UCLM-ESI

ÍNDICES MULTICLAVES

ESTE DOCUMENTO ES PARTE DEL TRABAJO SIGUIENTE:

ASIGNATURA: BASES DE DATOS (GESTIÓN)

ALUMNA: MARTA ISABEL GÓMEZ ALONSO

PROFESOR: FRANCISCO RUIZ
CURSO: 1999/2000

1. INTRODUCIÓN

1.1 Acceso por claves múltiples

Muchas aplicaciones exigen que los registros tengan más de una clave, lo que viene a complicar la organización. Para cierto tipo de consultas es ventajoso el uso de múltiples índices. Los registros de clave múltiple aparecen, en algunos sistemas de operaciones, pero se los encuentra más comúnmente en los sistemas de información, en los que han de admitirse muchos tipos de averiguaciones sobre diferentes aspectos de la información contenida en los archivos.

Últimamente ha habido mucho interés en las bases de datos que guardan datos multimedia, imágenes, sonido y vídeo. Hoy en día, los datos multimedia se suelen guardar fuera de las bases de datos, en sistemas de archivos.

Cuando el número de objetos multimedia es relativamente reducido, las prestaciones proporcionadas por las bases de datos no suelen ser importantes.

La funcionalidad de la base de datos se vuelve más importante cuando el número de objetos multimedia guardado es grande. Aspectos como las actualizaciones de las transacciones, las posibilidades de consulta y la creación de índices se vuelven importantes.

Los objetos multimedia suelen tener atributos de descripción, como los que indican el momento en que se crearon, la persona que los creó y la categoría a la que pertenecen.

Las bases de datos multimedia crecen en importancia. Problemas como la recuperación basada en la semejanza y la entrega de los datos a un ritmo garantizado son temas de investigación activa.

Los sistemas de recuperación de la información utilizan un modelo de datos más sencillo que los sistemas de bases de datos, pero proporcionan posibilidades de consulta más potentes dentro de ese modelo restringido.

Las consultas intentan encontrar los documentos que resultan de interés, especificando, por ejemplo, conjuntos de palabras clave. La consulta que el usuario tiene en mente no suele poderse formular con precisión; por tanto, los sistemas de recuperación de información ordenan las respuestas de acuerdo con su importancia potencial.

Los sistemas de información distribuidos que se ejecutan en Internet han tenido un desarrollo enorme en los últimos años. El sistema World Wide Web permite examinar esta información utilizando el paradigma del hipertexto. Las herramientas que integran HTML con SQL facilitan la labor de crear interfaces HTML para las bases de datos. Los sistemas distribuidos de recogida de información ayudan a los usuarios a buscar la información en el sistema Web.

En algunos tipos de consulta como ya hemos comentado es necesario procesar la información cumpliendo una combinación de valores de atributos. En un sistema con un solo índice caben tres posibilidades:

1. Utilizar el primer índice y buscar secuencialmente seleccionando los que cumplan la segunda condición.
2. Viceversa 2° índices.

3. Utilizar la intersección de los conjuntos que provocan la búsqueda de ambos índices.

Todo esto es complejo y requiere una programación cuidadosa, ya que el objetivo de una base de datos es que o no se tenga que programar la recuperación o que éste no sea muy costosa. Por esto se utilizan índices compuestos tanto por separado como combinando varios atributos en un solo índice, para lo cual el programa ordena los últimos atributos hasta llegar al primero. Vamos a ver dos ejemplos:

EJEMPLO 1:

Tenemos el archivo **Depósito** tiene dos índices:
- Nombre-sucursal
- Nombre-cliente

Considérese la consulta siguiente: **"Encontrar el saldo de todas las cuentas de Williams en la sucursal Perryridge"**. Como hemos comentado, existen tres posibles estrategias para procesar esta consulta :

1. Utilizar el índice por Nombre-sucursal para encontrar todos los registros pertenecientes a la sucural Perryridge. Examinar cada uno de estos registros para ver si Nombre-cliente es igual a Williams.

2. Utilizar el índice por Nombre-cliente para encontrar todos los registros pertenecientes a Williams. Examinar cada uno de estos registros para ver si Nombre-sucursal es igual a Perryridge.

3. Utilizar el índice por Nombre-sucursal para encontrar los punteros a todos los registros pertenecientes a Williams. Tomar la intersección de estos dos conjuntos de punteros. Aquellos punteros que estén en la intersección apuntan a registros pertenecientes tanto a Williams como a Perryridge.

Según las tres estrategias comentadas anteriormente, la tercera es la única de las tres que aprovecha la existencia de índices múltiples. Sin embargo, incluso esta estrategia puede ser pobre si se cumplen las tres condiciones siguientes:

1. Existe un gran número de registros que pertenecen a la sucursal Perryridge.
2. Existe un gran número de registros que pertenecen a Williams
3. Sólo hay un número pequeño de registros que pertenecen tanto a Williams como a la sucursal Perryridge.

EJEMPLO 2:

Tenemos el Archivo **Cuenta** tiene dos índices:
- Nombre-sucursal
- Saldo

Consideremos la consulta **"Encontrar todos los números de cuenta de la sucursal Pamplona con saldos igual a 200.000 pesetas ".**
Como ya hemos visto en el ejemplo1 hay tres posibles estrategias para procesar esta

consulta:

1. Usar el índice en Nombre-sucursal para encontrar todos los registros pertenecientes a la sucursal de Pamplona. Luego se examinan estos registros para ver si saldo es igual a 200.000.

2. Usar el índice en saldo para encontrar todos los registros pertenecientes a cuentas con saldos de 200.000 pesetas. Luego se examinan estos registros para ver si Nombre-sucursal es igual a Pamplona.

3. Usar el índice en Nombre-sucursal para encontrar punteros a registros pertenecientes a la sucursal Pamplona. Y también usar el índice en saldo para encontrar los punteros a todos los registros pertenecientes a cuentas con un saldo de 200.000. Se realiza la intersección de estos dos conjuntos de punteros. Aquellos punteros que están en la intersección apuntan a los registros pertenecientes a la vez a Pamplona y a las cuentas con un saldo de 200.000 pesetas.

Al igual que en el Ejemplo1 la tercera estrategia es la única de las tres que aprovecha la ventaja de tener varios índices. Sin embargo, incluso esta estrategia podría ser una pobre elección , al igual que en el ejemplo anterior, si sucediera lo siguiente:

1. Hay muchos registros pertenecientes a la sucursal Pamplona.
2. Hay muchos registros pertenecientes a cuentas con un saldo de 200.000 pesetas.
3. Hay solamente unos cuantos registros pertenecientes a ambos, a la sucursal Pamplona y a las cuentas con un saldo de 200.000 pesetas.

Si se cumplen estas condiciones, debemos examinar un gran número de punteros para producir un resultado pequeño.

Para acelerar el procesamiento de consultas con múltiples claves de búsqueda, una estrategia más eficiente para estos casos tanto en el ejemplo1 como el ejemplo2, es crear y utilizar una clave de búsqueda (Nombre-sucursal, saldo); esto es, la clave de búsqueda consistente en el nombre de la sucursal concatenado con el saldo de la cuenta , para el ejemplo2.

La estructura del índice es la misma que para cualquier otro índice, con la única diferencia de que la clave de búsqueda no es un simple atributo, sino una lista de atributos. Pueden mantenerse varias estructuras especiales que veremos mas adelante.

1.2 Indices Multiclave o Secundarios

En un archivo de varias claves es usual que una de éstas sirva para la identificación unívoca de los registros, a esta clave la llamaremos primaria, identificador exclusivo, y claves secundarias a las demás, pero ya explicaremos en el siguiente apartados dichos conceptos. No es normal que una clave secundaria identifique de forma exclusiva un registro, hay a menudo muchos registros con ese mismo valor de clave.

Una manera de hallar un registro a partir de una clave secundaria consiste en construir un índice para esa clave, los índices basados en un atributo que no es la clave

primaria se les llaman **índices secundarios o multiclave.**

En algunos sistemas de información resulta conveniente indizar muchos atributos diferentes, como hemos comentado en el apartado anterior, por lo que los índices secundarios suelen resultar numerosos.

Ejemplo:

Ejemplo típico de sistema de información con índices secundarios es el que sirve para obtener información acerca de las ventas de una compañía. El usuario puede llegar a plantear los siguientes interrogantes:

- Deme la lista de los 10 principales clientes de la Zona Sur.
- Ventas de este año y las cifras pendientes para los clientes de la Sucursal 74.
- Listado de los automóviles que tienen números de artículo comprendidos entre 1500 y 2000.
- Lista de los ítems pendientes con los clientes de la industria del transporte en el distrito 5.
- ¿Qué clientes de Bahía Blanca han pedido más de 500 unidades del artículo 721?

El nombre y la dirección postal del cliente forman, en conjunto una clave primaria para el direccionamiento de los registros de clientes. El número de artículo constituye una clave primaria para direccionar los registros de artículos, pero se necesitan claves secundarias para vincular estos dos archivos y para indizar ítems pendientes y categoría industrial.

En los sistemas de información en línea no es posible acumular grandes cantidades de averiguaciones y clasificarlas antes de procesarlas, como se haría en el procesamiento por lotes, por lo tanto deben preverse los medios para ir directamente de la averiguación planteada a los registros que permiten evacuarla.

Las opciones que se ofrecen al diseñador de un archivo de claves múltiples permiten llegar a una adecuada concertación entre el aumento de la cantidad de claves secundarias utilizadas, el aumento de la utilización de la memoria principal y el aumento del tiempo requerido para localizar los registros necesarios.

En el caso de estos archivos de claves múltiples, sólo una de las claves debe determinar la posición física del registro, en la mayoría es la clave primaria la que determina dicha posición. El método de direccionamiento de clave secundaria debe basarse por lo tanto en unas técnicas que no dependan de la posición física de los registros.

En apartados posteriores veremos con mas detalle cada una de las técnicas utilizadas para índices multiclave, veremos tres técnicas:

1. Basadas en vectores dimensionales, que será la técnica de rejilla.
2. Basadas en las funciones de asociación que es la técnica de función de asociación divididas.
3. Basadas en los índices, que tendremos cuatro técnicas:
   - 3.1 Archivos Multilista o Listas Múltiples
   - 3.2 Archivos Invertidos
   - 3.3 Cadenas en el Indice
   - 3.4 Arboles R

2. CONCEPTOS BÁSICOS

2.1 Definición

***Índice*:** Conjunto de anotaciones (una por cada registro) que contiene:

- Valor del campo clave
- Puntero (registro de dirección) que permite el acceso rápido al registro.

Los índices pueden ser simples o multiclave.

Índices Multiclave: En un índice vamos a tener asociadas k claves para las cuales se indicará el sitio de cada una.

Se estructuran de forma diferente a los primarios ya que los punteros no señalan directamente al archivo sino que lo hacen a cubetas que contienen punteros al archivo. De esta manera almacenan juntos los punteros de un valor de clave de búsqueda secundaria determinado, una revisión secuencial es rápida ya que los registros están almacenados físicamente en el orden de la clave. Naturalmente es imposible conseguir que el archivo esté ordenado físicamente. Lo que se hace es almacenar punteros en cubetas que apuntarán al registro de datos y éstos ya no los llevarán ni tampoco necesitarán revisiones secuenciales de la clave.

Además los índices secundarios o multiclave, suelen usar como entradas combinaciones de valores de atributos, tales como una combinación de Tipo-Artículo y Tamaño, en estos casos se dice que estos índices son "compuestos".

***Clave de búsqueda*:** Atributo o colección de atributos que se usan para buscar registros en un archivo. La clave de búsqueda se puede representar como una tupla de valores de la forma:

(a1, ... , an) donde los atributos indexados son A1,...,An.

El orden de los valores de la clave de búsqueda es el orden lexicográfico. Para el caso de dos atributos en la clave de búsqueda, (a1,a2) < (b1,b2) si:

a1 < b1 ó a1 = b1

y

a2 < b2 entonces el orden lexicográfico es básicamente el mismo orden alfabético de las palabras.

Ejemplo:
Tenemos una biblioteca, observamos que la mayoría de las bibliotecas mantienen varios catálogos de tarjetas: por autor, tema y título. El atributo o conjunto de atributos que se usa para buscar registros en un archivo se llama **clave de búsqueda**.
Esta definición de "clave" difiere de las de clave primaria y secundarias comentadas en el apartado 1.2. Este doble significado de la palabra clave está bien establecido en la práctica. Usando el concepto anterior de clave de búsqueda, vemos que si hay varios índices en un archivo existirán varias claves de búsqueda.

***Indexación*:** Permite el acceso rápido a los registros de un archivo de datos asociando cada búsqueda a una clave determinada. Para permitir el acceso aleatorio rápido a los registros de un archivo se utiliza una estructura de índice, pero la principal será aquella por la

cual se ordena secuencialmente el archivo de datos y que corresponde al índice primario, los demás son secundarios. Cada estructura de índice está asociada con una clave de búsqueda determinada. Si el archivo está ordenado secuencialmente y elegimos incluir varios índices en diferentes claves de búsqueda, el índice cuya clave de búsqueda especifica el orden secuencial del archivo es el índice primario.

Claves primaria y secundarias: En un archivo de varias claves es usual que una de éstas sirva para la identificación unívoca de los registros. Llamamos clave primaria a este identificador exclusivo, y claves secundarias a las demás. No es normal que una clave secundaria identifique también de forma exclusiva un registro.

> ***Ejemplo*:** En el apartado 1.2 hemos visto un ejemplo de un sistema de información acerca de las ventas de una compañía, en éste el Número-pieza podría ser una clave primaria puesto que identifica unívocamente un registro de pieza al no existir dos piezas que tengan el mismo número.
> Sin embargo, Tipo-pieza podría ser una clave secundaria, ya que pueden existir varias piezas del mismo tipo.
> El nombre y la dirección postal del cliente forman, en conjunto, una clave primaria para el direccionamiento de los registros de clientes.
> El número de artículo constituye una clave primaria para direccionar los registros de artículos, pero se necesitan claves secundarias para vincular ambos archivos y para indizar ítems tales como zona, ventas netas, número de artículos vendidos, número de ítems pendientes y categoría industrial.

Lista Enlazada: Un concepto fundamental en el enlace de un registro físico con otro es el uso de los punteros (un elemento de dato que contiene una dirección física, es decir, es un campo asociado con un registro de dato que se usa para encontrar un registro de dato relativo). Estas cadenas de punteros (lista de punteros, cada uno de los cuales apunta hacia el primer registro del archivo) nos suministran un conjunto de registros enlazados por dicha cadena de punteros se llama una lista enlazada.

2.2 Atributos que deben indizarse

Según comentamos en el apartado 1.2, tenemos una de las técnicas está basada en **índices**, cuando se diseña un archivo de claves múltiples para responder a una gran variedad de averiguaciones espontáneas acerca de los datos que el archivo contiene, el diseñador debe preguntarse para qué atributos deben crearse índices.

Tal vez sea conveniente evitar los índices que resultarían excesivamente largos. El diseñador preferirá quizás indizar un atributo que tiene un pequeño conjunto de valores posibles, de modo que su lista invertida sólo contenga un escaso número de entradas.
Desafortunadamente, los atributos que pertenecen a un conjunto reducido de valores no son en general muy selectivos.

Ejemplo: En el ejemplo de la biblioteca, el valor de atributo ficción se refiere quizás a la mitad de los libros almacenados en los estantes, de modo que un índice que sólo tenga este

valor como argumento sirve de muy poco.

Por el contrario, un atributo que dé como salida un número pequeño de ítems es muy probable que tenga muchos valores indizables.

Una manera de resolver este problema consiste en *combinar atributos* para la indización, precisamente como ellos serían combinados para fijar el criterio de la búsqueda. Los índices se basan a menudo en la combinación de dos o más valores de atributo.

En algunos tipos de consulta es necesario procesar la información cumpliendo una combinación de valores de atributos.

2.2.1 Arboles B +

Como comentamos en la estructura basada en índices tendremos distintas técnicas las basadas en:

Listas Invertidas: Un directorio en donde cada entrada contiene un puntero a todos los registros físicos con un valor específico. Es un archivo independiente, que generalmente contiene solamente dos datos:

1. Un valor de interés.
2. Todas las direcciones de los registros que tienen dicho valor.

Un refinamiento de la técnica de Lista Invertida son los **árboles B+**, se desarrollaron con el objetivo de proporcionar un método eficiente para el mantenimiento de una jerarquía de índices. El rendimiento de un método secuencial-indexado se degrada a medida que el archivo crece. Por otra parte, el rendimiento puede mejorarse con la reorganización periódica del archivo, aunque dichas reorganizaciones pueden consumir mucho tiempo y ser costosas. Un árbol B+ conserva su eficiencia a medida que el archivo crece o se reduce. Un árbol B+ consiste en una jerarquía de registros índices junto con un archivo de registros de datos. Los registros índices contienen claves y punteros que se usan para localizar los registros de datos.

La principal característica de este tipo de árboles es que todas las claves se encuentran en las hojas, y por ello, cualquier camino desde la raíz hasta un nodo tiene la misma longitud.
Como consecuencia de que todas las claves se encuentran en las hojas es posible mantener una lista enlazada apuntando al nodo de más a la izquierda en las hojas y que recorre todas las hojas. De esta manera es posible un recorrido ordenado de las claves que es una operación muy empleada en los ficheros indexados (listado ordenado de las claves), sin necesidad de recorrer el árbol, lo que redunda en una mayor eficiencia de dicho recorrido.

Conceptos Básicos del Arbol B

- ***Orden:*** Indicador del número de nodos que pueden conectarse a un nodo superior, es decir, número de claves que se pueden insertar en cada nodo.
- ***Nodo raíz:*** Nodo inicial
- ***Nivel:*** Todos los nodos que están a la misma distancia del nodo raíz.

- ***Peso:*** Número de niveles que tiene el árbol.
- ***Nodo terminal:*** Los nodos que están en el último nivel.
- ***Nodo rama:*** Nodos intermedios, tienen punteros apuntando a otros nodos.

Características de los árboles B+

Se define un árbol B+ de orden d de la siguiente forma:

1. Cada página, excepto la raíz, contiene entre d y 2d elementos (claves).
2. Cada página, excepto la raíz, tiene entre d+1 y 2d+1 descendientes. Se utiliza m para expresar el número de elementos en una página.
3. La página raíz tiene al menos dos descendientes.
4. Las páginas hojas están todas en el mismo nivel.
5. Todas las claves se encuentran en las páginas hojas.
6. Las claves de las páginas raíz e interiores se utilizan como índices, es decir, no tienen punteros a datos, solo hay punteros en las hojas.
7. Todas las claves que aparezcan en niveles intermedios también aparecen en la hoja y su puntero a datos en el último nivel (hojas).

Operaciones Básicas

1. **Búsqueda en árboles B+**

La operación de búsqueda en estos árboles es similar a la comentada en apartados anteriores en árboles B. La diferencia estriba en que al buscar en un árbol B podemos encontrar la clave en un nodo terminal (interior) y acabamos la búsqueda en ese momento, mientras que en los árboles B+ se ha de continuar aunque se encuentre la clave por la rama derecha de dicha clave hasta llegar a las hojas ya que es aquí donde se encuentra la referencia de donde se encuentra la información asociada a la clave.

2. **Inserción en árboles B+**

La inserción en árboles B+ es simple, similar a los árboles B. El único proceso de relativa dificultad es cuando se desea insertar una clave en una página llena (m=2d).

En este caso hemos de dividir la página en dos, distribuyendo las m+1 claves de la siguiente forma:

- Las d primeras claves en la página de la izquierda
- Las d+1 restantes en la de la derecha.

Una copia de la clave central sube a la página antecesora.

3. **Borrado en árboles B+**

La operación de borrado en los árboles B+ es más sencilla que en los árboles B y ello es debido a que las claves a eliminar están siempre en las hojas. Se distinguen los siguientes casos:

1. Si al eliminar una clave, m queda mayor o igual a d entonces hemos terminado la operación. Las claves de las páginas internas no se modifican aunque puedan contener una copia del elemento borrado.
2. Si al eliminar una clave, m queda menor que d entonces se redistribuyen las claves, tanto en el índice como en las hojas. Si al eliminar una clave y redistribuir las hojas el proceso debiera seguir hacia la raíz se continua, disminuyendo la altura del árbol.

2.2.2 Arbol R

Los árboles R son estructuras arborescentes equilibradas, similares a los árboles B+; sin embargo, en lugar de rangos de valores, se asocian cajas límite con cada nodo del árbol. El contenido de una caja límite de un nodo hoja es menor que el contenido de las cajas límite de sus nodos hijo. Las cajas límite asociadas con nodos hermano pueden solaparse.

Cuando realizamos la operación de inserción en un árbol de este tipo, se selecciona un nodo hoja para que lo contenga. Si el nodo ya está lleno, hay que llevar a cabo la división del nodo (y propagarla hacia arriba si hace falta), como vemos de manera parecida se realiza la operación de inserción en árboles B+. El algoritmo de inserción asegura que el árbol siga estando equilibrado y las cajas límite del nodo hoja y de los nodos internos sigan siendo consistentes.

2.2.3 Tablas de asociación

Una desventaja de la búsqueda de una clave es que hay que acceder a estructuras de datos, en algunas ocasiones complejas, para encontrarla.
Otra alternativa es la técnica de asociación o dispersión (hashing) que nos permite a través de una función localizar la clave deseada, esta estructura se empleará en la **técnica de función de asociación dividida**. Si suponemos que el índice correspondiente está distribuido en cubetas, la asociación nos proporcionará donde se encuentra la clave deseada.

La peor función de asociación es aquella que asigna todos los valores de la clave de búsqueda a una misma cubeta, luego la búsqueda de una clave concreta dentro de la cubeta requiere un tiempo bastante largo. Una función de asociación ideal es aquella que asigna una clave a cada cubeta.

Funciones de cálculo

Una función de asociación (hash) h es una función entre el conjunto de todas las claves de búsqueda y el conjunto de todas las direcciones de una cubeta.
El primer conjunto, que llamaremos K, puede ser tan grande como se quiera de tal manera que:

$$h(K) \longrightarrow \text{dirección}$$

Aunque el conjunto de las claves sea muy grande, el amacenado realmente es mucho más pequeño. Para que una función hash sea eficiente han de cumplirse los siguientes

requisitos:

1. El tiempo de búsqueda promedio de una clave ha de ser independiente del tamaño del fichero.
2. La distribución de los registros ha de ser uniforme y al azar.
    a. Uniformemente para que se asigne valores similares de la clave en la misma cubeta.
    b. Al azar para que cada cubeta tenga aproximadamente el mismo número de valores asignados.

La uniformidad no es fácil de conseguir ya que en una distribución de clave alfabética, existen más claves con G (García) que con X o Q (Quesada).
Por ello las funciones hash típicas hacen un cálculo sobre la representación binaria interna de la máquina de los caracteres de la clave de búsqueda. Este tipo de asociación se denomina abierta en contraposición de la cerrada, en la que solo existiría una cubeta y al función buscaría la clave dentro de esa cubeta. Esta última se utiliza para la creación de tablas de símbolos en compiladores y ensambladores, pero para sistemas de bases de datos se prefiere la asociatividad abierta porque la eliminación es complicada en la asociatividad cerrada por eso se usa para tablas fijas.

Funciones de asociación (Hash) estática

Ya vimos en apartados anteriores que una desventaja de los esquemas de índices es que debemos acceder a una estructura de índices para localizar los datos. La técnica de asociación nos permite evitar el acceso a una estructura de índices. El principio básico de la asociatividad es que, aunque el conjunto K de todas las claves de búsqueda posibles sea grande (quizá infinito), el conjunto { k1, k2,..,kn} de valores de clave de búsqueda realmente almacenados en la base de datos es mucho más pequeño que K. En el momento de hacer el diseño no sabemos cuáles son los valores de la clave de búsqueda que se van a almacenar en la base de datos, pero tenemos que hay demasiados valores posibles como para justificar la dirección de una cubeta a cada uno de ellos. Sin embargo, sabemos en el momento de hacer el diseño aproximadamente cuántos valores de clave de búsqueda van a ser almacenados en la base de datos. Elegimos el numero de cubetas para hacer la correspondencia con el número de valores de clave de búsqueda que esperamos tener almacenados en la base de datos. La función de asociación es la que define la asignación de valores de la clave de búsqueda a cubetas específicas.
Las funciones de asociación deben diseñarse con cuidado, una función de asociación bien diseñada da un tiempo de búsqueda promedio es una constante (pequeña), independiente del número de claves de búsqueda en el archivo. Esto se logra asegurándose de que, los registros se distribuyen uniformemente entre las cubetas. Para realizar una búsqueda del valor de clave de búsqueda Ki, simplemente calculamos h(ki) y buscamos la cubeta con esa dirección. La peor función de asociación posible asigna todos los valores de clave de búsqueda a la misma cubeta.
Luego, la gestión estática implica preveer inicialmente el número de claves posibles que se van a almacenar, puesto que las bases de datos crecen y decrecen sin poder saber, en muchos casos su tamaño durante el tiempo de uso, la utilización de la gestión estática suele estar limitada.

Funciones de asociación (Hash) dinámica

La mayoría de las bases de datos crecen conforme pasa el tiempo.
Existen varias técnicas de asociación que permiten modificar de manera dinámica la función de asociación para compensar el crecimiento o reducción de la base de datos. Estas técnicas se llaman dinámica. A continuación describiremos una forma da asociación dinámica llamada asociación extensible.

Asociación Extensible

La asociación extensible maneja los cambios en el tamaño de la base de datos dividiendo y fusionando cubetas conforme la base de datos crece y se reduce. Como resultado se mantiene la eficiencia de espacio. Además, puesto que la reorganización se realiza cada vez únicamente en una cubeta, el tiempo extra requerido es aceptablemente bajo.
La gran ventaja de la asociación extensible es que el rendimiento no disminuye según crece el archivo, requiriendo además un mínimo espacio adicional para albergar la tabla de direcciones de cubetas.
La desventaja es que hay que acceder primero a la tabla de cubetas antes que a la cubeta buscada (doble búsqueda) y esto conlleva una pequeña reducción en el rendimiento.

3.CONCEPTOS GENERALES

3.1 Ordenación de registros

La primera ordenación de registros es la secuencial del archivo por una clave de búsqueda permitiendo si los registros están ordenados una rápida recuperación de los mismos.

El puntero de cada registro apunta al siguiente en el orden de la clave de búsqueda, además los registros se almacenan físicamente en el orden que marca dicha clave.

Este orden es difícil mantenerlo cuando se manipula el archivo, es decir, cuando se insertan ó eliminan registros. Muchas de las dificultades que se presentan en el mantenimiento de una base de datos tienen su origen en las reorganizaciones periódicas a que se someten los registros.

3.2 Basadas en Vectores Dimensionales

3.2.1 Estructura de rejilla

Para acelerar el procesamiento de consultas con varias claves de búsqueda las cuales pueden implicar una o más operaciones de comparación una estructura que puede emplearse es la de archivos en retícula ó estructura de rejilla.

Una estructura de rejilla para estructuras que usan dos claves de búsqueda es un vector bidimensional indexado por los valores de las claves de búsqueda. El vector bidimensional de la figura 1 se llama **array en retícula** y los unidimensionales se llaman escalas lineales. El archivo en retícula tiene un único array en retícula y una escala lineal para cada atributo de la clave de búsqueda.

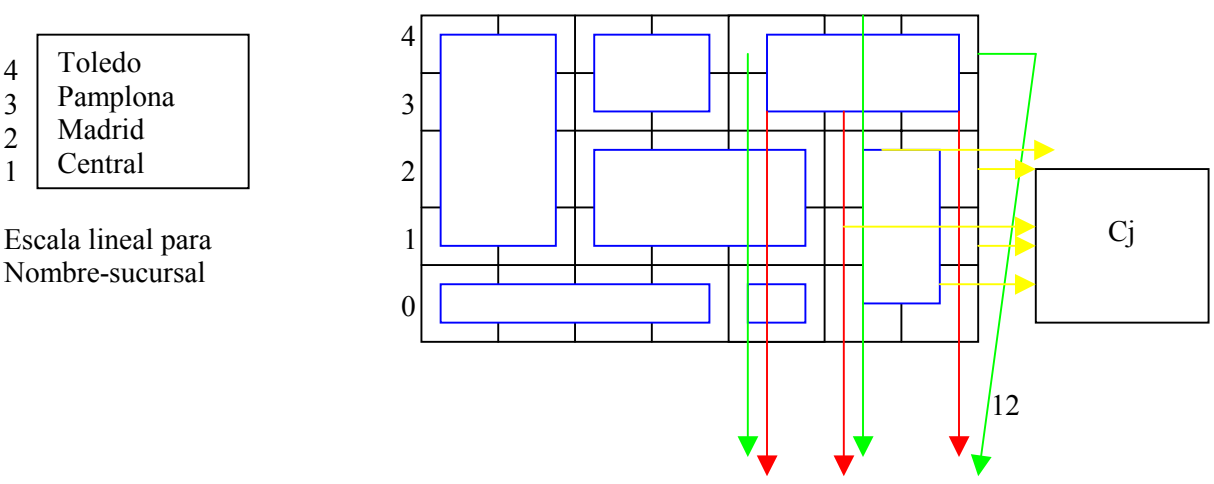
Las claves de búsqueda se asignan a las celdas como se describe a continuación. Cada celda en el array en retícula tiene un puntero a un cajón que contiene valores de las claves de búsqueda y punteros a los registros. Sólo se muestran en la figura 1 algunos de los cajones y punteros desde las celdas. Para conservar espacio se permite que varios elementos del vector puedan apuntar al mismo cajón. Los recuadros señalados con color azul de la figura 1 señalan las celdas que apuntan al mismo cajón.

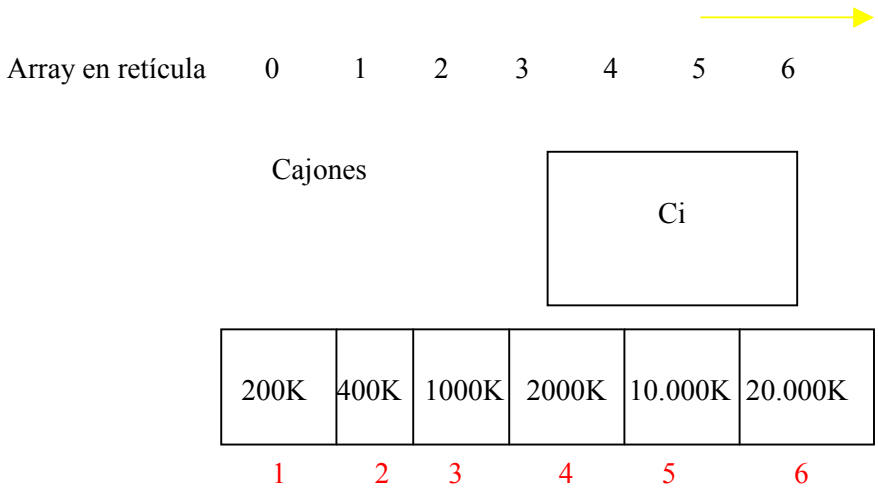
Supongamos que se quiere **insertar en el índice de archivo en retícula un registro** cuyo valor de la clave es ("Barcelona", 100.000.000), disponemos del archivo CUENTA, que vimos en el apartado 1.2 en el ejemplo2. Para encontrar la celda asignada a esta clave se localizan por separado la fila y la columna de la celda correspondiente.

1. Se utilizan las escalas lineales en Nombre-sucursal que era uno de los índices de los que disponía el archivo Cuenta, para localizar la fila de la celda asignada a la clave de búsqueda. Para ello se busca en el array para encontrar el menor elemento que es mayor que Barcelona. Según la figura1 es el primer elemento, así que la fila asignada a la clave de búsqueda es cero. Si fuera el **i-ésimo elemento**, la clave de búsqueda se asignaría a la **fila i-1**. Si la clave de búsqueda es mayor o igual que todos los elementos de la escala lineal, se le asignaría la **última fila**.

2. A continuación se utiliza la escala lineal en saldo (el otro índice del archivo), para encontrar de la misma manera qué columna le corresponde a la clave de búsqueda. En este caso, el saldo 100.000.000 tiene asignado la columna 6.

Figura 1: Archivo de rejilla para las claves Nombre-sucursal y Saldo del archivo CUENTA.





Escala lineal para saldo

Por tanto, el valor de la clave de búsqueda ("Barcelona" , 100.000.000) tiene asignado la celda de la fila 0, columna 6. De la misma manera, ("Daimiel", 12.000.000) tendría asignada la celda de la fila 1, columna 5. Ambas celdas apuntan al mismo cajón (como se indica en el recuadro de color azul), así que en los dos casos los valores de la clave de búsqueda y el puntero al registro están almacenados en el cajón con la etiqueta Cj de la figura1.

Para realizar la búsqueda que responda a la consulta del ejemplo2 citado en el apartado 1.2 , es decir, **"Encontrar los números de cuentas de la sucursal Pamplona con saldo igual a 200.000 pesetas "**, buscamos todas las filas con nombres de sucursal menores que Pamplona, utilizando la escala lineal de Nombre-sucursal. En este caso, estas filas son la 0, 1 y 2. La fila 3 y posteriores contienen nombres de sucursal mayores o iguales a Pamplona. De igual modo, se obtiene que sólo la columna 1 puede tener un saldo de 200.000. Así, solamente las celdas en la columna 1, filas 0,1 y 2 pueden contener entradas que satisfagan la condición de búsqueda.

A continuación hay que examinar todas las entradas de los cajones apuntados por estas tres celdas. En este caso solamente hay dos cajones, ya que dos de las celdas apuntan al mismo cajón, como se indica en los recuadros azules de la figura1. Los cajones podrían contener algunas claves de búsqueda que no satisfagan la condición requerida, de manera que cada clave de búsqueda del cajón se debe comprobar de nuevo para averiguar si satisface la condición o no. De cualquier modo, solamente hay que examinar un pequeño número de cajones para responder a la consulta planteada.

Las **escalas lineales se deben escoger** de tal manera que los registros estén uniformemente distribuidos a través de las celdas. Si el cajón llamémosle A queda lleno y se tiene que insertar una entrada en él, se asigna un cajón adicional B.

Si más de una celda apunta a A, se cambian los punteros a la celda de tal manera que algunos apunten a A y otros a B.

Las entradas en el cajón A y la nueva entrada se redistribuyen entre A y B basándose en las celdas que tengan asignados. Si sólo una celda apunta al cajón A, B se convierte en un cajón de desbordamiento de A.

Para mejorar el rendimiento en esta situación se tiene que reorganizar el archivo en

retícula con un array en **retícula extendido y escalas lineales extendidas.** Este proceso es como la expansión de la tabla de direcciones de los cajones en la **asociación extensible.**

Es conceptualmente sencillo **extender la aproximación del archivo en retícula a cualquier número de claves de búsqueda**. Si se quiere utilizar una estructura con n claves, hay que construir un array en retícula **n-dimensional,** que sea la rejilla, con n escalas lineales.

Así un simple índice de archivo en retícula o rejilla puede hacer el papel de tres índices distintos. Si cada índice se mantuviera por separado, los tres juntos ocuparían más espacio y el coste de su actualización sería mayor.

3.3 Basados en funciones de asociación

3.3.1 Función de asociación dividida

La asociación dividida es una extensión de la asociación que permite la indexación con varios atributos. Una estructura asociativa dividida permite consultas con atributos individuales, así como consultas que involucren varios atributos, aunque no permite consultas de rangos.

Supongamos que deseamos construir una estructura adecuada para consultas en el archivo **Depósito** que implican tanto a Nombre-cliente como a Nombre-sucursal. Construimos una estructura de asociación para la clave (Nombre-cliente, Nombre-sucursal).

La única diferencia entre la estructura que vamos a crear y las que vimos anteriormente es que imponemos una restricción adicional sobre la función de asociación h. Los valores de asociación se dividen en dos partes:

1. La primera depende sólo del valor de Nombre-cliente
2. La segunda depende sólo del valor de Nombre-sucursal.

La función de asociación se denomina **"dividida"** porque los valores de asociación se dividen en segmentos que dependen de cada elemento de la clave.

La **figura 2**, muestra un ejemplo de una función de asociación en la que los tres primeros bits dependen de Nombre-cliente y los tres últimos dependen de Nombre-sucursal. Así pues, los valores de asociación para (Hayes, Perryridge) y (Hayes, Mianus) coinciden en los tres primeros bits, y los valores de asociación para (Johnson,Downtown) y (Peterson,Downtown) coinciden en los tres últimos bits.

Figura 2. Función de asociación dividida para la clave (Nombre-cliente, Nombre-sucursal).

Valor de clave De búsqueda	Valor de asociación
(Green , Brighton)	101 111
(Hayes , Perryridge)	110 101
(Johnson , Downtown)	111 001
(Lyle , Perryridge)	000 101
(Peterson , Downtown)	010 001
(Smith , Mianus)	011 111
(Turner , Round Hill)	011 000
(Williams , Perryridge)	001 101
(Hayes , Mianus)	110 011

La función de asociación dividida de la Figura 1 puede usarse para responder a la consulta **"Encontrar el saldo de todas las cuentas de Williams en la sucursal Perryridge".**

Sencillamente calculamos **h(Williams , Perryridge)** y accedemos a la estructura de asociación. La misma estructura de asociación es apropiada para una consulta que implique únicamente una de las dos claves de búsqueda. Para encontrar todos los registros pertenecientes a la sucursal Perryridge, calculamos parte de la asociación dividida. Puesto que sólo tenemos el valor de Nombre-sucursal, sólo podemos calcular los tres últimos bits de la asociación. Para el valor Perryridge, estos tres bits son 101.

Accedemos a la estructura de asociación y examinamos aquellas cubetas para las que los tres últimos bits del valor de asociación son 101. En el ejemplo de la figura 2 accedemos a las cubetas para los valores 110 111, 000 101 y 001 101.

Como era el caso del archivo de rejilla, la asociación dividida se extiende a un número arbitrario de atributos. Podemos hacer varias mejoras en la asociación dividida si sabemos con qué frecuencia especificará el usuario cada uno de los atributos en una consulta. Las notas bibliográficas incluyen referencias a estas técnicas.

Utilizar la asociación dividida es lo mismo que usar archivos en retícula con dos diferencias:

1. La división por asociación se usa en valores de la clave, en vez de una división de un rango.
2. No hay directorio, en vez de esto el valor de la función de asociación proporciona directamente la dirección de un cajón.

Existen varias técnicas híbridas para el procesamiento de consultas multiclave. Estas técnicas podrían ser útiles en aplicaciones donde el implementador del sistema supiera que la mayoría de las consultas iban a ser de cierta manera.

3.4 Basadas en Indices

3.4.1 Organización de Listas Múltiples

Los archivos de listas múltiples se implementan con dos componentes:

1. Estructura con datos

Junto a cada atributo aparece un puntero al siguiente registro con igual valor del atributo.

Ejemplo:

Clave 1	Clave 2	Clave3
Numero-Artículo	Tipo-Producto	Color

		Sig		Sig
0132	Avión	Fin	Rojo	Fin
0133	Muñeca	Fin	Blanco	0134
0134	Auto.	0701	Blanco	Fin
0280	Cohete	Fin	Azul	0700
0700	Tren	Fin	Azul	Fin
0701	Auto.	Fin	Verde	Fin

2. Estructura Indice

Algunas búsquedas necesarias en una organización, como en el ejemplo planteado, se abrevian dividiendo las listas en secciones cortas, como en la **Fig. 3.** Se necesita entonces una entrada de índice para conocer el inicio de cada cadena. En ella hay un índice para cada atributo con una entrada por cada valor del atributo, que contiene una lista enlazada, ya definimos en apartados anteriores lo que es una lista enlazada, con los números de registro con dicho valor (una entrada por cada K registros).

La **Fig. 4** repite el archivo que venimos considerando, pero con las longitudes de lista limitadas a tres ítems (K=3) .

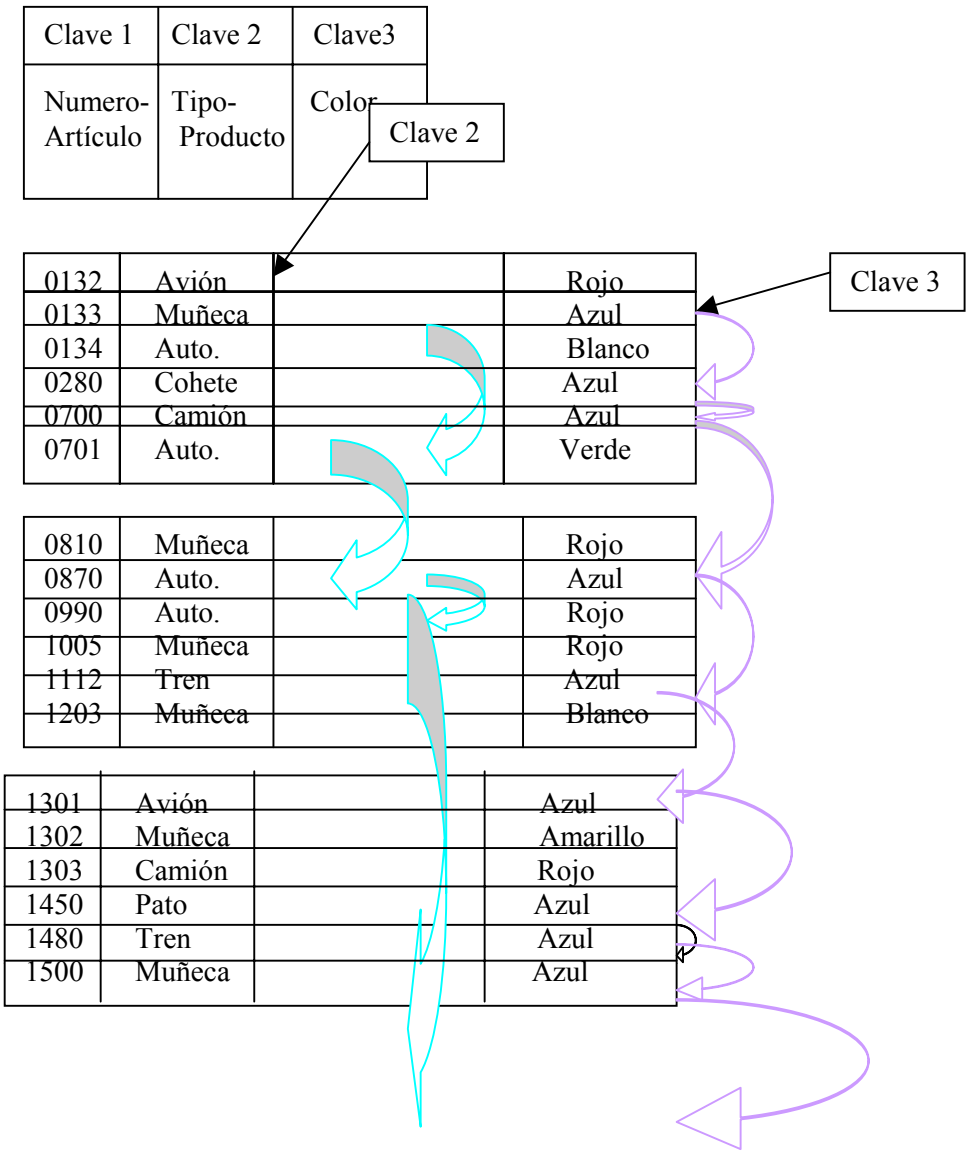
Esta organización se llama a veces organización de “listas múltiples” (lista: cadena de ítems), la longitud de las cadenas (K) es un parámetro arbitrario de la organización y se elige teniendo en cuenta las exigencias contrapuestas de

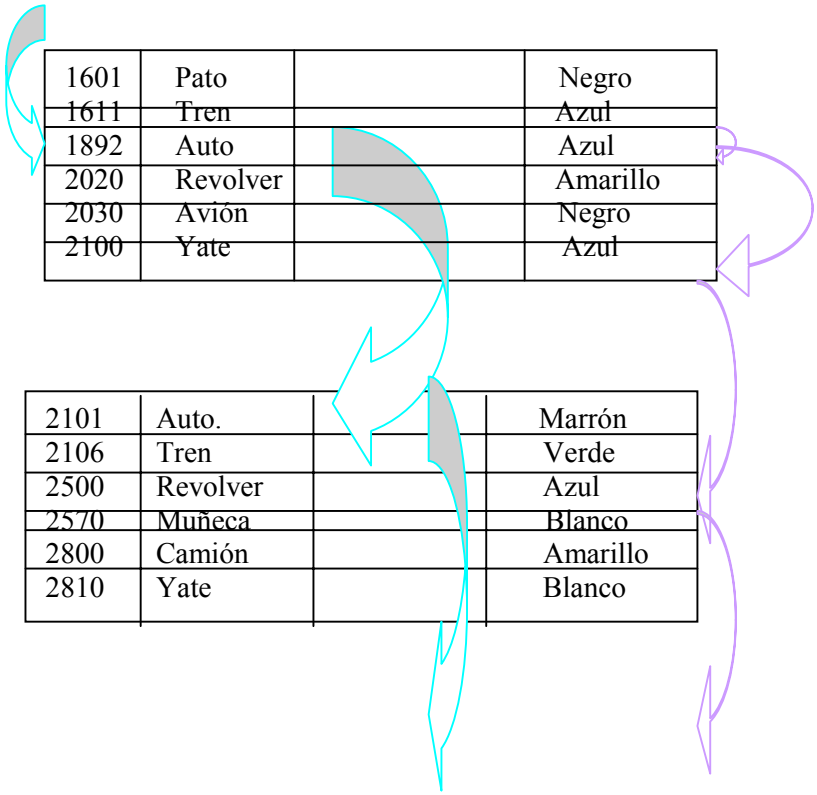
longitud de los índices y de duración de las búsquedas.

En el apartado 1.2 teníamos un ejemplo de un sistema de información, acerca de las ventas de una compañía en la que el usuario podría plantear ciertos interrogantes, ahora podrá responderse al usuario que nos pedía una lista de los automóviles que tenían Número-artículo comprendidos entre 1500 y 2000, sin el riesgo de tener que examinar muchos registros después del que corresponde al Número-artículo 1500 antes de dar con un eslabón de la cadena.
La computadora busca ahora la dirección del registro correspondiente al artículo 1500 y el índice AUTOMOVILES da la dirección de la cadena cuyo inicio precede a ese registro (la segunda cadena con flechas de color amarillo en la Fig. 4, siguiendo esta cadena se encuentra que sólo el artículo 1892 satisface el requerimiento).

Se advertirá que el diseñador debe decidir cuál es la combinación óptima en cada caso de número de entradas de índice y la rapidez, con que puede localizarse un ítem dado.

Fig. 3 :



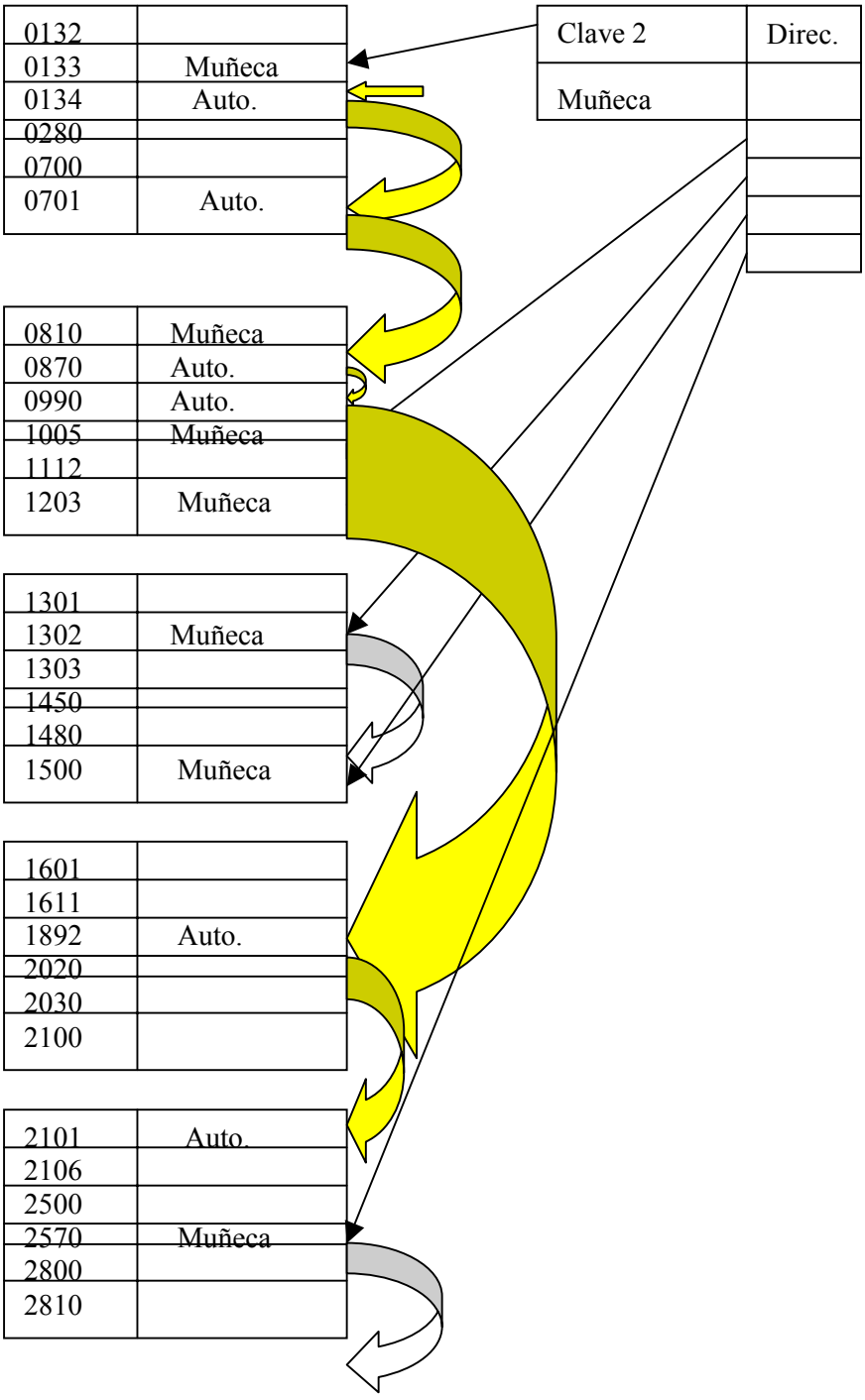


1601	Pato		Negro
1611	Tren		Azul
1892	Auto		Azul
2020	Revolver		Amarillo
2030	Avión		Negro
2100	Yate		Azul

2101	Auto.		Marrón
2106	Tren		Verde
2500	Revolver		Azul
2570	Muñeca		Blanco
2800	Camión		Amarillo
2810	Yate		Blanco

Fig. 4:

Clave 1	Clave 2
Número-Artículo	Tipo-Producto


0132
0133 Muñeca
0134 Auto.
0280
0700
0701 Auto.
0810 Muñeca
0870 Auto.
0990 Auto.
1005 Muñeca
1112
1203 Muñeca
1301
1302 Muñeca
1303
1450
1480
1500 Muñeca
1601
1611
1892 Auto.
2020
2030
2100
2101 Auto.
2106
2500
2570 Muñeca
2800
2810
Clave 2
Direc.
Muñeca

3.4.1.1 Multilista Celular

En los dos componentes de Lista Múltiple veíamos como las longitudes de las listas se limitaban de manera que no se extendiesen más allá de ciertas fronteras o zonas del hardware. Evitando así las largas búsquedas en que se incurre al seguir una lista de una zona a otra. Hemos visto que los registros se hallaban divididos en 5 grupos o zonas del hardware , estas zonas pueden ser cilindros, archivos de discos, tarjetas magnéticas, etc. En la **Figura 5** veremos el mismo archivo del fabricante de juguetes, utilizado en la figura 4, pero ahora las listas están previstas de modo que ninguna de ellas pasa de una zona a otra.

Llamamos multilista celular a aquellas que están sujetas a esta restricción. Hay una entrada de índice para cada valor de clave indizable y para cada zona de hardware que contiene ese valor. Según nuestro ejemplo, el hardware de la zona 3 no contiene ningún registro de muñeca y no hay por lo tanto entrada para esta zona en la tabla de muñecas.

Ejemplo:

Si queremos responder a un pedido de **Listado de muñecas Negras,** se examinarán :

1. Los índices para ver qué zonas del hardware contienen el atributo "Negro" y el atributo "muñeca". Si observamos la figura 5 nos damos cuenta que la zona 4 no contiene el atributo muñeca, y las zonas 1,3,5 no tienen el atributo "negro". Por lo tanto, solo hay que explorar la zona 2, pero sólo tiene dos registros con el atributo "negro".
2. Por lo tanto solo se examinará los registros que contienen el atributo negro para ver si hay muñecas negras.

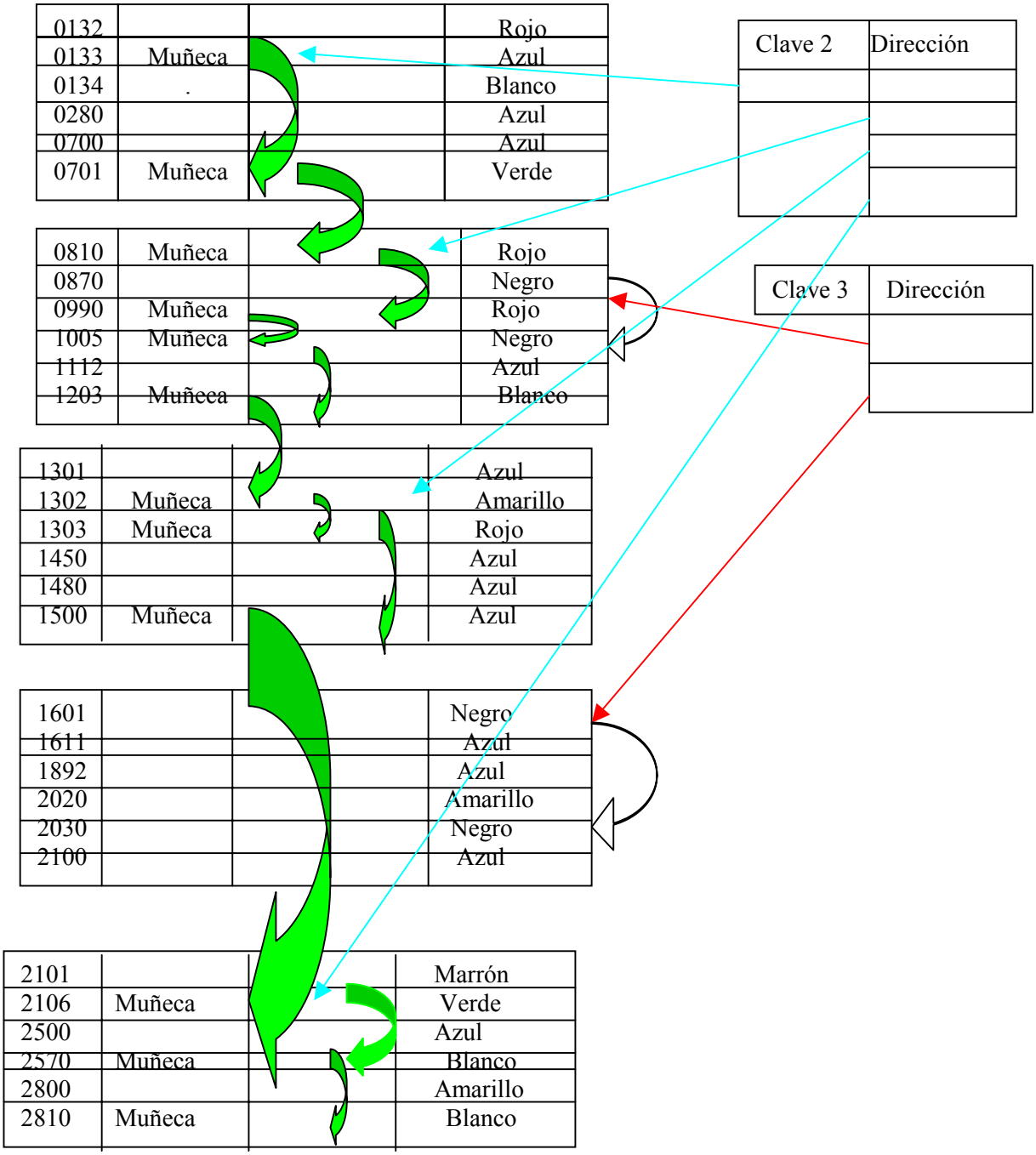
Multilista Celular en Paralelo

Esta técnica es un caso particular de Multilista celular:

Otra manera de distribuir los registros consiste en desparramarlos en módulos que pueden ser leídos simultáneamente, diseñando la distribución de manera de maximizar el número de pedazos de lista que se recorren en paralelo. Esta forma de operación se encontrará seguramente con mayor frecuencia en los sistemas del futuro que deban permitir la rápida exploración de grandes bases de datos.

Vamos a ver la Figura 5:

Clave 1	Clave 2	Clave 3
Numero-Artículo	Tipo-Producto	Color


0132
0133 Muñeca
0134 .
0280
0700
0701 Muñeca
Rojo
Azul
Blanco
Azul
Azul
Verde
Clave 2
Dirección
0810 Muñeca
0870
0990 Muñeca
1005 Muñeca
1112
1203 Muñeca
Rojo
Negro
Rojo
Negro
Azul
Blanco
Clave 3
Dirección
1301
1302 Muñeca
1303 Muñeca
1450
1480
1500 Muñeca
Azul
Amarillo
Rojo
Azul
Azul
Azul
1601
1611
1892
2020
2030
2100
Negro
Azul
Azul
Amarillo
Negro
Azul
2101
2106 Muñeca
2500
2570 Muñeca
2800
2810 Muñeca
Marrón
Verde
Azul
Blanco
Amarillo
Blanco

3.4.2 Lista Invertida

En los archivos de listas múltiples con listas de longitud controlada, ésta puede ir desde un eslabón por lista hasta un longitud suficiente para incluir todos los ítems. En el primer caso hablamos de una lista invertida, en ésta hay una entrada de índice de cada clave, para cada registro, como se ve en la **Fig. 5**.

De esta manera, se requiere un único acceso a los datos para cada registro que es la respuesta a una averiguación. Suponiendo que es posible examinar rápidamente los índices secundarios, la organización de lista invertida da la respuesta más rápida a las averiguaciones de tiempo real, porque no hay lista que seguir. En cambio, los índices pueden resultan tan extensos (habrá que almacenarlos en dispositivos secundarios) que su propia organización llega a constituir un problema).

En las listas invertidas, se indizan todos los valores de atributo de la clave secundaria, de modo que no hay cadenas.

Debe observarse que el archivo de listas invertida no es sino un caso particular de los archivos de listas múltiples, es decir, se trata de listas múltiples cuando k=1, en el índice se incluyen todos los registros o valores de la clave.

Se emplean en los sistemas de gestión de bases de datos (SGBD) Documentales llamados SRI (Sistemas de Recuperación de Información). Los ficheros de datos equivalen a una función del estilo:

Fichero (# registro) ⟶ (atributo : valor) , (atributo : valor) ...

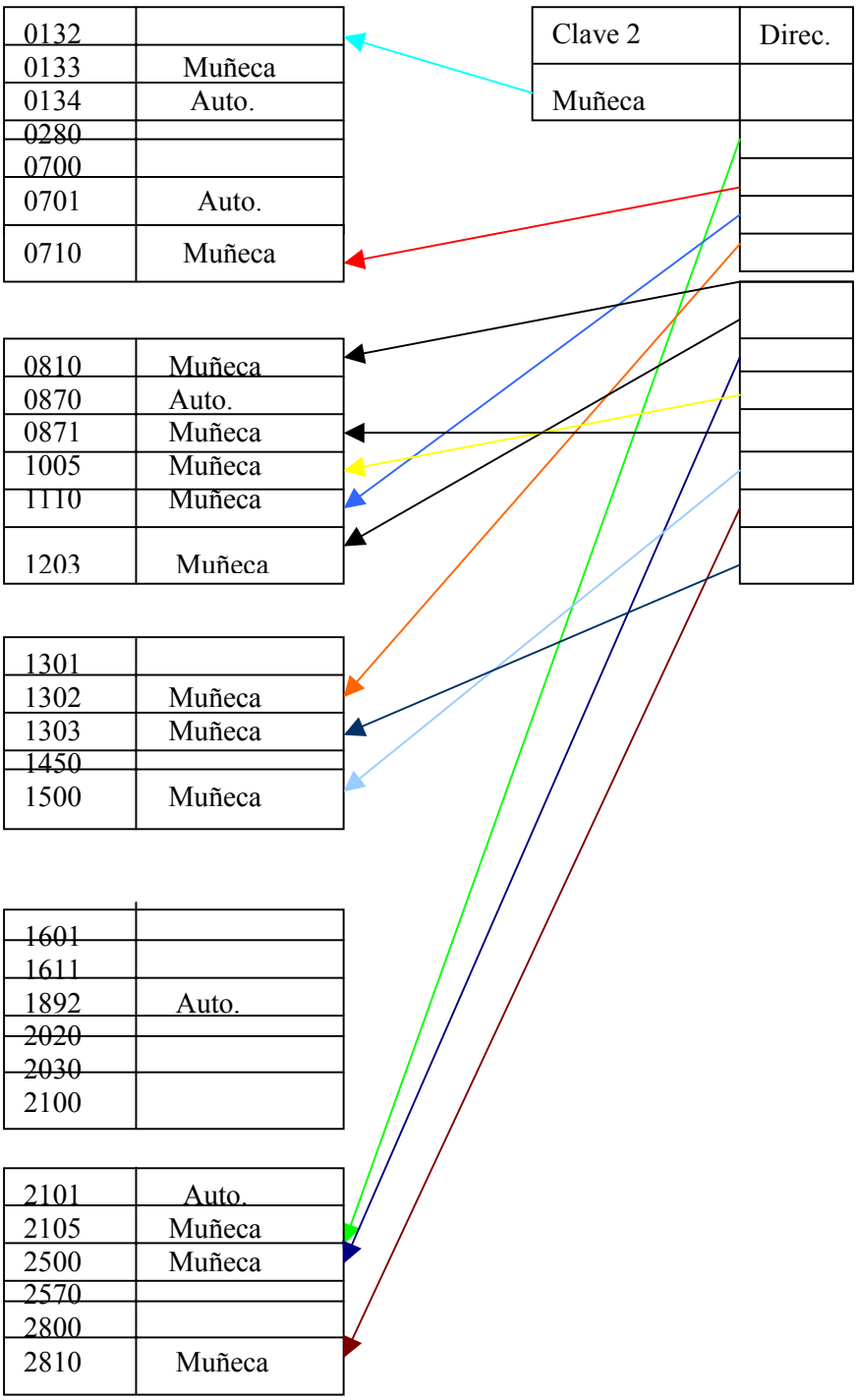
El fichero de índice representa la función inversa (de ahí el nombre de ficheros invertidos):

Fichero (atributo : valor) ⟶ # reg. # reg ...

Ejemplo:

Fig. 5: Listas Invertidas, se indizan todos los valores de atributo de la clave secundaria, de modo que no hay cadenas.

Clave 1	Clave 2
Número-Artículo	Tipo-Producto


0132
0133 Muñeca
0134 Auto.
0280
0700
0701 Auto.
0710 Muñeca
0810 Muñeca
0870 Auto.
0871 Muñeca
1005 Muñeca
1110 Muñeca
1203 Muñeca
1301
1302 Muñeca
1303 Muñeca
1450
1500 Muñeca
1601
1611
1892 Auto.
2020
2030
2100
2101 Auto.
2105 Muñeca
2500 Muñeca
2570
2800
2810 Muñeca
Clave 2
Direc.
Muñeca

3.4.2.1 Indices Indirectos

Los punteros que aparecen en las Figuras representadas en ejemplos anteriores, dan la dirección del registro a que apuntan. No obstante, los punteros de clave secundaria podrían ser punteros **"indirectos"**, es decir, dar la clave primaria del registro señalado, mejor que su dirección.

Este método de direccionamiento presenta el inconveniente de que el seguimiento de los punteros lleva más tiempo, puesto que es necesario recurrir al mecanismo de direccionamiento primario para convertir las claves primarias en direcciones. El uso de punteros indirectos tales como vimos en la Fig. 26.8 del apartado anterior, no causa pérdidas de tiempo graves en el caso de las Listas Invertidas. Llamaremos "**índice indirecto**" a aquel que contiene punteros indirectos.
El uso del índice indirecto ofrece *dos ventajas*:

1. Si el registro se cambia de lugar físico, no hay necesidad de actualizar el índice secundario, por lo general el uso de índices secundarios indirectos simplifica sustancialmente la tarea de mantenimiento del archivo.

2. Pueden responderse a muchos interrogantes sin tener que leer ningún registro de datos.

   Ejemplo: Según el ejemplo expuesto anteriormente, obtener el **listado de los automóviles cuyo número de artículo esté comprendido entre 1500 y 2000**, si la lista invertida de la Fig.5 contuviese números de artículos, en lugar de direcciones, bastaría con explorar el índice para contestar a la pregunta.

Para facilitar la atención de las averiguaciones que involucran claves múltiples, inclúyese una cuenta de los ítems en cada entrada, de modo que puedan emitirse en primer lugar las listas mas cortas.

3.4.2.2 Listas Invertidas Celulares

La principal desventaja de las listas invertidas reside en el tamaño de los índices. Es posible reducir este tamaño, en algunos archivos, almacenando no la dirección de cada registro, sino una indicación de la zona de hardware en la que reside aquél, en tal caso será necesario inspeccionar en la que reside aquél. Hay por lo tanto un problema entre el tamaño del índice y el número de registros que hay que leer. Este tipo de organización se llama organización de "**listas invertidas celulares**".

Con el objeto de minimizar el espacio ocupado por el índice, se hace referencia en éste a la celda, o zona de hardware, sea por medio de un número binario que pueda ser convertido en una dirección, sea por un único bit en una matriz de bytes, como la de la **Fig. 6**

Siguiendo con el ejemplo del sistema de información acerca de las ventas de una

compañía, para atender por ejemplo el pedido de una lista de todos los "patos negros", los bites correspondientes a "negro" 01010 y los bytes correspondientes a "pato" 00110, pueden combinarse en una instrucción **o inclusive** para obtener 00010, lo que revela que sólo la zona de hardware 4 contiene registros sobre patos negros. Podemos verlo en la Fig. 6

Fig. 6: Listas Invertidas Celulares

Clave 1	Clave 2	Clave3
Numero-Artículo	Tipo-Producto	Color

Zona de Hardware

		1	2	3	4	5
Clave 2	Auto.	1	1	0	1	1
	Muñeca	1	1	1	0	1
	Pato	0	0	1	1	0
	Elefante	0	0	1	0	0
	.					
	.					
Clave 3	Negro	0	1	0	1	0
	Azul	1	1	1	1	1
	Marrón	0	0	0	0	1
	Verde	1	1	0	0	1
	.					
	.					

Zona de Hardware 1

0132	Avión		Rojo
0133	Muñeca		Azul
0134	Auto.		Blanco
0280	Cohete		Azul
0700	Camión		Azul
0701	Auto.		Verde

Zona de Hardware 2

0810	Muñeca		Rojo
0870	Auto.		Azul
0990	Auto.		Rojo
1005	Muñeca		Rojo
1112	Tren		Azul
1203	Muñeca		Blanco

Zona de Hardware 3

1301	Avión		Azul
1302	Muñeca		Amarillo
1303	Camión		Rojo
1450	Pato		Azul
1480	Tren		Azul
1500	Muñeca		Azul

Zona de Hardware 4

1601	Pato		Negro
1611	Tren		Azul
1892	Auto		Azul
2020	Revolver		Amarillo
2030	Avión		Negro
2100	Yate		Azul

Zona de Hardware 5

2101	Auto.		Marrón
2106	Tren		Verde
2500	Revolver		Azul
2570	Muñeca		Blanco
2800	Camión		Amarillo
2810	Yate		Blanco

3.4.2.2.1 Listas Invertidas Celulares En Paralelo

Las posiciones de las celdas podrán seleccionarse de forma que dos o más de ellas puedan explorarse en paralelo, con el fin de reducir los tiempos de respuesta del sistema.

El archivo de claves múltiples puede distribuirse, por ejemplo, entre varios paquetes de discos, ocupando sólo unos pocos cilindros en cada paquete y con datos diferentemente organizados en los demás cilindros.
Las celdas a que se refieren los índices son cilindros, varios de los cuales pueden ser explorados en paralelo. Cada cilindro posee un índice propio para acelerar la búsqueda dentro de ese cilindro y para que no sea necesario leer todas las pistas de éste.

Para muchos archivos el cilindro sería una celda demasiado grande y se usarían entonces como celdas grupos menores de pistas dentro del mismo cilindro.

3.4.3 Cadenas en el Indice

Las cadenas consideradas hasta el momento se han armado siempre con punteros escritos en los registros de datos; es decir, hemos hablado de cadenas empotradas o de punteros empotrados.

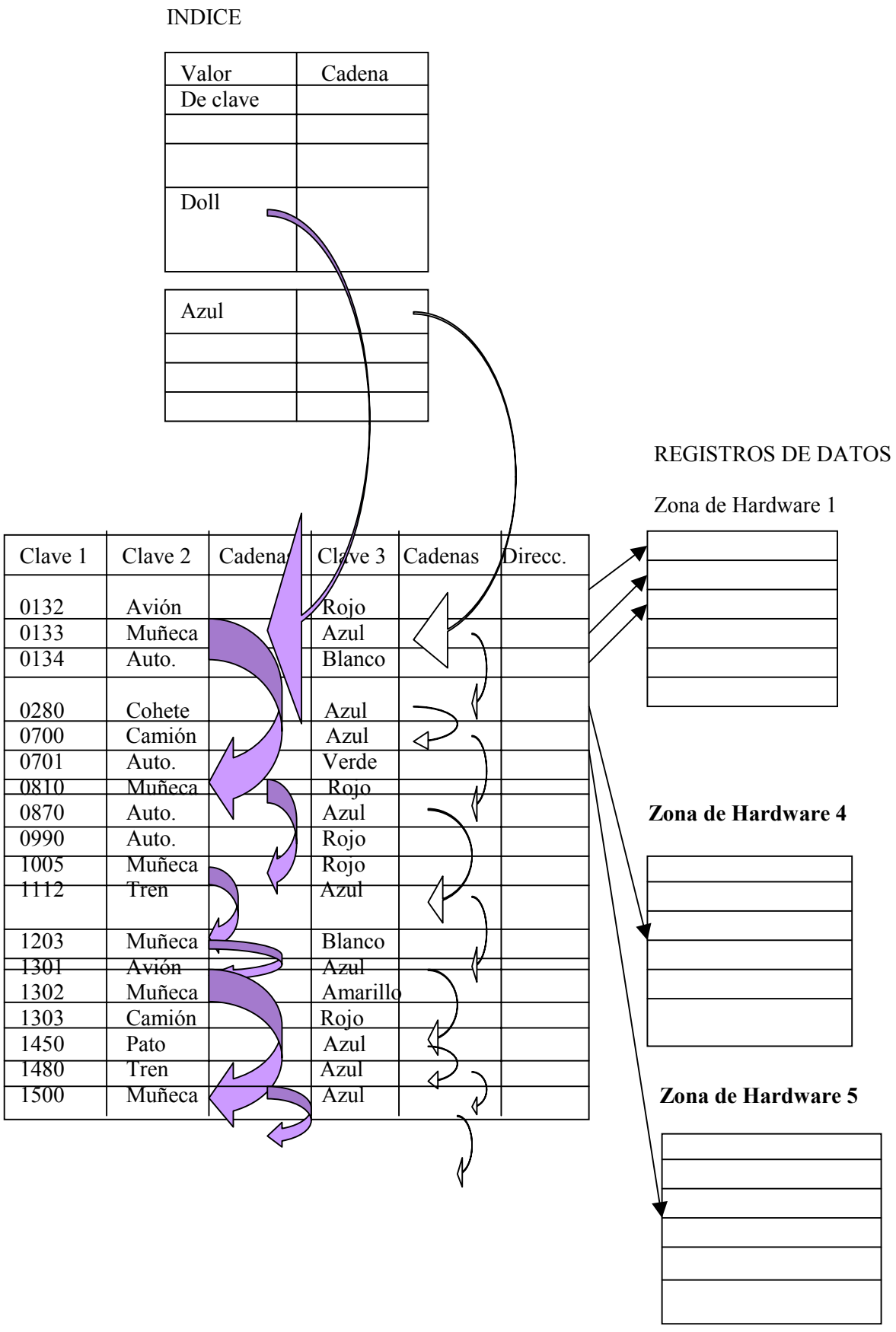
Desventaja de este tipo de cadena es el tiempo de acceso en que se incurre al seguir cada cadena.
Mejor que empotrar las cadenas en los registros de datos es, a menudo, introducirlas en los índices, donde será posible seguirlas sin tener que leer largos registros de datos.

Hay varias maneras de expresar las relaciones entre registros en los índices, la **Fig. 7** ejemplifica una organización con cadenas ininterrumpidas algo similar a la **Fig. 3 ,**que hemos visto en Listas multiples, excepto porque ahora las cadenas están almacenadas en el índice de la clave primaria en lugar de tendidas a través del archivo de datos.
Estas nuevas cadenas se siguen con muchas menos lecturas y búsquedas de pista que las de la Fig.3 anteriormente mencionadas.

Las cadenas de índice se dividirán, posiblemente, en trozos confinados a determinadas zonas del hardware.
La estructura del índice y los métodos de compactación que esa estructura admite constituyen un factor de primer orden en lo que se refiere a la eficiencia.

Ejemplo:

Fig.7: Cadenas de índice, las cadenas están en el índice, no en los registros de datos. Es posible sacar las claves de los registros de datos y organizar un archivo de tuplas de claves de cualquiera de las formas ilustradas por las figuras de apartados anteriores.


INDICE
Valor De clave
Cadena
Doll
Azul
REGISTROS DE DATOS
Zona de Hardware 1
Clave 1
Clave 2
Cadena
Clave 3
Cadenas
Direcc.
0132 Avión Rojo
0133 Muñeca Azul
0134 Auto. Blanco
0280 Cohete Azul
0700 Camión Azul
0701 Auto. Verde
0810 Muñeca Rojo
0870 Auto. Azul
0990 Auto. Rojo
1005 Muñeca Rojo
1112 Tren Azul
1203 Muñeca Blanco
1301 Avión Azul
1302 Muñeca Amarillo
1303 Camión Rojo
1450 Pato Azul
1480 Tren Azul
1500 Muñeca Azul
Zona de Hardware 4
Zona de Hardware 5

3.4.4 Arboles R

La estructura de almacenamiento denominada árbol R resulta útil para la creación de índices de rectángulos y de otros polígonos, como consecuencia puede servir para el procesamiento de consultas con varias claves de búsqueda.

Ejemplo: Encontrar todos los polígonos que intersectan a uno dado.

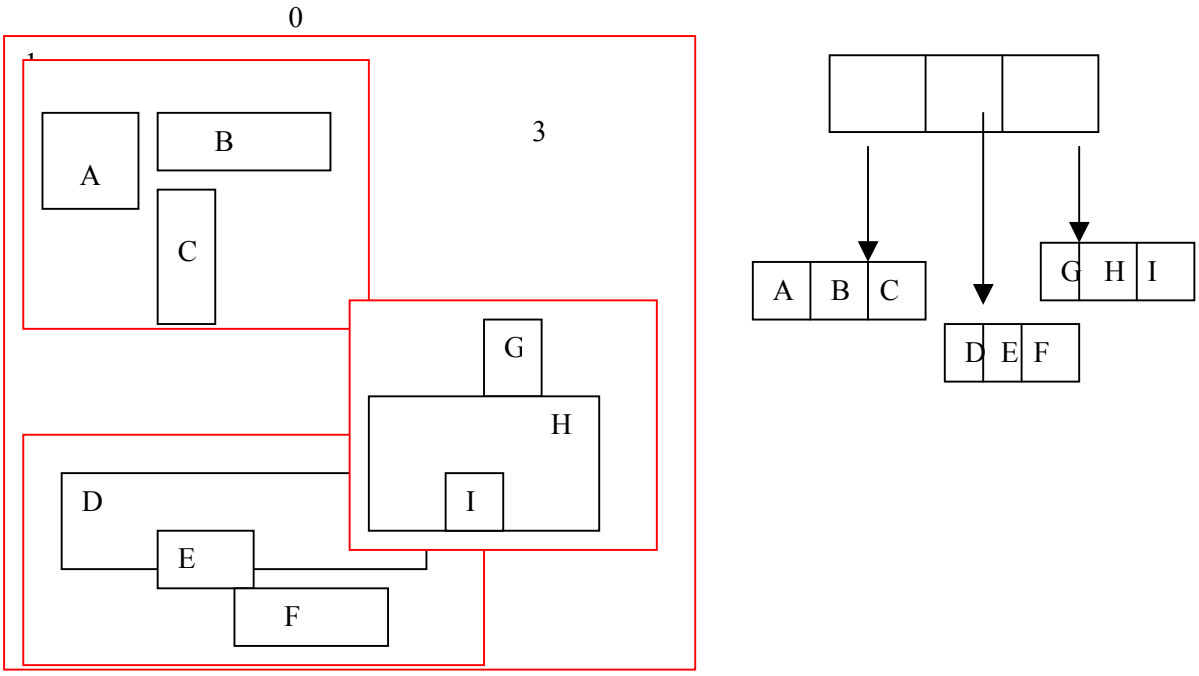
Como hemos comentado, éstos son útiles para la creación de rectángulos y otros polígonos y según las características de estos árboles comentados en apartados anteriores, los polígonos sólo se guardan en los nodos hoja . La caja límite de un nodo hoja es el menor rectángulo (paralelo a los ejes) que contiene las cajas límite de sus nodos hijo.

En la **figura 8** se muestra un ejemplo de un conjunto de rectángulos y las cajas límite (dibujadas de color rojo) de los nodos de un árbol R para el conjunto de rectángulos. El propio árbol R aparece a la derecha. Observamos que las cajas límite se muestran con espacio sobrante en su interior para que destaquen en la figura. En realidad, las cajas serían menores y cada lado tocaría al menos alguno de los objetos de las cajas límite de un nivel inferior.

La búsqueda de los polígonos que contienen un punto tiene que seguir, por tanto, todos los nodos hijos cuyas cajas límite asociadas contengan ese punto; como consecuencia, puede que haya que buscar por varios caminos.

Lo mismo ocurre en el ejemplo que hemos citado de encontrar todos los polígonos que intersectan a uno dado tiene que descender por todos los nodos en los que el rectángulo asociado intersecta el polígono.

Figura 8:



4. VENTAJAS E INCONVENIENTES DE LAS DISTINTAS TÉCNICAS

Según los diversos tipos de organización de claves múltiples, comentadas en apartados anteriores, vamos a considerar que tenemos una base de datos de solo 28 registros, **Fig.9** . En cada registro hay cinco atributos indizables:

- *Ao*: número de empleado (identificador único)
- *A1*: nombre del empleado
- *A2*: departamento en el que trabaja el empleado
- *A3*: código de capacitación
- *A4*: el salario

La parte restante y mayor del registro contiene detalles no indizables. Los registros, en casi todos los casos, están dispuestos según la secuencia ascendente de la clave primaria que es Numero-Empleado y ocupan cuatro zonas de hardware.
Las direcciones de los registros se escriben de la forma X.Y, donde:

- *X:* número de la zona hardware
- *Y:* número de orden del registro en esa zona

Depende de la técnica elegida se representará un diagrama de organización de estos datos. Admiten numerosas variantes. El lector debe imaginar cada diagrama extendido a un archivo más voluminoso, distribuido quizás en varios módulos de discos, con más claves secundarias y más valores de atributo.

Debe pensar también en cómo pueden diferir cada una de las técnicas en lo que se refiere al tiempo de respuesta en que incurren para responder a interrogantes por claves múltiples.
Además debe considerar la cuestión del mantenimiento o si requieren mucho tiempo, mientras que otras, el mantenimiento es más sencillo y rápido.

El atributo Salario cubre una gama continua de valores, para construir índices para este atributo, se lo "cuantiza" en rangos discretos, de $250 de longitud.
De igual modo, el atributo Nombre se distribuye en intervalos alfabéticos, en algunas ilustraciones se toma la primera letra del apellido como entrada para el índice, en la práctica sería preferible dividir los ítems de datos Salario y Nombre en intervalos elegidos de modo que cada uno de ellos contenga aproximadamente el mismo número de registros, o aproximadamente cadenas de igual longitud, en lugar de tomar como base, según el criterio adoptado, la igualdad de separación de claves.

Se supone que en el índice de nombre se usan entradas de longitud fija, por esta razón, la clave secundaria Nombre se trunca, limitándola a las cuatro primeras letras del apellido seguidas por las dos primeras iniciales. El resto del nombre de cada empleado se almacena en la parte no indizable del registro. Ocasionalmente, dos personas podrán tener así la misma clave Nombre; entonces será necesario leer los dos registros que les corresponden para saber quién es quien.

Supongamos que un archivo contiene 100.000 registros de 500 bytes cada uno, cada registro tiene una clave primaria de 10 bytes en total; el lector deberá considerar las cinco

preguntas siguientes y, teniéndolas presentes, ver las ventajas e inconvenientes de cada una de las técnicas :

1. ¿ Qué técnicas, entre las ilustradas, serían las más adecuadas desde el punto de vista de la rápida respuesta a las averiguaciones en un sistema de tiempo real, suponiendo que se usan discos como dispositivos de almacenamiento?

2. Suponemos que se dispone de una memoria de estado sólido de dos millones de bytes de capacidad, para servir como almacén intermedio en las operaciones de la base de datos. ¿Qué técnicas resultarían más adecuadas para un sistema de respuesta rápida?

3. Buscar la forma de insertar y eliminar registros en cada una de las técnicas comentadas. ¿Qué técnicas tienden a facilitar el mantenimiento, y cúales serían inconvenientes desde el punto de vista de esta operación?

4. En cada una de las técnicas, analizaremos el efecto de la destrucción de un puntero. ¿Qué tendríamos que hacer para reconstruir el puntero dañado? ¿Cuál sería la mejor manera de proteger la organización contra el efecto de la destrucción de un puntero?

5. Teniendo en cuenta el mantenimiento, la posibilidad de destrucción de los punteros, y los requerimientos de respuesta rápida, ¿Qué técnica debería preferirse en el caso de un archivo moderadamente volátil?

Vamos a ir viendo las ventajas e inconvenientes de cada una de las técnicas que hemos visto, considerando dichas preguntas.

Fig.9:

Ao	A1	A2	A3	A4	Detalles no indizados
Num.de Emplea.	Nombre	Depto	Cod. de Capac.	Salario	
07642	MARTJT	220	PL	1900	
07463	GREEJW	119	SE	2700	
07650	HALSPD	210	SE	2000	
07668	FEINPE	220	PL	1950	
07670	SCHAWE	119	AD	3100	
07611	MARSJJ	119	FI	1200	
07672	ALBEHA	210	SE	2100	
07700	LONDAJ	220	AD	3000	
07702	ANDEWF	119	FI	1000	
07710	MARTCH	220	PL	1750	
07715	FLINGA	119	AD	3000	
07716	MERLCH	220	FO	2200	
07760	JONEKB	119	PL	2200	
07761	REDFBB	119	SE	2650	
07780	BLANJE	220	FO	2100	
07805	ROPEES	220	PL	1900	
07806	KALNTD	119	MA	2300	
07815	EDWARB	220	PL	2040	
07850	DALLJE	119	FI	1050	
07883	JONETW	210	SE	2010	
07888	WEINSH	119	MA	2450	
07889	KLEINM	220	PL	1830	
07961	FREIHN	220	PL	1780	
07970	MANKCA	119	MA	2410	
07972	FIKETE	210	SE	2500	
08000	SCHEDR	210	FI	2100	
08001	FLANJE	119	PL	1920	
08100	JOOSWE	210	SE	3150	

Archivo en Retícula

Las estructuras de rejilla proporcionan una mejora importante en el tiempo de procesamiento para las consultas de claves múltiples. Sin embargo, requieren cierta cantidad de espacio y tiempo extra en las inserciones y eliminaciones de registros, es decir, implican un gasto adicional de espacio (el directorio en rejilla podría llegar a ser grande), así como degradación del rendimiento al insertar y borrar registros. Además, es difícil elegir una división en los rangos de las claves para que la distribución de las claves sea uniforme.

Si las inserciones en el archivo son frecuentes, la reorganización se tendrá que realizar periódicamente y eso puede tener un coste mayor.

Función de Asociación

Esta técnica permite encontrar la dirección de un elemento de datos directamente mediante el cálculo de una función con el valor de la clave de búsqueda del registro deseado. Ya que no se sabe en tiempo de diseño la manera precisa en la cual los valores de la clave de búsqueda se van a almacenar en el archivo, una buena función de asociación a elegir sería aquella que distribuya los valores de la clave de búsqueda a los cajones de una manera uniforme y aleatoria.

Listas Múltiples

Cada entrada del índice tiene asociado, por consecuencia, un número variable de direcciones de cadena.

El tiempo necesario para encontrar un registro a partir de una clave secundaria resultará así en muchos casos, pequeño.

Ejemplo : Para producir una lista de los empleados que tienen el código de capacitación MA en el departamento 210 la inspección de entradas de índice solo correspondería a la celdas que tienen esa combinación.

En este tipo de técnica tenemos una recuperación rápida, pero necesitamos un índice muchas veces mayor que en otras técnicas, en el caso de estructura índice, sin embargo en el caso de estructura con datos tenemos una recuperación lenta y un índice pequeño.

Listas Múltiples Celulares

Las listas se dividen en fragmentos, cada uno de los cuales reside en una zona del hardware. El contenido de una zona puede moverse sin mover el mecanismo de acceso; por lo tanto, se abrevia el tiempo total de búsqueda, es decir, al seguir una cadena no se requieren operaciones de búsqueda.

Listas Múltiples Celulares en Paralelo

La zonas del hardware se organizan de modo que se las pueda explorar simultáneamente, reduciéndose así el tiempo total de búsqueda. Entonces si los registros almacenados en las distintas celdas podemos leerlos simultáneamente, es entonces viable este tipo de inspección.

Lista Invertida

Hablamos de lista invertida cuando tenemos listas de longitud controlada, el índice contiene una entrada o un conjunto de entradas para cada valor de clave.

No hay cadenas en los registros de datos, pero los índices son sustancialmente más extensos, entonces en este tipo de técnica tenemos un índice mayor que en cualquier otro caso, para esto disponemos de unas "**técnicas de compresión de la clave"** ,que las comentaremos brevemente, en el apartado siguiente, sin embargo tenemos una recuperación más rápida.

En el índice de nombres se incluyen las claves de nombres completas en lugar de la inicial de cada apellido, como se hizo en las organización anterior. De modo similar, el índice de salario tiene como entradas los salarios reales en lugar de intervalos.

La inclusión de las claves completas en lugar de intervalos no aumenta el número de entradas de la lista invertida, pero si el tamaño del índice. Aumenta también, al mismo tiempo, el poder de resolución del índice y el número de averiguaciones que pueden satisfacerse sin leer los registros de datos.

Listas Invertidas con direccionamiento Indirecto

La organización es similar a la de lista invertida, excepto porque los índices A2, A3 y A4 no contienen direcciones de máquina, sino un número de serie de índice basado en la secuencia que siguen las entradas en el índice A1.

El uso del índice indirecto ofrece dos ventajas :

1. Mantenimiento más fácil: Al reorganizar los archivos y modificar las direcciones de máquina de los registros de datos, no hay necesidad de volver a escribir los índices A2, A3 y A4, sino sólo el índice A1.
2. Algunos interrogantes se contestan sin leer registros de datos: Esta organización permite responder a más interrogantes utilizando sólo los índices, sin tener que ir a los archivos de datos.

Para realizar la segunda ventaja el índice A1 incluye los nombres completos de los empleados.

Ejemplo: Si queremos saber, cuáles son los nombres de los empleados del Departamento 119 que ganan $3.000 o más, la inspección de los índices de departamentos y de salarios demuestra que satisfacen las dos condiciones planteadas por el interrogante los números de índice, por ejemplo 9 y 26. El índice A1 da entonces los nombres de dichos empleados.

Podría haberse usado la clave primaria Ao en lugar del número de serie de índice, pero este es más corto.

Si la averiguación incluyese el registro completo de cada uno de esos dos empleados, el hallar las direcciones de esos registros tomaría más tiempo que en el caso de la organización de lista invertida.

Lista Invertida celulares

Con la intención de reducir los voluminosos índices de la organización de lista invertida es posible recurrir a la organización de listas invertidas celulares ,esta organización se emplean registros de índice que contienen tuplas de las claves que necesitamos. De modo que los registros de datos se podrán encontrar con cualquier método de direccionamiento.

Dentro de ésta técnica los registros de datos principales pueden permanecer inalterados mientras se ejecutan operaciones de mantenimiento en el archivo de índices, relativamente pequeño. Además este archivo puede almacenarse en dispositivos de acceso directo, más rápidos y costosos que los que se usan para los registros principales. Al tener tuplas de claves disminuye el número de registros de índice que deben leerse al responder a averiguaciones de claves múltiples. Pero entonces se necesita un índice de nombres aparte y más largo y se complicará el mantenimiento del archivo de registros de índice.

Cadenas en el Indice

Esta organización las claves secundarias se han sacado de los registros de datos, los campos correspondientes a las claves secundarias se almacenan en los índices y no en los registros de datos, como ya explicamos en apartados anteriores, las cadenas están en el índice, donde será posible seguirlas sin tener que leer largos registros de datos, es decir, tienen una búsqueda más rápida.

El requerimiento total de espacio de almacenamiento permite ir a los registros de datos, donde se las puede seguir más rápidamente que en los voluminosos registros de datos.

Las operaciones de mantenimiento se llevan a cabo en el índice, más bien que en el archivo de datos.

Una desventaja , consiste en que si se dañan accidentalmente algunos punteros, no siempre resulta posible reasociar un registro de datos con sus claves secundarias, por esta razón en algunos casos, las claves secundarias se almacenan también en los registros de datos.

Arbol R

La eficiencia de almacenamiento es mayor que en otras técnicas, dado que los polígonos sólo se guardan una vez y se puede asegurar fácilmente que cada nodo esté como mínimo semilleno. Sin embargo, las consultas pueden resultar más lentas, dado que hay que buscar por varios caminos.

Debido a su gran eficiencia de almacenamiento y su parecido con los árboles B, los árboles R se han adoptado en los sistemas de bases de datos que trabajan con datos espaciales.

5. TÉCNICAS DE COMPRESIÓN DE LA CLAVE

Hemos visto que algunas técnicas los índices pueden ser muy extensos como en el caso de la **Lista Invertidas**, para ello se utilizan técnicas de compresión de las entradas.

Algunos índices se dejan comprimir por medio de la codificación binaria, en otros casos se logra la compresión suprimiendo partes innecesarias de las claves.

Vamos a comentar tres técnicas:

1. Compresión por el extremo posterior de la clave
2. Compresión del extremo frontal y del extremo posterior de la clave
3. Análisis o Parsing

5.1 Extremo posterior de la clave

Todo lo que realmente se necesita para localizar un registro es un número suficiente de caracteres de alto orden como para permitir distinguirlo del primer registro del cubo siguiente, es decir, se truncan los caracteres posteriores de las claves mas altas en el nivel más bajo del índice.

5.2 Extremo anterior y posterior de la clave

En esta técnica además de truncan los caracteres posteriores se trunca también los primeros caracteres de las claves. Esta técnica es eficaz, porque los caracteres que no se prestan a la compresión por el extremo posterior, sí se prestan a la compresión por el extremo anterior.

5.3 Análisis

Con esta técnica es posible logra aún una compresión más pronunciada, el nivel de índice más elevado contiene una porción de la clave. El nivel inmediato inferior contiene los caracteres siguientes, pero no repite los caracteres que se hallan en el nivel superior precedente, sin embargo la desventaja que reporta esta técnica es que algunos de los bloques de índices contendrán bastante más entradas en los bloques.