



Tema 5: Organización de Ficheros:

Organizaciones Auxiliares

- **Introducción**
 - Concepto de Índice
 - Motivación
 - Tipos de Índice
- **Ficheros Indizados (organizaciones indizadas)**
- **Índices Multinivel**
- **Indización por Árboles**
 - Árboles Binarios
 - Indización por Árboles B
- **Comparativa**



Tema 5.1: Introducción

Las organizaciones base suelen establecerse entorno a un proceso privilegiado (o unos pocos procesos).

Serial → privilegia inserciones

Secuencial → privilegia algún acceso ordenado (y alguna localización)

Direccionada → privilegia localizaciones a través de una clave

- Si existen otras claves de búsqueda (alternativas), la localización se hace muy pesada (búsqueda serial).

A menos que pudiéramos conocer la localización a partir del valor de la clave

... → Solución: almacenar esa información en un archivo aparte

Direccionamiento: establecer ubicación del registro por la clave

Indización: almacenar correspondencia de claves con la ubicación del registro



Tema 5.1: Introducción

- Ejemplos:

Asi-Fin	3		Dum-Mos	Los Tres Mosqueteros	Dumas	...
Dum-Con	4		Per-Cap	El Capitán Alatriste	Pérez-Reverte	...
Dum-Mos	1		Asi-Fin	El Fin de la Eternidad	Asimov	...
Per-Cap	2		Dum-Con	El Conde de Montecristo	Dumas	...
...

Asimov	•		Dum-Mos	Los Tres Mosqueteros	Dumas	...
Dumas	•		Per-Cap	El Capitán Alatriste	Pérez-Reverte	...
Dumas	•		Asi-Fin	El Fin de la Eternidad	Asimov	...
Pérez-Reverte	•		Dum-Con	El Conde de Montecristo	Dumas	...
...

- Un fichero puede contar con más de un índice (*indización múltiple*)



Tema 5.1.1: Concepto de *Índice*

ÍNDICE: *directorio cuya entrada se refiere a un solo registro*

- Es como un listado para traducir punteros (*lógicos a relativos*)

- “Almacenamiento auxiliar utilizado para localizar los registros”
 - Los índices se almacenan en un fichero (o varios) de índices.
 - Los registros se almacenan en un fichero de datos o *fichero principal*
 - El fichero principal mantendrá su *organización básica*
- El acceso selectivo a los registros a través del índice se hará mediante una clave que recibe el nombre de *clave de indización*.
- Es deseable que el índice esté almacenado en memoria principal (total o parcialmente) para optimizar el acceso (selectivo).



Tema 5.1.1: Concepto de *Índice*

¿Qué tipo de puntero es el adecuado en una entrada del índice?

- **La clave siempre es un puntero lógico**
- **El otro puntero suele ser relativo (dir. del cubo), porque...**
 - Los punteros físicos son rápidos, pero muy dependientes...
 - Los punteros lógicos son independientes, pero muy lentos...

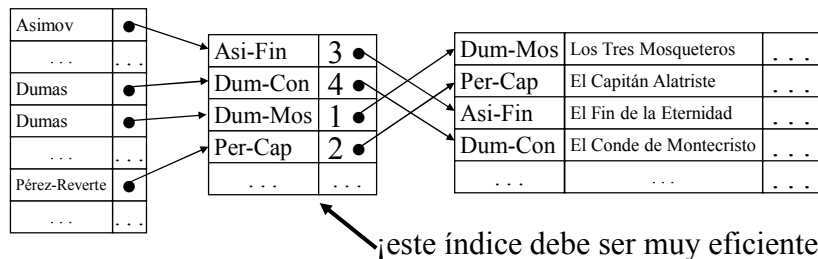
PERO...

- Si la organización base es secuencial, habrá que reorganizar el índice cada vez que se reorganice el área ordenada. (si se aplica partición celular, puede llegar a interesar puntero físico)
- Si el direccionamiento es virtual extensible o dinámico, es preferible utilizar la dirección virtual (puede ser mayor, pero ahorra actualizaciones en el índice)
- Para ganar independencia, se puede hacer un índice intermedio (con puntero relativo al índice intermedio, o incluso puntero lógico)



Tema 5.1.2: Índice Intermedio

- Si se cuenta con más de un índice, se puede considerar un índice intermedio y referir el resto a este. Interesa que este índice sea:
 - **primario** (cada valor de clave identifica unívocamente un registro)
 - de tamaño **reducido** (para que quepa en memoria)
 - casi **constante** (poco o nada volátil)





Tema 5.1.3: Beneficios al Indizar

1. Acceso por Claves Alternativas

Los procesos basados en localización por clave no privilegiada ganan notablemente en eficiencia

2. Aumento de la Tasa de Acierto (en M_{int})

Es el porcentaje de accesos a memoria intermedia que no precisan acceso al soporte. Con gestión de memoria intermedia adecuada, esta tasa en los accesos al índice es muy elevada...

(incluso puede llegar a ser 100% si está completamente contenido en M_{int} , lo que implica ahorrarse los accesos que supone el índice)

3. Reorganización Menos Costosa

Los procesos basados en un orden, aprovechan la ordenación física. Esa ordenación hay que mantenerla, pero es menos costoso mantenerla en un índice (que ocupa pocos bloques) que en el fichero de datos.



Tema 5.1.3: Inconvenientes al Indizar

1. **PROCESOS DE ACTUALIZACIÓN MÁS COSTOSOS**

En los procesos basados en localización se tendrá una ganancia:

- **consultas**
- borrados
- modificaciones
- inserciones (sólo con índice no denso)

Pero los índices hay que crearlos y mantenerlos actualizados, y eso supone un coste adicional en

- **inserciones**
- borrados
- modificaciones

2. Necesidad de Almacenamiento Auxiliar

3. Necesidad de Operaciones de Mantenimiento



Tema 5.2: Ficheros Indizados.

- Estructura de una Organización Indizada :

Fichero de Datos + Fichero de Índices
[+ Directorio] [+ Fichero de Desbordamientos] [+ ...]

- Parámetros de una Organización Indizada:

- *el tipo de índice*
- *la organización del índice*
 - *la gestión de desbordamientos del índice (en su caso)*
- *la organización del fichero de datos (organización base)*



Tema 5.2: Ficheros Indizados.

Taxonomía de Índices

- Según el carácter (identificativo) de la clave de indización:
 - *si es clave de identificación* → **índice primario**
 - *cualquier clave de búsqueda no unívoca* → **índice secundario**
- Según la correspondencia (biyectiva o no) entre entradas y registros:
 - **Denso** (1:1): existe una entrada del índice para cada registro
 - **No Denso** (1:n): varios registros indizados por una sola entrada
- Según el recubrimiento del índice:
 - **Exhaustivo**: todos los registros que deben tener entrada la tienen
 - **Parcial**: no se indizan todos los registros
(*se dejan aparte los que se acceden rara vez, los últimos en ser introducidos, etc.*)

→ *Si el índice exhaustivo falla, indica que el elemento buscado no existe; pero si un índice parcial falla, no aporta ningún valor informativo.*



Tema 5.2: Ficheros Indizados

Organización de Índices

- Organización del Índice:
 - Serial: no ordenado
 - Secuencial: ordenado
 - Direcccionado
 - Organización Base (Org. del Fichero de Datos):
 - Consecutiva (Serial o Secuencial)
(el comienzo de cada registro debe señalarse → marca de fin de registro)
 - No Consecutiva
 - Serial o Secuencial
 - Direccionada
- * para índices no densos, tanto índice como fichero de datos han de ser secuenciales



Tema 5.2: Ficheros Indizados.

tipo de índice	organización del índice	organización base
<i>denso</i>	<i>serial (desordenado)</i>	<i>serial (desordenado)</i>

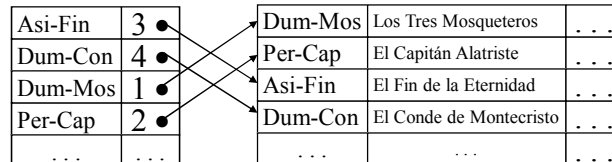
Per-Cap	2 ●	Dum-Mos	Los Tres Mosqueteros	...
Dum-Con	4 ●	Per-Cap	El Capitán Alatriste	...
Asi-Fin	3 ●	Asi-Fin	El Fin de la Eternidad	...
Dum-Mos	1 ●	Dum-Con	El Conde de Montecristo	...
...

- localización: serial sobre el índice (mucho tiempo de proceso)
si el índice cabe todo en memoria, pocos accesos (1)
- tamaño del índice: gen. grande, ya que cada registro tiene una entrada.



Tema 5.2: Ficheros Indizados.

tipo de índice	organización del índice	organización base
<i>denso</i>	<i>secuencial (ordenado)</i>	<i>serial (desordenado)</i>



- localización: ordenada sobre el índice (menor t. de proceso)
- el resto de consideraciones, no cambian (salvo que en este caso aparece la necesidad de reorganizar)



Tema 5.2: Ficheros Indizados.

tipo de índice	organización del índice	organización base
<i>denso</i>	<i>serial (desordenado)</i>	<i>secuencial (ordenado)</i>
	<i>secuencial (ordenado)</i>	

- con clave_indización≠clave_ordenación, aporta variedad en las claves de búsqueda, y además poder usar punteros simbólicos en el índice
- si clave_indización=clave_ordenación, los procesos selectivos a través misma clave no presentan mejoras, y encima precisa reorganizaciones.
 - sin embargo, si aporta la posibilidad de utilizar una técnica para procesos a la totalidad y otra para procesos selectivos.



Tema 5.2: Ficheros Indizados.

tipo de índice	organización del índice	organización base
<i>denso</i>	<i>serial (desordenado)</i>	<i>direccionado</i>
	<i>direccionado</i>	

- es necesario que cumpla: clave_indización \neq clave_direccionamiento
- puede mejorar las consultas a la totalidad ordenadas (pero no es el índice adecuado)

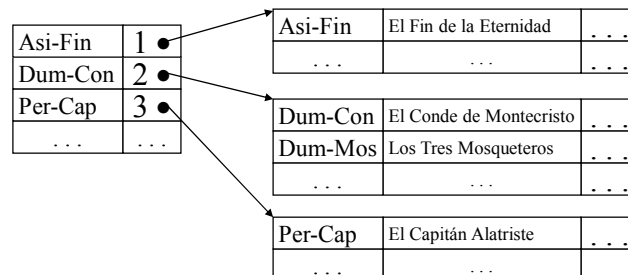
tipo de índice	organización del índice	organización base
<i>denso</i>	<i>secuencial (ordenado)</i>	<i>direccionado</i>

- si clave_indización = clave_direccionamiento se mejora mucho un proceso:
consulta a la totalidad ordenada a través de esa clave
(con clave_ordenación_lógica=clave_indización)



Tema 5.2: Ficheros Indizados.

tipo de índice	organización del índice	organización base
<i>NO denso</i>	<i>secuencial (ordenado)</i>	<i>secuencial (ordenado)</i>



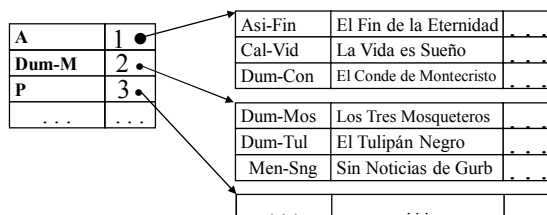
- localización:
 - búsqueda ordenada en el índice \rightarrow se obtiene dirección del cubo
 - se recupera el cubo, y se busca dentro del cubo ordenado



Tema 5.2: Ficheros Indizados.


Características de los Índices No Densos:

- Sólo puede aplicarse si la organización base es secuencial (ordenada), el índice es secuencial (ordenado), y *clave_indización = clave_ordenación*
→ sólo puede existir un índice no denso para cada archivo
- En lugar de utilizar toda la *clave* en la entrada, se pueden usar prefijos



Tema 5.2: Ficheros Indizados.

- Acceso a Índices No Densos:
 - Es un índice de tamaño muy reducido
 - ocupa pocas páginas, y se puede bloquear en memoria intermedia
 - al ser un índice tan eficiente, es idóneo para hacer *clustering* (índice intermedio entre la organización base y el resto de índices)
 - Aporta varias posibilidades de acceso:
 - procesos ordenados (a la totalidad): acceso serial (ordenado)
 - procesos selectivos (solución única): a través del índice
 - mixtos (selección de un rango): acc. indizado (1^{er} elemento) + serial
 - Al ser índice ordenado, surge el clásico problema de desbordamientos.


 Tema 5.2: Ficheros Indizados.
Índice NO Denso

- Inserción de un elemento que desborda y reorganización

Asi-Fin	1	•
Dum-Con	2	•
Per-Cap	3	•
...	...	

Dum-Con	El Conde de Montecristo	...
Dum-Mos	Los Tres Mosqueteros	...
Ner-Poe	Veinte Poemas de Amor	...


Men-Sin	Sin Noticias de Gurb	...
---------	----------------------	-----



Soluciones: las clásicas de org. secuenciales

- Gestión por Área Desordenada (de Desbordamiento)
 - tras el área ordenada, o en un fichero independiente (f. desbordamientos)
 - con o sin encadenamiento
- **Reorganización local:** ¡cuidado!
¡la reorganización puede provocar cambios en el índice!


© 2008 LaBDA – Universidad Carlos III Madrid FF - 19

 Tema 5.2: Ficheros Indizados.
Índice NO Denso

Asi-Fin	1	•
Dum-Con	2	•
Ner-Poe	3	•
...	...	

Dum-Con	El Conde de Montecristo	...
Dum-Mos	Los Tres Mosqueteros	...
Men-Sin	Sin Noticias de Gurb	...

Ner-Poe	Veinte Poemas de Amor	...
Per-Cap	El Capitán Alatriste	...
...




Reorganización local: al insertar en un cubo, desborda.

- el elemento desbordado se pasa al siguiente cubo (¡cambia la clave que indiza al cubo siguiente!, como en la figura)
- ¡Peligroso! Puede producir reorganizaciones en cascada

Soluciones:

- Introducir Espacio Libre Distribuido en cada cubo
- Intercalar cubos vacíos (de inicio o dinámicamente) → **partición celular**

© 2008 LaBDA – Universidad Carlos III Madrid FF - 20



Tema 5.2: Ficheros Indizados

Índices Direccionados


Índices altamente eficientes en procesos de selección unívoca (un solo acceso para recuperar la entrada), de conteo, y de existencia.

Inconvenientes:

- dejan de ser eficientes si se producen desbordamientos (no interesan si tienen más de uno o dos cubos de desbordamientos)
- son muy poco densos → no pueden mantenerse en memoria (sólo se mantendrán en memoria los cubos de desbordamientos)
- fuerte dependencia de la dispersión
- funcionan mejor si se actualiza poco → índices poco volátiles

* Utilizados como índices temporales (y para hacer *clustering*)


© 2008 LaBDa – Universidad Carlos III Madrid
FF - 21



Tema 5.2.1: Operaciones sobre Ficheros Indizados

- Operaciones sobre la totalidad del fichero:
 - **Creación**: hay que crear el índice (al crear el fichero o posteriormente)
 - **Borrado**: si se borra el fichero de datos, hay que borrar el índice
- se puede destruir el índice sin borrar el fichero de datos.
 - **Consulta**: generalmente, interesa hacerla sin contar con el índice
(excepción: proceso ordenado, con índice ordenado y f. de datos serial)
- Operaciones selectivas (sobre registros aislados):
 - **Localización**: se hará a través del índice
 - **Operaciones de Actualización**: pueden afectar al índice
(es necesario actualizarlo)


© 2008 LaBDa – Universidad Carlos III Madrid
FF - 22



Tema 5.2.1: Operaciones sobre Ficheros Indizados

- **Operaciones Selectivas (I):**
 - Recuperación:
 - Consulta Selectiva: localización + recuperación del registro
(se recupera un cubo y se busca dentro el registro)
 - Actualización (1): Inserción
 - índice denso: inserción en ambos (f. de datos e índice)
cada fichero (datos e índice) mantiene sus peculiaridades (orden, ...)
 - índice no denso: inserción sólo en fichero de datos
(puede producir desbordamientos, que a su vez pueden afectar al índice)


© 2008 LaBDA – Universidad Carlos III Madrid FF - 23




Tema 5.2.1: Operaciones sobre Ficheros Indizados

- **Operaciones Selectivas (II):**
 - Actualización (2): Borrado
 - índice consecutivo: borrado en índice → aparecen huecos
Los huecos pueden ser gestionados mediante una *lista de huecos*
 - si el fichero de datos es secuencial, la lista de huecos debe ser ordenada
 - si no lo es, la lista de huecos podría estar ordenada por tamaños
 - si los registros son de longitud fija, simplemente es una lista
A menudo se prefiere ‘cargar’ con los huecos y reorganizar periódicamente
 - índice no consecutivo: se genera espacio en el cubo (*no pasa nada*)
 - índice no denso: problemático si el elemento borrado era índice de su cubo
En este caso, se hace necesario **reorganizar** ese cubo y **actualizar** el índice.

© 2008 LaBDA – Universidad Carlos III Madrid FF - 24

	Tema 5.2.1: Operaciones sobre
Ficheros Indizados	
<ul style="list-style-type: none"> • <u>Operaciones Selectivas (III):</u> <ul style="list-style-type: none"> • <u>Actualización (3): Modificación</u> <ul style="list-style-type: none"> - <i>índice denso</i>: modificación en el fichero de datos <ul style="list-style-type: none"> · si cambia la clave, modificación en ambos (borrado y reinsertión) · si el fichero de datos tiene registros variables, o es secuencial, puede dar problemas: una solución es borrar y reinsertar - <i>índice no denso</i>: modificación en el fichero de datos <ul style="list-style-type: none"> · si cambia la clave y esta es el índice del cubo, será necesario modificar en ambos y comprobar si es necesaria una reorganización · también pueden dar problemas los registros variables (raro) 	
<p>© 2008 LaBDA – Universidad Carlos III Madrid FF - 25</p>	

	Tema 5.2.1: Operaciones sobre
Ficheros Indizados	
<ul style="list-style-type: none"> • <u>Gestión de Desbordamientos en Ficheros Índice:</u> <ul style="list-style-type: none"> • <u>Área (o Fichero) de Desbordamiento</u> <ul style="list-style-type: none"> - con organización serial - con encadenamiento - con punteros múltiples (el índice tiene una colección de punteros) - nueva entrada en el índice para cada cubo de desbordamiento • <u>Espacio Libre Distribuido:</u> procurar espacio libre en el área de datos <ul style="list-style-type: none"> - intercalar cubos completamente vacíos (se podría emplear también para secuencial indizado denso, pero no se hace) - dejar espacio en cada cubo: admite técnicas para mantener ese espacio <ul style="list-style-type: none"> · <i>rotaciones</i>: traspasar elementos de un cubo lleno a otro vecino con sitio · <i>particionamiento celular</i>: dividir un cubo lleno en dos semi-lenos 	
<p>© 2008 LaBDA – Universidad Carlos III Madrid FF - 26</p>	



Tema 5.3: Índices Multinivel

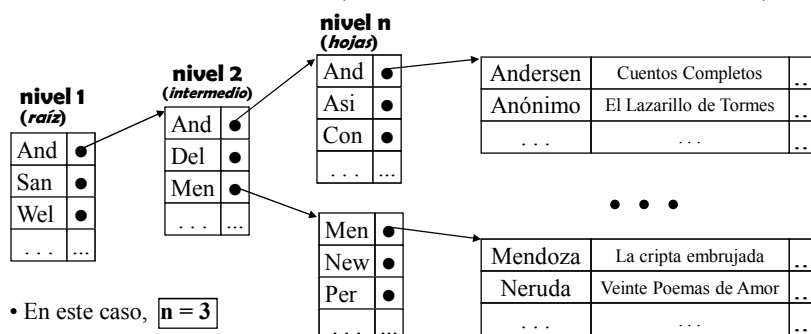
Índices muy grandes

- El índice denso puede ser muy grande → elevado coste de almacenamiento y de reorganización (si además es ordenado).
- Un índice no denso precisa menos entradas. Así, es apropiado cuando hay muchas entradas y el índice no cabría en memoria
- Pero si sigue sin caber, o si es denso, habrá que fraccionarlo.
¿Cómo organizar esos fragmentos de índice (páginas)?
- ¿Qué tal con un ‘índice de índices’? → **índice binivel**
- ¿Y si ese índice sigue siendo grande? → **índice multinivel**



Tema 5.3: Índices Multinivel

Son índices con n niveles (el nivel n es índice del nivel $n+1$)



- En este caso, $n = 3$
- Como el nivel n es ordenado, puede ser un índice ‘no denso’ (siempre que el archivo de datos sea secuencial, y $cl_ordenación = cl_indización$)
- Igual que en los *no densos*, las claves en las entradas pueden ser prefijos



Tema 5.3: Índices Multinivel: Características

- Se comportan mal en procesos de actualización (mala reorganización)
- Por ello, se usan sobre archivos constantes (no volátiles)
- **Corolario: el número de niveles es fijo (hasta que se reorganice)**
- Tiempo de Acceso:
 - Con un índice de este tipo, es ventajoso que todo el nivel 1 (raíz) se bloquee en memoria intermedia (se ahorra un acceso al soporte).
 - Si un cubo de cualquier nivel se lee en un acceso ($T_{\text{cubo}} = T_{\text{bloque}}$), el número de accesos para conseguir la dirección de los datos será n-1
 - Si el cubo de datos se lee en un acceso, el total serán n accesos
- **Problemas:**
 - el índice puede crecer y, si no aumenta el cubo, acabará desbordando
 - el número de accesos puede hacerse muy elevado
 - mantener el índice ordenado es muy costoso (mucho reorganización)



Tema 5.4: Indización en Árbol

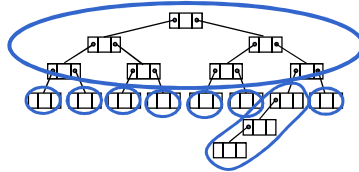
Caso particular de índice multinivel (en forma de árbol)

- **Árbol Binario:** cada entrada del índice es un nodo en un árbol
 - implementación: cada nodo contiene una clave, un puntero externo (a los datos) y dos punteros internos (a los nodos hijo, izquierdo y derecho)
 - ventajas: sencillez y uso eficiente en memoria (intermedia o principal)
 - inconveniente: desequilibrio
 - altísima degeneración → ineficiente en soporte secundario
 - se almacenan para leerlos completos, usarlos, y volcarlos completos
- **Árbol AVL:** árbol binario que contempla procesos de reorganización local
 - inconveniente: nodos lógicamente próximos no son físicamente vecinos
- **Árboles Binarios Paginados:** resuelven el problema de vecindad
 - inconveniente: bajísima densidad



Tema 5.4.1: Árboles Binarios Paginados

- **Concepto:** en cada página se almacena un subárbol de m niveles (completo)



- en este ejemplo,

$$m = 3$$

- De este modo, cada acceso procura m nodos válidos para la búsqueda
- **Ventaja:** se ahorran muchos accesos

$$\frac{\text{acceso binario: } \log_2 (n+1)}{\text{acceso binario paginado: } \log_{m+1} (n+1)}$$

- **Inconveniente:** disminuye notablemente la densidad de ocupación (p.e., si cabían 14 nodos/bloque, ocupaba 2 bloques y ahora... ¡ocupa 9 bloques!)



Tema 5.4.1: Árboles Binarios AVL Paginados

Su uso si puede ser ventajoso en soporte secundario, pero...

- No se garantiza una ocupación mínima de página (desperdicio de espacio).
- Nodos muy distantes (orden lógico) son vecinos (almacenados en la misma página) → ineficiencia en procesos ordenados de tasa de actividad elevada
- Los nodos afectados en las reorganizaciones locales tampoco son próximos en general → se producen más fallos de página.
- Se requieren el doble de punteros internos que entradas

Solución:

- Incluir varias entradas por nodo (y, por tanto, varios descendientes)
- Construir el árbol en orden ascendente (el separador pertenece al nodo que desborda)

} **árboles B**

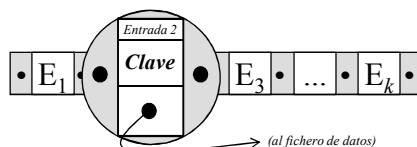


Tema 5.4.2: Indización en Árbol B

- *Propuestos por Bayer y McCreight*

Idea: ya que inicialmente no se conoce el elemento que es mejor separador, se comienza por las hojas. A medida que crezca, se construye hacia arriba.

- **Nodo:** cada nodo va a llenar la página; contiene entradas de índice (pares *clave indización-puntero a los datos*) y punteros (para apuntar nodos hijo)



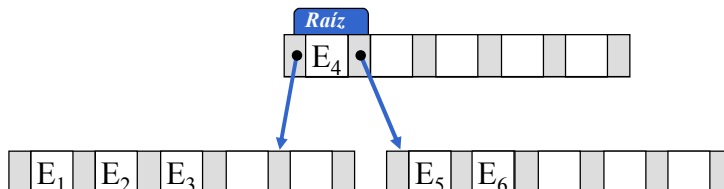
- **Orden del árbol:** indica la capacidad de los nodos (y por ende, del árbol)
 - según las entradas: el n° mínimo de claves de un nodo (*Bayer*)
 - según los punteros (hijos): n° máximo de hijos de un nodo (*Knuth*)




Tema 5.4.2: Indización en Árbol B

Árboles B : Partición y Promoción

- Las entradas dentro de un nodo van ordenadas
- Cuando un nodo *desborda*, se divide en dos y se promociona el elemento intermedio hacia el nivel superior (ese elemento se lleva dos punteros: uno hacia cada hijo, es decir, hacia cada uno de esos dos nuevos nodos)






Tema 5.4.2: Indización en Árbol B

Árboles B: otro ejemplo de Partición y Promoción

© 2008 LaBDa – Universidad Carlos III Madrid

FF - 35



Tema 5.4.2: Indización en Árbol B

Árboles B : Observaciones

- Si el árbol es de orden m , cualquier nodo tendrá a lo sumo m descendientes
- Si un nodo tiene m descendientes (no hoja), tendrá $m-1$ entradas
- Corolario: un nodo de un árbol de orden m tiene a lo sumo $k=m-1$ entradas
- En un nodo (T_{nodo} bytes) caben m punteros internos y k entradas, luego:


$$m \cdot T_{\text{ptro_interno}} + k \cdot T_{\text{entrada}} \leq T_{\text{nodo}}$$

(con esta fórmula y $k=m-1$ se podrá hallar el orden del árbol)

- El nodo raíz tiene al menos un elemento y, por tanto, al menos 2 hijos.
- Todas las hojas están al mismo nivel \rightarrow árbol '*plano*' (bushy)
- La inserción de una nueva entrada puede dar lugar a una (o más) particiones
- El borrado de una entrada puede dar lugar a una (o más) concatenaciones

© 2008 LaBDa – Universidad Carlos III Madrid

FF - 36



Tema 5.4.2: Indización en Árbol B

Árboles B : Propiedades

- Todos los nodos menos el raíz garantizan una **ocupación mínima**: $k_{\min} = \lfloor \frac{k}{2} \rfloor$

Corolario

- ¿Cuántos descendientes como mínimo tienen los nodos intermedios? (suponiendo política de 'dividir cuando desborda')


$$m_{\min} = k_{\min} + 1 \rightarrow m_{\min} = \lfloor \frac{m+1}{2} \rfloor$$

- Tamaño del fichero de índices
Se puede obtener una cota superior del fichero de índices

$N^{\circ}_{\max} \text{ nodos fichero} = n^{\circ} \text{ entradas fichero} / k_{\min}$

$T_{\max} \text{ fichero} = n^{\circ}_{\max} \text{ nodos fichero} \cdot T_{\text{nodo}}$

© 2008 LaBDa – Universidad Carlos III Madrid
FF - 37



Tema 5.4.2: Indización en Árbol B

Árboles B : Propiedades (II)

El n° de niveles (n) para un árbol de orden m y e entradas tiene **cota superior**

nivel	nodos	entradas	acumulado
1	1	1	1
2	2	$2 \cdot k_{\min}$	$1 + 2 \cdot k_{\min}$
3	$2 \cdot m_{\min}$	$2 \cdot m_{\min} \cdot k_{\min}$...
...
$\rightarrow n$	$2 \cdot m_{\min}^{n-2}$	$2 \cdot m_{\min}^{n-2} \cdot k_{\min}$	$(2 \cdot m_{\min}^{n-1}) - 1$
$n+1$	$2 \cdot m_{\min}^{n-1}$	$2 \cdot m_{\min}^{n-1} \cdot k_{\min}$	$(2 \cdot m_{\min}^n) - 1$

Cota Superior:

$n \leq 1 + \log_{\lfloor \frac{m+1}{2} \rfloor} \left(\frac{e+1}{2} \right)$

$\leq e$

$> e$ (este nivel es imposible)

© 2008 LaBDa – Universidad Carlos III Madrid
FF - 38



Tema 5.4.2: Indización en Árbol B

Árboles B : Propiedades (III)

- El número de accesos al soporte coincide, en general, con el nº de niveles:
 - Dado que la raíz estará siempre en memoria, contamos un acceso menos
 - Hay que contar un acceso más, desde la entrada de índice al fichero de datos
- El tiempo de acceso (máximo) a registro aleatorio dependerá del número de niveles (n) y del tiempo de acceso a un nodo:

$$t_{m\acute{a}x} = n \cdot t_{nodo}$$

- Este tiempo de acceso será cota superior del tiempo de acceso en cualquier momento de la vida del fichero (mientras no cambien sus condiciones).
- Puede calcularse una cota inferior (en base al número de niveles del árbol perfectamente construido), para conocer el tiempo de acceso óptimo (después de la creación del índice y de cada reorganización que sufra).



Tema 5.4.2: Indización en Árbol B

Árboles B : Valoración

Aspectos Positivos:

- Existe una cota superior razonable del número de accesos a soporte
- Generalmente, las operaciones (de inserción o borrado) requieren reestructurar una página. Y si son más, suelen ser pocas páginas.
- En el peor caso, las páginas están ocupadas a la mitad (aproximadamente)

Aspectos a Mejorar

- Si las entradas son grandes, el orden puede ser demasiado pequeño
- La densidad (mínima) de los nodos es muy mejorable
- En las hojas se desperdicia mucho espacio (no necesitan punteros)



Tema 5.4.2: Indización en Árbol B*

Idea: se pretende **aumentar la densidad** de los nodos

Para ello, en lugar de dividir un nodo en dos, se dividirán dos nodos en tres.
Así, en lugar de conseguir una ocupación mínima del 50% se obtendrá el 66%

- Cuando un nodo desborda, en lugar de dividir, se procurará ceder uno de sus elementos a su vecino (rotación). Si su vecino también está lleno, se dividen.
- Aparece otra ventaja: *un desbordamiento no siempre supone división/promoción*
- Por lo demás, el resto del funcionamiento es como el de los árboles B.

Propiedades

• Todos los nodos menos el raíz garantizan una ocupación mínima:

$$k_{\min} = \lfloor \frac{2k}{3} \rfloor$$

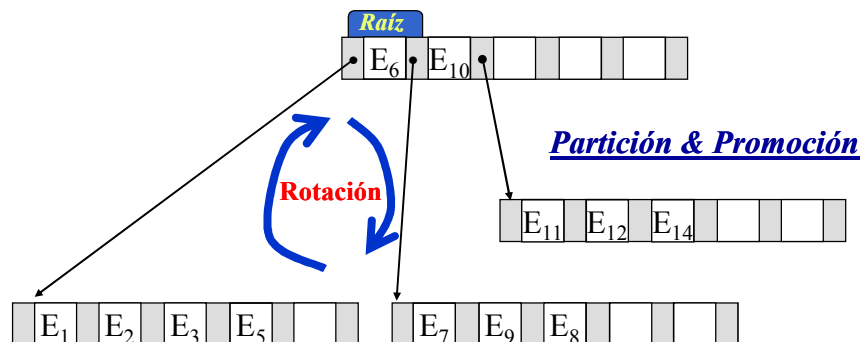
• Los nodos intermedios cuentan con $\frac{2k}{3} + 1$ descendientes \rightarrow
(suponiendo política de 'dividir cuando desborda')

$$m_{\min} = \lfloor \frac{2m+1}{3} \rfloor$$



Tema 5.4.2: Indización en Árbol B*

Rotación, Partición y Promoción (ejemplo)

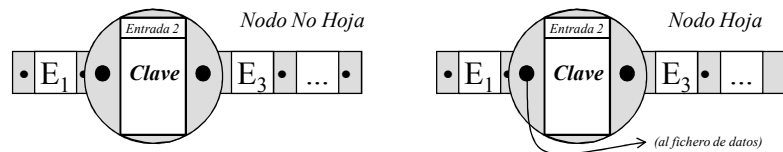




Tema 5.4.2: Indización en Árbol B⁺

Idea: se pretende reducir el tamaño de las entradas de índice

Se va a considerar que los nodos intermedios son separadores, y que todas las entradas de índice están en las hojas. En los nodos no hojas sólo es necesaria la clave (se ahorra el espacio que ocupa el puntero al f. de datos). Por tanto, en cada página caben más claves (el orden es mayor).




Por otro lado, las hojas, al no tener hijos, no utilizan esos punteros. Ahora los van a utilizar para apuntar al área de datos (puntero izqdo. ~ puntero externo)



Tema 5.4.2: Indización en Árbol B⁺

Partición y Promoción

- El funcionamiento es similar al de los árboles B
- Sólo cambia en la promoción de entradas de nodo hoja:
 - el **elemento** que **promociona desde una hoja**, debe **permanecer** en una hoja resultante de la partición
 - en los nodos no hoja, se opera como con los árboles B
- Observar que en los nodos hoja hay tantos punteros como claves. Sin embargo, se usa un puntero adicional para apuntar al siguiente nodo hoja. Este apuntamiento se realiza durante la partición, asignando:
 - al encadenamiento del nodo derecho, el puntero existente
 - al encadenamiento del izquierdo, la dirección del nodo nuevo
- Se consigue así un **encadenamiento de hojas** que proporciona un mecanismo de acceso alternativo eficiente para ciertos procesos.



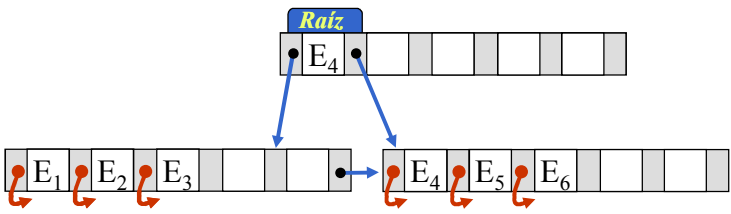
Tema 5.4.2: Indización en Árbol B⁺

Partición y Promoción


Ejemplo de Partición y Promoción: nodo hoja/raíz (primera partición)

Parámetros: Promoción por la izquierda (en este caso)

- En consecuencia, la condición de comparación en nodo no hoja será:
 - $C_{búsqueda} < E \rightarrow anterior;$
 - *e.o.c.* $\rightarrow siguiente$
 (en la búsqueda, si encaja en nodo no hoja se toma el siguiente)



© 2008 LaBDa – Universidad Carlos III Madrid
FF - 45

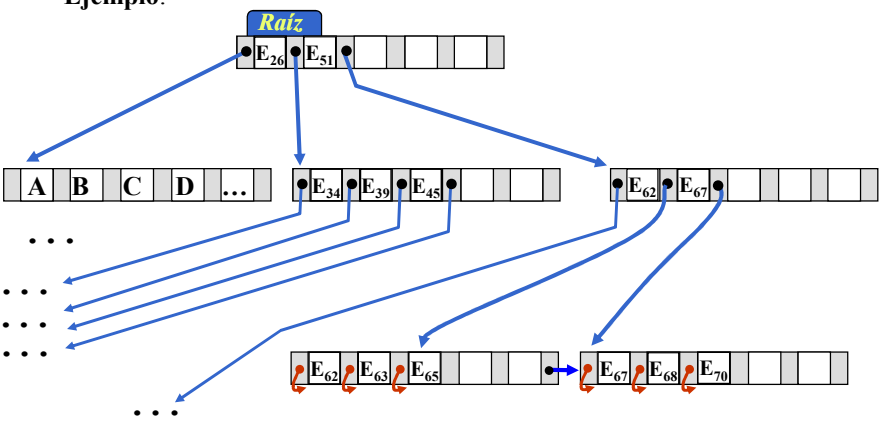


Tema 5.4.2: Indización en Árbol B⁺


Partición y Promoción

- En los nodos no hoja, se promociona igual que con los árboles B.

Ejemplo:



© 2008 LaBDa – Universidad Carlos III Madrid
FF - 46



Tema 5.4.2: Indización en Árbol B⁺


Propiedades (k_{\min} y m_{\min})

- Orden del árbol (m): se calcula para nodos no hoja, como en árboles B; teniendo en cuenta que las entradas esos nodos carecen de puntero externo

$$m \cdot T_{\text{puntero_interno}} + (m-1) \cdot T_{\text{clave}} \leq T_{\text{nodo}}$$
- Ocupación máxima (k) de los nodos hoja: si los tamaños de los punteros interno y externo son distintos, convendría calcularla por separado

$$k \cdot (T_{\text{clave}} + T_{\text{puntero_externo}}) + T_{\text{puntero_interno}} \leq T_{\text{nodo}}$$
- La ocupación mínima de las hojas será: $k_{\min} = \lfloor \frac{k+1}{2} \rfloor$
(suponiendo política de 'dividir cuando desborda')
- La ocupación mínima de los nodos intermedios será $\lfloor k/2 \rfloor \rightarrow$ $m_{\min} = \lfloor \frac{m+1}{2} \rfloor$
(la promoción en estos se opera como en los nodos de un árbol B)

© 2008 LaBDA – Universidad Carlos III Madrid
FF - 47



Tema 5.4.2: Indización en Árbol B⁺

Profundidad y Tamaño

Cálculo del número de niveles:

- El nivel de las hojas es el **nivel n** . ¿Cuántas hojas?
 $n^{\circ} \text{ hojas} = \lfloor e / k_{\min} \rfloor$
 $e = n^{\circ} \text{ total de entradas}$
- El n^o de nodos en el **nivel $n-1$** depende del número de nodos del nivel n

$$n^{\circ} \text{ nodos } (n-1) = \lfloor n^{\circ} \text{ nodos } (n) / m_{\min} \rfloor$$
- Cuando se llega a un nivel con un solo nodo (la raíz), este será el **nivel 1**.
(se tiene que el nivel $n-x=1$, y se puede despejar $n = \text{profundidad del árbol}$)
- **Tamaño máximo del fichero índice:**
se calcula como la suma de los nodos necesarios para cada nivel ($\sum_{i=1}^n \text{nodos}(i)$) multiplicado por el tamaño de un nodo.

© 2008 LaBDA – Universidad Carlos III Madrid
FF - 48



Tema 5.4.2: Indización en Árbol B⁺

Observaciones


- Los nodos intermedios son como un índice no denso (del nivel siguiente). Las entradas de estos nodos sólo necesitan parte de la clave (**prefijos**).
- Recordar que los nodos no hoja del árbol B⁺ se operan como nodos B
- El orden es mayor que el de los árboles B. Esto hace que existan menos nodos separadores. En consecuencia, la profundidad del árbol será menor (menor o igual en general) ¡aunque se repitan entradas!
- Todos los elementos tienen el mismo coste de acceso.
(aunque algunos elementos empeoren, la media del coste de acceso mejora)
- Se pueden combinar las aproximaciones B* y B⁺ (*árboles B⁺**)
- Son muy utilizados en el ámbito profesional (por ejemplo, los utiliza Oracle™)



Tema 5.4.2: Indización en Árbol B⁺

Encadenamiento de las Hojas

- El encadenamiento de hojas proporciona mecanismos de acceso alternativo.
- Es como tener varios índices (arbóreo y secuencial) en uno sólo.
- Resulta especialmente útil para procesos ordenados cuyo orden lógico coincida con la clave de indización
- Los procesos que se pueden beneficiar del encadenamiento son:
(ejemplos en base a un índice en árbol B⁺ con clave indización '*fecha* (dd-MM-AAAA)')
- **procesos a la totalidad ordenados**
Ejemplo: sacar un listado de todos los registros ordenados cronológicamente
- **procesos selectivos con tasa de actividad elevada**
Ejemplo: recuperar todos los registros con fecha en mes de 'Mayo'
- **procesos ordenados con varios resultados (*rangos*)** → acceso mixto
Ejemplo: recuperar todos los registros entre el 01-05-2005 y el 30-06-2005
- Los accesos mixtos consisten en recuperar la primera entrada (01-05-2005) a través del árbol, y el resto de entradas se recuperarán con el encadenamiento




Tema 5.4.2: Familias de Árboles B

Consideraciones Finales

- ✓ Se puede **organizar un fichero de datos en árbol**
(equivale a sustituir el puntero por el correspondiente registro de datos).
Sólo interesa si el orden es alto (registros pequeños y páginas grandes).
- ✓ Las mejoras logradas con árboles B⁺ y B* son combinables
- ✓ Existen variantes que proporcionan mayor eficiencia
 - menor número de accesos
 - menor profundidad del árbol
 - mayor aprovechamiento de espacio
 - mayor orden
 - mayor ocupación
 - mínimo coste de actualización

© 2008 LaBDa – Universidad Carlos III Madrid
FF - 51



Tema 5.5: Indización Mediante Claves Secundarias

- **Índice Secundario:** la diferencia es que la selección no es unívoca
(*pueden existir varias entradas de índice para cada valor de la clave*)
- Si el índice no está ordenado, habrá que recorrerlo entero siempre
En consecuencia, interesa contar con un índice denso ordenado
(accesible mediante búsqueda dicotómica extendida, ver FF-4-13).
- También se pueden utilizar *índices multinivel* (por ejemplo, en árbol)
teniendo siempre en cuenta que el nivel n tendrá que ser denso.
- **Tipos de Índice Secundario:** (atendiendo a la completitud)
 - *Índice Exhaustivo:* todos los registros indizados
 - *Índice Parcial:* sólo se indizan los *valores* más interesantes
(si no se encuentra ningún registro para el valor buscado, puede querer decir que no existe o que no está indizado → en estos casos, es obligado realizar un recorrido serial para completar la búsqueda).

© 2008 LaBDa – Universidad Carlos III Madrid
FF - 52



Tema 5.5: Indización Mediante Claves Secundarias

• Optimización:

- Habrá varias entradas con la misma clave... (valores repetidos)
- Sería muy interesante almacenar **sólo una vez el valor** de cada clave, y junto a ella todos sus punteros (*lista de punteros*).

ENTRADA \equiv **clave** · **long_lista** · (**puntero_externo**)^{long_lista}

• Corolarios: sólo habrá una entrada por valor de la clave...

- ↑ Al buscar, sólo hay que recorrer el índice hasta encontrar una entrada
- ↓ Para insertar, hay que buscar la entrada correspondiente
- ↓↓ Y en caso de que no exista, ¡hay que recorrer todo el fichero!

→ Con listas, conviene tener el índice ordenado, y siempre **no consecutivo**

- Esta optimización es aplicable a cualquier índice (árboles B, B⁺, etc.)




Tema 5.5: Comparativa Organizaciones Indizadas

Índices Seriales:

<i>Ventajas</i>	<i>Inconvenientes</i>
Sencillez (diseño & implementación)	Búsqueda 'pesada'
Inserción eficiente	La organización degenera (requiere procesos de mantenimiento)

Índices Dispersos:

<i>Ventajas</i>	<i>Inconvenientes</i>
Búsqueda muy rápida	Degeneran muchísimo
Buenos para índices temporales	Poco densos (tamaño grande)



Tema 5.5: Comparativa

Organizaciones Indizadas


Índices Secuenciales:

<i>Ventajas</i>	<i>Inconvenientes</i>
Inserción eficiente	Búsqueda 'menos pesada'
Procesos ordenados eficientes (con $cl_ord_lógica = cl_ord_física$)	La organización degenera mucho (requiere procesos de mantenimiento)
Índices secundarios por listas	

Índices (Secuenciales) No Densos:

<i>Ventajas</i>	<i>Inconvenientes</i>
Tamaño reducido (cabe en M_{int})	Determina organización base
Adecuados para hacer <i>clustering</i>	Sólo un índice no denso por archivo

© 2008 LaBDa – Universidad Carlos III Madrid
FF - 55



Tema 5.5: Comparativa

Organizaciones Indizadas

Índices Multinivel:

<i>Ventajas</i>	<i>Inconvenientes</i>
Índices grandes eficientes	Actualización requiere localización
Al crearlos, están bien equilibrados	Capacidad de crecimiento limitada (requiere procesos de reorganización)

Índices en Árbol Binario:

<i>Ventajas</i>	<i>Inconvenientes</i>
Coste equilibrado	Actualización requiere localización
Índices flexibles	Degeneración puede ser crítica
	Requieren paginación y rotaciones AVL, y entonces se hacen pesados y poco densos

© 2008 LaBDa – Universidad Carlos III Madrid
FF - 56



Tema 5.5: Comparativa Organizaciones Indizadas

Índices en Árbol de la familia B:

<i>Ventajas</i>	<i>Inconvenientes</i>
Coste equilibrado	Actualizaciones con 'pocos' accesos
Índices auto-mantenibles	Mejoran al reorganizarlos (opcional)
Reorganizaciones locales...	...que pueden requerir varios accesos
Reorganizaciones infrecuentes...	...que raramente sobrecargan el sistema
Razonablemente densos...	...pero no caben enteros en M_{int}
Buenos para primarios y secundarios...	
Útiles para varios procesos (sobre la misma clave)	
... ..	

* El SGBD Oracle® utiliza habitualmente índices en árbol B^+ con prefijos



Tema 5.6: Comparativa

Organizaciones Indizadas vs. Organizaciones Base

- Procesos que **mejora**: cualquiera que requiera localización
 - Procesos que **empeora**: actualizaciones (inserciones, ...)
- } *equilibrio*

Rasgos: tipo de índice, tamaño de página, gestión de desbordamientos

Factores que se optimizan: tiempo de respuesta, densidad,
y disminuye la necesidad de reorganizaciones

<i>Ventajas</i>	<i>Inconvenientes</i>
Manejable y optimizable	Precisa almacenamiento auxiliar
Mejora acceso por cl. alternativas	Acceso <i>casi</i> óptimo
Coste localización equilibrado	Empeora otras operaciones
Su uso es opcional	Complejidad (diseño y prg.)