros en el índice, relacionando el renglón de la descripción del primer archivo con todas las fichas que la contienen. Normalmente esta tabla puede crecer a tamaños enormes, pero como su tamaño de registro es mucho más pequeño que el original, el espacio en disco no queda impactado fuertemente.

Índice general

Todos los índices quedarán en un solo archivo, y por lo tanto hay que incluir una columna que indique de cuál campo fue tomada esa información.

RENGLÓN	CAMPO	DESCRIPCIÓN	FICHA
0001	TIT	ANALISIS DE LOS INDICES DE PRODUCTIVI-DAD EN L	00002
0002	NOM	ARGUELLES ROMO JULIO	00001
0003	ESC	FACULTAD DE INGENIERIA	00001
0004	ESC	FACULTAD DE INGENIERIA	00002
0005	NOM	ELIZALDE TOPETE JAIME	00002
0006	NOM	RUIZ VELASCO Y ROMO MIGUEL AGUSTIN	00002
0007	TIT	SISTEMAS PARA LA PLANEACION Y CONTROL DEL MAN	00001
0008	NOM	VEYTIA FERNANDEZ MARIO	00002
0009	NOM	VOUTSSAS MARQUEZ JUAN	00001

Para el control de las repeticiones de registros idénticos romperemos el archivo en dos: uno que contiene la información sin repeticiones, y usaremos una tabla auxiliar:

RENGLÓN	CAMPO	DESCRIPCIÓN	OCURRE
0001	TIT	ANALISIS DE LOS INDICES DE PRODUCTIVI-DAD EN L	00001
0002	NOM	ARGUELLES ROMO JULIO	00001
0003	ESC	FACULTAD DE INGENIERIA	00002
0004	NOM	ELIZALDE TOPETE JAIME	00001
0005	NOM	RUIZ VELASCO Y ROMO MIGUEL AGUSTIN	00001
0006	TIT	SISTEMAS PARA LA PLANEACION Y CONTROL DEL MAN	00001
0007	NOM	VEYTIA FERNANDEZ MARIO	00001
0008	NOM	VOUTSSAS MARQUEZ JUAN	00001

RENGLÓN	FICHA
0001	00002
0002	00001
0003	00001
0003	00002
0004	00002
0005	00002
0006	00001
0007	00002
0008	00001

El campo OCURRencia contiene la cantidad de veces que se repitió ese campo. La tabla auxiliar es leída por el programa y se interpreta de la siguiente manera:

- ✓ El texto del renglón 1 se encuentra en la ficha 2
- ✓ El texto del renglón 2 se encuentra en la ficha 1
- ✓ El texto del renglón 3 se encuentra en la ficha 1
- ✓ El texto del renglón 3 se encuentra también en la ficha 2
- ✓ El texto del renglón 4 se encuentra en la ficha 2
- ✓ Etcétera.

Como parte del ejemplo, el orden del archivo de índice se hizo con base en el campo DESCRIPCIÓN, pero lo que posiblemente sea mejor en este caso sería hacer una llave compleja de CAMPO+DESCRIPCIÓN con el fin de tener agrupados todos los registros del mismo campo, tal como se realiza para los desarrollos en CD-ROM, donde la información queda completamente estática. Para bancos de consulta en-línea que estén continuamente modificándose, hay que reorganizar cada determinado tiempo el o los archivos de índice con el fin de agilizar los procesos de búsqueda de los usuarios. De este modo se disminuye el consumo de recursos de cómputo.

RECUPERACIÓN POR ÍNDICES

Es el procedimiento que le permite al usuario consultar los índices en forma ordenada sobre la información que en cada campo de la base bibliográfica quedó indizada. Quien genera la base de datos define de antemano cuáles campos deben ser indizados, ya que no todos los campos de un catálogo son útiles en forma de índice. Por ejemplo, no es lógico crear un índice por el número de edición en un catálogo. Se da el caso entonces de que algunos campos de la base bibliográfica no hayan sido indizados, y por lo tanto dichos campos *NO* podrán ser consultados por el usuario en esa aplicación.

La definición de cada base bibliográfica debe definir explícitamente cuáles campos serán indizados y cuáles no. Este es el gran factor de eficiencia de este enfoque, el cual considera que las preguntas *NO* programadas degradan el sistema.

Ejemplo del acceso a los índices por parte del usuario:

- ❖ El usuario debe seleccionar qué índice desea consultar.

Ejemplo: Índice de NOMbre.

- ❖ Como se trata de un índice el usuario deberá proporcionar el punto inicial deseado de acceso al archivo

Ejemplo: teclear - ELI

- El sistema le responderá enviando la siguiente información:

OCURRE	DESCRIPCIÓN
00001	ELIZALDE TOPETE JAIME
00001	RUIZ VELASCO Y ROMO MIGUEL AGUSTIN
00001	VEYTIA FERNANDEZ MARIO
00001	VOUTSSAS MARQUEZ JUAN

- Con el cursor marcando el primer elemento, puede moverse hacia arriba, o hacia abajo. También es posible usar el *ratón* para apuntar el registro.

Ejemplo: Una vez que se ha encontrado el registro, teclear *enter* para ver el contenido completo de esa ficha:

CAMPO	DESCRIPCIÓN
FIC	000002
ESC	Facultad de Ingeniería
TIT	Análisis de los índices De Productividad En La Industria
NOM	Elizalde Topete Jaime
NOM	Ruiz Velasco Y Romo Miguel Agustín
NOM	Veytia Fernández Mario

La ficha que ve desplegada el usuario respeta la tipografía original; no fue afectada por la normalización; sólo los índices.

Técnicamente lo hizo de la siguiente manera:

- ✓ El usuario seleccionó el índice de NOMbre.
- ✓ El sistema abre el índice de NOMbre junto con su *índice* sobre el campo DESCRIPCIÓN.
- ✓ El usuario teclea el punto de inicio deseado en la tabla: ELI.
- ✓ El sistema realiza una búsqueda directa y encuentra que dicha información empieza en el renglón # 2.
- ✓ El sistema invoca un procedimiento de despliegue de tablas, asocia dicho archivo en el renglón encontrado, y la rutina desplegadora de tablas empieza a desplegar la información en la pantalla a partir de ese punto inicial.

- ✓ El usuario se mueve hacia arriba o hacia abajo con el cursor o el *ratón* hasta ubicarse en el registro de su interés en el índice.
- ✓ El usuario oprime *enter* dado que quiere ver esa ficha.
- ✓ La rutina de despliegue de tablas manda llamar a la rutina que despliega la ficha donde, en este caso, este registro apunta a la ficha única que es la #2, y ésta es desplegada en su totalidad. Si la ocurrencia de este registro fuera múltiple, la rutina de despliegue de fichas le dará la opción de ver las fichas: Siguiente, Anterior, Primera, Última, o el número de secuencia del 1 al *n* de las *n* fichas encontradas.

ALMACENAMIENTO POR PALABRAS

Ya se dijo que es el procedimiento que nos permite generar las bases de consulta de palabras tomando como información origen la base de datos bibliográfica.

A diferencia del procedimiento de almacenamiento por índices, las distintas palabras que componen un registro son indizadas una por una, ficha por ficha, lo cual implica un procedimiento enorme; sin embargo esto nos permite posteriormente buscar cualquier palabra cuyo campo haya sido indizado.

Al igual que en índices, no todos los campos son deseables de indizarse; el administrador de la base decide previamente a qué campos del registro se le aplicará el indizado por palabras. Para cada campo por utilizar en la generación por palabras, tomaremos cada una de ellas aunadas a la clave del campo y le asignaremos un número consecutivo a cada palabra.

Previamente al almacenamiento se realiza un proceso de normalización de las palabras de los campos con objeto de maximizar la recuperación. Como el proceso de normalización también se aplica al momento en que el usuario está buscando algo, se garantiza que las entradas sean las mismas. Recuérdese que estas conversiones sólo se aplican a las palabras que van a indizarse; el texto original de la ficha se respeta y será visto por el usuario al momento del despliegue de la misma forma que fue grabado originalmente.

El primer paso es la conversión de todas las palabras a mayúsculas, lo que permite recuperarlas independientemente de cómo el usuario las use al momento de buscar. También procederemos a eliminar los diacríticos (acentos, diéresis, tildes, cedillas, etcétera) para recuperar las palabras, independientemente de si el usuario introduce las palabras con sus diacríticos o sin ellos. Dicho procedimiento de normalización deberá ser aplicado idénticamente al momento de realizar la consulta para ser congruente.

Ejemplo de información bibliográfica (dos registros de tesis):

CAMPO	DESCRIPCION
FIC	000001
ESC	Facultad de Ingeniería
TIT	Sistema para la planeación y control del mantenimiento en grandes instalaciones.
NOM	Argüelles Romo Julio
NOM	Voutssás Márquez Juan
@@	
FIC	000002
ESC	Facultad de Ingeniería
TIT	Análisis de los índices de Productividad En La Industria
NOM	Elizalde Topete Jaime
NOM	Ruiz Velasco Y Romo Miguel Agustín
NOM	Veytia Fernández Mario
@@	

Para el presente ejemplo procederemos a obtener las palabras de cada uno de los registros para el campo ESCuela;

RENGL	CAMPO	DESCRIPCIÓN	FICHA
00001	ESC	FACULTAD	00001
00002	ESC	DE	00001
00003	ESC	INGENIERIA	00001
00004	ESC	FACULTAD	00002
00005	ESC	DE	00002
00006	ESC	INGENIERIA	00002

En este ejemplo se observa que las palabras “FACULTAD”, “DE”, e “INGENIERIA” se repitieron dos veces. Es obvio que en un banco de este tipo habrá palabras que se repetirán cientos, miles, decenas de miles y hasta cientos de miles de veces. Pensemos en la palabra “FACULTAD” en un banco como el de tesis de la UNAM, y encontraremos que su ocurrencia se acerca a las doscientas mil veces. Si escribiéramos esas doscientas mil palabras en una lista de recuperación ocuparían mucho espacio.

Como podemos apreciar en este pequeño ejemplo, las repeticiones de las palabras en el campo_ESCuela generan un espacio enorme en almacenamiento, por lo cual es muy recomendable realizar una optimización que consiste en:

El archivo maestro de palabras.

PALABRA	CAMPO	DESCRIPCIÓN	OCURRE
00001	ESC	FACULTAD	2
00002	ESC	DE	2
00003	ESC	INGENIERIA	2

El archivo de recuperación de referencias de palabras y fichas.

PALABRA	FICHA
00001	000001
00002	000001
00003	000001
00001	000002
00002	000002
00003	000002

Este último archivo debe interpretarse, al ser leído renglón por renglón como: "la palabra 1 existe en la ficha 1"; "la palabra 2 existe en la ficha 1"; "la palabra 3 existe en la ficha 1"; "la palabra 1 existe en la ficha 2", etcétera ¿Cuál es la palabra 1? refirámonos al archivo maestro de palabras, la palabra 1 es "FACULTAD", la palabra 3 es "INGENIERIA", etcétera.

Así con tales estructuras ahorraremos enormes espacios de almacenamiento y por tanto obtendremos mucha mayor velocidad de acceso, ya que físicamente cada palabra sólo está escrita una sola vez en el archivo maestro de palabras. En palabras-ficha hacemos referencia a esa palabra sólo con un número.

Para ejemplificar con un banco de datos real, en el disco compacto de TesiUNAM 1992, el archivo de palabras tenía 180,393 registros de palabras distintas y el de referencias de palabras-ficha tenía 8'400,000, es decir, se indizaron casi ocho millones y medio de palabras por las cuales se puede recuperar, pero sólo ciento ochenta mil son distintas. Hay que notar la proporción de los registros. Sin este tipo de simplificaciones se hubiera requerido un espacio de almacenamiento superior a la capacidad máxima del CD-ROM en que se editó.

Otra consideración que debemos tener en mente cuando realizamos el proceso de generación de palabras es eliminar las palabras altamente repetitivas y que por lo general no ayudan a la recuperación; estas palabras son los artículos, pronombres,

proposiciones, conjunciones, contracciones e interjecciones gramaticales. A estas palabras se les conoce como *"stop words"* o desechables:

EL, LA, LOS, LAS, LO, UN, UNA, UNOS, UNAS, UNO, YO, TU, EL, Y, E, NI, QUE, O, A, ANTE, BAJO, CABE, CON, CONTRA, DE, DESDE, EN, ENTRE, HACIA, HASTA, PARA, POR, SEGUN, SIN, SO, SOBRE, TRAS, AL, DEL, AH, OH.

Estas palabras se encuentran almacenadas en una tabla previamente construida al efecto, la cual puede ser modificada al momento de generar las palabras con objeto de agregar o suprimir tales palabras "desechables". En algunos casos en que la información de las fichas venía en inglés, la tabla fue modificada a ese idioma para suprimir las palabras no deseadas en esa lengua (a, an, the, I, you, it, this, that, and, to, etcétera).

De ahí, en el momento de indizar cualquier campo, al ser analizadas una a una las palabras se cotejan contra la tabla; y si existen ahí son eliminadas de la lista que va a indizarse. Además en cualquier campo las palabras a indizarse deberán tener tres caracteres o más. Normalmente las palabras de uno o dos caracteres no aportan información importante para la recuperación de la información, salvo en casos especiales. Por supuesto si en el momento de eliminar palabras no deseadas el campo quedase vacío, se respetarán todas las palabras que lo integren.

Así al aplicar el proceso de normalización a los campos de las fichas ejemplificadas los archivos resultantes de palabras para el presente ejemplo quedan como:

El archivo de palabras:

PALABRA	CAMPO	DESCRIPCION	OCURRE
00001	ESC	FACULTAD	2
00002	ESC	INGENIERIA	2
00003	TIT	SISTEMA	1
00004	TIT	PLANEACION	1
00005	TIT	CONTROL	1
00006	TIT	MANTENIMIENTO	1
00007	TIT	GRANDES	1
00008	TIT	INSTALACIONES	1
00009	NOM	ARGUELLES	1
00010	NOM	ROMO	2
00011	NOM	JULIO	1
00012	NOM	VOUTSSAS	1
00013	NOM	MARQUEZ	1

00014	NOM	JUAN	1
00015	TIT	ANALISIS	1
00016	TIT	INDICES	1
00017	TIT	PRODUCTIVIDAD	1
00018	TIT	INDUSTRIA	1
00019	NOM	ELIZALDE	1
00020	NOM	TOPETE	1
00021	NOM	JAIME	1
00022	NOM	RUIZ	1
00023	NOM	VELASCO	1
00024	NOM	MIGUEL	1
00025	NOM	AGUSTIN	1
00026	NOM	VEYTIA	1
00027	NOM	FERNANDEZ	1
00028	NOM	MARIO	1

(Todas las demás palabras que lo integraban han quedado eliminadas del índice)

El archivo de recuperación de referencias de palabras-ficha:

PALABRA	*FICHA*
00001	**000001**
00002	**000001**
00003	000001
00004	000001
00005	000001
00006	000001
00007	000001
00008	000001
00009	000001
00010	**000001**
00011	000001
00012	000001
00013	000001
00014	000001
00001	**000002**

00002	**000002**
00015	000002
00016	000002
00017	000002
00018	000002
00019	000002
00020	000002
00021	000002
00022	000002
00023	000002
00010	**000002**
00024	000002
00025	000002
00026	000002
00027	000002
00028	000002

Las palabras eliminadas por estar en la lista de *stop words* o por su tamaño son: DE, PARA, LA, Y, DEL, LOS.

Nótese como las palabras 00001, 00002 y 00010 son mencionadas dos veces en esta tabla; ello quiere decir que existen en las dos fichas. Este hecho se repetirá tantas veces como sea necesario y millones de veces si es preciso, hasta agotar la totalidad de las palabras de todas las fichas que van a indizarse. Al final queda un archivo maestro de palabras que contiene cada una de las palabras *distintas* que aparecen en el banco de datos ya normalizadas, así como un archivo de palabras-ficha que contiene todas y cada una de las veces que cada palabra ocurre en una ficha dada. Estos dos archivos se generan una sola vez al momento de crear el banco de datos y permiten la ulterior recuperación por palabras.

RECUPERACIÓN POR PALABRAS

Es el proceso que realiza el usuario final y por medio del cual trata de encontrar la ocurrencia o existencia de alguna o algunas palabras que se encuentren dentro de los campos de las fichas. La idea es facilitarle al máximo las oportunidades para encontrar su información.

El archivo de palabras tiene dos tipos distintos de acceso que sirven para:

- Hacer una búsqueda libre (todos los campos).
- Hacer una búsqueda sólo dentro de un campo dado.

Búsqueda libre quiere decir que la palabra será buscada en todos y cada uno de los campos indizados. Por ejemplo, la palabra "CASO", buscada en forma libre, significa que deseamos las ocurrencias de esa palabra, ya sea como nombre de un autor, un título, etcétera.

Si acotamos que sólo deseamos la búsqueda en un campo, ello significa que tal palabra sólo será buscada en el campo especificado y que ignorará sus demás ocurrencias; por ejemplo "CASO" sólo como autor y no como título ni nada más.

La(s) palabra(s) a buscar puede(n) ser además de dos tipos:

- Palabra completa: ELIZALDE
- Palabra truncada: ELIZ* (terminada con un asterisco); esto significa "todas la palabras que comiencen con ELIZ, independientemente de su terminación". A esto se le conoce como palabra "comodín" o *"wildcard"*.

De acuerdo con los dos tipos de condiciones de búsqueda tendremos las siguientes posibilidades para cada palabra:

CONDICIONES	LIBRE	CAMPO
COMPLETA	$LIBre ELIZALDE	$NOMbre ELIZALDE
TRUNCADA	$LIBre ELIZ*	$NOMbre ELIZ*

La "llave" o campo por el que se desea recuperar se introduce usando el signo "$" previamente al campo; esto le indica al sistema que ese texto es una llave y no parte de lo que ese está buscando. Ejemplo: $NOM indica al sistema que se busca por la llave "NOMBRE". El prefijo $LIBre es el asignado automáticamentel en la línea de búsqueda, de manera que se asume por omisión o *"default"* y no se requiere teclearlo y, como ya hemos visto, significa "buscar en todos los campos indizados"

$LIBre ELIZALDE es lo mismo que teclear ELIZALDE

A menos que se haya utilizado otro prefijo modificador de campo; ejemplo:

ELIZALDE $ESCuela INGENIERIA
$LIBre ELIZALDE $ESCuela INGENIERIA
$ESCuela INGENIERIA $LIBre ELIZALDE

Algoritmo de cómo se realiza una búsqueda de una sola palabra:
Caso $LIBre ELIZALDE

- ✓ El índice del archivo de palabras - palab
- ✓ Buscar en el archivo de palabras = "ELIZALDE "
- ✓ Encontró la palabra = 00019
- ✓ Buscar en el archivo de recuperación = 00019

- ✓ Encontró la ficha = 000002
- ✓ Escribe en el archivo de respuestas lo encontrado
- ✓ Despliega la ficha encontrada del archivo de respuestas

Caso $LIBre IN* (ejemplo de palabra comodín o *wildcard*)
El índice del archivo de palabras - palab

- ✓ Buscar en el archivo de palabras = "IN"
- ✓ Encontró las palabras = 00002, 00008, 00016 y 00018
- ✓ Buscar en el archivo de recuperación = 00002
- ✓ Encontró las fichas = 000001 y 000002
- ✓ Escribe en el archivo de respuestas lo encontrado
- ✓ Buscar en el archivo de recuperación = 00008
- ✓ Encontró la ficha = 000001
- ✓ Agrega en el archivo de respuestas lo encontrado
- ✓ Buscar en el archivo de recuperación = 00016
- ✓ Encontró la ficha = 000002
- ✓ Agrega en el archivo de respuestas lo encontrado
- ✓ Buscar en el archivo de recuperación = 00018
- ✓ Encontró la ficha = 000002
- ✓ Agrega en el archivo de respuestas lo encontrado
- ✓ Despliega las fichas encontradas del archivo de respuestas

En este caso, se encontraron las palabras "INGENIERIA", "INSTALACIONES", "INDICES" e "INDUSTRIA". Las fichas encontradas al final referidas por estas palabras, y por lo tanto desplegadas serán la 00001 y la 00002.

Caso $NOMbre JUAN

- ✓ El índice del archivo de palabras - campo+palab.
- ✓ Buscar en el archivo de palabras = "nomJUAN "
- ✓ Encontró la palabra = 00014
- ✓ Buscar en el archivo de recuperación = 00014
- ✓ Encontró la ficha = 000001
- ✓ Escribe en el archivo de respuestas lo encontrado

- ✓ Despliega la ficha encontrada del archivo de respuestas

Caso $NOMbre ROM*

- ✓ El índice del archivo de palabras - campo+palab.
- ✓ Buscar en el archivo de palabras = “nomROM”
- ✓ Encontró la palabra = 00010
- ✓ Buscar en el archivo de recuperación = 00010
- ✓ Encontró la ficha = 000001y 000002
- ✓ Escribe en el archivo de respuestas lo encontrado
- ✓ Despliega la ficha encontrada del archivo de respuestas

Caso de búsqueda que implica más de una palabra

Para el caso de búsqueda de dos o más palabras, el procedimiento se vuelve mucho más complejo.

Para ello se crean dos archivos temporales llamados *preguntas* y *respuestas*, los cuales tendrán la información que se vaya obteniendo en los procesos internos de búsqueda y alternadamente llegarán hasta el resultado final (si lo hubiera). Después de cada respuesta estos archivos se “limpian”; es decir, se dejan en blanco para ser utilizados una y otra vez durante las búsquedas, y al terminar la sesión del usuario son destruidos para no crear espacio ocioso.

También deberemos utilizar un archivo temporal de búsquedas donde se anotarán las palabras encontradas y cuántos registros tiene cada una de ellas. Una vez que sabemos todas las palabras que el usuario desea encontrar obtenemos las ocurrencias de todas ellas y se ordena el archivo de búsqueda en modo ascendente según las ocurrencias totales para cada palabra, de manera que el primer registro es la ocurrencia que menos veces existe en el archivo de referencias, el segundo registro es el que sigue en número de ocurrencias, y así sucesivamente hasta que el último registro es el que más ocurrencias tiene. En el caso de palabras truncadas (comodines o *wildcards*), podrán existir múltiples ocurrencias de palabras y se anotará la suma de ocurrencias en el campo de totales de esa palabra.

El motivo de ponerlas en orden creciente de ocurrencias es que las intersecciones se hacen una a una efectuando el producto cartesiano del primer conjunto con el segundo, el conjunto resultante con el tercero, etcétera. Este orden es de suma importancia para la eficiencia del proceso.

Para ilustrar este concepto supongamos que al analizar una palabra dada por el usuario ésta se encuentra en 100,000 fichas; una segunda palabra se encuentra 10,000 veces en igual número de fichas y una tercera palabra ocurre 100 veces en otras tantas fichas, pero sólo diez fichas tienen esas tres palabras *al mismo tiempo*. Para resolver la intersección debemos verificar en cuáles fichas aparece cada palabra de

forma individual y luego comparar en cuáles fichas coincide que esas palabras aparezcan simultáneamente (esto es, la intersección). Si no cuidamos el orden y realizáramos la intersección como ha sido presentada al principio de este párrafo, deberíamos verificar de una lista de 100,000 fichas cuáles de ellas coinciden con la lista de 10,000 fichas. Esto es una comparación de una lista de 100,000 con una de 10,000. El producto es 1'000,000,000 (mil millones de comparaciones). El resultado es el número de fichas en que coincide que las dos palabras aparecen en la misma ficha. Esta lista deberá ser intersectada con la lista de 100 fichas. El número máximo de intersecciones posibles sería de 10,000, ya que es el número de fichas que existen en la segunda lista. Digamos que el resultado de esta primera intersección son mil fichas; éstas a su vez se intersectarán con la lista de cien; lo que nos lleva a 1000 x 100 igual a 100,000 comparaciones para al final obtener las 10 fichas que coinciden con la intersección final. Al final, debieron hacerse mil millones cien mil comparaciones para obtener el resultado.

Si nosotros comenzamos por la lista más pequeña con la que le sigue en número de ocurrencias, intersectamos la lista de 100 con la de 10,000; esto es, su producto son 1'000,000 (un millón) de comparaciones. Esta lista resultado *NO puede ser mayor* a 100 elementos, ya que la primera lista es de 100 y por lo tanto no puede haber más fichas que coincidan. Si ahora hacemos la intersección de esta lista con la de 100,000 fichas, el máximo que podría hacerse es 100 X 100,000 igual a 10'000,000 (diez millones) de comparaciones. Sumadas a la intersección anterior (un millón) se observa que el máximo posible serían diez más uno, once millones de comparaciones para llegar al mismo resultado. Compárese este numero con el del orden de intersección anterior (mil millones cien mil) y se tendrá una idea de lo que el tiempo de búsqueda se reduce al ordenar de menor a mayor las listas. Como en todo producto cartesiano, el orden sí altera el producto. Es importante recalcar que el orden adecuado es aquel en el cual el algoritmo trabaja formando las listas de palabras en función del número de ocurrencias en ellas. *El orden en que el usuario teclee las palabras al tiempo de buscar es totalmente irrelevante.*

Ilustraremos el algoritmo de resolución de las listas intersectadas con un ejemplo:

Caso $NOMbre Márquez Voutssás Juan
(A propósito se intercambió el orden de las palabras para observar que el orden en que se teclee no afecta el resultado)

Al aplicar el mismo procedimiento de normalización (se traduce a mayúsculas y sin diacríticos) que se usó al momento de indizar el texto introducido por el usuario, queda como:

$NOM MARQUEZ VOUTSSAS JUAN

Se genera el archivo de búsqueda

SECUENCIAL	PALABRA	OCURRE	TOTAL
1	MARQUEZ	1	1
2	VOUTSSAS	1	1
3	JUAN	1	1

- ✓ El índice del archivo de palabras - palab.
- ✓ Buscar en el archivo de palabras = “MARQUEZ ”
- ✓ Encontró la palabra = 00013
- ✓ Buscar en el archivo de recuperación = 00013
- ✓ Encontró la ficha = 000001
- ✓ Escribe en el archivo de respuestas lo encontrado
- ✓ Pasa la información de respuestas a preguntas.
- ✓ Borra el archivo de respuestas
- ✓ Buscar en el archivo de palabras = “VOUTSSAS ”
- ✓ Encontró la palabra = 00012
- ✓ Buscar en el archivo de recuperación = 00012
- ✓ Encontró la ficha = 000001
- ✓ Intersecta el archivo de preguntas
- ✓ Pasa la información al archivo de respuestas
- ✓ Borra este registro del archivo de preguntas
- ✓ Pasa la información de respuestas a preguntas.
- ✓ Borra el archivo de respuestas
- ✓ Buscar en el archivo de palabras = “JUAN ”
- ✓ Encontró la palabra = 00014
- ✓ Buscar en el archivo de recuperación = 00014
- ✓ Encontró la ficha = 000001
- ✓ Intersecta el archivo de preguntas
- ✓ Pasa la información al archivo de respuestas
- ✓ Borra este registro del archivo de preguntas
- ✓ Despliega las fichas encontradas del archivo de respuestas

Caso $NOMbre ROM* $ESCuela Ingeniería $LIBre Facultad $TITulo IN*
Se normaliza a mayúsculas y sin diacríticos:
$NOM ROM* $ESC INGENIERIA $LIB FACULTAD $TIT IN*

Se genera el archivo de búsqueda

SECUENCIAL	PALABRA	OCURRE	TOTAL
1	nomROMO	2	2
2	escINGENIERIA	2	2
3	FACULTAD	2	2
4	titINSTALACIONES	1	2
4	titINDUSTRIA	1	2

- ✓ El índice del archivo de palabras - campo+palab.
- ✓ Buscar en el archivo de palabras = “nomROMO ”
- ✓ Encontró la palabra = 00010
- ✓ Buscar en el archivo de recuperación = 00010
- ✓ Encontró la ficha = 000001 y 000002
- ✓ Escribe en el archivo de respuestas lo encontrado
- ✓ Pasa la información de respuestas a preguntas.
- ✓ Borra el archivo de respuestas
- ✓ Buscar en el archivo de palabras = “escINGENIERIA ”
- ✓ Encontró la palabra = 00002
- ✓ Buscar en el archivo de recuperación = 00002
- ✓ Encontró la ficha = 000001 y 000002
- ✓ Intersecta el archivo de preguntas
- ✓ Pasa la información al archivo de respuestas
- ✓ Borra este registro del archivo de preguntas
- ✓ Pasa la información de respuestas a preguntas.
- ✓ Borra el archivo de respuestas
- ✓ El índice del archivo de palabras - palab.
- ✓ Buscar en el archivo de palabras = “FACULTAD”
- ✓ Encontró la palabra = 00001

- ✓ Buscar en el archivo de recuperación = 00001
- ✓ Encontró la ficha = 000001 y 000002
- ✓ Intersecta el archivo de preguntas
- ✓ Pasa la información al archivo de respuestas
- ✓ Borra este registro del archivo de preguntas
- ✓ Pasa la información de respuestas a preguntas.
- ✓ Borra el archivo de respuestas
- ✓ El índice del archivo de palabras - campo+palab.
- ✓ Buscar en el archivo de palabras = “titINSTALACIONES ”
- ✓ Encontró la palabra = 00008
- ✓ Buscar en el archivo de recuperación = 00008
- ✓ Encontró la ficha = 000001
- ✓ Intersecta el archivo de preguntas
- ✓ Pasa la información al archivo de respuestas
- ✓ Borra este registro del archivo de preguntas
- ✓ Buscar en el archivo de palabras = “titINDUSTRIA ”
- ✓ Encontró la palabra = 00018
- ✓ Buscar en el archivo de recuperación = 00018
- ✓ Encontró la ficha = 000002
- ✓ Intersecta el archivo de preguntas
- ✓ Pasa la información al archivo de respuestas
- ✓ Borra este registro del archivo de preguntas
- ✓ Despliega las fichas encontradas del archivo de respuestas

Estos procedimientos han estado funcionando satisfactoriamente durante muchos años. Aunque parezcan muy complejos, tienen la versatilidad de buscar tanto palabras completas o truncadas como de hacer búsquedas libres o en el campo específico. Optimizan bastante el espacio para ser creados y trabajan a una muy alta velocidad.

Como característica adicional los algoritmos han sido probados intensivamente y como conclusión se ha verificado que las búsquedas con este algoritmo son totalmente determinísticas, o sea que lo buscado es encontrado exactamente y que no pasa lo que con otros buscadores de información que recuperan datos *NO* solicitados. Así con éste método el nivel de *“ruido”* en la información recuperada es prácticamente cero.

CONCLUSIONES

Cuando se ha deseado crear y explotar bases de datos bibliográficas en un computador hace tiempo que el personal técnico ha enfrentado el problema de las proporciones de la plataforma de cómputo de que se dispone. No todas los desarrolladores de bancos de datos electrónicos pueden tener acceso a computadores de plataformas poderosas dentro del mercado de cómputo. Muchas veces se tienen bancos de datos susceptibles de ser creados pero el equipo disponible muestra no ser suficiente para contender aceptablemente con las dimensiones del mismo; otras pareciera que el equipo que se requiere necesita ser de mayor capacidad y costo superior al alcance de las bibliotecas. En otros casos el banco de datos resultante ocupa tantos recursos que cae fuera del alcance de la computadora promedio del gran público potencialmente usuario, y el banco queda restringido a ser usado en plataformas de mucha capacidad y alto precio dentro de instituciones de cierta capacidad económica.

Tradicionalmente para la creación y manejo de este tipo de bancos de datos en computadoras personales se ha utilizado una generación de programas que fue conocida de modo genérico como "paquetes dBase" o "paquetes .dbf", debido a la marca que las popularizó en el mercado y al estándar de archivo de base de datos que creó (archivos dbf). Si bien las marcas y variantes fueron múltiples, las características en lo general y, lo más importante, las capacidades de esos paquetes, eran similares. Como estaban construidos para aplicaciones promedio de bases de datos, para tamaños promedio y para explotaciones promedio, no contemplaban una serie de factores que son de suma importancia en bases de datos bibliográficas, las cuales tienen características muy propias. Por lo tanto los rendimientos fueron promedio y existió un claro límite del rendimiento en cuanto a velocidad de recuperación, espacio ocupado por cada ficha, etcétera. Los desarrolladores de bases de datos en estos computadores de algún modo conocían estos límites de rendimiento y cuando éstos eran ignorados las consecuencias eran evidentes.

A favor de estos paquetes es necesario decir que su aprendizaje es sencillo y rápido y que con un poco de entrenamiento cualquier persona puede aprender a crear un banco de datos en poco tiempo. La creación de un banco de datos se vuelve tarea simple y además, por lo estandarizado del formato, permite el intercambio de datos entre bancos basados en él. Además el costo de adquisición y mantenimiento del paquete es bajo.

Es cierto también que existen en el mercado paquetes manejadores de bases de datos muy poderosos que incluyen herramientas de las denominadas "motores de búsqueda" (*search engines*) muy sofisticadas y permiten manejar numerosos datos en tiempos muy razonables, pero por lo general estos paquetes son instalados en plataformas poderosas no del tipo personal y quedan fuera del alcance del usuario promedio.

La pregunta fue ¿pueden llevarse los rendimientos de los "paquetes" diseñados para las plataformas personales *sensiblemente* más allá de sus límites típicos, sin perder el atractivo que representa el trabajar en una plataforma sencilla y económica?

Por supuesto estos paquetes deben ser modificados por medio de programación adicionada por parte del personal técnico de cómputo al servicio de una biblioteca. Pero ¿qué debe modificarse y cómo?

Analizando las características de los paquetes existentes en el mercado, la experimentación de diversas técnicas fue llevando al desarrollo de un modelo nuevo, en el que se destacan las siguientes premisas básicas:

- Puede mantenerse la utilización de las estructuras básicas de estos paquetes (independientemente de su marca); es decir, los archivos .dbf y sus índices asociados a la marca del paquete (.cdx, .ndx, .ntx, etc.), ya que son confiables, fáciles de aprender y de utilizar.
- Es necesario entender y utilizar los conceptos típicos de la información bibliográfica explicados aquí cuyo manejo es a veces particularmente complicado.
- Es conveniente agregar algunos o todos los procedimientos de optimización sobre el manejo de información bibliográfica presentados a lo largo de este trabajo al manejo de dicha información.
- Los procedimientos son presentados en forma de algoritmo y no de lenguaje de programación, a fin de que cualquiera pueda desarrollar y agregar su propia programación siguiendo el modelo.

Dadas las limitaciones de recursos de cómputo que existen en plataformas de tipo personal, el manejo de bases de datos bibliográficas puede ser mucho más rentable en su actualización y consulta utilizando archivos tipo ”binario” combinados con archivos *.dbf y sus índices (*.cdx, *.ndx, *.ntx, etc.) modificados con los algoritmos presentados en este trabajo, que utilizando motores de bases de datos tipo Oracle, Sybase, Informix, Interbase, etcétera, dado que estos motores tienen poca flexibilidad para manejar este tipo de información y normalmente son manejados por comandos SQL que son difíciles de integrar a lenguajes de programación tipo C, Pascal, Fortran, Cobol, etcétera, y normalmente programan instrucciones QUERY de SQL. Emplear motores de bases de datos -si hubiera acceso desde lenguajes de 3ª generación como C ó C++ (compilables, no intérpretes como Perl) a las bases de datos-, sería una excelente alternativa si sólo se emplearan las bases de datos como esqueletos y todo el manejo a través de programación. A la fecha no hemos encontrado que estos tipos de motores de búsqueda para bases de datos puedan dar acceso a sus archivos vía programación de 3ª generación, ya que las bases de datos jerárquicas, relacionales o de red tienen principios demasiado rígidos y por tanto no cumplen, o no eficientemente, con las características requeridas para el manejo de información bibliográfica.

El gran inconveniente de esta metodología es que normalmente hay que utilizar lenguajes de programación como dBase, Clipper o FoxPro, que tienen muchas limitaciones. Si queremos hacer que estos procedimientos corran verdaderamente mucho más rápido existen otros paquetes, como CodeBase (comercial) o Xbase (gratuito), que junto con un compilador C o C++ son capaces de manejar los archivos

*.dbf y sus índices con una eficiencia muy notable; y por lo menos 5 veces más rápido que con los paquetes dBase genéricos.

Los motores de búsqueda normalmente tienen costos de licencia muy elevados y por lo general no son instalables en las plataformas más pequeñas de tipo computadora personal *Intel-Windows*. En plataformas más potentes basadas en Unix (Solaris, Linux, FreeBSD, etcétera) estas aplicaciones tienen costos de licencia prefijados por cada usuario concurrente.

Sin utilizar motores de búsqueda las únicas limitaciones son los recursos del sistema. Al ser más eficientes los procesos de consulta este tipo de sistemas puede atender a más usuarios en forma concurrente sin que noten los demás usuarios que el sistema tiene una carga provocada por el alto acceso de los otros usuarios, y por supuesto sin invertir enormes cantidades de dinero en licencias de uso ni tampoco depender de los cambios que pudiesen introducir esas empresas.

Existen otros sistemas en el mercado que habilitan al usuario a hacer preguntas no planeadas por el desarrollador, con costos terriblemente altos en procesamiento y saturación del equipo de cómputo *(overhead)*, al tener que "barrer" completamente todo el contenido del banco de datos *original* para cada búsqueda no planeada de cada usuario (consultas tipo *query* o SQL), lo que en muchos casos genera enormes necesidades adicionales de equipamiento totalmente artificiales que carecen de algoritmos eficientes de búsqueda de la información. Estos sistemas resuelven las preguntas, pero lo logran por medio de la "fuerza bruta" de mucho equipo.

Con una buena planeación por parte del desarrollador en cuanto al diseño de los campos que serán recuperables vía índices o palabras, se debe pensar que al realizar sus consultas el usuario jamás ocupe el archivo bibliográfico original en los procedimientos de búsqueda; éstas siempre serán realizadas sobre las estructuras optimizadas como las enunciadas aquí. El archivo bibliográfico exclusivamente será usado en el último instante del proceso solamente para desplegar la información original ante el usuario.

Todos los algoritmos aquí abordados fueron exhaustivamente probados y funcionan realmente en la práctica. Al final de este trabajo se presenta una lista de bancos de datos editados y en explotación real basados exclusivamente en estos algoritmos sin ayuda extra de ninguna pieza de programación adicional (salvo las usadas para el despliegue de imágenes, las cuales nada tienen que ver con el manejo de fichas). El lector puede replicar la creación de cualquiera de esos bancos en una PC utilizando paquetes comerciales en lugar de estos algoritmos y se dará cuenta de la enorme diferencia de espacio ocupado y tiempos de respuesta, además de la calidad lograda. Más aún, la mayoría de tales blancos fueron editados en *cd-rom*, cuyos tiempos de acceso son del orden de 20 a 100 veces más lentos que los de un disco duro, dependiendo de la velocidad de lectura de la tornamesa *cd*, y aun así responden a velocidades satisfactorias. Cuando estas técnicas son aplicadas a bancos de datos que se instalan y explotan en un disco duro cualquiera, sus tiempos de respuesta son todavía mucho más espectaculares dada la diferencia de velocidades existente entre estos dispositivos.

La conclusión final de este desarrollo es que estas técnicas definitivamente sí permiten extender las capacidades de equipos de cómputo "modestos" al facilitar la creación y explotación de bancos de datos bibliográficos mucho mayores y que sobrepasan sus capacidades esperadas promedio sin menoscabo en la calidad de información que se espera de un banco de este tipo. Llevando el experimento aún más lejos, al extrapolar estos procedimientos a servidores *Unix* de tamaño modesto (es decir, las plataformas que definimos como "grandes" al inicio de este trabajo) su rendimiento se vio también incrementado notablemente al poder crear y explotar bancos de datos de dimensiones fuera de lo normal en nuestro medio bibliotecario. En un prototipo desarrollado en una computadora *Sun* modelo *Sparc 20* con sistema operativo *Solaris*, que es una de las más pequeñas en su serie, se creó el banco de datos de tablas de contenido de la empresa *Swets* con la integración de estas técnicas y *sin* ningún motor de búsqueda especializado comercial, lo cual permitió que cualquier consulta tuviera tiempo de respuesta prácticamente instantánea, del rango de unos cuantos segundos, en un banco que integra dos millones de fichas por año, y llegó a un total de *nueve millones de fichas.* Aun como prototipo éste llegó a ser el banco de datos más grande de la UNAM por el número de registros existente en-línea, instalado en un computador de sólo 20,000 dólares. Como puede verse estas técnicas incrementan sensiblemente el rendimiento del equipo de cómputo al ser utilizadas en la creación y explotación de bancos de datos bibliográficos, independientemente de la plataforma en que se desarrollen.

LISTA DE BANCOS DE DATOS EDITADOS CON EL ALGORITMO:

- ✓ Universidad Nacional Autónoma de México (1992*). ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1992-1993.* 2ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Dirección General de Servicios de Cómputo para la Administración. [dos bancos, 20,000 registros]

- ✓ Universidad Nacional Autónoma de México (1992). *Catálogo de libros de las bibliotecas de la UNAM [LIBRUNAM].* Disco Compacto. México : UNAM, Dirección General de Bibliotecas; Dirección General de Servicios de Cómputo para la Administración. [un banco, 480, 588 registros]

- ✓ Universidad Nacional Autónoma de México (1992). *Tesiunam: Catálogo de tesis de la UNAM y otras instituciones.* Disco compacto y guía. México : UNAM, Dirección General de Bibliotecas, Dirección General de Servicios de Cómputo para la Administración. [un banco, 180,000 registros]

- ✓ Universidad Nacional Autónoma de México (1993). *BIBLAT: Bibliografía sobre América Latina e Información.* Disco Compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [cuatro bancos, 290,000 registros].

- ✓ Universidad Nacional Autónoma de México (1993). *Catálogo de libros de las bibliotecas de la UNAM [LIBRUNAM].* Disco compacto. México : UNAM, Dirección General de Bibliotecas; Dirección General de Servicios de Cómputo para la Administración. [un banco, 500,000 registros]

- ✓ Universidad Nacional Autónoma de México (1993). IRESIE: Banco de Datos sobre Educación 1996. 2ª. ed. Disco compacto y guía. México : UNAM, Centro de Investigaciones y Servicios Educativos. Dirección General de Servicios de Cómputo para la Administración. [cuatro bancos, 43,000 registros]

- ✓ Universidad Nacional Autónoma de México (1993). *SERIUNAM: Banco de datos de publicaciones periódicas 1993.* Disco compacto y guía. México : UNAM, Dirección General de Bibliotecas; Dirección General de Servicios de Cómputo para la Administración. [un banco, registros]

- ✓ El Colegio de México (1994). *CD-Colmex: Catálogo de la Biblioteca "Daniel Cosío Villegas" 1940-1993*. Disco compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco, 31,609 registros bibliográficos con 3'164,132 acervos]

- ✓ Universidad Nacional Autónoma de México (1994). *BIBLAT: Bibliografía sobre América Latina e Información*. 2ª. ed. Disco Compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [cuatro bancos, 300,000 registros].

- ✓ Universidad Nacional Autónoma de México (1994). *ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1993-1994*. 3ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Centro de Información Científica y Humanística. [dos bancos, 27,000 registros]

- ✓ El Colegio de México (1995). *CD-Colmex: Catálogo de la Biblioteca "Daniel Cosío Villegas" 1940-1995*. 2ª. ed. Disco compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco, 296,000 registros]

- ✓ El Colegio Mexiquense (1995). *Bibliografía del Estado de México*. Disco compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco, 8,550 registros con imágenes]

- ✓ Secretaría de Salud (1995). *RENCIS: Catálogo Colectivo de Publicaciones Seriadas*. 4ª. ed. Disco compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco, 6959 registros, 20,565 acervos]

- ✓ Universidad Nacional Autónoma de México (1995*). ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1994-1995.* 4ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Centro de Información Científica y Humanística. [dos bancos, 28,000 registros]

- ✓ Universidad Nacional Autónoma de México (1995). *Ciencias Sociales y Humanidades.* Disco compacto y guía. México : UNAM, Coordinación de Humanidades. Centro de Información Científica y Humanística. [dieciocho bancos, 77,000 registros]

- ✓ Universidad Nacional Autónoma de México (1995). *Shock Waves in Medicine.* Dos disquetes 3.5 " y guía. México : UNAM, Instituto de Física, Centro de Información Científica y Humanística. [un banco, 3000 registros]

- ✓ Universidad Nacional Autónoma de México (1995). *SERIUNAM: Banco de datos de publicaciones periódicas 1995*. 2ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Bibliotecas; Centro de Información Científica y Humanística. [un banco, 42029 registros bibliográficos con 5'499,258 acervos]

- ✓ Fideicomiso para la Cultura México-USA (1996). *Abimex: Antigua Bibliografía Mexicana.* Disco compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco con imágenes, 10,417 registros].

- ✓ Secretaría de salud (1996). *Bibliomex Salud: Bibliografía Mexicana en Biomedicina y Salud.* Disco Compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco, 14,391 registros].

- ✓ Universidad Nacional Autónoma de México (1996*). ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1995-1996.* 5ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Centro de Información Científica y Humanística. [dos bancos, 34,000 registros]

- ✓ Universidad Nacional Autónoma de México (1996). IRESIE: Banco de Datos sobre Educación 1996. 3ª. ed. Disco compacto y guía. México : UNAM, Centro de Investigaciones y Servicios Educativos. Centro de Información Científica y Humanística. [cuatro bancos, 50,000 registros]

- ✓ Universidad Nacional Autónoma de México (1997). *ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1996-1997*. 6ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Centro de Información Científica y Humanística. [dos bancos, 38,000 registros]

- ✓ Secretaría de Salud (1997). *RENCIS: Catálogo Colectivo de Publicaciones Seriadas.* 6ª. ed. Disco compacto y guía. México : UNAM, Centro de Información Científica y Humanística. [un banco, 5,335 registros, 18,339 acervos]

- ✓ Universidad Nacional Autónoma de México (1998). *ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1997-1998*. 7ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Centro de Información Científica y Humanística. [dos bancos, 42,000 registros]

- ✓ Secretaría de Salud (1999). *RENCIS: Catálogo Colectivo de Publicaciones Seriadas.* 8ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Servicios de Cómputo Académico. [un banco, 8,926 registros, 29,791 acervos]

- ✓ Universidad Nacional Autónoma de México (1999). *ARIES: Acervo de Recursos de Instituciones de Educación Superior. 1999-2000*. [8ª] 7ª. ed. Disco compacto y guía. México : UNAM, Dirección General de Intercambio Académico. Dirección General de Servicios de Cómputo Académico. [dos bancos, 44,000 registros]

REFERENCIAS BIBLIOGRÁFICAS

Keuffel, Warren. *Battle of the modeling techniques.* DBMS On-line. August 1996.
http://www.dbmsmag.com/9608d14.html

Knuth, Donald. *The Art of Computer Programming, vol. 1: Fundamental Algorithms* ; Reading, Mass., 1973: *Addison-Wesley, 2nd ed,*

Knuth, Donald. *The Art of Computer Programming: vol. 3: Sorting and Searching*; Reading, Mass., 1973 : Addison-Wesley.

Martin, James. *Database Analysis and design*. Prentice-Hall., 1992.

National Institute for Standards and Technology (). *Dictionary of Algorithms, Data Structures and Problems.*,1999.
http://hissa.nist.gov/dads/terms.html

Standish, T. *Data Structure Techniques.* Addison-Wesley.,1980. pp. 191-210.

Standish, Thomas. *Data structures, Algorithms and software principles*; Addison-Wesley., 1994. 748 pp.

Sunday, Daniel. A very fast substring search algorithm. En: *Communications of the ACM*, vol. 33, No. 8, August 1998. pp. 132-142.

University of Western Australia. *Data Structures and Algorithms.*, 1998. *http://swww.ee.uwa.edu.au/~plsd210/ds/ds_ToC.html* Capítulos 3: Data Structures. Capítulo 4: Searching. Capítulo 5: Searching Revisited.

Voutssás, J. Ruiz-Velasco, M. *Algoritmo de Compresión para el Almacenamiento de Información Bibliográfica.* México, 1999 : UNAM, Centro Universitario de Investigaciones Bibliotecológicas.