

1.3. ORGANIZACIÓN DE ARCHIVOS Y MÉTODOS DE ACCESO

Pasamos a describir las cuatro organizaciones básicas de archivos en las que se pueden sustentar los diferentes métodos de acceso que se emplean en los SGBD. Haremos una serie de suposiciones y simplificaciones que nos van a permitir hacer las estimaciones de los parámetros objeto de la comparativa. Existen en la literatura otras arquitecturas adicionales a las que aquí vamos a examinar, pero éstas no son más que formas mixtas o variaciones sobre las que vamos a exponer, pudiendo el lector extrapolar los resultados obtenidos para las cuatro arquitecturas básicas a otros tipos de organización.

1.3.1. ARCHIVO SECUENCIAL FÍSICO (ASF)

En este modelo de archivo, los datos se colocan según su orden de llegada sin tener en cuenta el significado de su contenido. Haremos la suposición más general sobre la estructura de los registros, esto es, los registros son de estructura variable, y por ende, de longitud variable. Como ya se comentó anteriormente, están formados por pares de la forma $(Atr_id, valor)$. La Figura 1.8 ilustra un ejemplo de un ASF. En ella aparecen los identificadores para facilitar la interpretación por parte del lector de los datos reflejados, pero en el sistema éstos están codificados⁶.

DNI=01086214; NOMBRE=Abad; CATEGORIA=PCD; COD_DEP=1
COD_ASIG=MP1; NOMBRE=Metodología de la Programación; CREDITOS=4.5; CARACTER=T; CURSO=1
COD_AULA=2.1; CAPACIDAD=75
DNI=01007223; NOMBRE=Acosta; CATEGORIA=TU; COD_DEP=1
COD_ASIG=BD1; NOMBRE=Bases de Datos; CREDITOS=3; CARACTER=T; CURSO=2

FIGURA 1.8 Archivo secuencial físico con registros de longitud variable.

⁵Archivos de este tipo son los que genera el SGBD relacional de Oracle cuando se crea un clúster para albergar varias tablas.

⁶La definición de los registros y los tipos asociados a sus campos estarán en otro lugar, en el diccionario de datos.

1.3.1.1. Espacio ocupado por registro

El primero de los parámetros a estimar es el espacio ocupado por registro, que coincide con el cálculo ya descrito en el Apartado 1.2.1 para registros de tamaño variable. A saber,

$$R = a'(A + V + 2)$$

1.3.1.2. Recuperación de registros en un ASF

Consideremos un fichero con n registros en el que tratamos de localizar uno en concreto por el valor de una clave de búsqueda. En el peor de los casos, puede ser necesario leer todo el fichero para localizar el registro deseado. Como todos los registros tienen igual probabilidad de aparición en cualquier posición del archivo, como mínimo se leerá 1 y como máximo n (todos). La media será la suma de los tiempos necesarios para recuperar cada registro dividido entre el número de registros.

Si el tiempo necesario para leer el primer registro es 1 unidad, el tiempo para leer el segundo es 2 unidades, y así sucesivamente; entonces tenemos:

$$T_F = \sum_{i=1}^n \frac{i}{n} \cdot T = \frac{(n+1)}{2} \cdot T \cong \frac{n}{2} \cdot T \text{ si } n \gg 1$$

1.3.1.3. Obtención del siguiente registro

El sucesor potencial del último registro recuperado según el valor de la clave de búsqueda puede estar en cualquier parte. Como la posición se desconoce, el tiempo para encontrar un sucesor será el de localizar cualquier registro, esto es:

$$T_N = T_F$$

1.3.1.4. Inserción de un registro

Dado que no hay una estructura única en un ASF, el registro se insertará simplemente al final del archivo.

$$T_I = T_W$$

donde T_W es el tiempo de escritura del registro correspondiente sin condiciones de orden.

1.3.1.5. Actualización de un registro

Para llevar a cabo la actualización de un registro hay que localizarlo, y en el peor de los casos, marcarlo como borrado e insertar el nuevo, que deberá colocarse al final del fichero. Este caso se da cuando se produce un incremento de tamaño en el registro nuevo respecto del antiguo. Por tanto, el tiempo necesario para llevar a cabo dicha actualización vendrá dado por la expresión:

$$T_U = T_F + T_W + T_I$$

cuyos términos se corresponden, respectivamente, con la localización, la escritura de la marca de *borrado* y la inserción del nuevo registro.

Si el registro no cambia de tamaño o éste es menor, se sobreescribe sobre él mismo y la expresión se reduce a:

$$T_U = T_F + T_W$$

1.3.1.6. Lectura exhaustiva de un ASF

Sí el archivo es de tamaño n y no es relevante su contenido,

$$T_X = n \cdot T$$

donde T es el tiempo necesario para localizar un registro cualquiera sin tener en cuenta su contenido. Si se desea que la lectura se haga ordenadamente de acuerdo con el valor de algún atributo,

$$T_X = n \cdot T_F$$

lo que supone un costo excesivamente elevado que se evita ordenando previamente el fichero de acuerdo con el atributo de búsqueda.

1.3.1.7. Reorganización de un ASF

Si actualizaciones y borrados se llevan a cabo marcando los registros obsoletos, periódicamente habrá que eliminarlos físicamente para liberar el espacio que éstos

están ocupando. Esto se logra copiando el archivo en otro, pero omitiendo los registros marcados. Si tenemos en cuenta los siguientes parámetros desde la carga inicial del fichero de n registros

- **O**, el número de registros añadidos, y
- **d**, el número de registros marcados para borrar,

el archivo final tendrá un total de $(n+O-d)$ registros y, por tanto, el tiempo necesario para copiarlos será:

$$T_Y = (n + O) \cdot T + (n + O - d) \cdot T_W$$

esto es, el tiempo necesario para leerlos todos, más el tiempo necesario para copiar los definitivos.

1.3.2. ARCHIVO SECUENCIAL LÓGICO (ASL)

En esta organización de archivos, los registros se ordenan según una secuencia específica que viene determinada por el contenido de un campo que denominaremos clave física. Como simplificación, haremos la suposición de que los registros son de longitud fija, por tanto, todos los registros tienen el mismo número de atributos, del mismo tipo y en el mismo orden. De esta forma, los nombres de los atributos solo se escriben una vez en la cabecera del fichero. Este tipo de archivos facilita la mezcla de información proveniente de distintos ficheros donde la clave sea la misma. Se utilizan frecuentemente en organizaciones de tipo “*secuencial-indexado*” que luego veremos. En la Figura 1.9 se muestra un fichero que implementa este mecanismo de almacenamiento.

La clave física puede estar formada por un solo atributo o por varios; éste es el caso que se produce cuando ningún atributo por sí solo identifica únicamente al registro. Por ejemplo, si varios registros coinciden en el valor del *primer apellido*, podrían utilizarse también el *segundo apellido* y el *nombre* como clave física. La lectura por orden físico según la clave es una lectura en serie secuencial.

Cuando el archivo tiene un tamaño considerable, es muy costoso abrir un hueco a un nuevo registro, para mantenerlo ordenado por valor de clave física, por lo que se suele disponer de una estructura adicional conocida como fichero de desbordamiento

DNI	NOMBRE	CATEGORIA	COD_DEP
01007223	Acosta	TU	1
01347123	Miras	PCD	3
02818234	Pérez	TEU	2
12815413	López	PAD	7
...

FIGURA 1.9 Estructura de un archivo secuencial lógico.

o de overflow *no ordenado*, que se gestiona como un ASF. De este modo, los nuevos registros suelen insertarse aparte, en el fichero de desbordamiento, hasta que haya un número suficiente de ellos y se haga la inserción definitiva en el archivo *principal maestro*⁷ para todos a la vez.

1.3.2.1. Espacio ocupado por registro

Dado que los registros tienen un tamaño fijo, el espacio necesario para almacenar los códigos de los atributos es insignificante frente al tamaño del propio fichero. Sin embargo, se les reservará espacio a los valores omitidos o no definidos, esto es, tanto si se conoce como si no se conoce el contenido de un campo, el espacio que éste ocupa quedará igualmente reservado. El tamaño del registro será constante, y ya fue estimado como:

$$R = \sum_i V_i$$

Si hay muchos valores desconocidos o indefinidos, la densidad del archivo será baja. En una situación así, podría replantearse el diseño físico de la base de datos.

⁷Tanto el archivo inicial como el archivo de desbordamiento son de datos, pero denominaremos principal al primero para distinguirlo del de desbordamiento.

En lo que sigue se va a hacer un análisis de los parámetros con los dos planteamientos considerados, esto es, en primer lugar con solo un archivo de datos y, posteriormente, con archivo principal más un archivo de desbordamiento.

1.3.2.2. Recuperación de registros en un ASL

Si el *atributo de búsqueda no coincide con la clave física* por la que están almacenados los registros, para localizar uno se tendrá que leer la mitad del archivo en promedio. Por tanto:

$$T_F = \frac{n}{2} \cdot T$$

Si en el archivo principal se han producido varias inserciones de registros, y éstos han sido colocados en el fichero de desbordamiento, también deberá mirarse en este archivo cuando se lleve a cabo una recuperación de registro. Si llamamos **O** al número de registros que se han almacenado en el fichero de desbordamiento, el tiempo total medio empleado en recuperar un registro será:

$$T_F = \frac{n}{2} \cdot T + \frac{O}{2} \cdot T$$

esto es, la suma de los tiempos empleados en las búsquedas en ambos ficheros.

Si el *atributo de búsqueda coincide con la clave física* por la que están almacenados los registros, la localización podría llevarse a cabo mediante una búsqueda binaria⁸; en ese caso el tiempo aproximado de recuperación se verá reducido a:

$$T_F \cong \log_2(n) \cdot T$$

con solo un archivo principal, ya que en cada paso el espacio de búsqueda se reduce a la mitad. Si hubiese área de desbordamiento, a ese tiempo habría que añadirle el que se consuma en recorrer los registros de dicha área. Se deja al lector la búsqueda de la expresión para este caso.

⁸Sería válido éste o cualquier otro mecanismo de búsqueda eficiente, dada una ordenación, cuyo orden de complejidad sea comparable.

1.3.2.3. Obtención del siguiente registro en un ASL

En un ASL, el siguiente registro a uno dado está inmediatamente accesible y puede estar incluso en el mismo bloque. Si con frecuencia se va a solicitar el siguiente registro, el sistema deberá programarse para que el bloque leído en el buffer se conserve allí. La probabilidad de encontrar al registro sucesor en el mismo bloque viene determinada por el número de registros por bloque (Bfr). Por tanto, en el $100/Bfr\%$ de los casos, se necesitará leer el siguiente bloque. Si no tenemos en cuenta el tiempo que se pierde en el cambio de bloque (no se ha tenido en cuenta hasta ahora), el tiempo esperado para leer el siguiente registro será el mismo que el necesario para leer un registro arbitrario cualquiera (sin condiciones), esto es:

$$T_N = T$$

considerando que no hay un fichero de desbordamiento. La expresión correspondiente al caso en que dicho fichero exista, sería:

$$T_N = T + O \cdot T$$

Recordemos que no hay orden en el archivo de desbordamiento. Por ello, no es hasta leer los O registros de este archivo que se sabe con certeza cuál es el registro siguiente por valor de clave; de ahí, el segundo sumando en la expresión.

1.3.2.4. Inserción en un ASL

Hay que conservar la secuencia de los datos según la clave. En el caso de que se manipulen pocos registros, podrían moverse todos los registros una posición desde el punto de inserción con el fin de abrir un hueco. Esto implica las siguientes operaciones:

1. localizar el punto donde hay que insertar el nuevo registro mediante sucesivas lecturas,
2. leer y escribir todos los registros restantes, y
3. escribir el registro en cuestión.

Teniendo en cuenta que éste sea el modo en que se lleve a cabo la inserción, la expresión obtenida es:

$$T_I = T_F + \frac{n}{2} \cdot T + \frac{n}{2} \cdot T_W [+T_W]$$

donde se considera que, en promedio, cada fase implica la mitad de los registros del fichero. El tiempo consumido por la última operación de escritura (la del registro a insertar) puede considerarse insignificante, frente a tener que escribir la mitad de los registros del fichero, por lo que puede omitirse. De nuevo, la expresión obtenida es la que se corresponde con un ASL sin archivo de desbordamiento.

La existencia del fichero de desbordamiento nos permitiría recolectar todos los registros a insertar en un fichero aparte según su orden de llegada (operación poco costosa) y realizar la inserción real de todos ellos en el fichero principal de forma simultánea con posterioridad, cuando esto sea necesario o conveniente (por ejemplo, aprovechando una operación de reorganización). En primera instancia, la inserción de un registro en el fichero de desbordamiento supondría un tiempo:

$$T_I = T_W$$

que es un resultado irreal, teniendo en cuenta que falta completar la segunda parte del proceso, esto es, su posterior inclusión en el archivo principal. Suponiendo que **O** sea el número de registros de este fichero de desbordamiento, el costo de ampliar el archivo maestro estará en función de **O** y del tiempo que consume la reorganización del mismo (T_Y), por tanto,

$$T_I = \frac{T_Y}{O}$$

donde T_Y es el tiempo de reorganización de *todo* el archivo⁹. Si dividimos T_Y entre el número de registros involucrados en la reorganización, obtendremos el tiempo consumido por *un solo* registro. Este mecanismo resulta más eficiente cuantos más registros se inserten de golpe, siendo muy ineficiente en el caso de pocos registros.

1.3.2.5. Actualización de un registro en un ASL

Teniendo en cuenta la organización secuencial con un solo archivo y que la actualización no involucre un cambio en el campo clave, el tiempo necesario para llevar

⁹Se le ha añadido el peso de la reorganización a la inserción.

a cabo esta operación sobre un registro será:

$$T_U = T_F + T_W$$

Si se produce un cambio en la clave, el proceso involucra, además, el borrado de un registro.

T_U = T_F + T_W + T_I → es un reg. nuevo, por que la clave es diferente, por eso es como insertar un nuevo registro

1.3.2.6. Lectura exhaustiva de un ASL

Si se utiliza un único archivo de datos, la expresión para el tiempo de lectura de todo el archivo será:

$$T_X = n \cdot T$$

puesto que el siguiente registro según el orden lógico es también el siguiente según el orden físico.

En el caso de utilizarse un archivo de desbordamiento, primero deberá ordenarse este último según la clave y luego leerse ambos archivos simultáneamente de forma alternativa. Así,

$$T_X = T_C(O) + (n + O) \cdot T$$

siendo T_C el tiempo necesario para ordenar los O registros del fichero de desbordamiento.

1.3.2.7. Reorganización de un ASL

Esta operación tiene sentido cuando existe un archivo de desbordamiento junto con el principal o bien cuando se ha producido un elevado número de operaciones de borrado con solo un archivo de datos.

En el primer caso, debe realizarse la mezcla de ambos ficheros y el de desbordamiento deberá estar previamente ordenado según el mismo campo clave que el archivo principal. La mezcla se realizará sobre un tercer archivo que pasará a ser el nuevo archivo principal. Durante el proceso serán omitidos los registros marcados como borrados de manera que, con el mismo esfuerzo computacional, se compactará también el espacio ocupado. El tiempo total será la suma del tiempo de ordenación

del fichero de desbordamiento, más el tiempo de realizar la mezcla (caso de que proceda) la cual requiere, a su vez, la suma de los tiempos para leer ambos ficheros más el de escritura en el nuevo archivo secuencial; por tanto,

$$T_Y = T_C(O) + (n + O) \cdot T + (n + O - d) \cdot T_W$$

donde d es el número de registros borrados de forma lógica.

1.3.3. ARCHIVO SECUENCIAL INDEXADO (ASI)

El método de búsqueda basado en un índice tiene por objeto acelerar el tiempo de acceso a los datos por una clave de búsqueda. El principio de funcionamiento es similar al del índice de un libro en la búsqueda de información. Hay que buscar en el índice la entrada correspondiente y éste nos indicará el número de página donde hemos de empezar a leer. Extrapolando esta idea a los ficheros de datos, el índice tendrá algún valor de dato relevante junto con una dirección o apuntador sobre la localización de dicho valor en el fichero. En función de si el apuntador que se encuentra en el índice indica una dirección de registro o de bloque, nuestros índices se clasifican en *densos* y *no densos*, respectivamente.

Sea cual sea el tipo de índice de que se trate, la estructura de un ASI consiste en:

- 1. Un fichero de datos, que es un ASL, con una posible área de desbordamiento. Mantendremos la misma simplificación de que los registros son de longitud fija. La gestión del área de desbordamiento será diferente a la vista para el caso del ASL debido a razones que se expondrán más adelante. En cualquier caso, los datos en el área de desbordamiento se mantienen lógicamente ordenados.
- 2. Un fichero índice compuesto por registros de longitud fija, que no es otra cosa que un ASL. Por simplicidad supondremos que éste no tiene área de desbordamiento. Los registros del índice constan todos de un campo clave por el que se mantienen ordenados (clave física) y un campo apuntador que contiene una dirección que será de utilidad para recuperar un registro en el archivo maestro.

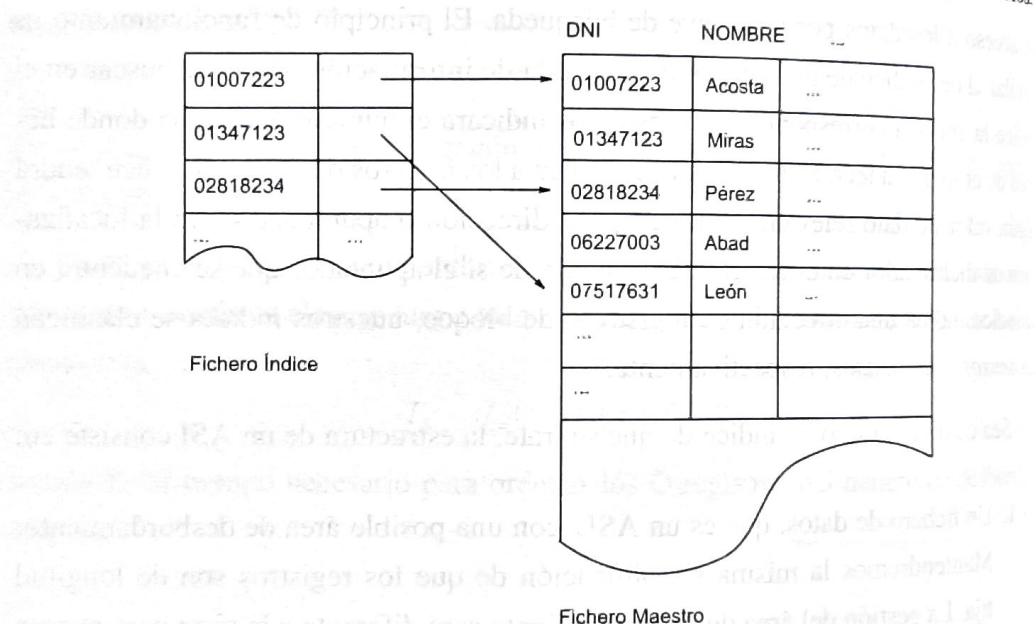
Denominaremos archivo maestro al ASL de datos para distinguirlo del de índice. De nuevo vamos a considerar que la clave física del archivo maestro es única, esto es, no hay dos registros con el mismo valor para ella.

1.3.3.1. ASI denso

1.3. ORGAN

Con los mismos componentes de la estructura que acabamos de describir, denominaremos índice denso a aquel en el que el número de entradas o registros coincide con el número de registros en el archivo maestro (suponemos, inicialmente, que no existe área de desbordamiento).

En la Figura 1.10 presentamos un ejemplo de esta estructura. Tenemos un fichero secuencial ordenado por una clave que llamaremos *física* sobre la que se ha montado un índice. Para el índice hemos considerado como clave de búsqueda la propia clave física del fichero de datos; a este tipo de índices se los denomina **índices primarios**.



FIG

FIGURA 1.10 Índice denso primario.

Cuando se monta un índice primario sobre un fichero de datos se establece una correspondencia directa entre los valores del campo clave y el propio registro al que pertenece ese valor, a través del apuntador. Los registros del índice se almacenan ordenados por el campo clave, lo que facilita la búsqueda por valor de clave o por intervalos de valores.

Es posible, sin embargo, crear diversos índices sobre otros campos que no sean la clave física del fichero de datos. En estos índices, llamados secundarios, la clave de búsqueda por la que se mantiene ordenado el índice realiza la misma tarea que el

campo cla
índice secu

puede obs
secundari
una secue
número d
primario,
registros e

- Esi

Los
ma
Co
ció
sor
El
ser

campo clave, aunque sus valores pueden repetirse. En la Figura 1.11 se ha creado un índice secundario sobre otro campo numérico, el *código del departamento*. Como se

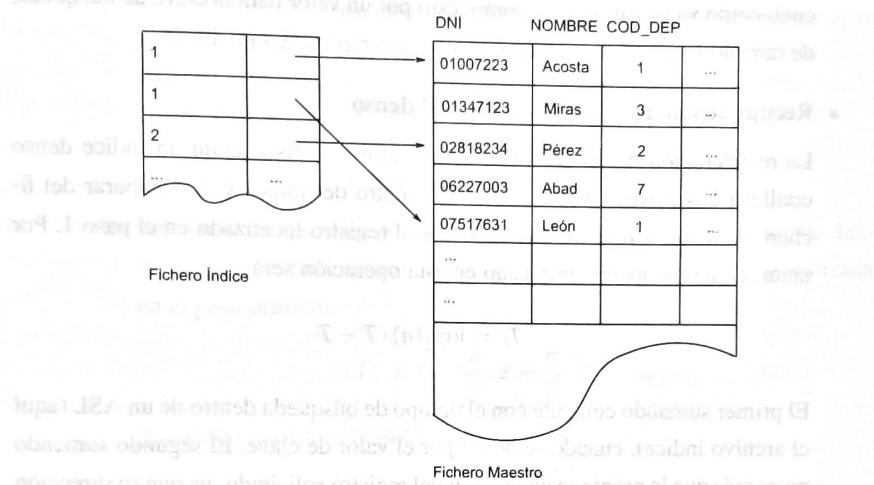


FIGURA 1.11 Índice denso sobre el campo departamento del fichero de profesores.

puede observar, si se lista de forma completa el fichero de datos utilizando el índice secundario, nos veremos obligados a ir “saltando” de bloque en bloque para forzar una secuencia que nada tiene que ver con la secuencia física de los registros y el número de lecturas de bloques de disco se dispara. Sin embargo, utilizando el índice primario, el orden sería el mismo que el orden en que se encuentran almacenados los registros en el fichero de datos y el recorrido será, por tanto, más rápido.

■ Espacio ocupado por registro

Los registros del fichero índice son iguales en número a los que posee el fichero maestro, pero de menor tamaño, ya que solo almacenan pares clave-apuntador. Conforme aumenta el tamaño de los registros del fichero de datos, la proporción de ahorro de espacio aumenta. En cualquier caso, las búsquedas en índices son más rápidas que en los ficheros de datos.

El tamaño global ocupado por un registro de datos en una estructura como ésta será pues:

$$R = \sum_i V_i + (V_k + P)$$

donde el primer sumando coincide con el espacio ocupado por registro de datos en el archivo maestro y $V_k + P$ es el tamaño de registro en el fichero índice, el cual, como ya sabemos, está compuesto por un valor para la clave de búsqueda de tamaño V_k , junto con un apuntador o referencia de tamaño P .

■ Recuperación de registros en un ASI denso

La recuperación de un registro por un valor concreto según un índice denso conlleva dos pasos: 1) buscar el valor dentro del índice y 2) recuperar del fichero de datos el registro apuntado por el registro localizado en el paso 1. Por tanto, el tiempo medio empleado en esta operación será:

$$T_F = \log_2(n) \cdot T + T$$

El primer sumando coincide con el tiempo de búsqueda dentro de un ASL (aquí el archivo índice), cuando se busca por el valor de clave. El segundo sumando no es más que la propia recuperación del registro solicitado, ya que su dirección ha sido aportada por el registro de índice.

■ Obtención del siguiente registro en un ASI denso

La obtención del siguiente registro a uno ya localizado es muy directa, ya que se trata de leer el siguiente registro de índice y posicionarse directamente sobre el fichero de datos para realizar la lectura del registro completo. El tiempo necesario para esta operación viene dado por la expresión:

$$T_N = T + T = 2 \cdot T$$

■ Inserción de un registro en un ASI denso

La operación de inserción de un nuevo registro de datos supone, por un lado, las mismas operaciones que para la inserción en el fichero secuencial pero, por otro lado, es necesario actualizar también el índice, esto es, insertar una nueva entrada en el fichero asociado al mismo.

La forma de proceder para la inserción de un registro en un índice denso es exactamente igual que la inserción en cualquier ASL: buscar el sitio que le corresponde, insertarlo en caso de haber espacio, o bien la redistribución de registros existentes.

En primer lugar vamos a considerar la inserción en un ASI, sin área de desbordamiento; en este caso deberán llevarse a cabo los siguientes pasos:

1. Insertar el registro completo en el fichero de datos o fichero maestro. Esta operación consume un tiempo:

$$T_{I_1} = T_F + \frac{n}{2} \cdot T + \frac{n}{2} \cdot T_W [+T_W]$$

cuya expresión ya comentamos para el ASL, y

2. Insertar el campo clave y el apuntador en el sitio que les corresponde dentro del índice. Esta operación consume el mismo tiempo que la realizada en el paso primero.

$$T_{I_2} = T_F + \frac{n}{2} \cdot T + \frac{n}{2} \cdot T_W [+T_W]$$

quedando la expresión definitiva:

$$T_I = T_{I_1} + T_{I_2} = 2 \cdot (T_F + \frac{n}{2} \cdot T + \frac{n}{2} \cdot T_W [+T_W])$$

Si consideramos que el ASL dispone de un archivo de desbordamiento, la expresión para T_I será:

$$T_I = T_{I_2} + T_W$$

que se corresponden, respectivamente, con la inserción en el índice y en el archivo de desbordamiento.

■ Actualización de un registro en un ASI denso

Si la actualización solo involucra campos distintos al que se ha utilizado como clave de búsqueda del índice, la operación es idéntica a la que se lleva a cabo en un ASL y consume el mismo tiempo, esto es:

$$T_U = T_F + T_W$$

En el caso de que la modificación afecte a la clave de búsqueda, la actualización de datos deberá llevarse a cabo en ambos ficheros y, por tanto, el tiempo medio consumido será:

$$T_U = 2 \cdot (T_F + T_W) + T_{I_1} + T_{I_2}$$

que se corresponde con la anulación de la entrada en ambos archivos y las respectivas inserciones de los registros con nueva clave.

Lo mismo puede decirse de la operación de borrado de un registro. Esta supone, por un lado, el borrado de un registro en el fichero de datos y, por otro lado, el borrado de una entrada en el índice. Como los borrados se llevarán a cabo a través de la escritura de una marca, esta operación consume el mismo tiempo que la actualización de un registro.

■ Lectura exhaustiva de un ASI denso

Si el recorrido que se ha solicitado es por la clave física del fichero de datos, el índice no es necesario y la expresión coincide, de nuevo, con la de un ASL, esto es, $T_X = n \cdot T$. Por otro lado, si el recorrido se debe realizar según el valor de otro campo, habremos de recorrer el correspondiente índice para poder acceder según el orden requerido. En este caso,

$$T_X = n \cdot T + n \cdot T = 2 \cdot (n + T)$$

ya que el índice nos va dando las direcciones de los registros de forma directa. Si consideramos que hay archivo de desbordamiento la expresión sería:

$$T_X = (n + O) \cdot T + n \cdot T + O \cdot T = 2 \cdot (n + O) \cdot T$$

que se corresponde con el tiempo de recuperación de $n + O$ entradas del índice, n entradas del archivo principal de datos y O entradas del archivo de desbordamiento.

■ Reorganización de un ASI denso

Esta operación se plantea cuando se ha producido un número elevado de borrados y, por tanto, hay una proporción relativamente alta de espacio desperdiciado. Los huecos se habrán generado tanto en el fichero índice como en el fichero principal. Si, además, se ha empleado un fichero de desbordamiento, habrá que incorporar estos registros al nuevo fichero maestro en el lugar que les corresponde. Por tanto, el tiempo necesario para llevar a cabo la reorganización completa cuando el índice es primario será:

$$T_Y = T_C(O) + (n + O) \cdot T + (n + O - d) \cdot T_W + (n + O - d) \cdot T_W$$

cuyos sumandos se corresponden con:

- $T_C(O)$ es el tiempo necesario para ordenar el fichero de desbordamiento,
- $(n + O) \cdot T + (n + O - d) \cdot T_W$ es el tiempo necesario para leer todos los registros más el de escribir solo los válidos y, por último,
- $(n + O - d) \cdot T_W$ es el tiempo necesario para crear el índice de nuevo.

Si el índice no fuera primario, habría que añadir un tiempo extra proporcional a $n^2 \cdot T$ dedicado a la ordenación temporal del fichero maestro según el campo por el que se deseé crear el índice.

Como conclusión, es importante señalar que, si bien los índices aceleran el acceso a los datos, las operaciones de borrado e inserción se ralentizan, ya que se han de mantener los índices actualizados a la vez que los datos. Es por ello que hay que considerar la conveniencia de crear uno o varios índices por fichero y si está justificado por la frecuencia de las consultas y/u operaciones de mantenimiento de los datos. En los SGBD es muy frecuente que el propio sistema cree y mantenga índices densos por clave primaria para todas sus tablas, al ser éste el atributo (o atributos) más solicitado y el más frecuentemente involucrado en las consultas que, por otro lado, son las operaciones más frecuentes.

1.3.3.2. ASI no denso

Aunque lo ideal sería mantener los índices en memoria principal, éstos, por lo general, siguen siendo muy grandes, pues en un índice denso existe el mismo número de registros que en el fichero de datos que indexa, aunque éstos sean de menor tamaño. Para tratar de reducir el tamaño de los índices, se idearon los denominados *índices no densos*, que pasamos a describir.

Los registros de los índices no densos (al igual que en el caso de los densos) están compuestos por dos campos: la clave de búsqueda (que en este caso debe coincidir con la clave física) y la dirección de comienzo de la página donde se encuentran los registros cuya clave se encuentra en el rango indicado.

Ahora bien, al contrario de lo que sucede en los índices densos, el número de registros que contiene el no denso se reduce al número de páginas de datos del fichero principal (ver Figura 1.12). Por tanto, para crear y mantener esta estructura, es necesario ir agrupando los registros del fichero maestro en páginas de igual tamaño de

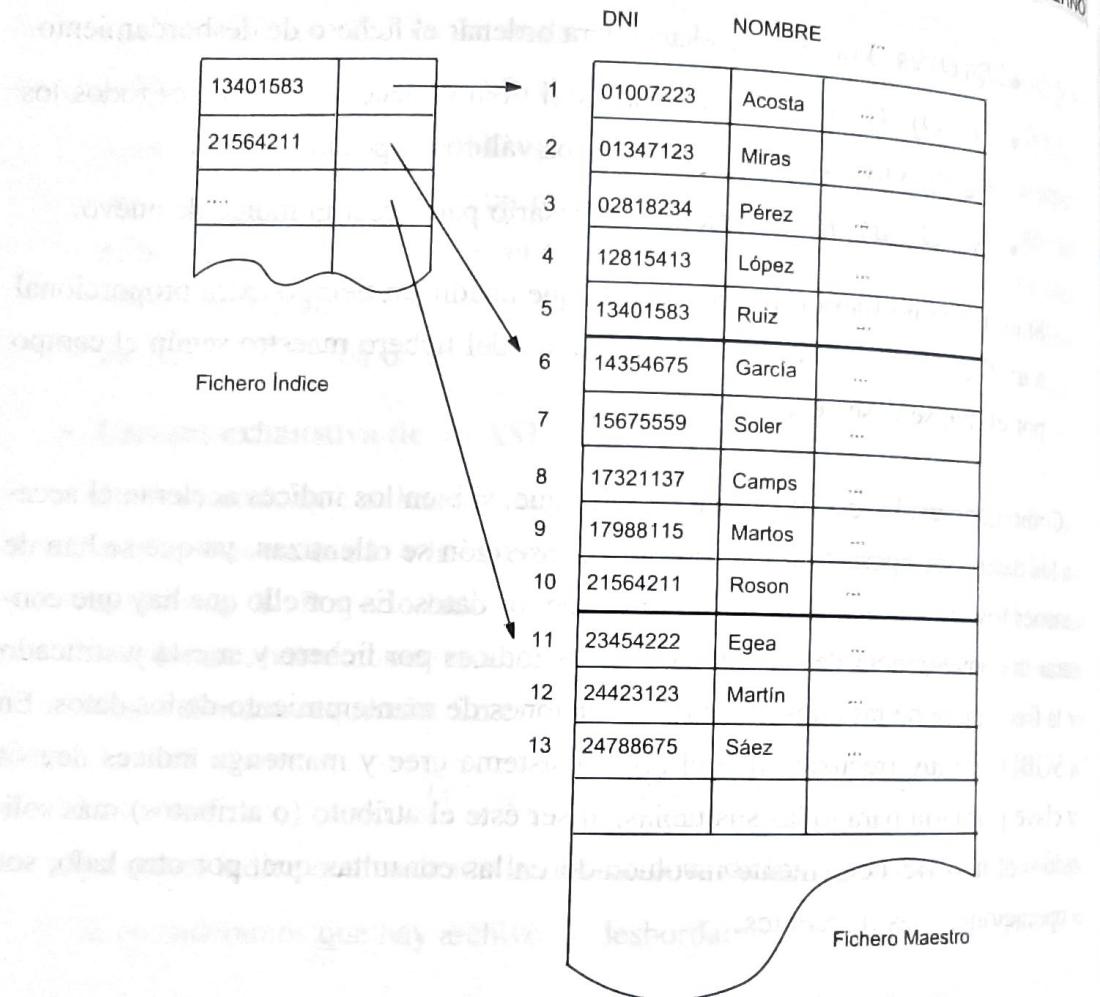


FIGURA 1.12 ASI no denso.

manera que el índice, en vez de permitir localizar registros de forma individual, ahora lo que nos indica es la página en la que *posiblemente* se encuentre un determinado registro, pues no se puede garantizar su existencia.¹⁰

Así, suponiendo que se tienen n registros en el archivo de datos, el número de entradas en el índice será $\lceil n/Bfr \rceil$, donde Bfr es el ya conocido factor de bloqueo del archivo de datos¹⁰.

Dada la drástica reducción en el número de registros con respecto a los del fichero principal, el acceso secuencial al índice no denso suele ser muy rápido.

¹⁰ $\lceil \cdot \rceil$ es el operador *menor entero mayor* que pues también deben direccionarse páginas que no están completas.

Existen dos diferencias principales respecto a cómo se realiza la búsqueda en un índice denso:

1. Una vez encontrada la página donde podría encontrarse el registro que se está buscando (se selecciona en el índice la entrada inmediatamente inferior al valor de la clave de búsqueda que se está consultando) se carga en memoria el bloque que la contiene y se hace una búsqueda secuencial dentro de la misma hasta hallar el registro requerido. Esta búsqueda local no requiere ningún coste adicional de E/S a disco, una vez transferido el bloque a memoria principal.
2. Cuando se realiza una búsqueda en el índice no denso, no se tiene garantía de encontrar el registro deseado hasta después de consultar la página de datos correspondiente. Por ejemplo, si buscamos el registro de valor 03234786 en el índice no denso de la Figura 1.13, tendríamos que cargar la página 0, mientras que la misma búsqueda en un índice denso se detendría ahí, al no haberse encontrado una entrada en el índice con el valor requerido. Como consecuencia de todo ello, las consultas de tipo existencial podrán resolverse directamente consultando un índice denso, pero necesitarán de la lectura de bloques de datos en el caso de uno no denso.

A pesar de las ventajas que aportan los índices no densos sobre un fichero, éstos solo se pueden definir sobre la clave física, esto es, solo puede haber un índice no denso por fichero de datos.

Volviendo a la Figura 1.12, en ella puede verse un fichero secuencial indexado de forma no densa, en el que cada página contiene cinco registros. Para leer los datos de forma consecutiva, se irán leyendo secuencialmente los registros del fichero sin tener en cuenta el índice, pero para localizar un registro determinado, buscaremos en el índice una entrada cuyo valor de clave sea menor o igual (o mayor) que el del registro que buscamos, y el índice nos indicará la página en la que éste debería estar. Una vez localizada la página en cuestión, debe hacerse un barrido secuencial de la misma hasta localizar el registro requerido.

Dentro de cada página, los registros deben estar siempre ordenados según la clave. Para el caso en el que se realicen inserciones, es frecuente dejar algún espacio libre al final de cada página de manera que puedan desplazarse los registros hacia abajo para abrir nuevos huecos a los registros entrantes. A veces se opta también por

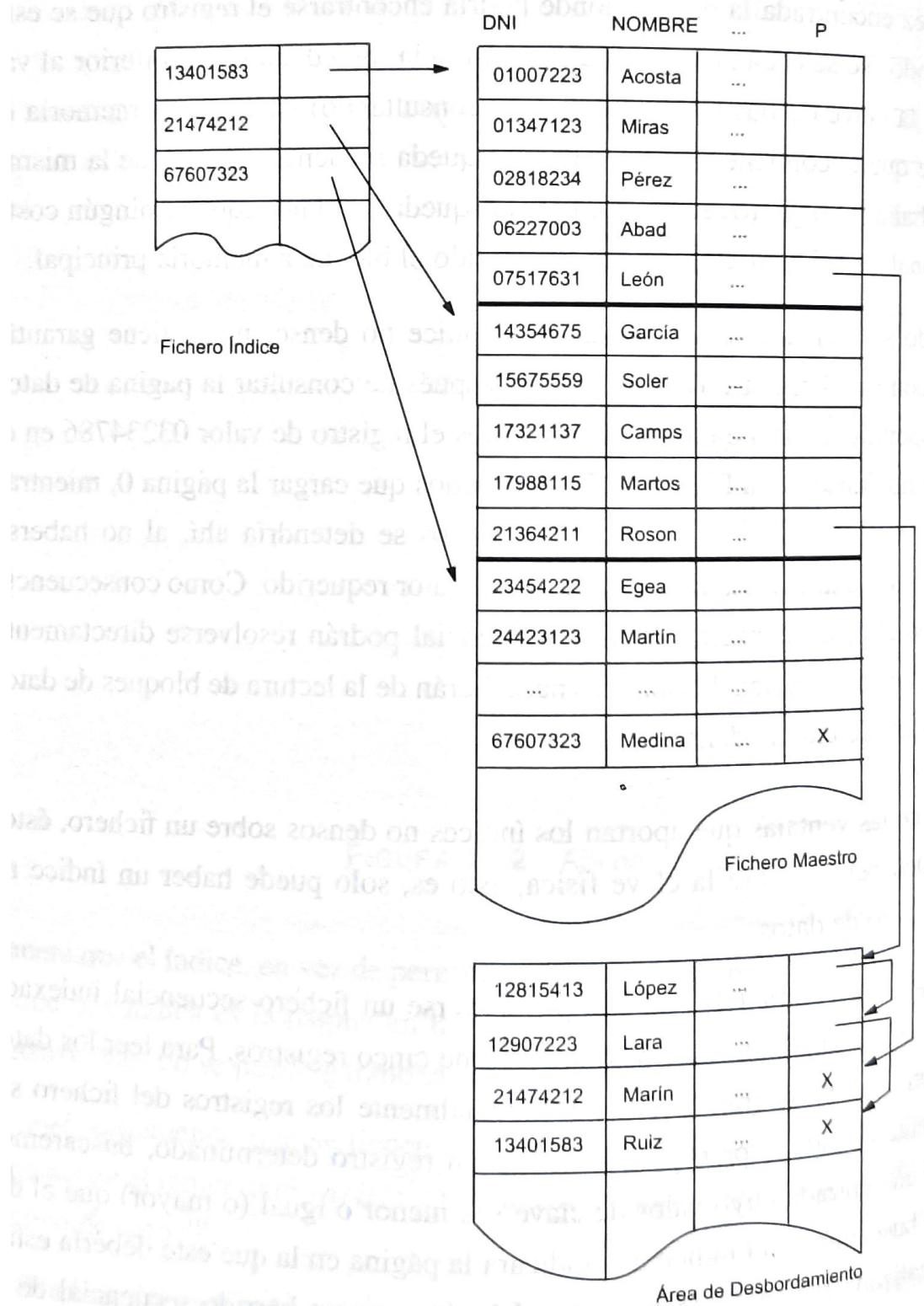


FIGURA 1.13 ASI no denso con direccionamiento a cadena de desbordamiento.

mantener espacio adicional en un fichero de desbordamiento para ubicar registros a los que les corresponden páginas que están completamente llenas. Como es de suponer, la utilización de este fichero tiene el inconveniente de que requiere un acceso separado, con el consiguiente tiempo de localización e inserción. Por otro lado, la asignación de espacio en cada bloque es factible solo si los bloques son grandes y las inserciones están más o menos distribuidas a lo largo del fichero ya que, de otro modo, es muy fácil que el espacio se acabe pronto en algunos bloques mientras que otros nunca lleguen a ser accedidos para inserción.

Si se utiliza un archivo de desbordamiento, todos los registros del mismo que pertenecen lógicamente a una misma página del fichero principal (independientemente de su ubicación real), se eslabonan en una cadena. En la Figura 1.13 puede encontrarse una ilustración de esta organización.

Como puede observarse, la estructura de un registro del archivo principal de datos contiene un campo adicional *P*, que puede albergar una dirección en el archivo de desbordamiento. Para la inserción de un nuevo registro, éste se eslabona en la cadena de acuerdo con su valor de campo clave, de manera que se conserve el orden. Como puede observarse se mantiene el orden tanto lógico como físico de los registros que se encuentran en el fichero maestro. Los nuevos registros insertados se pondrán en el lugar que les corresponda según la secuencia lógica. Si el nuevo registro ha de ocupar el lugar de otro registro en el fichero principal porque no hay sitio en la página, el que fuera el último registro se desplaza al área de desbordamiento.

La dirección del primer registro de desbordamiento correspondiente a una determinada página del fichero maestro, debe ser almacenada en algún lugar dentro de la propia página. En la Figura 1.13 esta dirección ha sido almacenada en el último registro de cada página¹¹.

1.3.3.3. ASI multinivel

Los índices son efectivos cuando los archivos son bastante grandes y/o los registros de datos son grandes ya que, en proporción, éste contendrá muchos menos

¹¹ Existen otras soluciones, como por ejemplo añadir un campo más a los registros de índice y anotar ahí la dirección del primer registro de desbordamiento.

bloques. Si el índice es, a su vez, de gran tamaño, éste puede también indexarse, denominándose a esta estructura *índice multinivel*. En lo que sigue, consideraremos los índices multinivel en la forma más general con m niveles, siendo éste un número natural mayor o igual que 1. Se deja al lector ajustar las expresiones que aquí se exponen para el caso particular de un ASI no denso de un solo nivel.

Denominaremos y al número de registros de las páginas de índice, y se calcula como el cociente entre el tamaño del bloque, B , y el espacio que ocupa cada registro de índice ($V + P$). Esto es,

$$y = \lfloor \frac{B - C}{V + P} \rfloor$$

donde V es el tamaño ocupado por el campo clave (que es el único campo de datos que se escribe en el índice) y P es el espacio que ocupa el apuntador.

Nótese que, al ser los registros de índice ($V + P$) menores que los registros de datos ($R + P$), siempre se cumple $Bfr < y$.

Vamos a evaluar el número de niveles óptimo de un índice para un tamaño de memoria m .

Ejemplo 1.4

- Supóngase un archivo con 1.000.000 de registros y que el tamaño de bloque $B = 2$ KB, el tamaño del campo clave $V = 14$ bytes, el tamaño del apuntador $P = 6$ bytes, la cabecera $C = 0$ bytes, los registros tienen una longitud total $R = 200$ bytes y el espacio en memoria para el buffer de ese archivo es 4 KB.

- El factor de bloqueo es $Bfr = \lfloor 2048/200 \rfloor = 10$. Si cada bloque contiene 10 registros, tendremos 100.000 bloques de datos. Podemos suponer que la clave del índice del primer nivel coincide con la clave física por la que están ordenados los datos en el archivo maestro. Luego utilizaremos un ASI no denso.

Como ya sabemos, nuestro índice de primer nivel tendrá 100.000 entradas que coincide con el número de bloques de datos. Dado que cada anotación en el índice ocupa un total de $V + P = 14 + 6 = 20$ bytes, el espacio total ocupado por este índice será de 20×100.000 bytes, esto es, aproximadamente 2 MB. Como el número de bytes necesario es excesivo, no caben en memoria, habrá que construir un índice no denso de segundo nivel. Para ello, se calculará el número de registros de índice que

caben en cada bloque:

$$y = \frac{B}{V + P} = \left\lfloor \frac{2048}{20} \right\rfloor = 102$$

Como hay 100.000 registros y caben 102 en cada bloque, el número de bloques necesarios o entradas en el nuevo índice será de $100.000 / 102 = 981$. Por tanto, el índice de segundo nivel ocupará $20 \text{ bytes} \times 981 = 19620 \text{ bytes}$ o lo que es lo mismo aproximadamente 19 KB. Como esta cifra sigue resultando aún excesiva (lo es considerando el tamaño del buffer) se creará un índice no denso de tercer nivel. Tenemos, pues, 981 registros agrupados en páginas de 102 registros cada una, por lo que necesitaremos 10 bloques o entradas en el nuevo índice, que ocupará, definitivamente, $20 \text{ bytes} \times 10 = 200 \text{ bytes}$. Dados los requerimientos planteados, hemos desarrollado una estructura de índice multinivel de tres niveles. Obsérvese que el tiempo de recuperación de un registro requiere conocer el número de niveles de índice utilizado.

La estructura del ASI recién planteada puede verse como un árbol general, donde cada nodo del árbol representa una página del índice no denso, que indexa las páginas de nivel inferior y donde las hojas son páginas o bloques de datos del archivo maestro.

Según esta nomenclatura, llamaremos **índice raíz** al índice de nivel superior.

Espacio ocupado por registro en un ASI multinivel

Suponemos que se dispone inicialmente de n registros cuyo tamaño es:

$$r = P + \sum_i V_i$$

donde, siguiendo con nuestra notación habitual, P es el campo de direccionamiento para el área de desbordamiento (ya comentado para un ASI no denso con desbordamiento), O es el número de registros añadidos tras la carga inicial de los n registros del fichero maestro y d el número de registros marcados para borrar.

El índice de primer nivel contiene una anotación por cada bloque del fichero maestro; así pues, este nivel contendrá un número de i_1 entradas, donde:

$$i_1 = \lceil \frac{n}{Bfr} \rceil \text{ entradas}$$

Los tamaños para los siguientes índices se determinan por medio del valor y , que, a su vez, está determinado por el tamaño del bloque, B y el tamaño del registro de

índice $V + P$. En general, para un índice de nivel k , el número de entradas será:

$$i_k = \lceil \frac{i_{k-1}}{y} \rceil$$

y el número de bloques que ocupa será:

$$b_k = \lceil \frac{i_k}{y} \rceil = i_{k+1} \text{ bloques}$$

→ Por tanto, el tamaño total ocupado para índices será el número total de bloques necesarios multiplicado por el tamaño de dichos bloques:

$$I = (b_1 + b_2 + \dots + b_m) \cdot B$$

Entonces, el espacio medio que debe reservarse para la gestión de *un solo registro* de datos en esta organización es:

$$R = r + \frac{o}{n} \cdot r + I/n$$

donde r es el espacio ocupado por cada registro en el archivo maestro, $\frac{o}{n} \cdot r$ es la porción de espacio ocupado para O registros de desbordamiento e I/n es la porción de espacio de indexación que corresponde a cada registro del fichero maestro.

Recuperación de registros en un ASI multinivel

Supondremos que el índice raíz está en memoria principal. El proceso consistirá en una revisión de éste, una lectura por cada nivel restante de índice y una lectura del registro en el archivo principal.

$$T_F = T_M + (m - 1) \cdot T_{F_i} + T'_F$$

donde T_M es el tiempo de revisar la tabla maestra, $(m - 1) \cdot T_{F_i}$ es el tiempo de realizar una localización en cada uno de los índices restantes y, por último, T'_F es el tiempo consumido por la lectura del registro en cuestión en el fichero principal dentro de la página direccionalada.

Sin embargo, si ha habido inserciones, puede que el registro buscado se encuentre en el área de desbordamiento.

Por tanto, en el peor de los casos, el tiempo total consumido en el acceso a un determinado registro, considerando que existe área de desbordamiento, sería:

$$T_F = T_M + (m - 1) \cdot T_{F_i} + T'_F + T_{F_Cadena}$$

donde T_{F_Cadena} es el tiempo que se tarda en localizar el registro solicitado partiendo de la primera dirección de búsqueda; una vez en el área de desbordamiento, habrá que seguir la cadena de registros a través de sus apuntadores, lo que producirá un retardo.

Obtención del siguiente registro en un ASI multinivel

Para localizar el siguiente registro, se comienza desde el último registro leído y se ignora el índice.

Si el predecesor no es el último de su página, se procederá a leer el siguiente registro en la forma habitual. Si el sucesor está en el fichero maestro, se lee directamente (tiempo T); si, por el contrario, está en el fichero de desbordamiento, se sigue el apuntador. En cambio, si el predecesor fuera el último registro de la página, habría que localizar el primer registro de la página siguiente. En cualquier caso, la expresión

$$T_N \simeq T$$

recoge las consideraciones anteriores.

Inserción de un registro en un ASI multinivel

La inserción de un registro en esta estructura puede implicar la realización de varias operaciones, dependiendo de las consideraciones que se hagan. Por ello, vamos a presentar la siguiente casuística. Para facilitar su comprensión conviene tener presente la Figura 1.14.

- *Caso I*

El registro se inserta en una página del archivo principal. En este caso podemos considerar:

1. Existe espacio suficiente en el bloque que le corresponde lo que da lugar a la expresión más simple:

$$T_I \cong T_F + 1/2 \cdot Bfr(T + T_W) + T_W$$

cuyos sumandos se corresponden con

- el tiempo de localizar en el índice la página en la que debe realizarse la inserción, (*ordenación*)

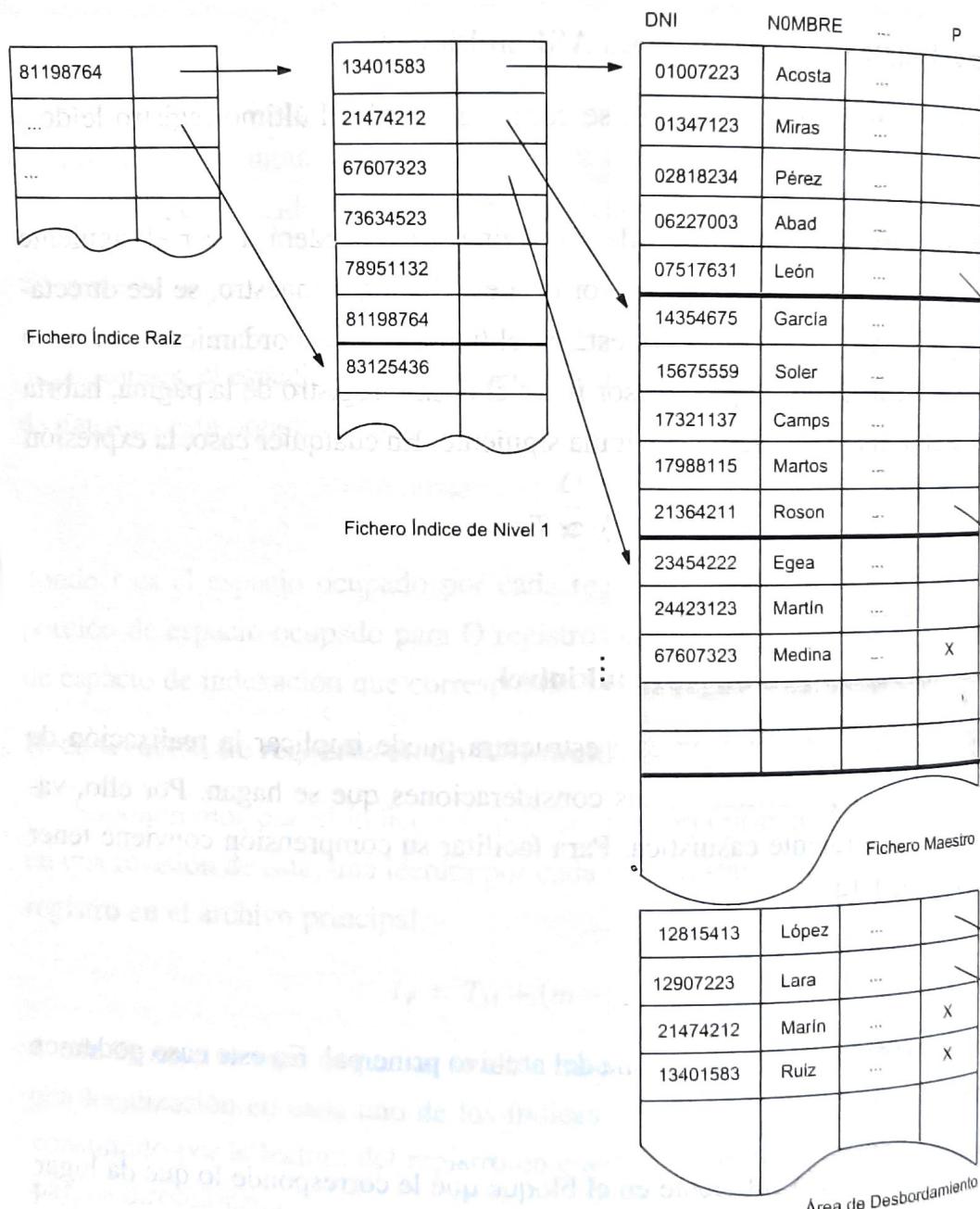


FIGURA 1.14 ASI multinivel.

- la lectura y escritura de, en término medio, la mitad de los registros de la página, con el fin de abrir un hueco (el tiempo empleado es mínimo teniendo en cuenta que todos los registros se encuentran en memoria), y
- el tiempo necesario para escribir el registro.

Por ejemplo, si se inserta el registro de clave 45123586, éste se ubicará en la tercera página del fichero maestro desplazando al actual último registro para conservar el orden físico.

2. Se inserta en el archivo principal pero no hay sitio. En esta situación, la inserción provoca el desplazamiento del último registro del bloque al fichero de desbordamiento, que queda enlazado a su página mediante un apuntador.

$$T_I \cong T_F + 1/2 \cdot Bfr(T + T_W) + 2 \cdot T_W$$

Esta operación requiere la localización del lugar de la inserción, el desplazamiento de la mitad de los registros (en término medio) y la escritura del último registro en el fichero de desbordamiento.

Por ejemplo, si se inserta el registro de clave 15876543, al que corresponde la segunda página, desplazaríamos el registro de clave 21364211 al área de desbordamiento.

■ Caso 2

El registro se inserta en el archivo de desbordamiento, en cualquier posición dentro de una cadena. En ese caso la expresión sería:

$$T_I \cong T_F + 2 \cdot T_W$$

donde los dos tiempos de escritura que figuran se corresponden con la escritura del registro al final del archivo de desbordamiento, y la actualización del apuntador del registro anterior.

Por ejemplo, si se inserta el registro de clave 13000011, al que corresponde la primera página, éste se ubicaría directamente en el área de desbordamiento, llevándose a cabo el reajuste correspondiente de apuntadores.

Actualización de un registro en un ASI multinivel

Un registro actualizado cuyo campo clave sigue intacto, puede colocarse en el lugar de la versión anterior, de manera que el proceso se evalúa como una recuperación guiada de una escritura:

$$T_U = T_F + T_W$$

La **eliminación** de un registro, por medio de la escritura de una marca, se lleva a cabo por este mismo proceso y consume el mismo tiempo.

En el caso en el que se haya modificado el valor del campo clave, debe cambiarse la ubicación del registro y, por tanto, la versión antigua del registro será eliminada; el nuevo registro será insertado en el lugar que le corresponda. En el registro antiguo se pondrá la marca de borrado y el nuevo se insertará en su sitio:

$$T_U = T_F + T_W + T_I$$

Lectura exhaustiva en un ASI multinivel

En este caso el índice puede ignorarse. No tendremos en cuenta el retardo producido al cambiar de los bloques del área principal a los de desbordamiento. Desde este punto de vista, el tiempo invertido en ese proceso, T_X , será proporcional al número total de registros presentes ($n + O$).

$$T_X \cong (n + O) \cdot T$$

Reorganización de un ASI multinivel

Seguimos considerando una organización en la que los registros se encadenan en un área de desbordamiento según el orden marcado por el campo clave.

Cuando las áreas de desbordamiento alcancen un tamaño excesivo o antes de que esto suceda, será necesaria una reorganización del archivo. También puede realizarse cuando, debido a la creación de largas cadenas, los tiempos de recuperación se vuelvan excesivos. Tal reorganización consiste en leer el archivo secuencialmente y volver a escribirlo, dejando fuera todos los registros marcados como eliminados y escribiendo los restantes en las áreas principales del nuevo archivo. Durante este procedimiento se crearán, a su vez, nuevos índices en base a los valores de los nuevos bloques. La frecuencia con que se lleve a cabo esta operación depende de la actividad *Learning ParallelDB*

de inserción dentro del fichero. Ya que esta organización requiere, por lo general, mucho tiempo, se suele llevar a cabo antes de que el problema se agudice para evitar tener que hacerlo en épocas de mucha actividad. Se pueden establecer medidas de periodicidad, como excederse un cierto límite de registros en el área de desbordamiento, o una vez al mes... o cualquier otro criterio adecuado al contenido y utilización del sistema.

La reorganización de un ASI supone, pues, la lectura del archivo completo, la escritura de todos los registros menos los borrados y la organización de un nuevo índice, lo que supone un tiempo

$$T_Y = T_X + (n + O - d) \cdot T_W + k \cdot I$$

donde k es una constante de proporcionalidad e I es el tamaño del índice.

Después de una reorganización, el área de desbordamiento estará vacía.

Criterios para la selección de índices

Para terminar el estudio de los ASI, sean del tipo que sean, son varias las consideraciones a tener en cuenta a la hora de plantearse una organización de este tipo:

- ▪ En primer lugar, hay que hacer un estudio de las consultas más frecuentes. Es obvio que no se monta un índice si no hay consultas que se beneficien de él, o si se realizan con poca frecuencia.
- ▪ Los atributos candidatos a ser clave de búsqueda en el índice son los que están en la cláusula WHERE¹². Suelen ser candidatos aquellos atributos que sirven para reunir tablas en las consultas.
- ▪ Hay que tener en cuenta la selectividad del atributo sobre el que se monta el índice, esto es, en una columna con valores repetidos, una búsqueda por valor nos debe dar un abanico de salida inferior al 5%; en otro caso es casi

¹²Recordemos que los tipos de consultas que se benefician son por valor de igualdad y por rango.

preferible una búsqueda secuencial¹³. Dicho de otra forma, utilizar atributos que no tengan muchos valores repetidos.

- Cuando se plantea un índice multiatributo es importante el orden de los atributos en el índice, de manera que se ajuste a las consultas más frecuentes.
- Por último y no por ello menos importante, hay que evaluar el coste de mantenimiento de un índice; no se suelen indexar columnas que son actualizadas frecuentemente ya que obligan a actualizar también el índice.

1.3.4. ARCHIVO DE ACCESO DIRECTO (AAD)

Este método de organización intenta explotar la capacidad proporcionada por las unidades de disco y dispositivos similares para lograr acceso a cualquier dirección de archivo de forma directa. Para lograr el direccionamiento directo se utiliza el valor de la clave del registro para determinar su localización en el archivo. Los métodos de acceso directo transforman la clave mediante un algoritmo de computación y así poder utilizarla como dirección. Esta organización utiliza un espacio extra en el archivo principal para conservar huecos que reduzcan la probabilidad y el costo de insertar más registros en el archivo. En la Figura 1.15 se muestra una organización de archivos de este tipo.

Existen diversos métodos de conversión de clave a dirección. Un valor de campo clave puede variar sobre un amplio intervalo de posibles valores, limitado solo por el tamaño del propio campo clave **V**. El número de claves legales para una clave numérica es 10^V (10 dígitos posibles tomados de **V** en **V**) y de 26^V para valores alfabéticos (26 caracteres). El fichero dispondrá de **m** celdas para alojar **n** datos, donde $m \geq n$. Si la clave utiliza **b** símbolos distintos, se cumplirá que $b^V \gg m$ y el algoritmo elegido deberá transformar cada posible combinación del conjunto b^V en un valor entre 0 y $m - 1$. Para esta organización, los parámetros descritos m, n, b y V son fundamentales y han de ser considerados *a priori* por el diseñador de la BD.

¹³ Ya se mencionó el caso extremo de un índice sobre un atributo de tipo booleano donde todas las entradas del índice se aglutan en torno a dos o tres valores, si consideramos NULL.

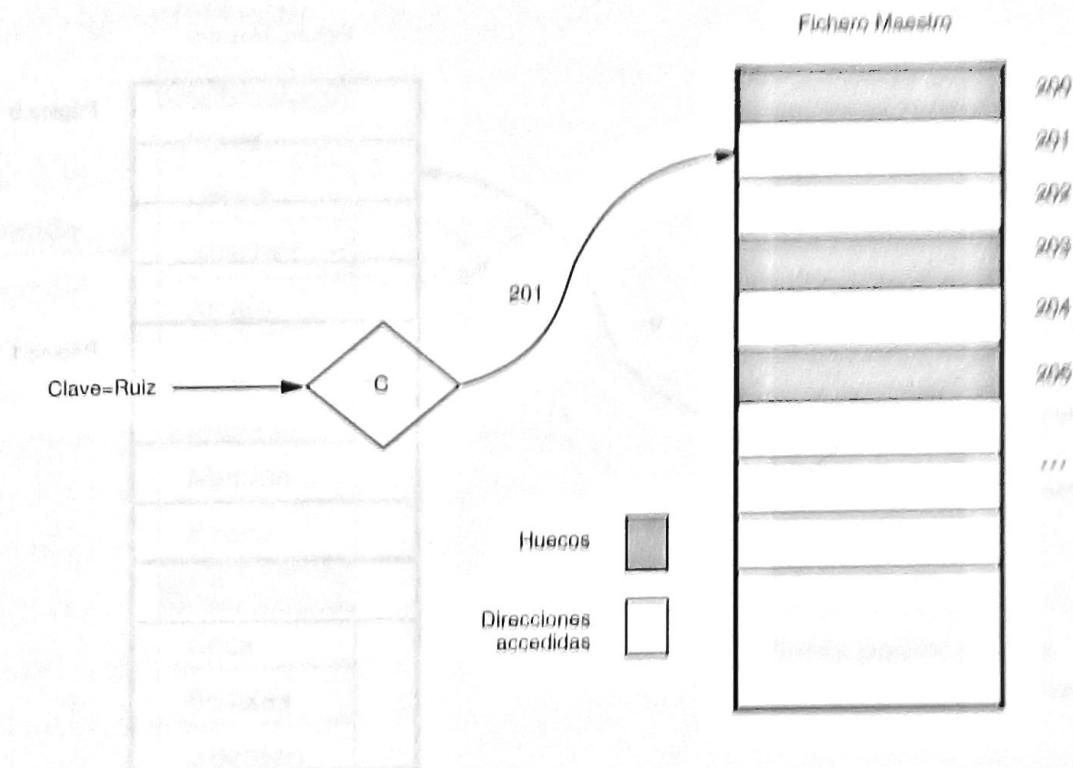


FIGURA 1.15 Estructura de un archivo de acceso directo.

Sean cuales sean los valores estimados o elegidos, en esta estructura surgen inevitablemente dos problemas bien conocidos:

- **Colisiones:** Se producen cuando dos valores distintos de clave conducen a la misma posición en el fichero. Este problema puede solucionarse habilitando zonas de desbordamiento y/o utilizando el algoritmo de direccionamiento para apuntar a páginas.

En la Figura 1.16 puede verse una ilustración de este tipo de direccionamiento. En este caso, los registros que dan lugar a la misma dirección de bloque se colocan secuencialmente dentro del mismo, de manera que para una posterior localización habría que leer secuencialmente, como máximo, tantos registros como contenga el bloque en el instante de la búsqueda.

- **Huecos:** Se producen cuando no existe ningún valor de campo clave que dirija una determinada posición (el hueco). Estos huecos se aprovechan, a veces, para posteriores inserciones en el fichero o cuando se produce una colisión.

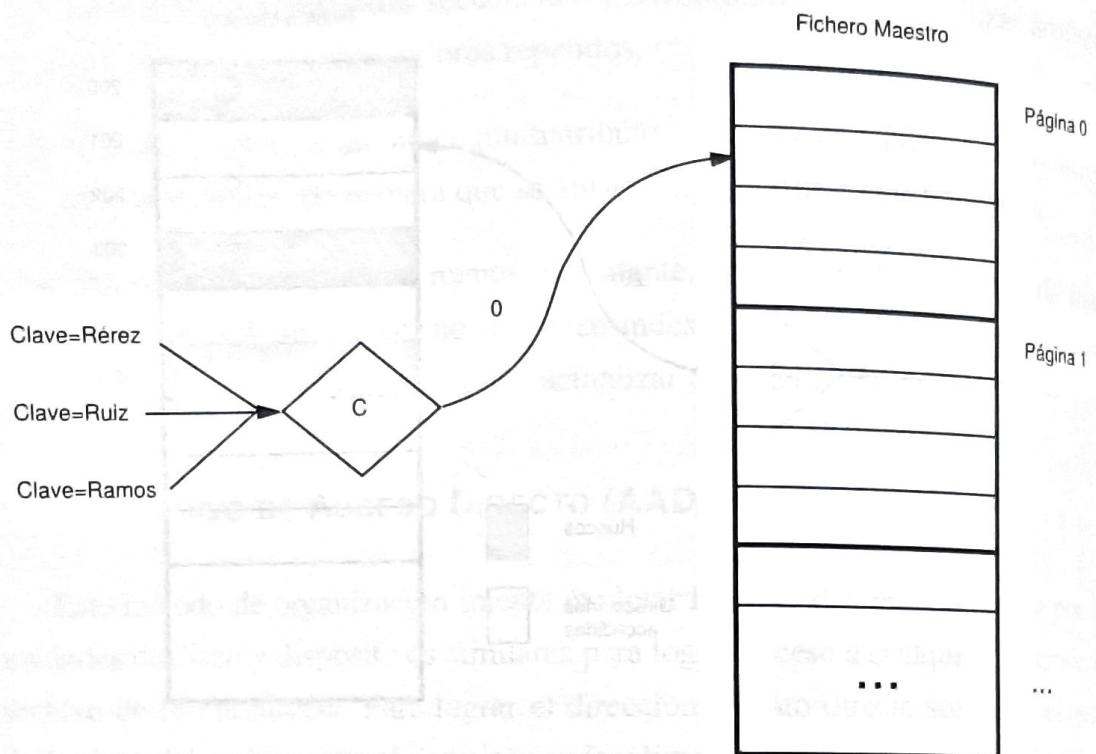


FIGURA 1.16 Hashing paginado.

1.3.4.1. Mecanismos de solución a las colisiones

Existen, en general, dos formas de solucionar el problema de las colisiones dependiendo del uso o no de estructuras adicionales de desbordamiento, ajenas al fichero principal donde se guardarán los datos inicialmente.

- **Direccionamiento Cerrado.** El espacio de direcciones calculadas pertenece a un único archivo. Direccionamientos de este tipo son:
 - *Búsqueda Lineal*. Cuando ocurre una colisión, el registro se almacena en la primera posición libre dentro del mismo bloque. Este método es sencillo y puede utilizarse siempre que $n < m$. El problema es que, cuando un bloque se utiliza con mucha frecuencia, la recuperación de un registro determinado necesita llevar a cabo la lectura previa de muchos otros registros.
 - *Realeatorización*. El registro a insertar se aloja en una posición aleatoria que se calcula mediante un nuevo método de transformación clave.

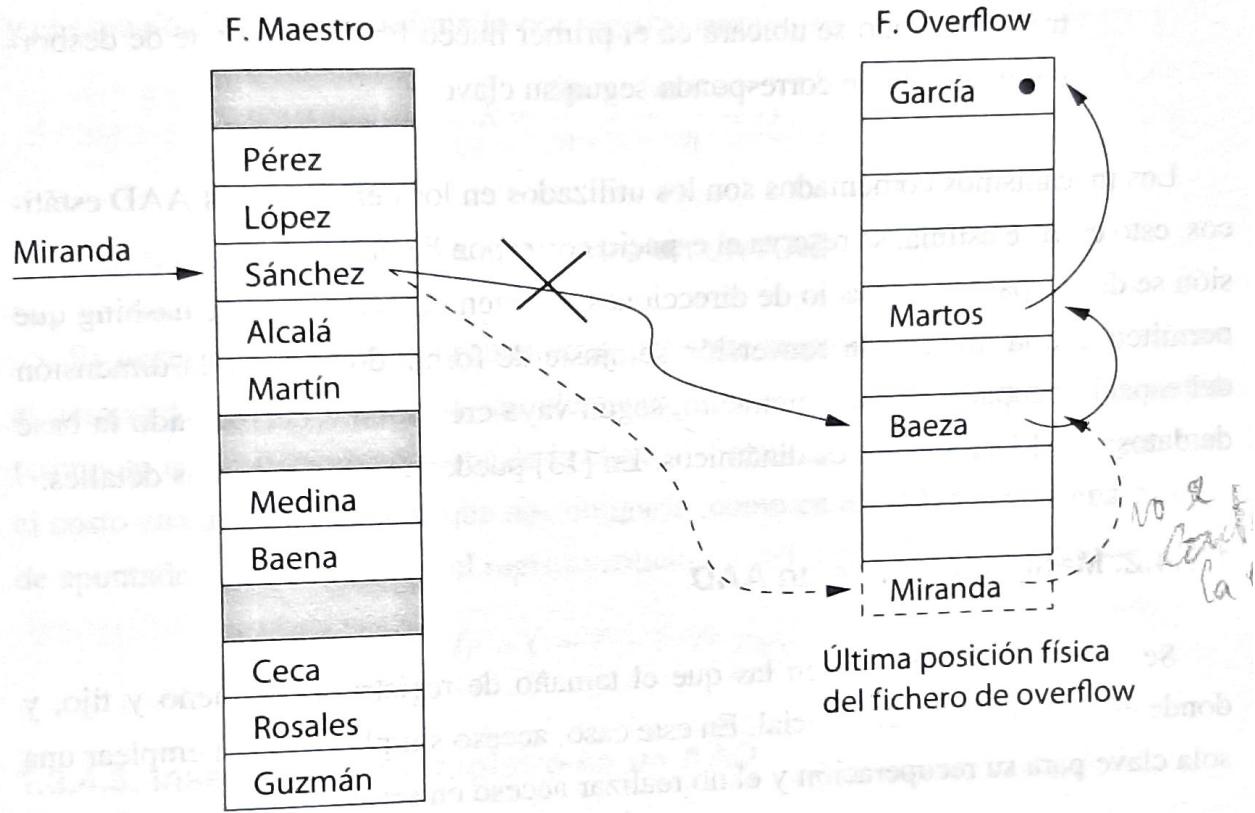


FIGURA 1.17 Inserción en AAD con área de desbordamiento.

dirección (dentro del archivo). Pueden ocurrir nuevas colisiones y es posible que haya que repetir el procedimiento, ya que un proceso como éste no garantiza que se llegue fácilmente a un hueco. Esto conlleva, además, que sea necesario realizar varias lecturas para recuperar dicho registro.

- **Direccionamiento Abierto.** El espacio de direcciones calculadas pertenece a más de un fichero, concretamente al fichero principal y al fichero de desbordamiento. El direccionamiento abierto puede implementarse de dos maneras:

– *Listas enlazadas.* Este método es el que se muestra en la Figura 1.17 y consiste en colocar todos los registros colisionados en el archivo de desbordamiento utilizando un encadenamiento semejante al de los ASI. El archivo de desbordamiento crece conforme se va necesitando espacio, por lo que no es necesario reservar *a priori* un número de registros mayor que el esperado ($n = m$).

– *Bloques de desbordamiento.* En este caso, se adjudican bloques de desbordamiento comunes a un intervalo de claves, de manera que si la posición que corresponde a un determinado registro ya está ocupada, el regis-

tro colisionado se ubicará en el primer hueco libre del bloque de desbordamiento que le corresponda según su clave.

Los mecanismos comentados son los utilizados en los denominados AAD estáticos, esto es, n se estima, se reserva el espacio correspondiente y la función de conversión se define para ese espacio de direcciones. Existen otras técnicas de hashing que permiten que la función de conversión se ajuste de forma dinámica a la dimensión del espacio ocupado en cada momento, según vaya creciendo o decreciendo la base de datos; de ahí su nombre de dinámicos. En [13] pueden encontrarse más detalles.

1.3.4.2. Manipulación de un AAD

Se usan en aplicaciones en las que el tamaño de registro es pequeño y fijo, y donde el acceso rápido es esencial. En este caso, acceso simple significa emplear una sola clave para su recuperación y el no realizar acceso en serie.

El parámetro inicial en su evaluación es el número de huecos en disco de que dispone (m) para el almacenamiento de n registros. Notaremos por c el número de registros que provocan colisiones.

Analizaremos el caso concreto de un AAD con direccionamiento abierto mediante listas enlazadas que consta de un archivo principal y un área separada de desbordamiento que contendrá hasta c registros. Los registros son de longitud fija y contienen un campo apuntador único. No hay claves idénticas, de forma que las colisiones se deben a la aleatorización. Además, $n < m$, de forma que una distribución perfecta no debería llevar a desbordamientos y $c = 0$. Los registros de desbordamiento irán encadenados en una lista de enlaces que comienza en el área principal del fichero maestro.

1.3.4.3. Espacio ocupado por registro

Suponemos como siempre que los registros de datos son de longitud r fija, igual a $r = P + \sum_i V_i$, considerando un campo apuntador en cada registro para la zona de desbordamiento. En ese caso el tamaño total del archivo junto con el área de desbordamiento es:

$$S_F = m \cdot r + c \cdot r = (m + c) \cdot r$$

y el espacio de memoria asignado por registro sería:

$$R = \frac{S_F}{n} = \frac{(m+c)}{n} \cdot r$$

1.3.4.4. Recuperación de un registro en un AAD

Es necesario conocer la probabilidad p de que se produzca una colisión, ya que el tiempo de recuperación será, simplemente, el necesario para encontrar la posición (cómputo de la dirección a partir de la clave, C), realizar la recuperación, T , y sumar el costo en caso de colisión, que nos obligaría, como en el ASI, a seguir una cadena de apuntadores hasta dar con el registro solicitado; así pues, el tiempo estimado será:

$$T_F = C + T + p \cdot T_{F_Cadena}$$

1.3.4.5. Inserción de un registro en un AAD

Como en los casos anteriores, en primer lugar hay que localizar la posición en la que se va a realizar la inserción. En el caso en el que esta posición esté ya ocupada, el procedimiento consistirá en enganchar el nuevo registro como segundo miembro de la cadena de registros colisionados. En este caso, habrá que modificar el apuntador del registro del área principal para darle la dirección del nuevo registro insertado, que se colocará en el área de desbordamiento en la última posición física, evitando así el desplazamiento innecesario de registros, como se muestra en la Figura 1.17.

Estas nuevas escrituras suponen un tiempo total de inserción:

$$T_I = C + 2 \cdot T_W$$

Una de las escrituras proviene de actualizar el campo de dirección P en el registro del fichero principal y la otra consiste en la escritura en el archivo de desbordamiento.

1.3.4.6. Actualización de un registro en un AAD

La actualización de un registro, por cualquier campo que no sea la clave, consiste en encontrarlo y volver a escribirlo en el mismo sitio. Así pues,

$$T_U = T_F + T_W$$

Si se modifica el valor del campo clave será necesario realizar secuencialmente una eliminación y una inserción.

1.3.4.7. Lectura exhaustiva de un AAD

La lectura completa de este tipo de archivos se lleva a cabo rara vez, puesto que la posición física de un registro no guarda relación con la de sus *vecinos* inmediatos según el valor de la clave. En el caso de querer recorrer el fichero secuencialmente, el tiempo estimado será:

$$T_X \cong (m + c) \cdot T$$

1.3.4.8. Reorganización de un AAD

La reorganización se realiza si ha aumentado mucho el número total de registros que se van a almacenar y no se tomaron medidas de previsión para su ampliación. Dicha reorganización es también beneficiosa cuando se han realizado muchas eliminaciones, con lo que las rutas de búsqueda resultan largas y enrevesadas. La reorganización de este tipo de archivos consistirá en leerlo completamente e ir cargándolo nuevamente según indique el nuevo algoritmo de transformación clave-dirección utilizado. Por tanto,

$$T_Y = T_X + T_{Carga}$$

donde $T_{Carga} = \sum_{i=1}^n T_I(i)$ siendo n el número de registros a escribir y T_I el tiempo empleado en la inserción, que, en este caso, depende del número total de registros insertados hasta el momento y del espacio total asignado para su manejo n/m .

1.4. |

1.3.5. LAS ORGANIZACIONES VISTAS. RESUMEN

Para terminar el apartado de las organizaciones, vamos a resumir algunas de las características que las distinguen entre sí, y que nos van a permitir tener un criterio cualitativo. Para tener un criterio cuantitativo será necesario realizar una comparativa de todos los parámetros que han sido estimados para cada una de estas organizaciones. A esto habrá que añadir algunas consideraciones adicionales y/o restricciones que se verán a continuación.

ASI versus ASL

Una gran ventaja de los ASI frente a los ASL es que se puede conocer el número máximo de bloques a visitar para localizar cualquier registro; aun más, se puede saber con exactitud el número de bloques a consultar, ya que la implementación de los ASI en muchos SGBD, se realiza mediante árboles B. *Grosso modo*, éstos garantizan que la longitud del camino entre el nodo raíz y los nodos hojas es la misma para todos los valores de la clave. Al mismo tiempo, resulta un mecanismo simple y eficiente. Todos los ASL tienen la restricción de que solo un atributo clave determina el orden principal del archivo, de manera que los otros atributos no son adecuados como argumentos de búsqueda. En el caso de los ASI densos, como el índice direccional de registros, se permite realizar un acceso secuencial por cualquier campo de interés. Si el direccionamiento es a bloques o páginas, la restricción es la misma que en los ASL.

ASI versus AAD

Partiendo del hecho de que queremos acelerar ciertas consultas:

- El acceso directo es más adecuado cuando se consulta por valor de igualdad. Algunas consultas que se pueden ver beneficiadas son aquellas que contengan alguna reunión natural sobre un atributo con dicho acceso.
- Para consultas por rango son adecuados los ASI.

1.4. EVALUACIÓN DEL SISTEMA

En las secciones anteriores, se ha analizado el matiz técnico de algunos sistemas de archivo. Sin embargo, resultaría de gran interés el analizar cómo se traducirían los parámetros calculados a costos financieros y beneficios económicos.

La mayoría de los factores que afectan al beneficio que puede obtenerse del sistema, solo pueden estimarse; el hecho de que sean solo estimaciones no disminuye la importancia de que se asignen valores cuantitativos a todos los parámetros importantes en el diseño de un sistema. El empleo de valores específicos, documentados

En las siguientes secciones nos centraremos en los siguientes puntos:

- Estimación del uso del sistema.
- Beneficios del sistema.
- Costos de almacenamiento.
- Costos de explotación.
- Análisis comparativo de costo/beneficio.

1.4.1. ESTIMACIÓN DEL USO DEL SISTEMA

Para obtener beneficios de una BD, ésta tiene que usarse. Los beneficios obtenidos, la efectividad con la que se obtengan y el costo de la operación de la BD están relacionados con la carga sobre el sistema. Por *carga* se entiende tanto las demandas de almacenamiento sobre la BD como las operaciones solicitadas por las transacciones.

1.4.1.1. Carga por demandas de almacenamiento

Un aspecto fundamental de la carga es el *volumen de datos* almacenados que se requiere para una operación efectiva de la BD. La estimación de la carga del sistema en términos de volumen, se controla de acuerdo con el número de registros, el número total de atributos a considerar y el promedio de atributos que hay en cada registro. También afectan las longitudes de los campos de datos y los descriptores o nombres de los atributos. Los diferentes diseños de archivo combinarán estos parámetros comunes de formas distintas; así pues, el primer paso a dar será tomar una decisión de diseño.

Carga por recuperación de información

La recuperación de datos es el objetivo central de cualquier sistema de información. La carga operativa se calcula mediante la estimación del número de solicitudes realizadas a un archivo a través de un conjunto de transacciones durante un periodo de tiempo. También debe estudiarse la distribución de estas solicitudes durante un

periodo diario o mensual para conocer cuáles son los períodos de alta demanda. La carga y la duración de los intervalos de fuerte actividad sirven para verificar que el sistema propuesto es el adecuado también en esos momentos críticos.

Para un sistema de BD que deba producir, además de datos, información (informes, resúmenes, gráficos, estadísticas...) la estimación es mucho más difícil. La amplitud de su uso depende de la satisfacción del usuario y ésta depende, a su vez, de factores humanos, de las instalaciones y de la bondad del sistema. Por ejemplo, un informe sobre la historia clínica de un paciente pierde rápidamente su valor si el esfuerzo para obtenerlo es mayor que el interés que tiene el médico por dicho informe pasado cierto tiempo. En este caso, el beneficio medio del sistema será bastante bajo.

Puede resultar necesario un cuidadoso análisis del potencial de producción de información de un sistema y de la satisfacción que ésta información proporciona a los usuarios. Es necesario un diseño flexible para permitir que el sistema crezca y se adapte según las necesidades. Con el fin de evaluar el sistema, se tendrá en cuenta la frecuencia con que se realizan los distintos tipos de operación, esto es,

- *Frecuencia de solicitud de una localización.*
- *Frecuencia de solicitud del registro siguiente.*
- *Frecuencia de solicitud de una búsqueda exhaustiva.*

tanto en términos promedios durante largos períodos de tiempo como para cortos períodos de alta actividad.

Carga por actualización

De la misma manera, será conveniente realizar un análisis estimatorio de cuál va a ser la carga del sistema por causa de actualizaciones en los datos. En este punto habrá que tener en cuenta:

- *Frecuencia de adición de un nuevo registro.*
- *Frecuencia de actualización de un registro existente.*
- *Frecuencia de eliminación de un registro.*
- *Frecuencia de ampliación de un registro.*