Unidad 1: Planificación del Almacenamiento e Indexación

Bases de Datos Avanzadas, Sesión 1:

Planificación del Almacenamiento de Registros

Iván González Diego
Dept. Ciencias de la Computación
Universidad de Alcalá

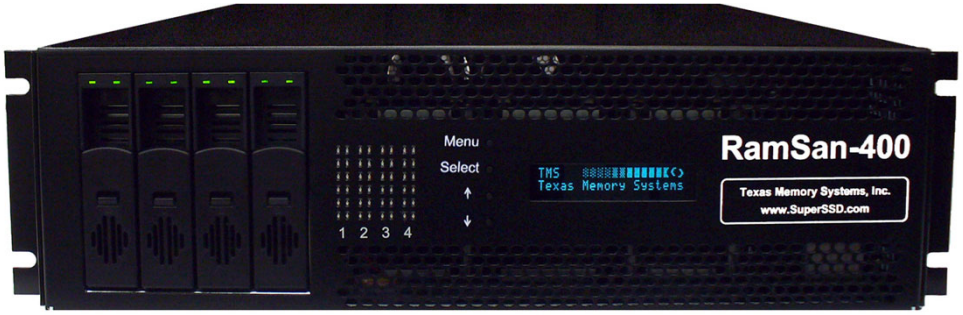
INDICE

- *Organización de los archivos.*
- *Estimación del tamaño de un archivo*
- *Secuenciales o Cluster Index*
- *Arboles B^+*
- *Hash*
- *Agrupaciones*
- *Estimación de número de registros y coste*
- *Ejemplo*

Referencias: Silberschatz 4ª Ed. Pp 249 - 315
Elmasri, 3ª Ed. Pp 105 - 181

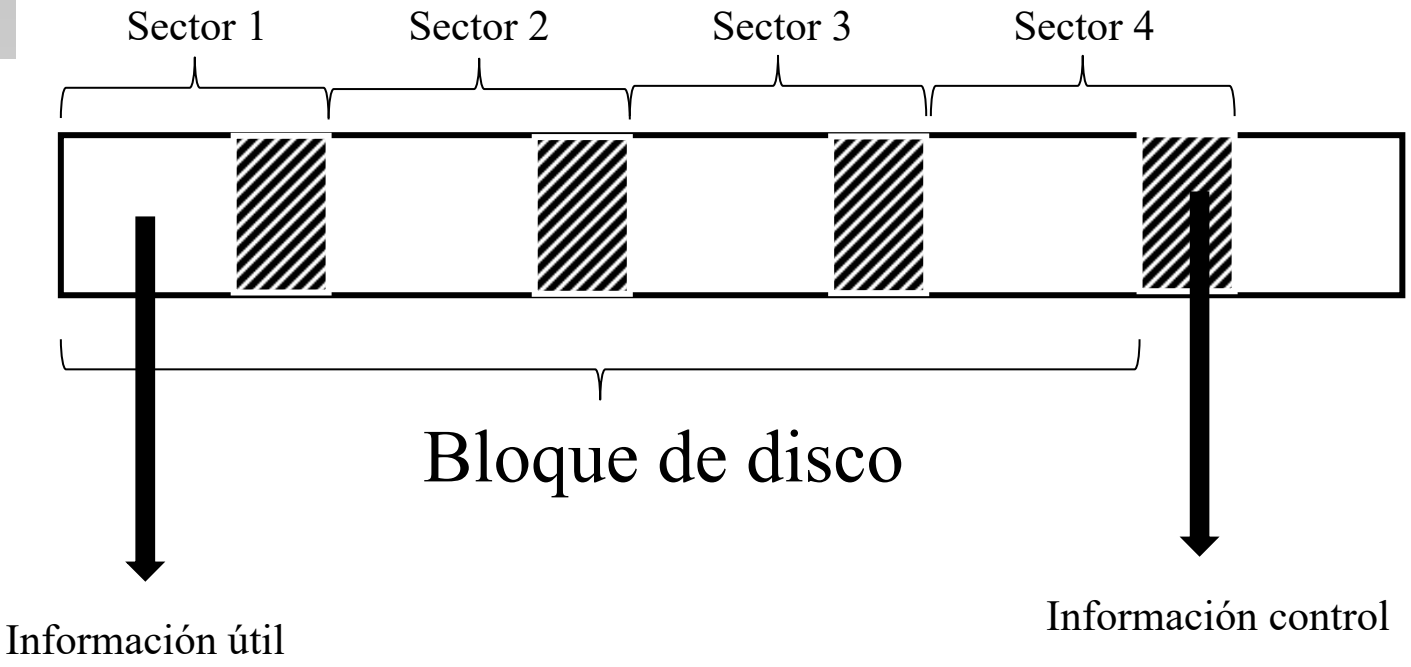
Organización de los archivos

- *Vista lógico ⇒ archivos son secuencias de registros que se deberían de corresponder con bloques de disco.*



Acceso a los datos: bloques de disco

- *Operación I/O ⇒ dirección disco ⇒ numero bloque*
- *Bloque ⇒ secuencia continua de sectores de una pista*
- *Datos se transfieren en bloques*
- *Ejemplo: SO 4 sectores de disco / bloque*



Organización de los archivos

- *Regla ⇒ Un registro en un Bloque de Datos ⇒ un acceso disco*
- *Rara vez un registro va a ocupar exactamente un bloque de disco*
- *Registros ⇒ Tamaño fijo o variable (cadenas texto, arrays, etc).*

```
Type deposito = record
        nombre_sucursal: char(22);
        numero_cuenta: char(10);
        saldo: real;
end
```

```
type Lista_cuentas = record
    nombre_sucursal: char (22);
    información_cuenta: array [1..∞] of record
          numero_cuenta: char(10);
          saldo: real;
       end
end
```

Organización de los archivos

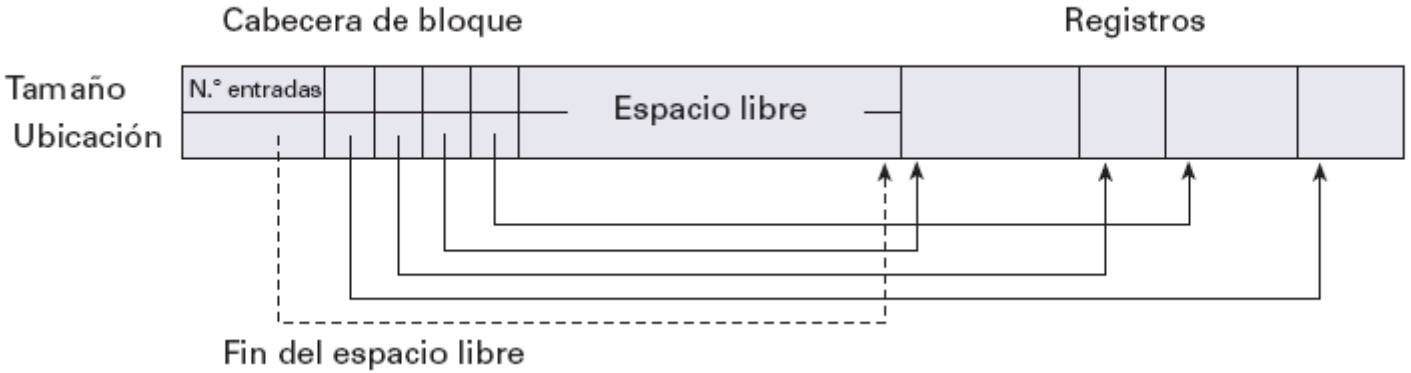
UA

- *Estructura fija*

registro 0	C-102	Navacerrada	400
registro 1	C-305	Collado Mediano	350
registro 2	C-215	Becerril	700
registro 3	C-101	Centro	500
registro 4	C-222	Moralzarzal	700
registro 5	C-201	Navacerrada	900
registro 6	C-217	Galapagar	750
registro 7	C-110	Centro	600
registro 8	C-218	Navacerrada	700

0	Navacerrada	C-102	400	C-201	900	C-218	700
1	Collado Mediano	C-305	350	⊥	⊥	⊥	⊥
2	Becerril	C-215	700	⊥	⊥	⊥	⊥
3	Centro	C-101	500	C-110	600	⊥	⊥
4	Moralzarzal	C-222	700	⊥	⊥	⊥	⊥
5	Galapagar	C-217	750	⊥	⊥	⊥	⊥

- *Estructura página de ranuras*



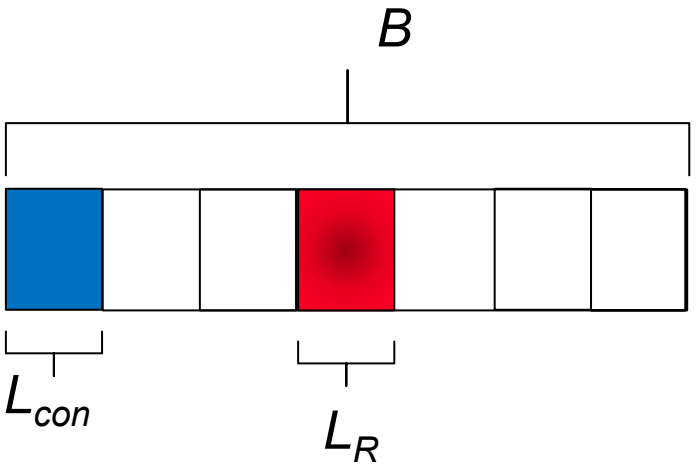
Estimación tamaño de un archivo

- *Secuencia lógica de bloques que se almacenan en disco.*
- *B → tamaño del bloque (bytes)*
- *n_R → número de registros del archivo*
- *$L_R = \Sigma L_{Ci}$ → longitud del registro (suma de bytes de cada campo C_i)*
- *f_R → factor de bloque (número de registros enteros que caben en 1 bloque)*

$$f_R = \lfloor B/L_R \rfloor$$

- *b_R → Número de bloques del archivo*

$$b_R = \lceil n_R / f_R \rceil$$



- *Tamaño en disco (bytes) → $b_R * B$*
- *Puede haber información de control*

$$f_R = \lfloor (B-L_{con})/L_R \rfloor$$

Organización de los registros en los archivos

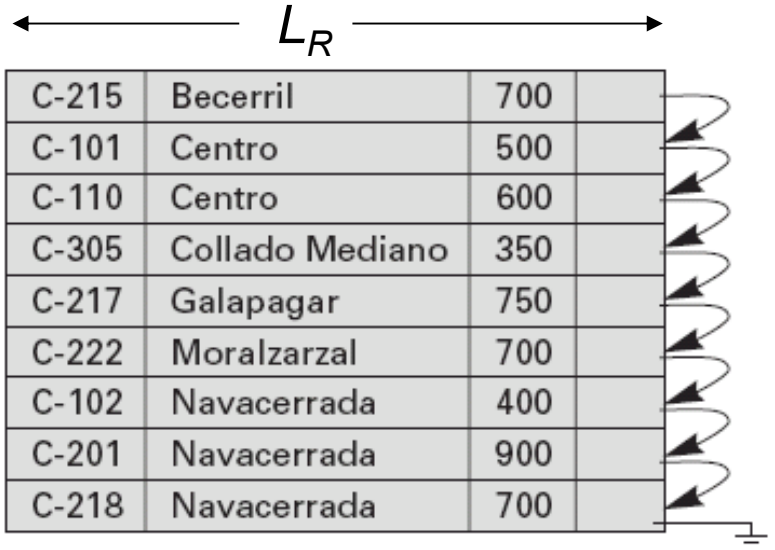
- *Varios tipos:*
  - *Montículo: No hay ordenación. Se colocan los registros donde hay espacio libre.*
  - *Archivos secuenciales: Se colocan ordenados a partir de una clave de búsqueda*
  - *Organización por árboles B^+*
  - *Organización asociativa hash:*
    - *Función hash que relaciona algún atributo de cada registro.*
    - *Dependiendo del resultado, se sitúa el registro en un bloque.*
  - *Agrupaciones:*
    - *Múltiples relaciones (tablas) en el mismo archivo.*
    - *Registros de distintas relaciones se guardan en el mismo bloque.*

Organización de los registros en los archivos: Archivos secuenciales

- *Están diseñados para maximizar la capacidad de procesamiento relacionada con un atributo determinado de los registros, que suele ser la clave de búsqueda*
- *Los registros se vinculan con punteros.*
- *Permite la lectura de registros en orden ⇒ útil en ciertos algoritmos → Búsqueda binaria*
- *Difícil mantener el orden físico cuando se insertan o borran registros.*
- *Ocupa:*

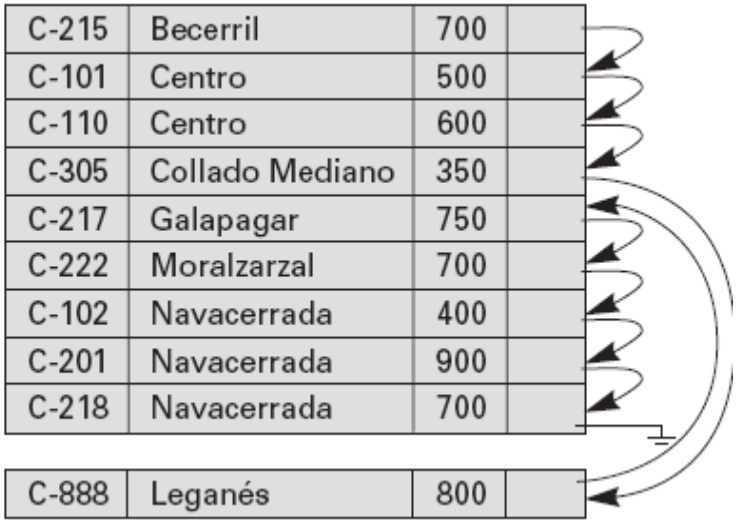
$$f_R = \lfloor B/L_R \rfloor$$

$$b_R = \lceil n_R / f_R \rceil$$



Organización de los registros en los archivos: Archivos secuenciales

- *Inserción:*
  - *Localizar el registro que precede al que se va a insertar según el orden de la clave.*
  - *Si existe un hueco libre **en el bloque** de ese registro ⇒ se insertará ahí el nuevo registro*
  - *Si no hay hueco ⇒ se inserta el registro en un **bloque de desbordamiento***
  - *Actualización de los punteros necesarios:*
    - *Puntero del nuevo registro.*
    - *Puntero del registro anterior.*

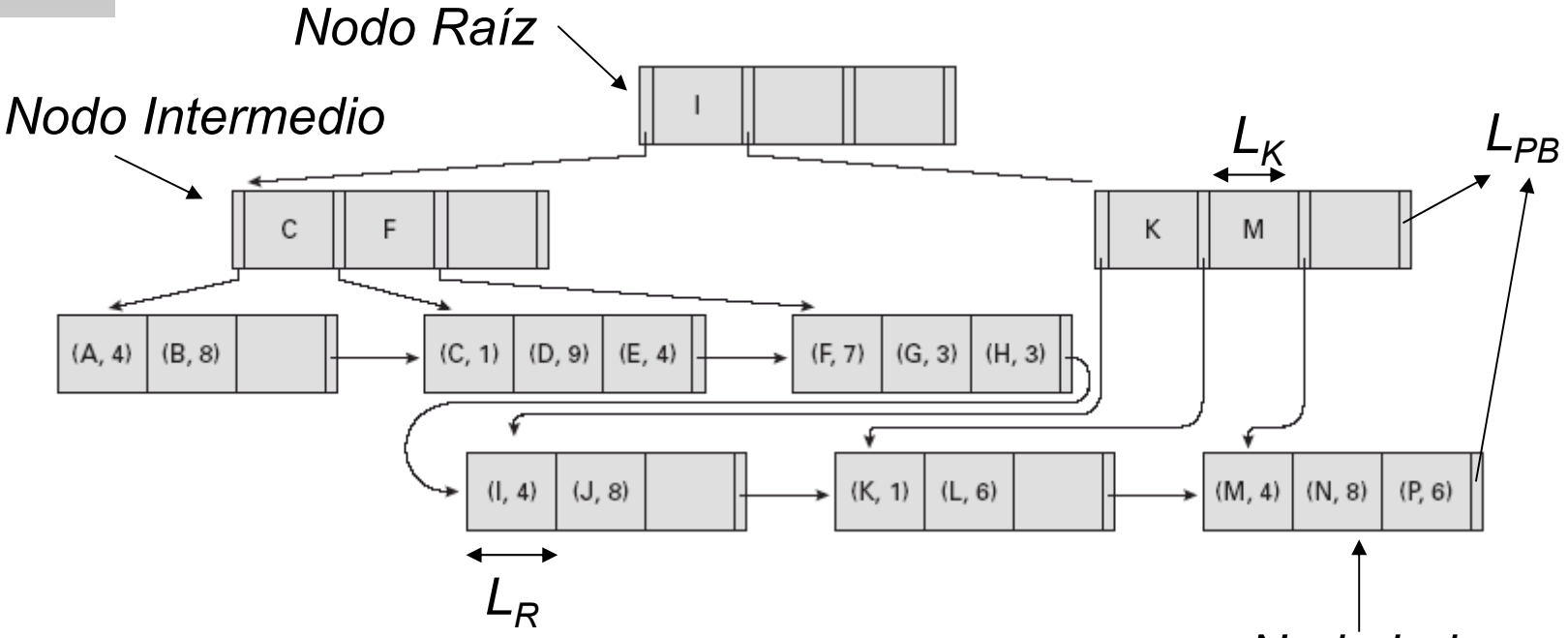


C-215	Becerril	700	
C-101	Centro	500	
C-110	Centro	600	
C-305	Collado Mediano	350	
C-217	Galapagar	750	
C-222	Moralzarzal	700	
C-102	Navacerrada	400	
C-201	Navacerrada	900	
C-218	Navacerrada	700	
C-888	Leganés	800	

- *Problema:*
  - *Puede perderse el orden por completo ⇒ Reordenar el fichero.*

Organización de Archivos con Árbol B^+

- También es posible utilizar árboles B^+ para organizar los registros de los archivos.
- Parecido al árbol binario pero con más valores en cada nodo.
- Los nodos hojas almacenan registros. (1 nodo = 1 bloque)



L_{PB} → Longitud Puntero Bloque

Organización de Archivos con Árbol B^+

Nodo Raíz / Intermedio: $n*L_{PB}+(n-1)*L_K \leq B$

Nodo Hoja: $n_h*L_R + L_{PB} \leq B$

- *Número de boques del archivo:*
  - *Número de nodos hoja:* $N_{hoja} = \lceil n_R / n_h \rceil$
  - Número bloques intermedio 1: $N_{Int1} = \lceil N_{hoja} / n \rceil$
  - Número bloques intermedio 2: $N_{Int2} = \lceil N_{Int1} / n \rceil$
  - Número bloques intermedio 3: $N_{Int3} = \lceil N_{Int2} / n \rceil$
  - ……
  - *Número de bloques raíz: 1*

Asociación Estática

- *Es una técnica basada en la utilización de funciones de asociación.*
- *Sea K el conjunto de los valores de clave de búsqueda.*
- *Se B el conjunto de todas las direcciones de los cajones.*
- *Una función h es una función de K a B tal que:*

$$h(k)=B$$

- *Número de cajones N_c → Número de cajones: Número de valores diferentes de la función de asociación.*
- *Ejemplo: h(x)= x mod 10 → 0,1,2,3,4,5,6,7,8,9 → 10 cajones*

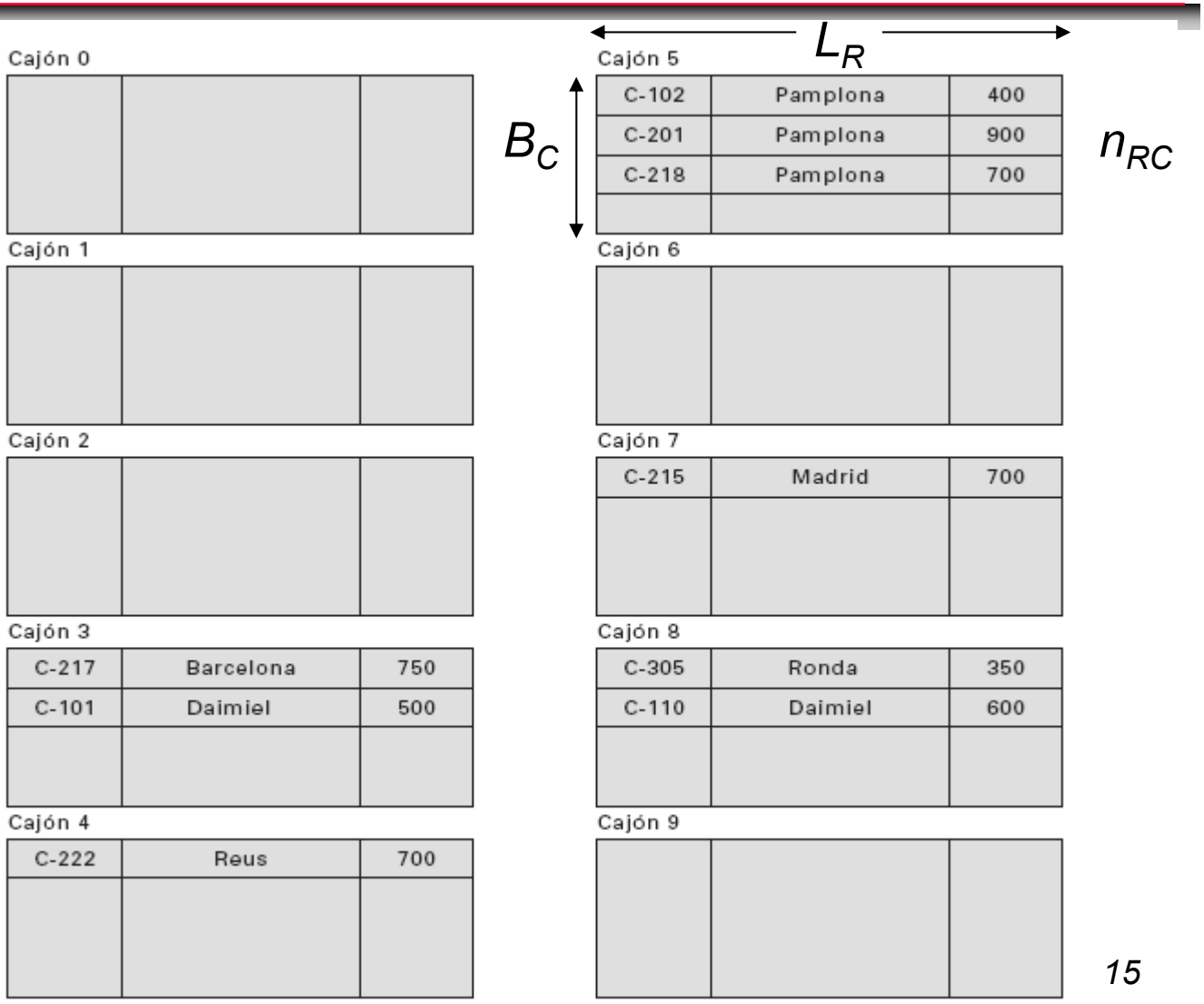
  h(x)= $\lfloor x/100 \rfloor$, sabiendo que min(x)=0, max(x)=399

  0,1,2,3 → 4 cajones

Funciones de Asociación

- *Distribución Uniforme:*
  - *Cada cajón tiene asignado el mismo número de valores de la clave de búsqueda dentro del conjunto de todos los valores posibles.*
- *Distribución Aleatoria:*
  - *En el caso medio, cada cajón tendrá casi el mismo número de valores asignados a él, sin tener en cuenta la distribución actual de los valores de la clave de búsqueda.*
  - *El valor de asociación no será correlativo a ninguna orden exterior visible en los valores de la clave de búsqueda.*

Organización de archivos por asociación



Si equiprobalidad:

$n_{RC} = n_R/N_c$

n_{RC} = nº registros /cajón

$f_{rc} = \lfloor B/L_R \rfloor$

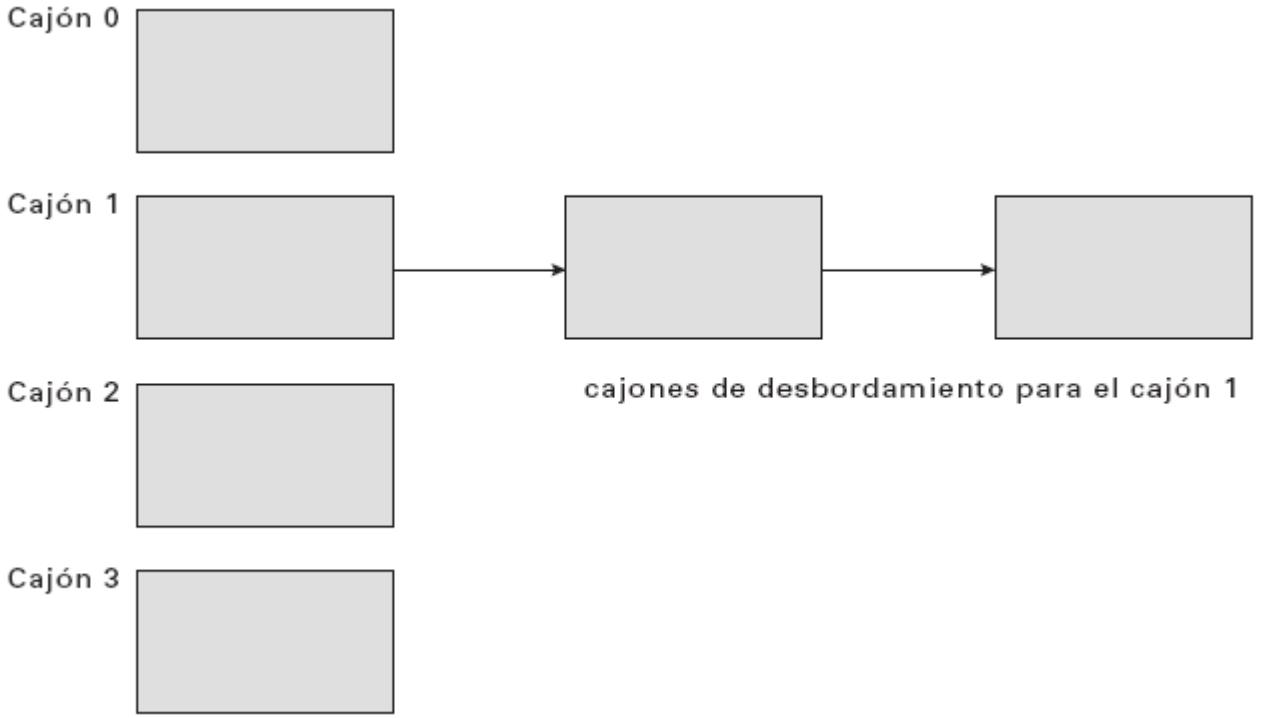
$B_c = \lceil n_{RC} / fr \rceil$

Número bloques: $N_C * B_c$

Gestión del desbordamiento de cajones

- *Cajones insuficientes*
- *El número de cajones debe ser escogido tal que $n_c > n_r / f_r$*
- *Atasco*
  - *Algunos cajones tienen asignados más registros que otros.*
  - *Puede ocurrir:*
    - *Varios registros podrían tener la misma clave de búsqueda.*
    - *La función de asociación elegida podría producir una distribución irregular de las claves de búsqueda.*
- *En general $n_c=(n_r / n_f)*(1+d)$ donde d suele ser un factor de 0.2 aproximadamente.*
- *Cajones de desbordamiento + lista enlazada*

Gestión del desbordamiento de cajones



Organización de los registros en los archivos: Agrupaciones

- *Utilizado para bases de datos de tamaño relativamente grande en el que se utiliza un único archivo.*
- *La gestión del archivo y su estructura pasa del S.O. al SGBD.*
- *Se optimiza su estructura para la ejecución de consultas.*
- *Su estructura está diseñada para poder mezclar los datos de distintas relaciones dentro del mismo bloque de disco ⇒ puede recuperar datos de varias relaciones en una lectura ⇒ mejora la ejecución de ciertas consultas.*
- *Sistema inteligente de bases de datos ⇒ capaz de deducir posibles mezclas de datos de forma automática.*
- *Mezcla de tuplas de distintas relaciones ⇒ punteros de situación ⇒ permitiendo acceder a las tuplas de las relaciones de forma independiente cuando es necesario.*

Organización de los registros en los archivos: Agrupaciones

nombre-cliente	*número-cuenta*
López	C-102
López	C-220
López	C-503
Abril	C-305

nombre-cliente	*calle-cliente*	*ciudad-cliente*
López	Principal	Arganzuela
Abril	Preciados	Valsaín

select *número-cuenta*, *nombre-cliente*, *calle-cliente*, *ciudad-cliente*
from *impositor*, *cliente*
where *impositor.nombre-cliente* = *cliente.nombre-cliente*

select *
from *cliente*

López	Mayor	Arganzuela
López	C-102	
López	C-220	
López	C-503	
Abril	Preciados	Valsaín
Abril	C-305	

López	Mayor	Arganzuela	
López	C-102		
López	C-220		
López	C-503		
Abril	Preciados	Valsaín	
Abril	C-305		

Estimación registros/bloques a recuperar del archivo de datos

- *V(A) → Número de valores diferentes campo A en archivo / tabla R*
- *Número registros a recuperar → n_{rc} , ejemplo $\sigma_{A=1}(R)$? (Igualdad)*
- *Si el campo es clave → n_{rc}=1 ,*
  - *Archivo no ordenado por A, mejor caso 1 bloque, peor b_R, media b_R / 2*
  - *Archivo ordenado por A, búsqueda binaria :* $\lceil \log_2 (b_R) \rceil$
- *Si el campo no es clave → n_{rc}=n_R / V(A)*
  - Si el archivo no ordenado por A, → leer b_R bloques (peor caso)
  - *Si el archivo datos ordenado → leer :* $\lceil \log_2 (b_R) \rceil + \lceil n_{rc} / f_R \rceil$ -1 bloques.
- *Ejemplo: f_R =2 (registros /bloque), n_R=6, b_R=3, $\sigma_{valor=A}(R)$?*

A
A
A
A
B
B

Ordenado

n_{rc}=6 / 2=3 reg.

Leer = $\lceil \log_2 (3) \rceil + \lceil 3/2 \rceil$ -1 = 3 bl.

A
B
A
A
B
A

No Ordenado

n_{rc}=6 / 2=3 reg.

Leer = 3 bl.

Estimación registros/bloques a recuperar del archivo de datos

- $\sigma_{A>=1}(R)$ ó $\sigma_{A<=1}(R)$ ó $\sigma_{A>1 \wedge A<3}(R)$? (Comparación)

  $n_{rc} = n_v * n_r / V(A,R)$ (Nv → Número de valores diferentes que cumplen la condición)

- $\sigma_{A<>1}(R)$? (Distinto) → $n_{rc} = n_r - n_r(\sigma_{A=1}(R))$

- $\sigma_{A=1 \wedge B=3}(R)$? → $n_{rc} = n_r / (V(A,R) * V(B,R))$

- $\sigma_{A=1 \vee B=3}(R)$? → $n_{rc} = n_r / V(A,R) + n_r / V(B,R)$ (Aproximada)

- Coste? → Depende si hay ordenación o no.

Ejemplo

- *Considerar un archivo de datos que contiene información sobre estudiantes, donde se almacena:*
  - *Número de carnet: 20 bytes, es la PK*
  - *Nombre del alumno: 40 bytes.*
  - *Código de carrera: 2 bytes, in (0,1,2,3,4,5,6,7,8,9)*
  - *Edad: 16 bytes.*
  - *Índice académico: 32 bytes.*
- *Si hay 100.000 registros y cada bloque es de 512 bytes. ¿Cuál es el espacio necesario para almacenar el archivo?*
- *Si el bloque está ocupado al 65%, ¿Cuál será el espacio del disco ahora?. Redondear al entero más cercano.*
- *Coste y registros para* $\sigma_{carnet=2345}(r)$?
- *Coste y registros para* $\sigma_{codigo\_carrrera=2}(r)$?
- Si está ordenado el archivo por carnet, $\sigma_{carnet=2345}(r)$?
- Si está ordenado el archivo por código carrera, $\sigma_{codigo\_carrrera=2}(r)$?

Ejemplo

- *La longitud del registro será:*
  LR = 20+40+2+16+32 = 110 bytes.

- *Número de registros / bloque = Factor de bloque = ⌊512/ 110⌋ = ⌊4.65⌋ = 4*

- Número de bloques = ⌈100.000/4⌉ = 25.000 bloques.

- Si cada bloque ocupa 512 bytes = 25.000*512 = 12.20 MB

Ejemplo

- *Si el bloque está al 65%, entonces el número de registros / bloque ahora será : 4*0.65 = 2.63 $\Rightarrow$ 3 registros*
- Número de bloques = $\lceil 100.000/3 \rceil$ = 33.334 bloques.
- Si cada bloque ocupa 512 bytes = 33.334*512 = 16.27 MB
- SI ahora se utiliza la lista enlazada entre registros para seguir un orden lógico del fichero y cada puntero ocupa 12 bytes, ¿Cuál es el tamaño ocupado por el fichero ?

Ejemplo

- *La longitud del registro será:*
  LR = 20+40+2+16+32+12 = 122 bytes.
- *Número de registros / bloque = Factor de bloque = ⌊512/ 122⌋ = ⌊4.19⌋ = 4*
- Número de bloques = ⌈100.000/4⌉ = 25.000 bloques.
- Si cada bloque ocupa 512 bytes = 25.000*512 = 12.20 MB

Ejemplo

- $\sigma_{carnet=2345}(r)$? No hay ordenación

  Número de valores diferentes V(carnet)=100.000, es PK

  Número de registros a recuperar:

  $n_{rc} = n_R / V(carnet) = 100000/100000 = 1$

  Número de bloques a leer: No hay ordenación: Búsqueda secuencial, peor caso, leer todos los bloques : B_R = 25.000 bloq.

- $\sigma_{codigo\_carrrera=2}(r)$? No hay ordenación

  Número de valores diferentes V(codigo_carrera)=10, no es PK

  Número de registros a recuperar:

  $n_{rc} = n_R / V(código\_carrera) = 100000/10 = 10.000$

  Número de bloques a leer: No hay ordenación: Búsqueda secuencial, peor caso, leer todos los bloques : B_R = 25.000 bloq.

Ejemplo

- $\sigma_{carnet=2345}(r)$? Hay ordenación

  Número de valores diferentes V(carnet)=100.000, es PK

  Número de registros a recuperar:

  $n_{rc} = n_R / V(carnet) = 100000/100000 = 1$

  Número de bloques a leer: hay ordenación: Búsqueda binaria, $\lceil \log_2 (b_R) \rceil + \lceil n_{rc} / f_R \rceil - 1 = \lceil \log_2 (25000) \rceil + \lceil 1 / 4 \rceil - 1 = 15$ bl.

$\sigma_{codigo\_carrrera=2}(r)$? Hay ordenación

Número de valores diferentes V(código_carrera)=10, no es PK

Número de registros a recuperar:

$n_{rc} = n_R / V(codigo\_carrera) = 100000/10 = 10.000$

Número de bloques a leer: hay ordenación: Búsqueda binaria, $\lceil \log_2 (b_R) \rceil + \lceil n_{rc} / f_R \rceil - 1 = \lceil \log_2 (25000) \rceil + \lceil 10.000 / 4 \rceil - 1 =$ 2514 bloq.