Tema 9: Contenidos

9.1 Introducción

9.2 El índice en la organización indexada

9.3 Organización indexada simple

9.4 Índices de mayor tamaño

9.5 Índices no densos

9.6 Organización del índice en árbol B

9.7 Árboles B+

9.8 Árboles B+ frente a árboles B

9.9 Estimación de la estructura del índice no denso

9.10 Ocupación de la organización indexada

9.11 Acceso a los archivos indexados

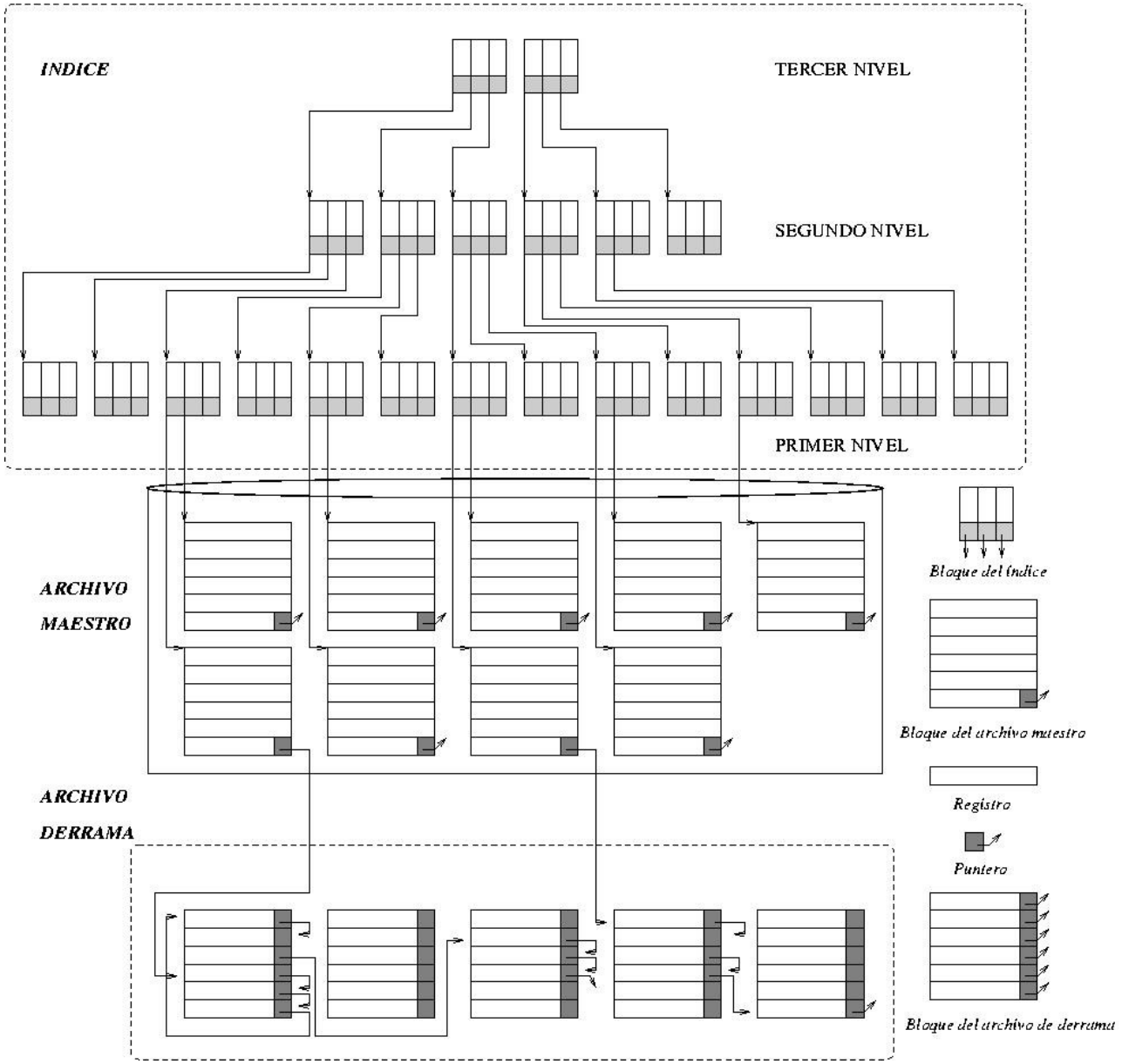
Contenidos extraídos del libro:

Ficheros. Organizaciones clásicas para el almacenamiento del a información

Autores: *Irene Luque Ruiz, Juan Antonio Romero del Castillo y Miguel Ángel Gómez-Nieto*
Editorial: Servicio de Publicaciones de la Universidad de Córdoba, 1998.
ISBN 84-7801-468-3

9.1 Introducción

La org. Secuencial Indexada



Problemas organizaciones anteriores

- **Estructura física**
  - Existencia de la zona de *derrama*
  - Bajo rendimiento de las operaciones de mantenimiento
  - Alto *envejecimiento*
  - Frecuentes reorganizaciones
  - Estructura compleja y rígida
  - Alto coste de mantenimiento
- **Funcionales**
  - Posibilidad de accesos rápidos por otros atributos
  - Acceso rápido ordenado por otros atributos
  - Predicados de búsqueda en los que intervengan varios atributos

Organización indexada

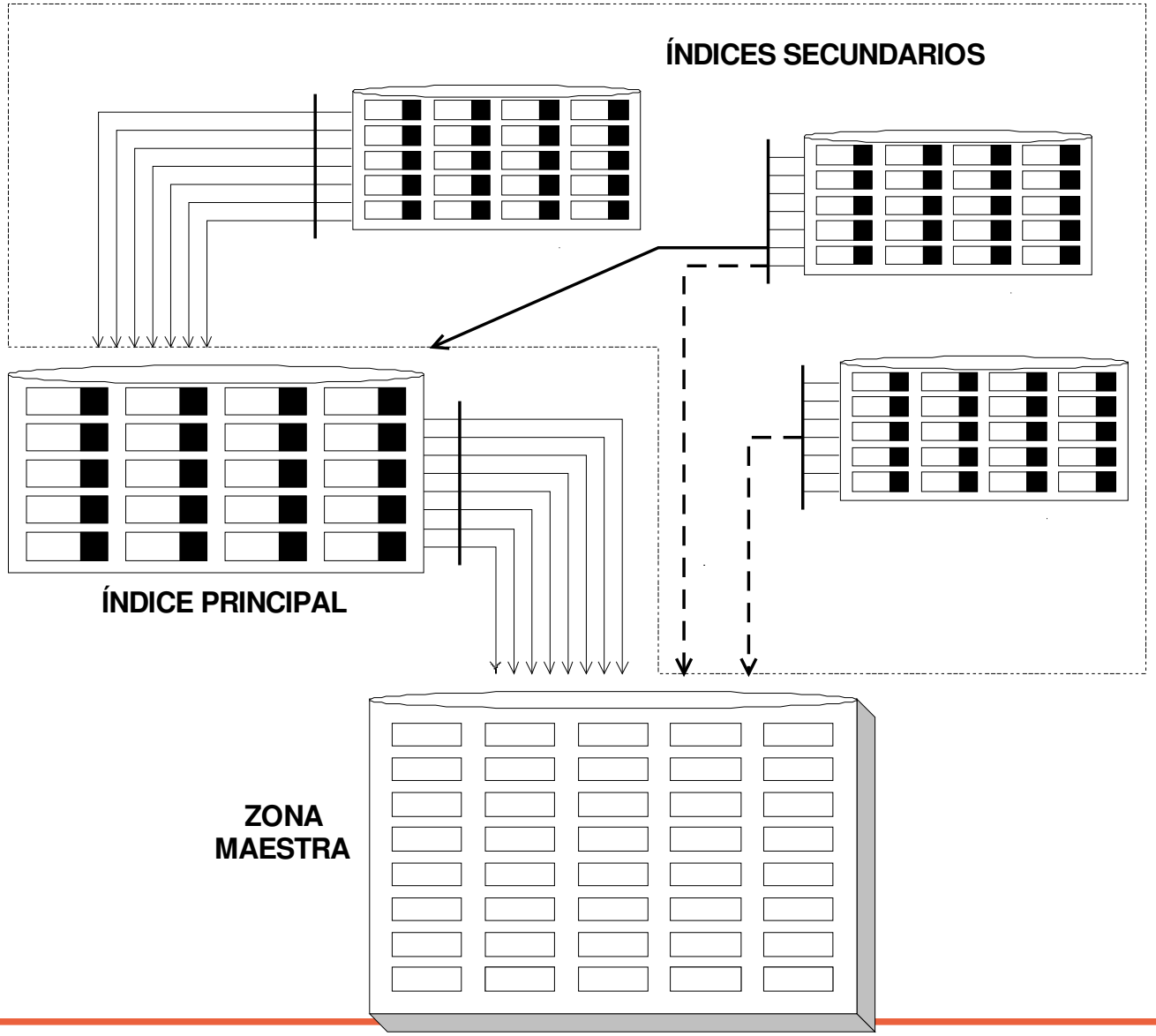
Objetivos:

- Rapidez en accesos exactos, genéricos por uno o varios atributos
- Predicados de búsqueda con varios atributos
- Alta disponibilidad de la información
- Flexibilidad en la estructura que permita actualizaciones dinámicas
- Intentando la sencillez del mantenimiento

Principios de diseño:

- Referencia básica en la organización secuencial indexada:
  - Eliminar zona de derrama eliminando la restricción de orden en la zona maestra
  - El índice debe ser actualizable dinámicamente

9.2 El índice en la organización indexada



Organización indexada: estructura básica

Estructura básica de la organización indexada:

- El índice como vía de cualquier acceso
  - El índice principal
  - Los índices secundarios
- La zona maestra de estructura uniforme
  - Sin ninguna restricción de orden
  - Siguiendo el modelo apilo

EL ÍNDICE

Organización secuencial indexada

el índice no se actualiza ⇨

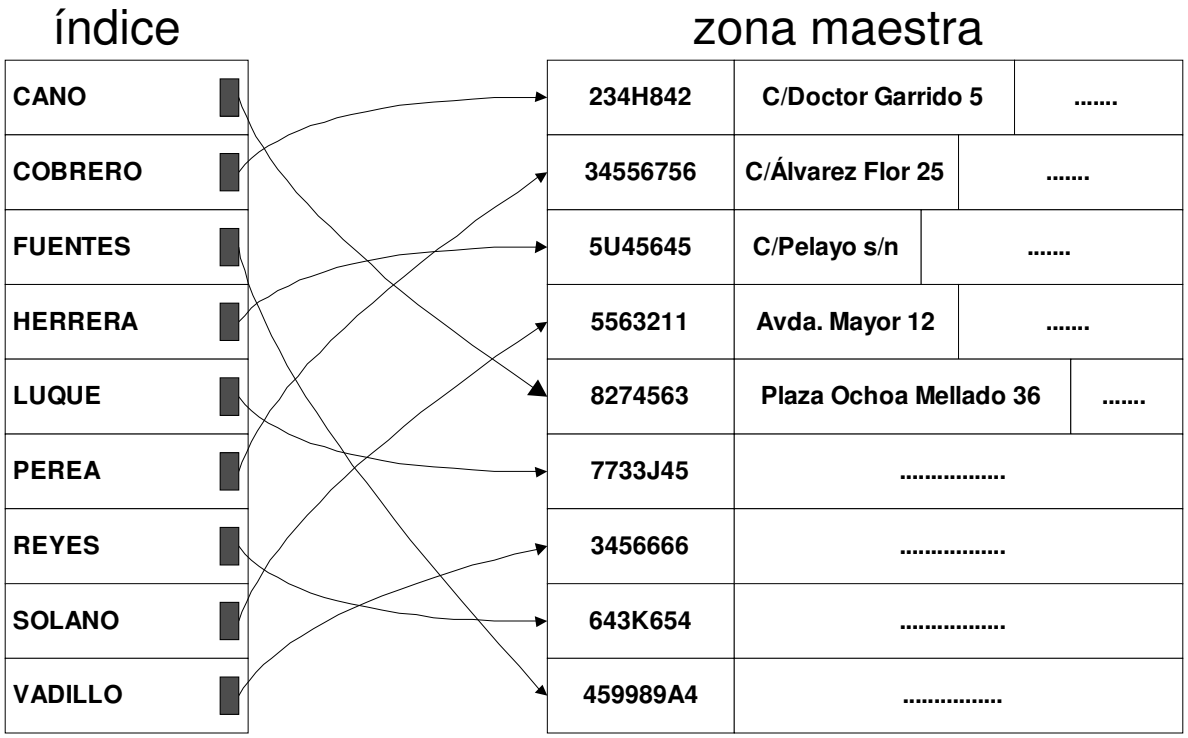
- Deterioro de la estructura: zona de derrama
- Retardos en el mantenimiento
- Necesaria reorganización

Organización indexada

índice actualizado ⇨

- Estructura siempre actualizada
- No hay **RETARDOS** por deterioro: no hay derrama
- **NO** necesaria **reorganización**

9.3 Organización indexada simple



- Se puede optimizar la búsqueda debido a su orden y simpleza
- Inserciones muy simples en zona maestra

Organización indexada simple

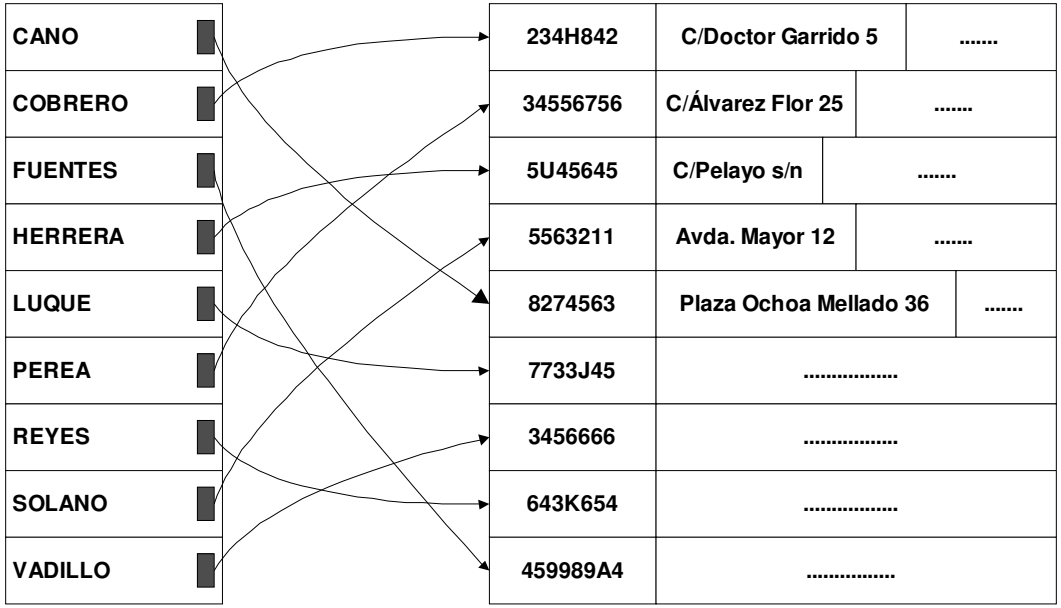
ocupación

$$\Gamma_M = R \times n$$

$$\Gamma_I = R_I \times n$$

acceso

- lectura
- lectura consecutiva
- inserción
- actualización
- borrado
- lectura exhaustiva
- lectura ordenada
- reorganización



$$T_{TABLA}, T_{TB}, T_{APILO}$$

Organización indexada simple

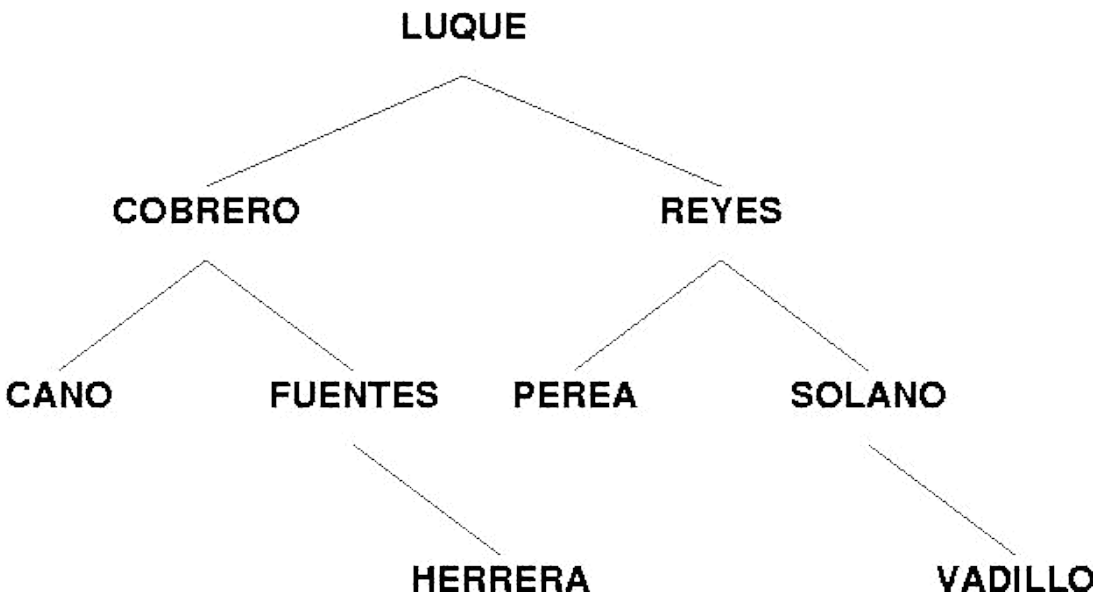
- Si la tabla *Argumento-Puntero* crece, su manejo en memoria se hace prohibitivo y se hace necesario recurrir al disco.

- Aún así presenta utilidad en ciertas aplicaciones:
  - Posibilita búsquedas binarias a pesar de que los registros de la zona maestra sean de longitud variable.
  - El fichero de índices es generalmente de menor tamaño. Mantener y ordenar el índice sigue siendo más rentable que mantener y ordenar el fichero maestro.
  - Pueden mantenerse varios índices sin modificar zona maestra.
  - Indices selectivos.

- Siguiente paso:
  ⇒ **modo eficaz de almacenamiento del índice en el disco**

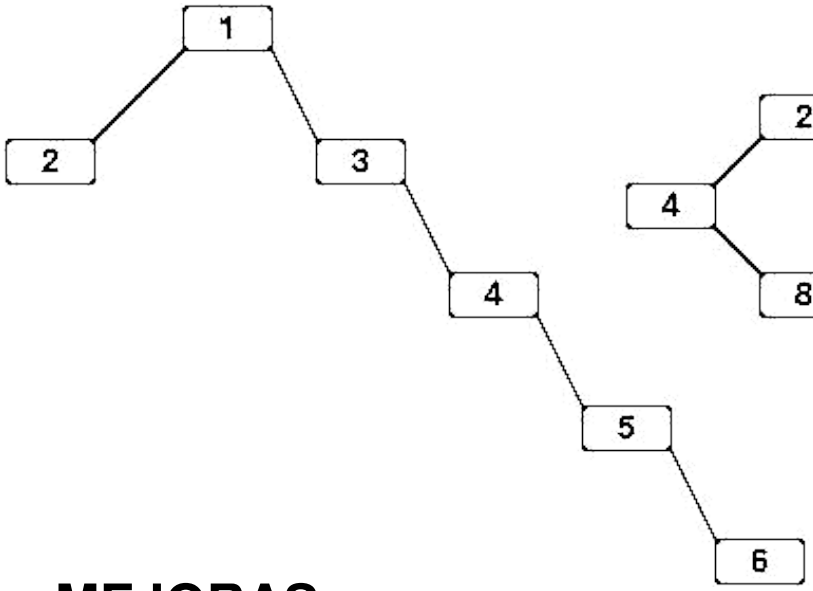
9.4 Índices de mayor tamaño

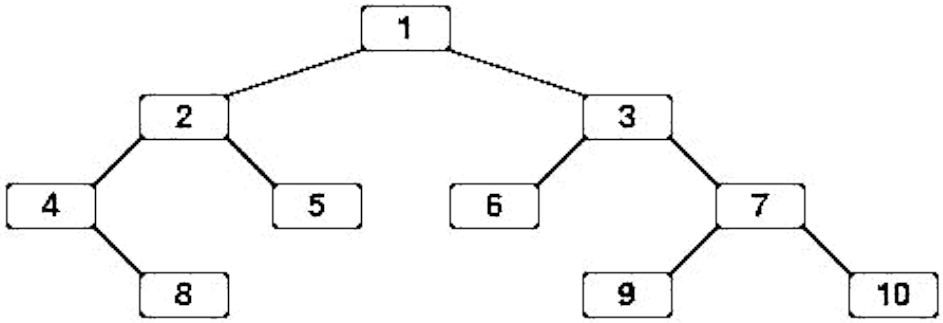
Índices en árbol binario



#reg	clave	h.lzq	h.der
1	COBRERO	3	5
2	REYES	4	7
3	CANO		
4	PEREA		
5	FUENTES		6
6	HERRERA		
7	SOLANO		9
8	LUQUE	1	2
9	VADILLO		

Índices en árbol binario





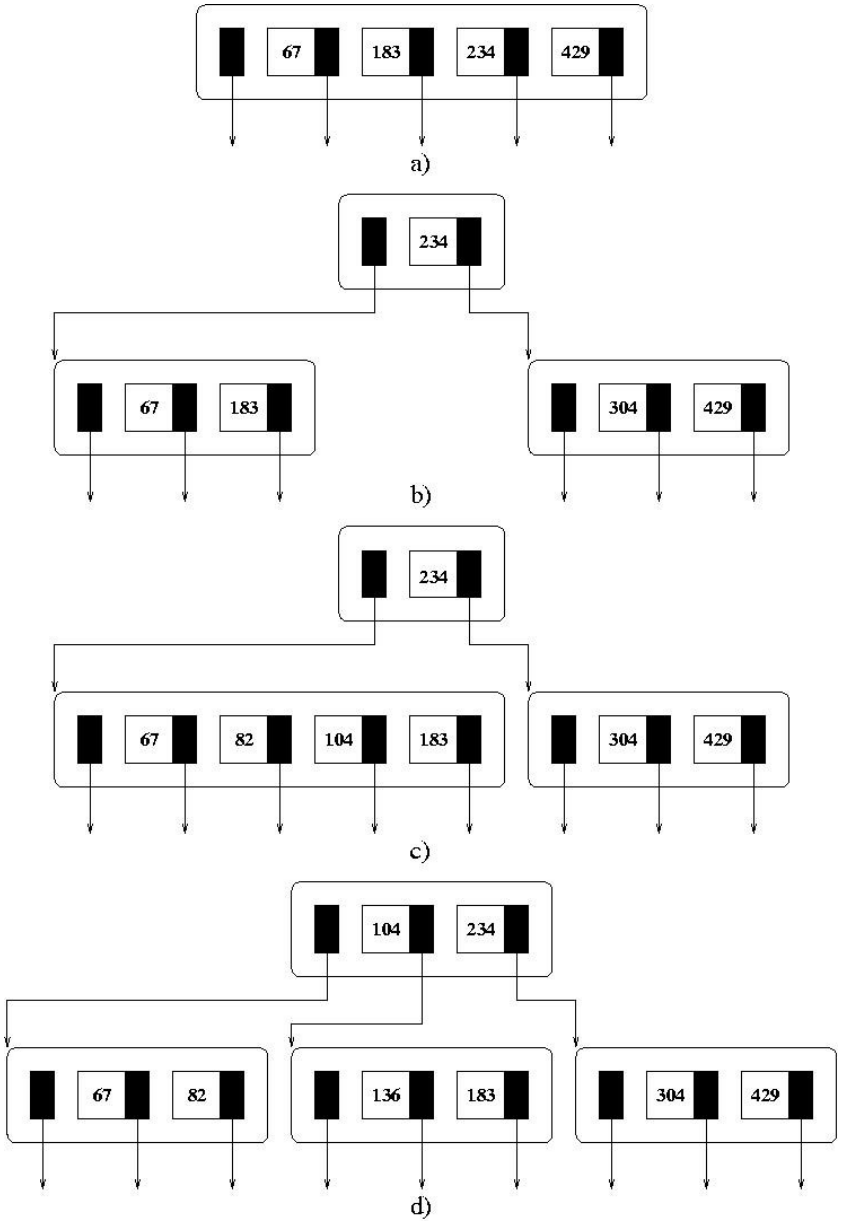
MEJORAS:

- Árboles equilibrados
- Árboles binarios paginados
- Estructuras en árbol de *Bayer* y *McCreight* (**árboles B**)

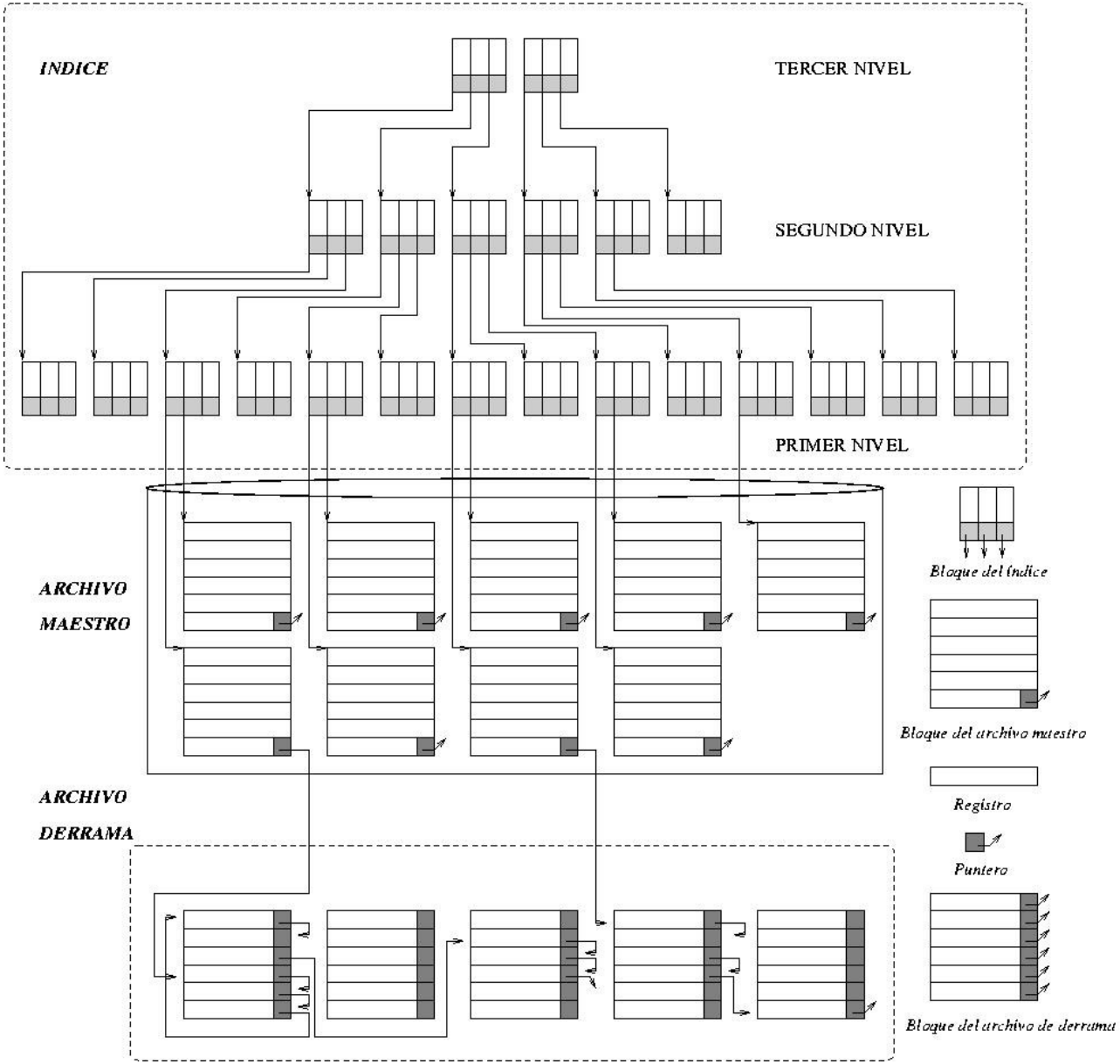
Árbol B:

desarrollo balanceado

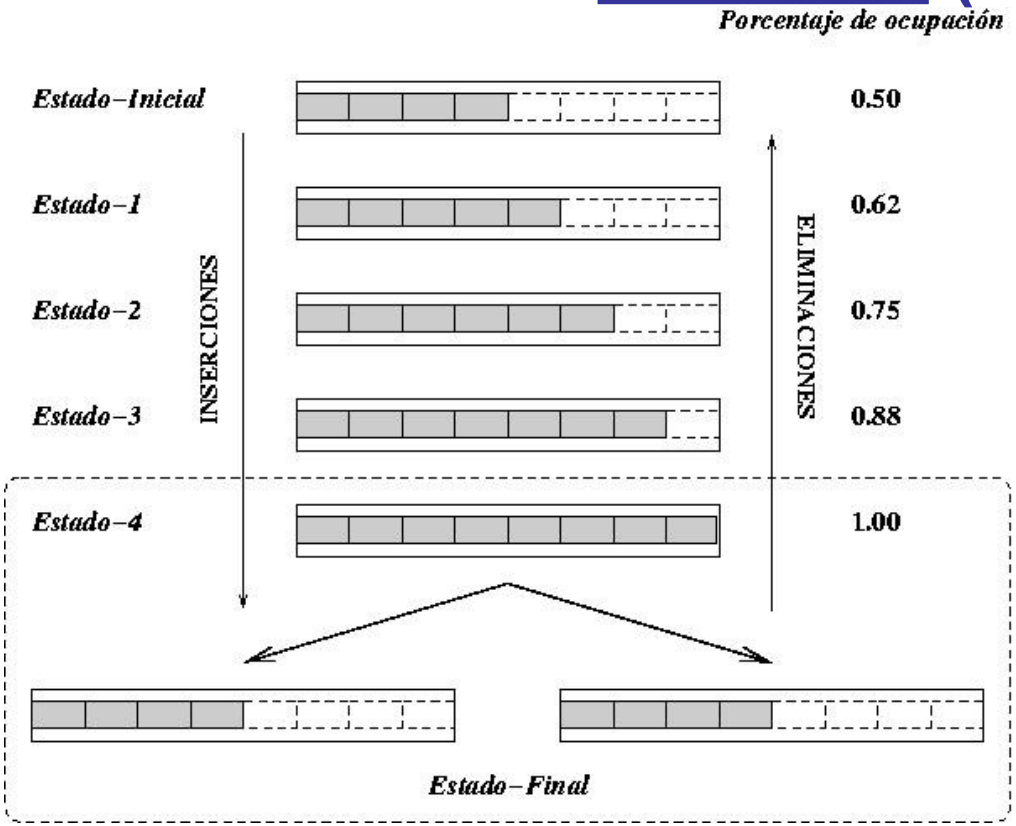
- En inserciones
- Y en el resto de operaciones



El índice denso de la Org. Sec. indexada



9.5 Índice no denso (árboles B)



$$\frac{F_{B_I}}{2} \leq \bar{F}_{B_I} \leq F_{B_I}$$

$$\bar{F}_{B_I} = 0.69\, F_{B_I}$$

- El índice siempre está actualizado. No existen entradas obsoletas.
- Las incorporaciones y eliminaciones son simples de manejar.
- La densidad de los bloques es uniforme.
- El desperdicio de espacio es controlado.

Características del Árbol B

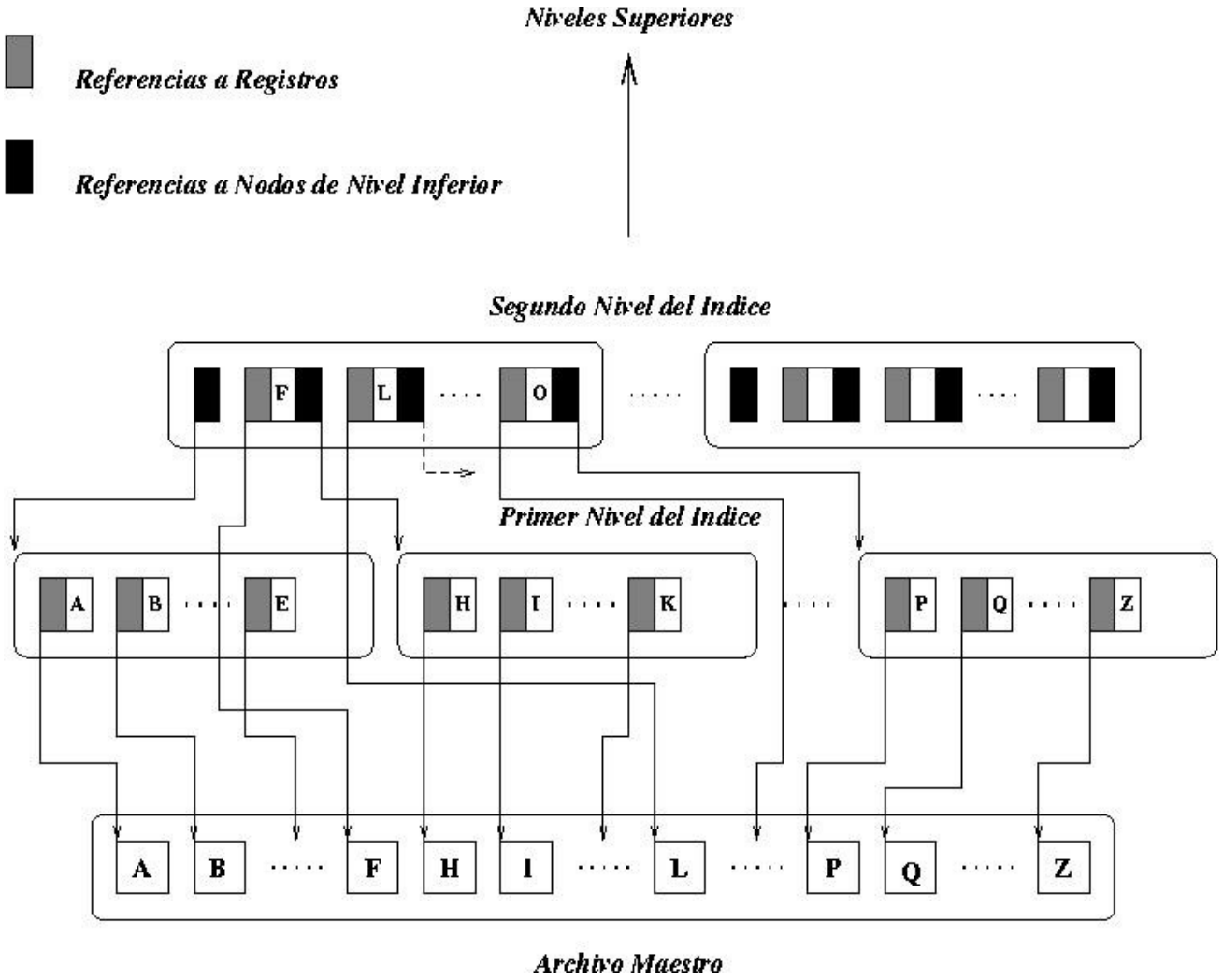
- Árbol balanceado
- Coste de mantenimiento reducido:
  - Simpleza de operaciones de **inserción** y **eliminación**
  - **Localidad** en los casos que requieren reorganización
  - El propio **balanceo** del árbol facilita el mantenimiento.

Características del Árbol B

Reducción de los accesos al disco

- La buena selección de la clave promocionada facilita el **balanceo del árbol** y las posteriores operaciones de búsqueda.
- Mantenimiento del **equilibrio** independiente de la secuencia de inserción.
- La estructura **multivía**.
- **Reparto** uniforme de las claves entre los nodos.

9.6 Organización del índice en Árbol B



9.6 Organización del índice en Árbol B

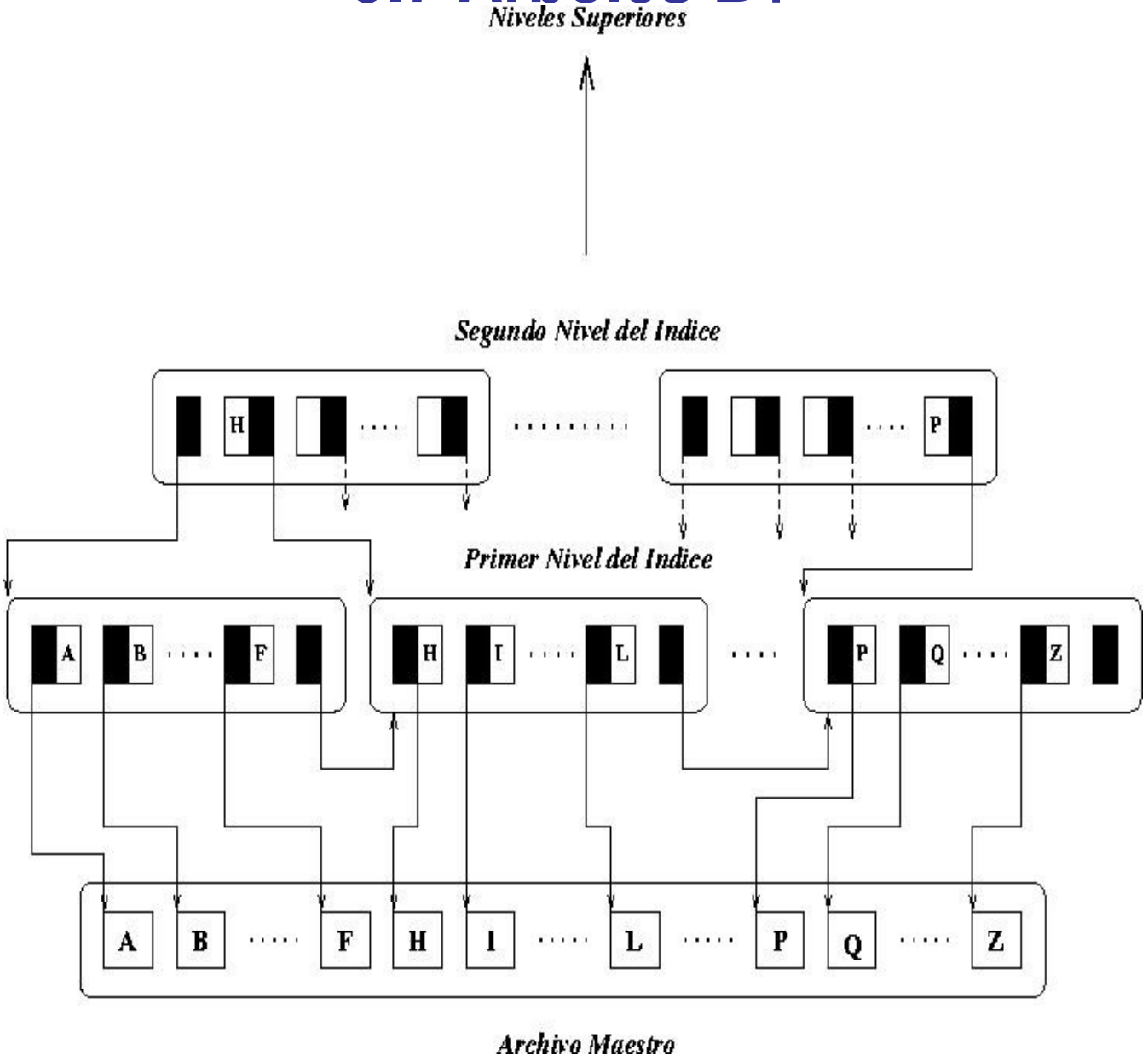
Definición de un bloque del árbol B:

```
Define Estructura Bloque-índice-B
  Puntero a nivel anterior: *ptr;
  Cuerpo-índice-B: array [F_BI/2 ··· F_BI];
Fin Estructura.
```

donde:

```
Define Estructura Cuerpo-índice-B
  Puntero a zona maestra:    *ptr;
  Entrada del índice:        string;
  Puntero al nivel anterior: *ptr;
Fin Estructura.
```

9.7 Árboles B+


Niveles Superiores
Segundo Nivel del Indice
H
P
Primer Nivel del Indice
A
B
F
H
I
L
P
Q
Z
A
B
F
H
I
L
P
Q
Z
Archivo Maestro

9.7 Árboles B+

Definición de un bloque del árbol B+:

```
Define Estructura Bloque-índice-B+
  Puntero a nivel anterior: *ptr;
  Cuerpo-índice-B: array [F_BI / 2 ··· F_BI];
Fin Estructura.
```

donde:

```
Define Estructura Cuerpo-índice-B+
  Puntero a zona maestra:     *ptr;
  Entrada del índice:         string;
  Puntero al nivel anterior: *ptr;
Fin Estructura.
```

Ejemplo: distribución de entradas en el índice B+

zona maestra

$$F_B = \frac{B}{R} = \frac{512}{64} = 8$$

$$b = \frac{n}{F_B} = \frac{100000}{8} = 12500$$

$$\Gamma_M = b \times B = 12.500 \times 512 = 6.4 \ Mbytes$$

B = 512 bytes

P = 6 bytes

V = 10 bytes

R = 64 bytes

n = 100.000

primer nivel

$$F_{B_I} = \lfloor \frac{B}{V+P} \rfloor = \lfloor \frac{512}{10+6} \rfloor = 32 \ entradas \ por \ bloque$$

primer nivel (corrección)

$$F_{B_I} = \lfloor \frac{B-P}{V+P} \rfloor = \lfloor \frac{512-6}{10+6} \rfloor = 31 \ entradas \ por \ bloque$$

promedio

$$\bar{F}_{B_I} = \lfloor ln\,2 \times F_{B_I} \rfloor = \lfloor 0.69 \times 31 \rfloor = 21 \ entradas \ por \ bloque$$

segundo nivel

$$F_{B_I} = 31 \ entradas \ por \ bloque$$

$$\bar{F}_{B_I} = 21 \ entradas \ por \ bloque$$

9.8 Árboles B frente a árboles B+

Árbol B

- Eliminación de redundancia en las entradas del índice.
- No es necesario alcanzar una hoja para acceder al puntero a la zona maestra.
- Espacio requerido por cada bloque es mayor.
- Mantenimiento más costoso.
- Recorrido secuencial supone recorrer el árbol por completo.

Árbol B+

- Recorrido secuencial ordenado más eficiente.
- Operaciones de mantenimiento más simples y eficientes.
- Es necesario alcanzar una hoja para acceder al puntero de la zona maestra.
- Duplicación de las entradas del índice.

Árbol B+ frente a org. anteriores

Organizaciones anteriores:

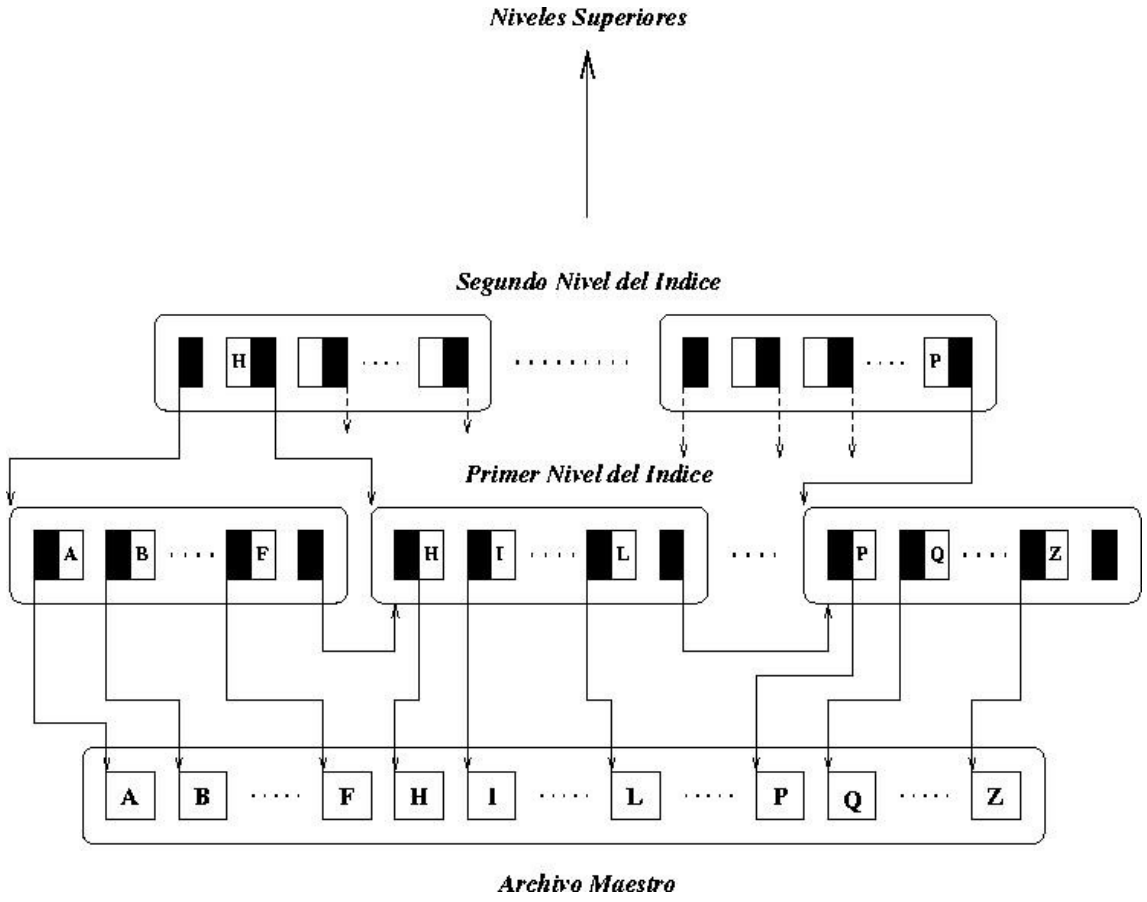
- Recorrido ordenado de la zona maestra es muy costoso con un **árbol B**.
- La organización **secuencial indexada** es poco flexible y no permite varios índices.
- La **organización secuencial** no permite accesos eficientes por una clave.

Árbol B+

- Mantenimiento simple y eficiente (árbol B).
- Acceso eficiente a un registro mediante una o varias claves (árbol B).
- Procesamiento eficiente en secuencia ordenada por una o varias claves de los registros

Árbol B+ frente a org. anteriores

*Las características comunes con el árbol B y el acceso eficiente ordenado, hacen del **árbol B+** la estructura más utilizada en las organizaciones indexadas*



9.9 Estimación del índice no denso

Árbol B

B = 2048 bytes

P = 6 bytes

V = 26 bytes

R = 64 bytes

n = 1.600.000

razón de salida máxima

$$F_{B_I} = \lfloor \frac{B-P}{V+P+P} \rfloor = \lfloor \frac{2048-6}{26+6+6} \rfloor = 53 \ entradas \ por \ bloque$$

razón de salida promedio

$$\bar{F}_{B_I} = 0.69 \times F_{B_I} = 0.69 \times 53 = 36 \ entradas \ por \ bloque$$

altura del índice

$$h = log_{\bar{F}_{B_I}+1} n = log_{37} 1.600.000 = 4$$

9.9 Estimación del índice no denso

Árbol B+

B = 2048 bytes

P = 6 bytes

V = 26 bytes

R = 64 bytes

n = 1.600.000

razón de salida máxima

$$F_{B_I} = \lfloor \frac{B-P}{V+P} \rfloor = \lfloor \frac{2048-6}{26+6} \rfloor = 63 \ entradas \ por \ bloque$$

razón de salida promedio

$$\bar{F}_{B_I} = 0.69 \times F_{B_I} = 0.69 \times 63 = 43 \ entradas \ por \ bloque$$

altura del índice

$$h = log_{\bar{F}_{B_I}+1} n = log_{44} 1.600.000 = 4$$

9.10 Ocupación de la organización indexada

Árbol B

capacidad promedio

Ocupación del índice en árbol B			
Nivel	Bloques	Entradas	Apuntadores
4	1	36	37
3	37	1332	1369
2	1369	49284	50653
1	50653	1823508	1874161
		(1549348)	

capacidad máxima

Ocupación del índice en árbol B			
Nivel	Bloques	Entradas	Apuntadores
4	1	53	54
3	54	2862	2916
2	2916	154548	157464
1	157464	8345592	8503056
		(1442537)	

Ocupación de la organización en árbol B+

$$E_{I_1} = n$$

primer nivel:

$$b_{I_1} = \frac{E_{I_1}}{\bar{F}_{B_I}}$$

$$\Gamma_{I_1} = b_{I_1} \times B$$

---

segundo nivel:

$$b_{I_2} = \frac{b_{I_1}}{\bar{F}_{B_I} + 1}$$

$$\Gamma_{I_2} = b_{I_2} \times B$$

último nivel:

$$b_{I_h} = \frac{b_{I_{h-1}}}{\bar{F}_{B_I} + 1} = 1$$

$$\Gamma_{I_h} = b_{I_h} \times B$$

Ocupación de la organización en árbol B+

$$\Gamma_I = \sum_{i=1}^{h} \Gamma_i =$$

ocupación total:

$$= \left(b_{I_1} + b_{I_2} + \cdots + b_{I_{h-1}} + b_{I_h} \right) \times B =$$

$$= \left(1 + b_{I_{h-1}} + \cdots + b_{I_2} + b_{I_1} \right) \times B =$$

$$= B + \Gamma_{I_{h-1}} + \cdots + \Gamma_{I_2} + \Gamma_{I_1}$$

Ocupación de la organización en árbol B+

capacidad promedio

Ocupación del índice en árbol B			
Nivel	Bloques	Entradas	Apuntadores
4	1	43	44
3	44	1892	1936
2	1936	83248	85184
1	85184	3662912	3748096

capacidad máxima

Ocupación del índice en árbol B			
Nivel	Bloques	Entradas	Apuntadores
4	1	63	64
3	64	4032	4096
2	4096	258048	262144
1	262144	16515072	16777216

ajuste de la capacidad

Ocupación del índice en árbol B			
Nivel	Bloques	Entradas	Apuntadores
1	37210	1600000	1600000(<1637240) + 37209 (nivel 1)
2	846	37210	37210 (< 37224)
3	20	846	846(< 880)
4	1	20	20(< 44)

9.11 Acceso a los ficheros indexados

1) Lectura $T_L = h(t_l + t_r + T_{tB}) + t_l + t_t + T_{tB} = (h+1)(t_l + t_r + T_{tB})$

```
Algoritmo BUSQUEDA_ARBOLB+(raiz,clave;;nodo,posicion)
Inicio
    nodo=null
    posicion=-1
    p=raiz
    mientras (p != null)
        leer_bloque(p,bloque)
        i=busca_clave(bloque,clave)
        si((i < n) y (clave = bloque.k(i))) entonces
                nodo=p
                posicion=i
        finsi
        p=bloque.hijo(i+1)
    finmientras
Fin
```

n: número de claves del bloque

Acceso a los ficheros indexados

2) Lectura Consecutiva

Siguiente entrada en el mismo bloque:

$$T_{LC} = t_l + t_r + T_{tB}$$

Siguiente entrada en siguiente bloque con:

$$T_{LC} = 2\,(t_l + t_r + T_{tB})$$

puede aproximarse al tiempo en acceder a la zona maestra debido al conjunto secuencial

Acceso a los ficheros indexados

3) Inserción

a) Inserción del registro en la zona maestra: $T_{Im\,aestra}$

b) Para cada uno de los a´ índices afectados: T_{indice}

c) Tiempo empleado en las divisiones: $T_{division}$

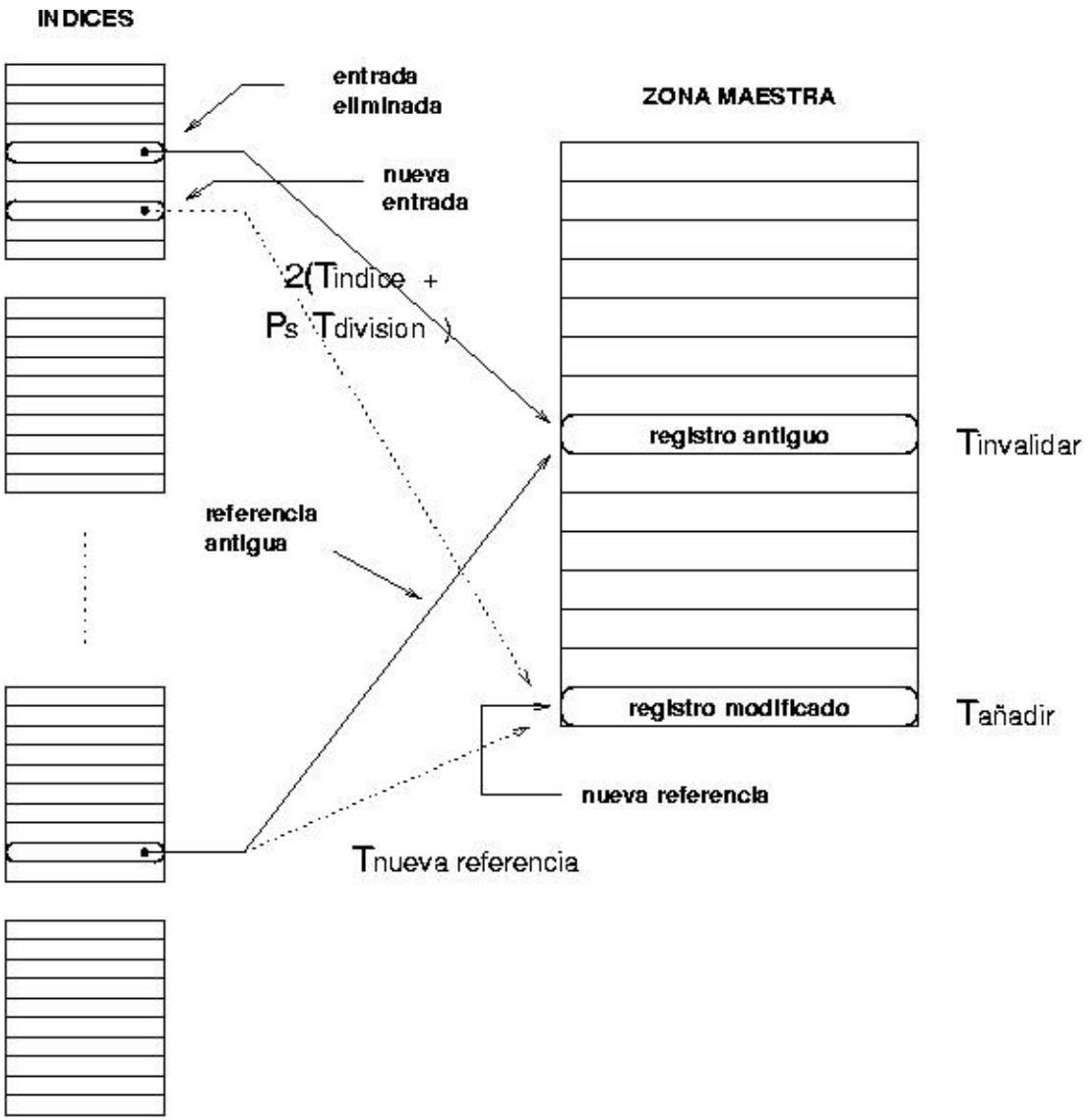
$$P_s = \frac{1}{F_{B_I}/2}$$

Total:

$$T_I = T_{Im\,aestra} + a'(T_{indice} + P_s T_{division}) =$$
$$= (t_l + t_r + T_{tB} + T_{RE}) +$$
$$+ a'\left(T_L + T_{RE} + 2\frac{1}{F_{B_I}/2}(t_l + t_r + T_{tB} + T_{RE})\right)$$

Acceso a los ficheros indexados

4) Actualización



Acceso a los ficheros indexados

4) Actualización

a) Borrado del registro en la zona maestra: $T_{invalidar}$

b) Adición del nuevo registro a la zona maestra: $T_{añadir}$

c) Actualización del índice afectado: T_{indice}

d) División y consolidación de bloques de índice: $T_{division}$, P_s

e) Modificación de las referencias en otros índices: $T_{nueva\ referencia}$

Total:

$$T_A = a_{actualizacion}\, 2\,(T_{indice} + P_s\, T_{division}) + T_{invalidar} + \\ + T_{añadir} + (a' - a_{actualizacion})\, T_{nueva\ referencia}$$

Acceso a los ficheros indexados

5) Lectura Ordenada

Mediante índice exhaustivo:

$$T_{LT} = T_L + (n-1)T_{LC}$$

Zona maestra ordenada en bloques de $\bar{F}_B$

$$T_{LT} = \frac{n}{\bar{F}_B}(t_l + t_r + T_{tB})$$

Zona maestra ordenada en bloques de F_B

$$T_{LT} = \frac{n}{F_B}(t_l + t_r + T_{tB})$$

Acceso a los ficheros indexados

6) La Reorganización puede necesitar una o varias de las siguientes operaciones:

a) Reorganizar la zona maestra por haber reg. marcados para borrado

b) Reorganizar algún índice corrupto o dañado o por tener marcas para borrado

Reorganizar la zona maestra

1. leer la zona maestra
2. ordenarla
3. construir el nuevo índice

$$T_{RO_M} = T_{LT}(r) + T_{clasificacion}(r') + \frac{\Gamma_I(r')}{t'}$$

Acceso a los ficheros indexados

para cada índice a reconstruir:

1. Lectura del fichero maestro

2. Creación del nuevo índice ordenado por el atributo correspondiente

$$T_{RO_i} = T_{LT} + \frac{\Gamma_I(r')}{t'}$$

$$T_{RO} = T_{RO_M} + a\ T_{RO_i}$$

a: número de índices secundarios a reconstruir

Resumen

- La organización indexada es la más frecuente y más útil.
- La idea fundamental: el índice.
- Estructuras para el índice:
  - La estructura de índice simple en tabla es simple y eficiente (problemas pequeños).
  - Las estructuras en árbol mejoran las búsquedas (*desbalanceo*).
  - El árbol B: estructura equilibrada a bajo coste computacional.
  - El árbol B+: mejora el acceso secuencial a los registros.
- Se incluyen en la mayoría de los sistemas por su:
  - Eficiencia en el acceso a registros por una o varias claves.
  - Bajo coste computacional.

Fin