
# *Unidad 1: Planificación del Almacenamiento e Indexación*

*Bases de Datos Avanzadas, Sesión 2-3: Estructuras Índices*

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


# INDICE

- *Introducción.*
- *Clasificación*
- *Índices Secuenciales (ordenados)*
- *Árboles $B^+$*
- *Árboles B*
- *Índices Asociativos*
- *Índices Multiclave*
- *Retícula*
- *Asociación Dividida*
- *Mapas de Bits*

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


# *Indización de datos*

- *Mejorar el tiempo de búsqueda de los datos de la base de datos.*
- *Existen dos tipos básicos*
  - *Ordenados ⇒ basados en la disposición ordenada de valores*
  - *Asociativos (hash) ⇒ distribución uniforme de valores por funciones hash entre cajones (buckets).*

- *Criterios de valoración:*
  - *Tipos de acceso: búsqueda único valor o rango de valores.*
  - *Tiempo de acceso: tiempo en buscar un elemento de datos o varios*
  - *Tiempo de inserción: buscar + actualizar*
  - *Tiempo de borrado: buscar + actualizar*
  - *Espacio adicional requerido por el índice.*
- *Coste de Buscar datos (C)= Coste Buscar Índice ($C_{indice}$) + Coste Buscar archivo de Datos ($C_{datos}$)*


# Clasificación de índices

- Consideraciones respecto al campo de búsqueda.
- V(A) → número de valores diferentes para el campo A en archivo / tabla
- $n_R$ → número de registros del archivo (tabla) R

- Archivo - Ordenado respecto al campo de búsqueda → primario
  - No ordenado respecto al campo de búsqueda → secundario

- Valores del campo: - Campo clave → 1 valor dentro del archivo / tabla
  - Campo no clave → valores repetidos

- $n_{Ri}$ → número registros en el índice (nunca entradas/valores duplicados)

- Registro índice → Valor campo + Puntero (a Registro datos ó Bloque)

  $L_{Ri} = L_K + L_{PB}$ ó $L_{Ri} = L_K + L_{PR}$


# Clasificación de índices

- Clasificación:
  - Primario + campo clave → $n_{Ri} = V(A) = n_R$
  - Primario + campo no clave → $n_{Ri} = V(A)$ ( sólo se apunta al primero)
  - Secundario + campo clave → $n_{Ri} = V(A) = n_R$
  - Secundario + campo no clave → cajones punteros → $n_{Ri} = V(A)$ (se apunta a cada cajón)

Cajones de punteros contienen siempre Punteros a Registro


# *Estimación registros/bloques a recuperar del archivo de datos ($C_{datos}$)*

- *Número registros a recuperar → $n_{rc}$*
- *Si el campo es clave → $n_{rc}$=1 → leer un bloque del archivo datos.*
- *Si el campo no es clave → $n_{rc}$= $N_v$ * $n_R$ / V(A)*

*donde $N_v$ → Nº valores diferentes que cumplen la condición*

- *Si el archivo datos ordenado (primario) → leer* $\lceil n_{rc} / f_R \rceil$ bloques datos
- Si el archivo no está ordenado (secundario) → leer $n_{rc}$ bloques datos (peor caso)

# *Estimación registros/bloques a recuperar del archivo de datos ($C_{datos}$)*

- *Ejemplo: $f_R$ =2 (registros /bloque), $n_R$=6, $b_R$=3, $\sigma_{campo=A}(R)$ ?*

*Archivo de datos*

| *Campo* |
|---|
| A |
| A |
| A |
| A |
| B |
| B |

*V(Campo)=2*

| *Campo* |
|---|
| A |
| B |
| A |
| A |
| B |
| A |

*Primario (Ordenado)*

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

*Leer* = $\lceil 3 / 2 \rceil$ =2 bl. datos

*Secundario (No Ordenado)*

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

*Leer* = 3 bl. datos


# Índices secuenciales (Índice ordenado)

- El índice es un archivo secuencial que siempre está ordenado.
- Dos tipos básicos:
  - Índice Primario
    - archivo datos ordenado respecto a una clave de búsqueda.
    - Suele ser la clave primaria pero no tiene por qué.
    - Índice primario ≠ índice sobre clave primaria.
    - Otro nombre: índice con agrupación (cluster index)
  - Índice secundario
    - Archivo datos no ordenado respecto a la clave de búsqueda.
    - Otro nombre: índice sin agrupación (non cluster index)

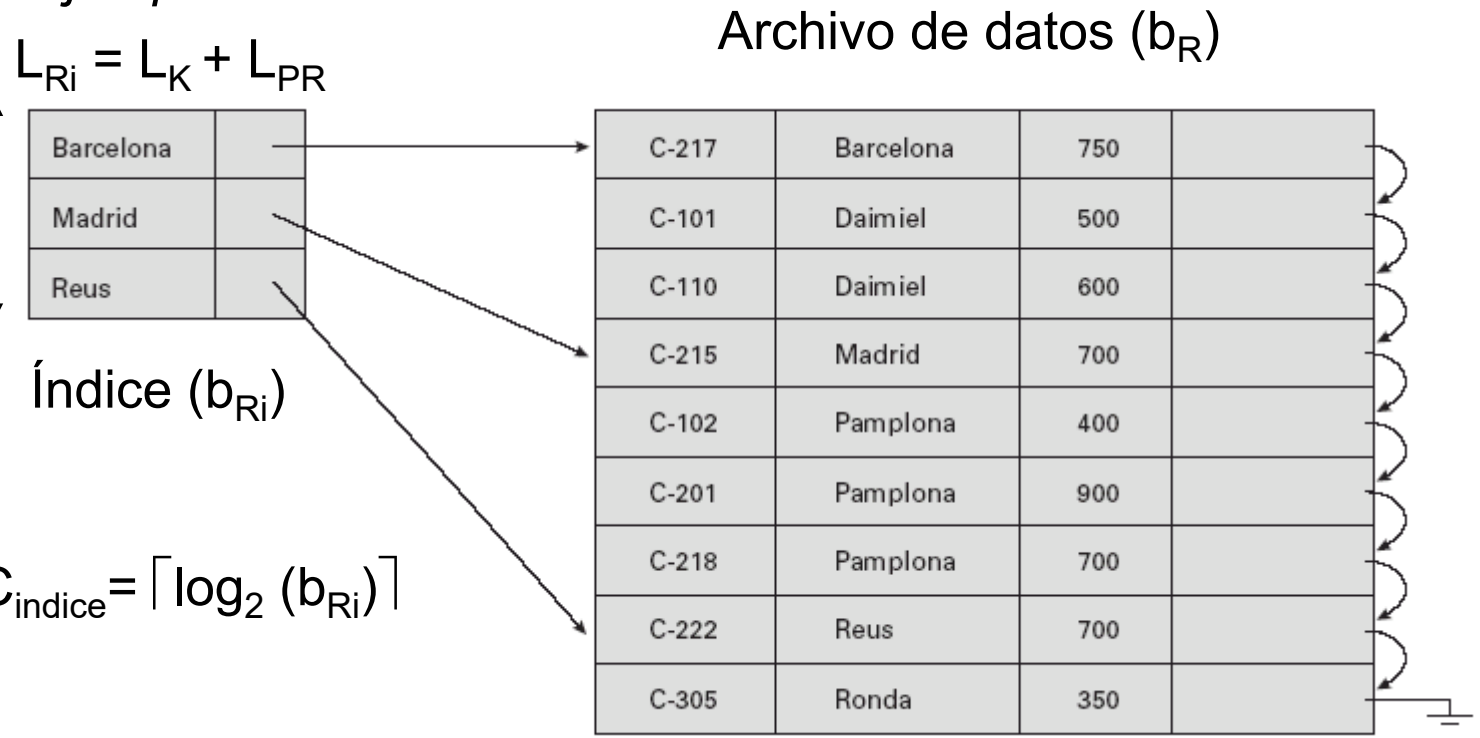

# *Índices secuenciales: Primarios*

- *Índice primario:*
  - *Los ficheros se ordenan secuencialmente a partir de una clave de búsqueda principal.*
  - *Estos archivos se llaman archivos secuenciales indexados.*
  - *Muy utilizados en bases de datos que exigen procesamiento secuencial y acceso directo a sus registros*
  - *Condicionan ciertos tipos de indexación $\Rightarrow$ índice disperso.*

    *Índice denso*

# *Índices secuenciales: Primarios*

- *Ejemplo:*



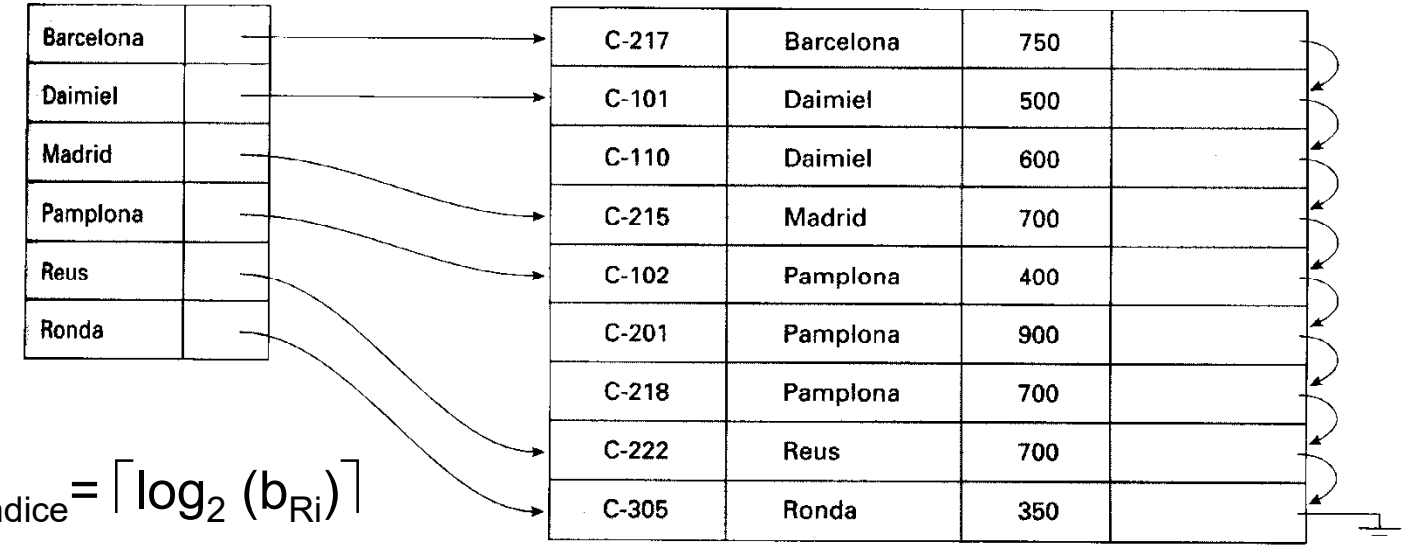

- *Registro índice: valor de la clave de búsqueda + punteros a registros con ese valor de la clave de búsqueda.*
- *Puntero a registro: numero de bloque + offset dentro del bloque*


# *Índices secuenciales: Densos*

- *Índice Denso:*
  - *Se crea un nuevo registro índice para cada valor de la clave de búsqueda en el archivo.*
  - *El registro índice contiene la clave de búsqueda y un puntero al primer registro de la base de datos con ese valor*
  - *Destaca que esta aproximación es correcta si los ficheros de la base de datos están ordenados con la estructura del índice primario.*

# Índices secuenciales: Densos

- *Su funcionamiento es sencillo: Al realizar una búsqueda en la base de datos de accede al índice y se busca el elemento que coincida con la clave de búsqueda.*
- *Una vez encontrado, se puede acceder de forma directa al registro que contiene esa información, y a partir de ese registro, si hay otros con la misma clave de búsqueda, aparecerán a continuación.*



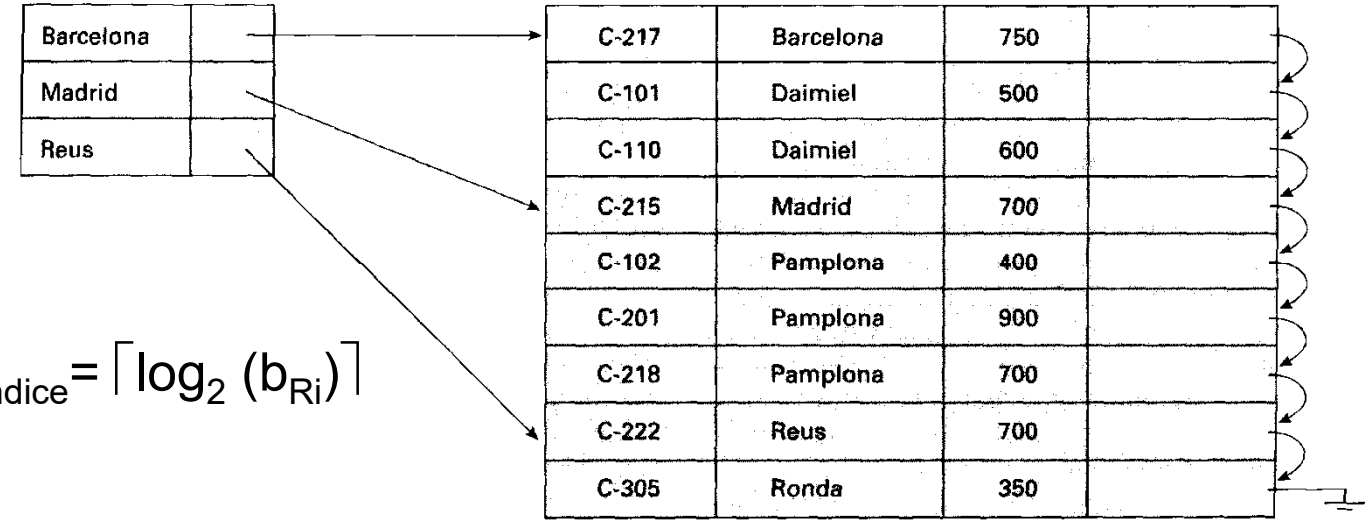

$C_{indice} = \lceil \log_2 (b_{Ri}) \rceil$


# Índices secuenciales: Dispersos

- *Índice Disperso:*
  - *Sólo se crea el registro índice para ciertos valores → una entrada por cada bloque del archivo de datos*
  - *Al igual que los densos ⇒ cada registro índice contiene un valor de la clave de búsqueda y un puntero al primer registro con ese valor de clave*
  - *Utilizan menos espacio al no almacenar todas las claves de búsqueda*
  - *Menor mantenimiento para las inserciones y borrados.*
  - *Se basan en los índices primarios.*

# Índices secuenciales: Dispersos

- *Para localizar un registro:*
  - *Se busca la entrada del índice con el valor más grande que sea menor ó igual al valor que se está buscando.*
  - *Una vez encontrado ⇒ se pasa a recorrer la base de datos hasta encontrar el registro exacto que se busca.*



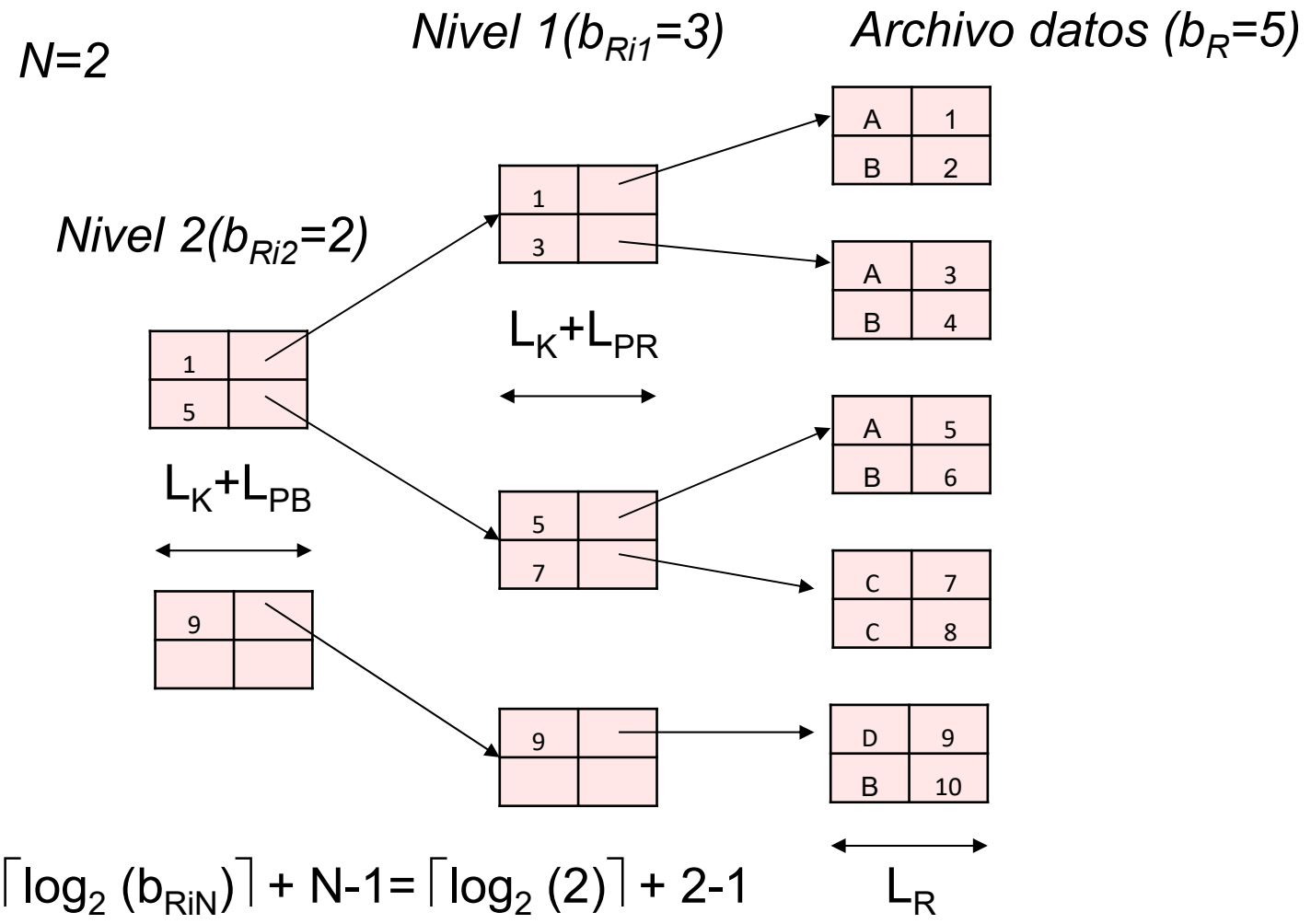

$$C_{indice} = \lceil \log_2 (b_{Ri}) \rceil$$


# *Índices secuenciales: Multinivel*

- *Índices Multinivel → Hacer índices dispersos sobre índices de 1 nivel (los anteriores). Como si fuese un árbol hasta el nivel dado.*

  - *Se utilizan para grandes cantidades de datos.*
  - *Utilizan bloques de índices, generando una estructura en árbol con dos niveles de indirección.*
  - *Cada bloque de índices corresponde normalmente con un bloque de disco ⇒ se optimizan al máximo los accesos.*
  - *El bloque principal ó índice externo se suele cargar en la memoria principal o en la memoria de intercambio, al ser el más utilizado.*
  - *Dependiendo de la cantidad de datos, se puede aumentar el nivel de indirección.*

# Índices secuenciales: Multinivel (N niveles)

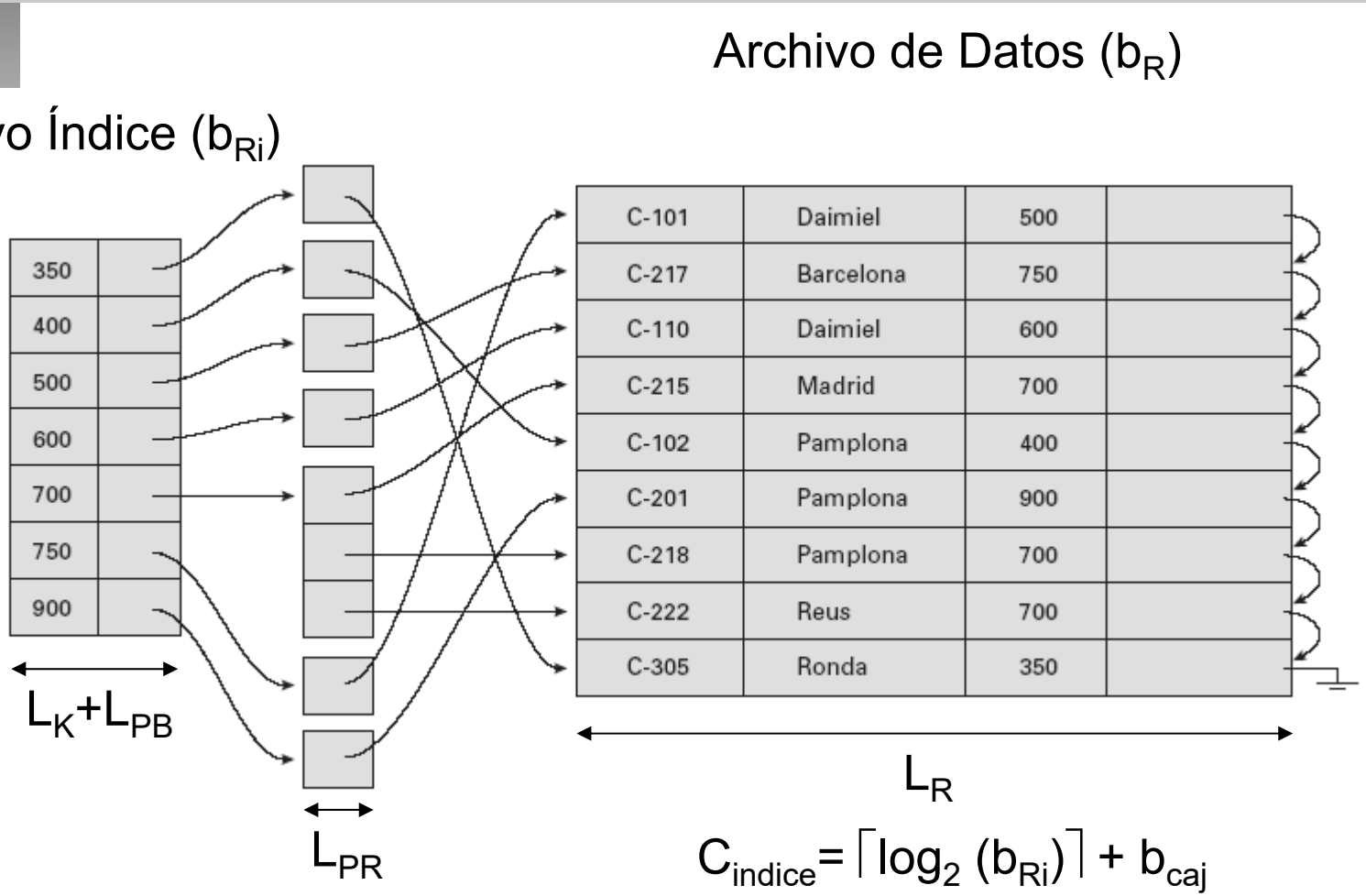



$C_{indice} = \lceil \log_2 (b_{RiN}) \rceil + N-1 = \lceil \log_2 (2) \rceil + 2-1$


# *Índices secuenciales secundarios*

- *Los índices secundarios están estructurados de manera diferente a los primarios.*
- *Es necesario mantener referencias a todos los registros de una clave de búsqueda ⇒ Siempre Densos*
- *Los índices secundarios sobre claves candidatas son similares a los índices primarios, excepto que los registros apuntados por los valores del índice no se encuentran almacenados de forma secuencial.*
- *Se suelen utilizar bloques de punteros asociados a cada una de las claves de búsqueda.*

# *Índices secuenciales secundarios*




Cajones de Punteros ($b_{caj}$)

1 cajón tiene $n_{Ri}$ / V(campo) Punteros


# *Ejemplos*

- *Ejemplos de índices secuenciales:*

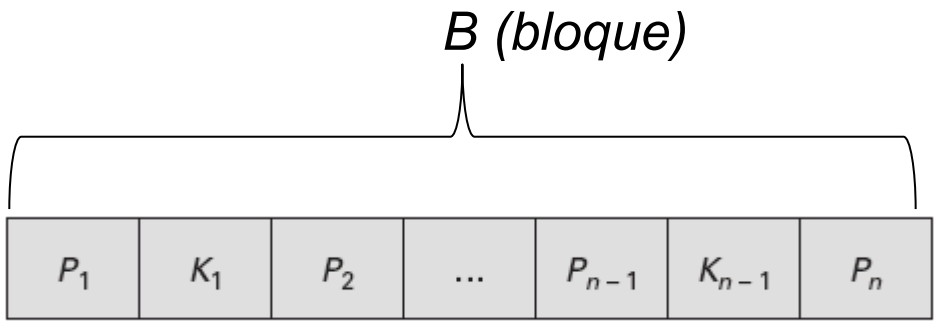
  *Ejemplos_indices_secuenciales.ppt*


# *Índices de Árbol $B^+$*

- *Los archivos secuenciales indexados tienen problemas a la hora de realizar búsquedas $\Rightarrow$ a medida que crece el fichero se va degradando el rendimiento.*
- *Las estructuras de índice de árbol $B^+$ mantienen su eficiencia a pesar de la inserción y borrado de datos.*
- *Los árboles $B^+$ son estructuras arbóreas equilibradas, donde los caminos de la raíz a cada nodo hoja es fijo*
- *Cada nodo no hoja tiene entre $\lceil n/2 \rceil$ y n hijos, donde n se fija para cada árbol en particular.*

# Índices de Árbol $B^+$

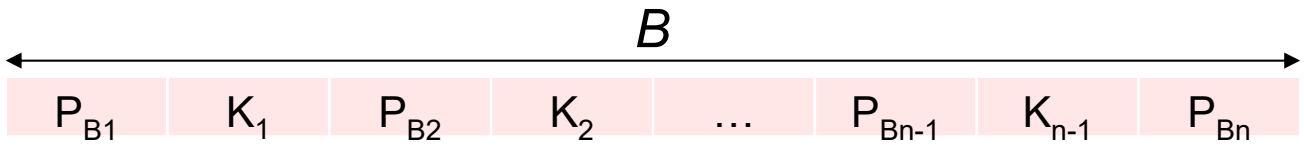
- Nodo típico de Árbol $B^+$, $P_i$ → puntero , $K_i$ → valor campo , $K_i < K_{i+1}$




- n punteros (grado de salida):
  - Intermedios son punteros a bloques
  - Hoja son punteros a registros ó bloques.
- n-1 valores de la clave de búsqueda
- Nodos hoja contengan al menos $\lceil (n-1)/2 \rceil$ valores.
- Nodos intermedios al menos $\lceil n/2 \rceil$ punteros
- Nodo raíz al menos 2 punteros.
- Característica árbol: balanceado.

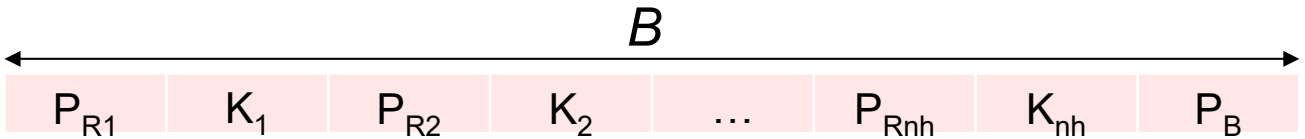

# Índices de Árbol $B^+$

- *Diseño de los nodos*
- *Nodo intermedio / Raíz → Punteros a bloque + valores campo*




$$n*L_{PB} + (n-1)*L_K \leq B$$

- *Nodo hoja → Punteros a registro o bloque + valores campo + puntero a bloque*



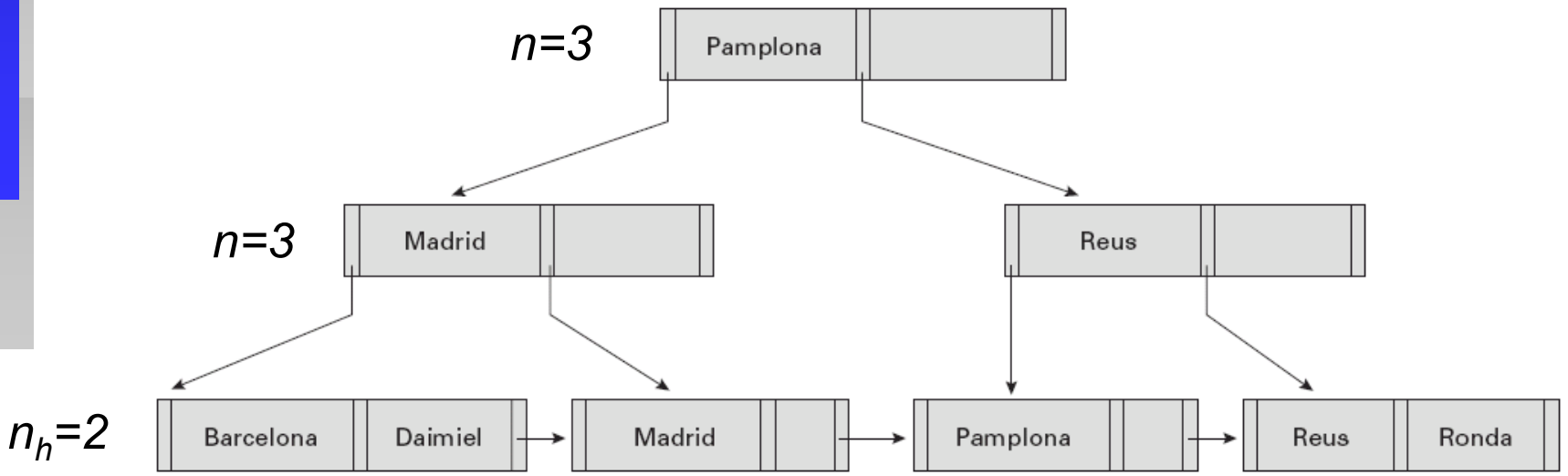

$$n_h*(L_{PR} + L_K) + L_{PB} \leq B$$

$L_{PR}$ /$L_{PB}$ → *longitud puntero a registro /bloque (bytes)*

$L_K$ → *longitud de campo de búsqueda del índice (bytes)*

# Índices de Árbol $B^+$

- *Estructura de ejemplo de un árbol $B^+$*
- *Cada nodo del árbol contiene n punteros a bloques, junto con n-1 claves de búsqueda. N=3 niveles, n=3 punteros en cada nodo*

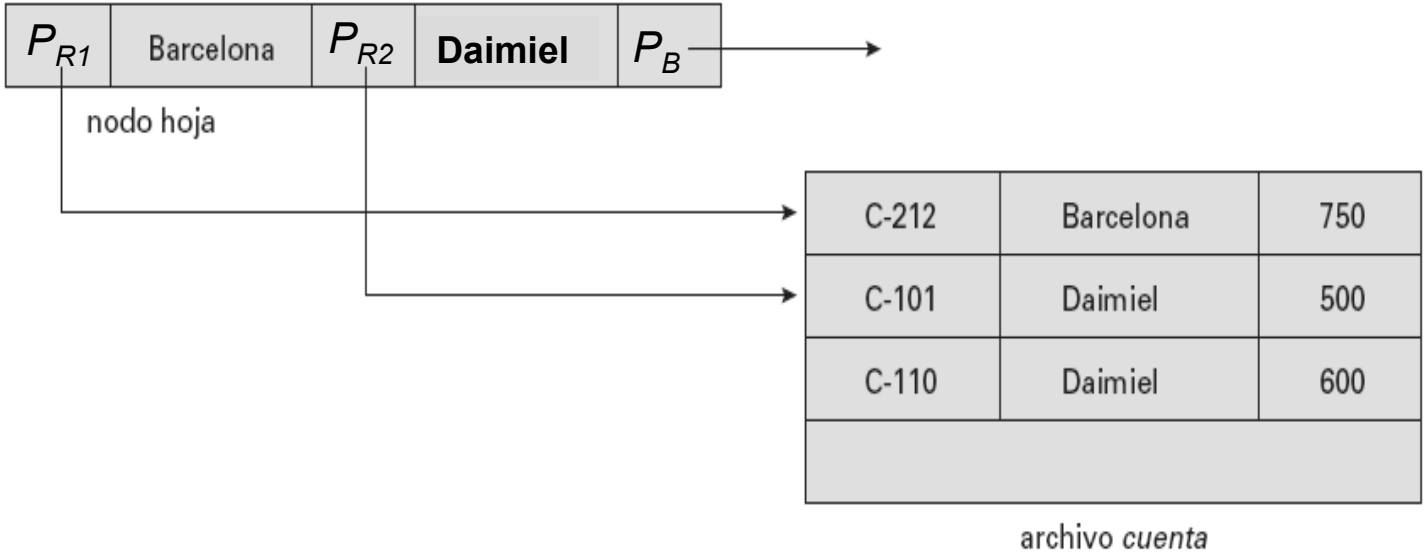



$n_{Ri}$=V(campo) = 6, Número nodos hojas= $b_{hojas}$= $\lceil n_{Ri} / n_h \rceil = \lceil 6/ 2 \rceil = 3$

Número nodos intermedio 1= $\lceil b_{hojas} / n \rceil = \lceil 3 / 3 \rceil = 1$ , es la raíz ya.

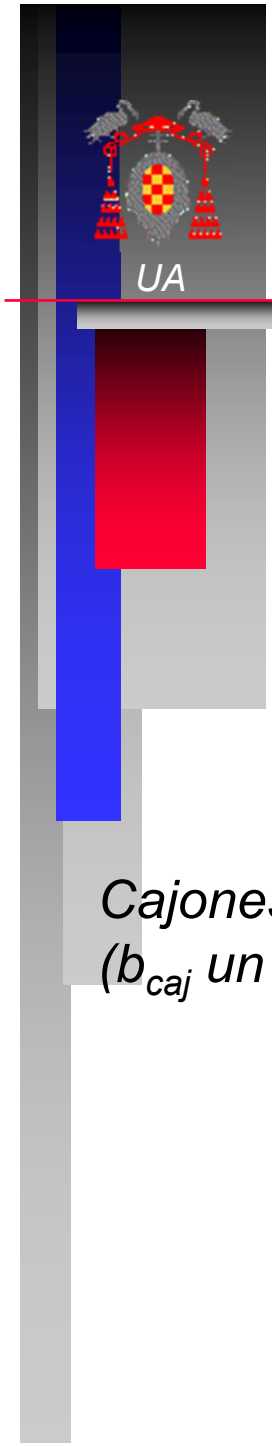
$C_{indice}$ = N = 2 niveles en esta estimación.

# Índices de Árbol $B^+$

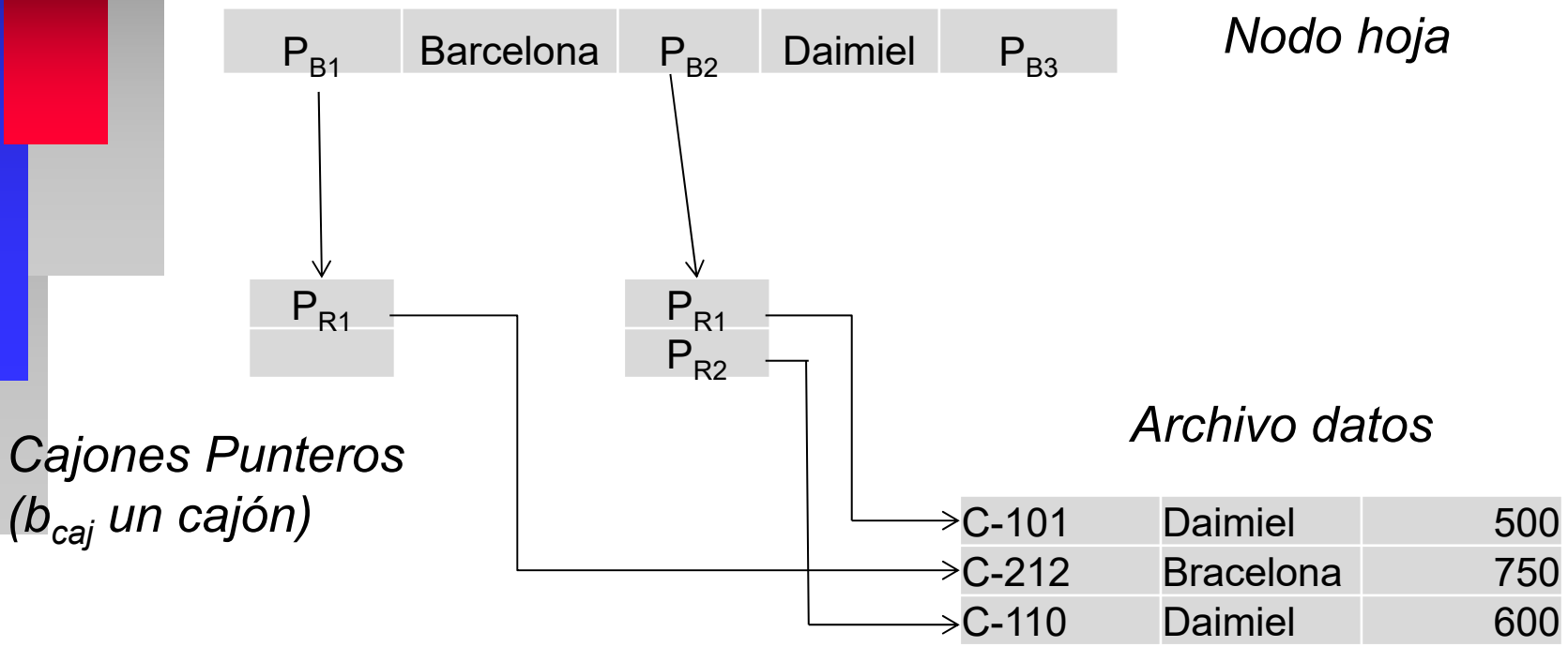



*Índice primario + campo no clave*

$C_{indice} = N$




# Índices de Árbol $B^+$




*Índice secundario + campo no clave → cajones punteros*

$C_{indice} = N + b_{caj}$


# *Ejemplos*

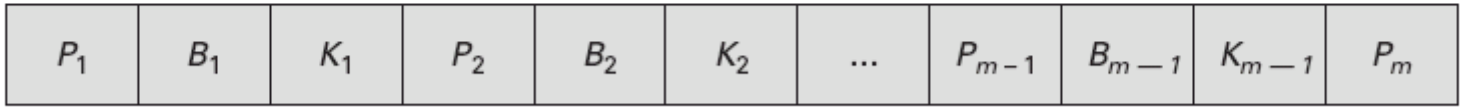
- *Ejemplo de árboles B+*

Ejemplos_arboles_Bmas.ppt


# *Archivos de índices de Árbol B*

- *Son similares a los árboles B+ excepto que el árbol B elimina el almacenamiento redundante de los valores de la clave de búsqueda.*
- *Esta operación se realiza añadiendo un campo más a los nodos internos del árbol, y eliminando por tanto esos datos de los nodos hojas finales, lo que crea una reestructuración completa del árbol*


(a)

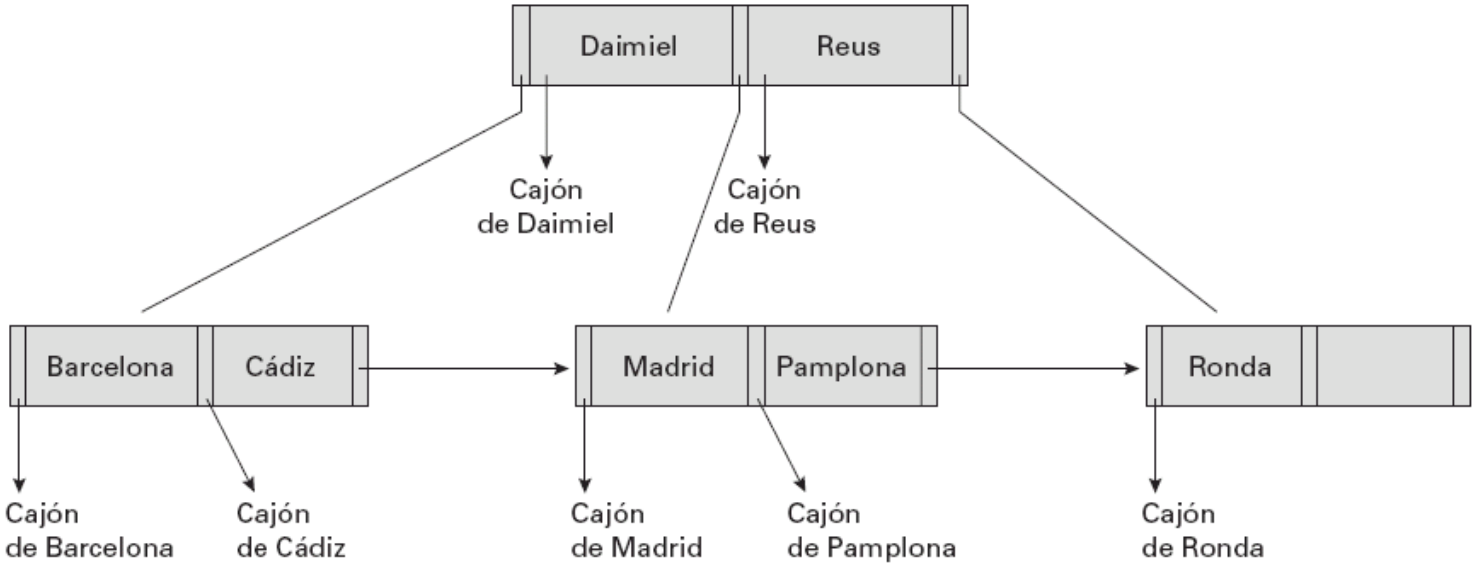



(b)

$m*L_{PB} + (m-1)*(L_K+L_{PR}) \leq B$ , nodo Intermedio

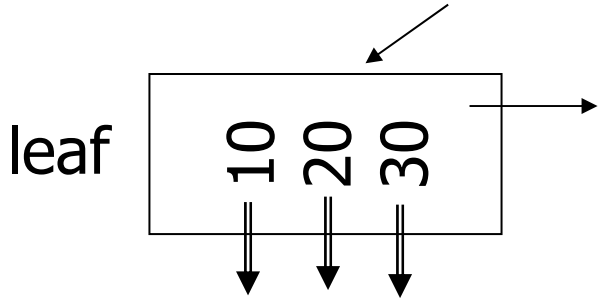
# *Archivos de índices de Árbol B*

- *En los árboles B es más complicado el borrado, pero más simple la inserción.*



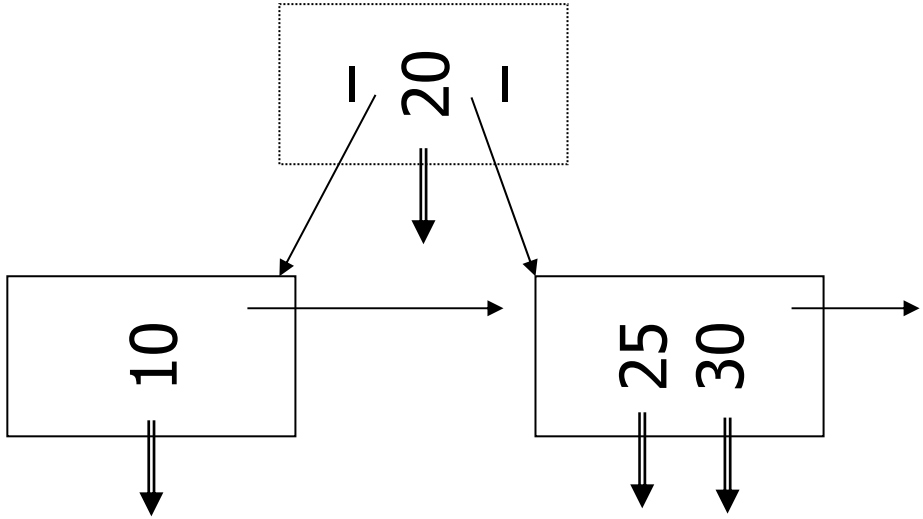

# *Ejemplo*

- *Say we insert record with key = 25*




n=4

- *Afterwards:*




# *Asociación Estática*

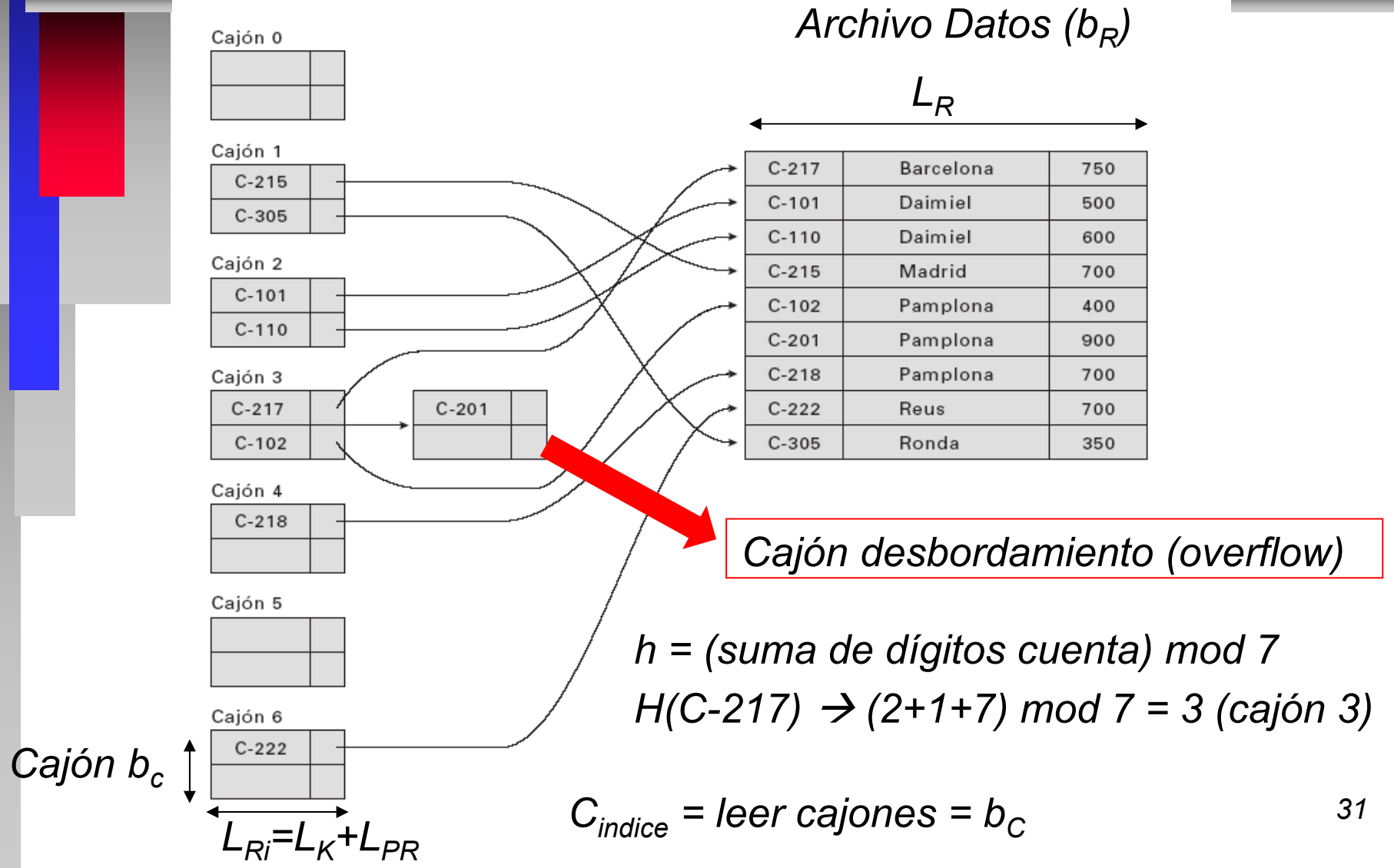
- *Es una técnica basada en la utilización de funciones de asociación.*

- *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:*

$$h(k)=B$$

- *Igual que archivos hash, pero es un índice. Mismas propiedades*
- *Los cajones guardan valor del campo + Punteros*
- *Los cajones están todos creados (estáticos).*

# Índices asociativos




$h$ = (suma de dígitos cuenta) mod 7

$H(C\text{-}217) \rightarrow (2+1+7) \bmod 7 = 3$ (cajón 3)

$C_{indice}$ = leer cajones = $b_C$


# *Ejemplo*

- *Ejemplo de índices asociativos:*

  *ejemplos_indices_asociativos.ppt*

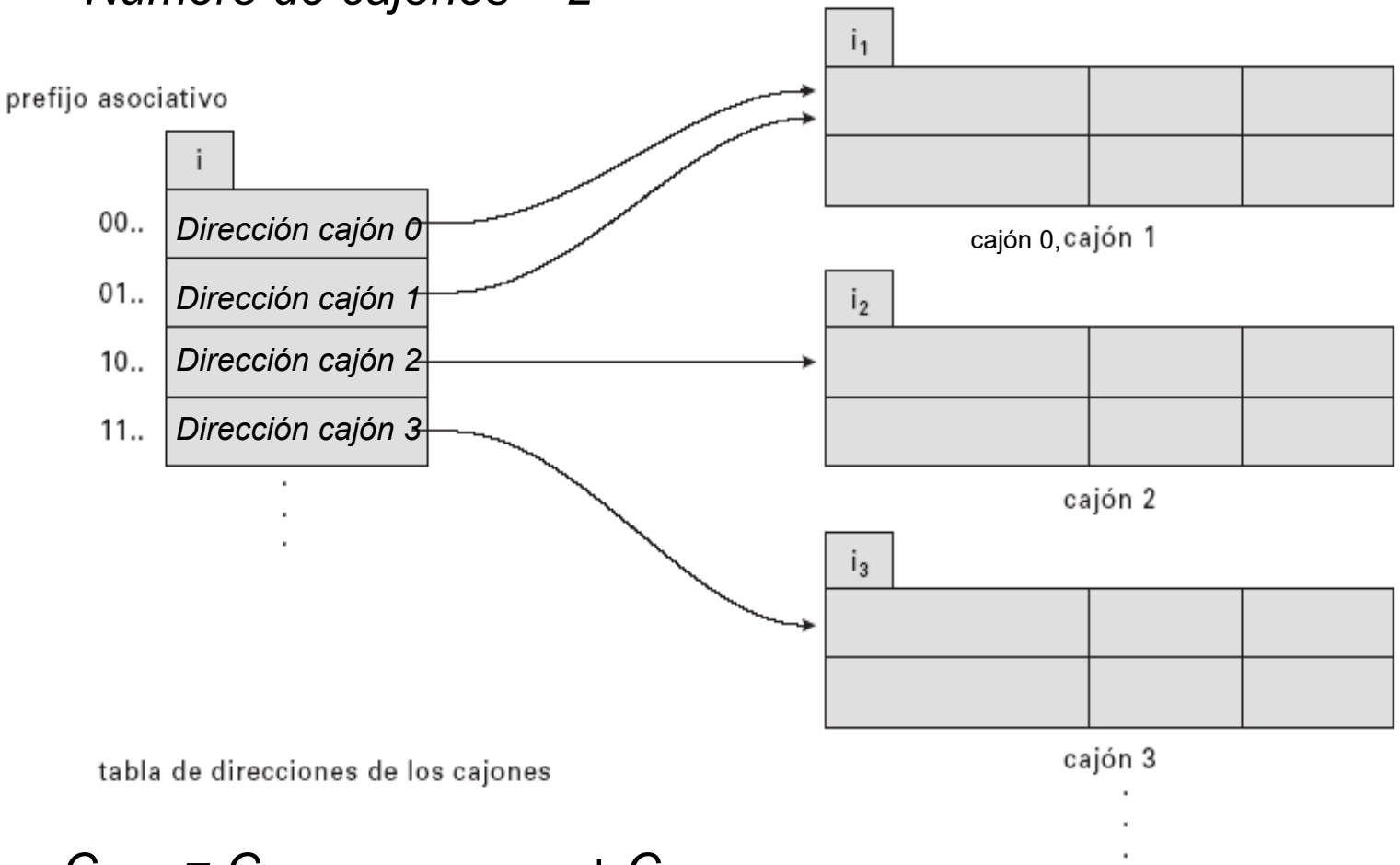

# *Asociación Dinámica*

- *Las técnicas de asociación estática tienen el problema de que poco a poco la base de datos va aumentando su tamaño ⇒ irremediablemente se llegan a situaciones de desbordamiento.*
- *Las técnicas de asociación dinámica permiten paliar esos problemas ⇒ modificando la función de asociación dinámicamente para acomodarse al aumento o disminución de la base de datos.*
- *En esta parte del tema se estudiará la estructura asociativa general dinámica.*

# *Asociación Dinámica*

UA

*Número de cajones = $2^i$*




$C_{indice} = C_{leer\ tabla\ direcciones} + C_{leer\ cajón}$


# *Asociación dinámica: Ejemplo*

- *Ejemplo asociación dinámica:*

*ejemplos_indices_asociativos_dinamicos.pptx*


# *Definición de índices en SQL*

- *La normal SQL no permite el control de los índices de la base de datos.*
- *Aún así, la mayoría de sistemas permiten la gestión de índices:*
- *Ejemplos:*
  - *Create index <nombre-indice> on <nombre-relación> (<lista-atrbutos>)*
  - *Create index indice-s on sucursal(nombre sucursal).*
  - *Create unique index indice on tabla(clave-unica).*
  - *Drop index <nombre-indice>*


# *Accesos Multiclave*

- *Muchas consultas realizadas a la base de datos constan de condiciones en varios índices:*

  *Select * from tabla where Campo1>10 and Campo2<7*

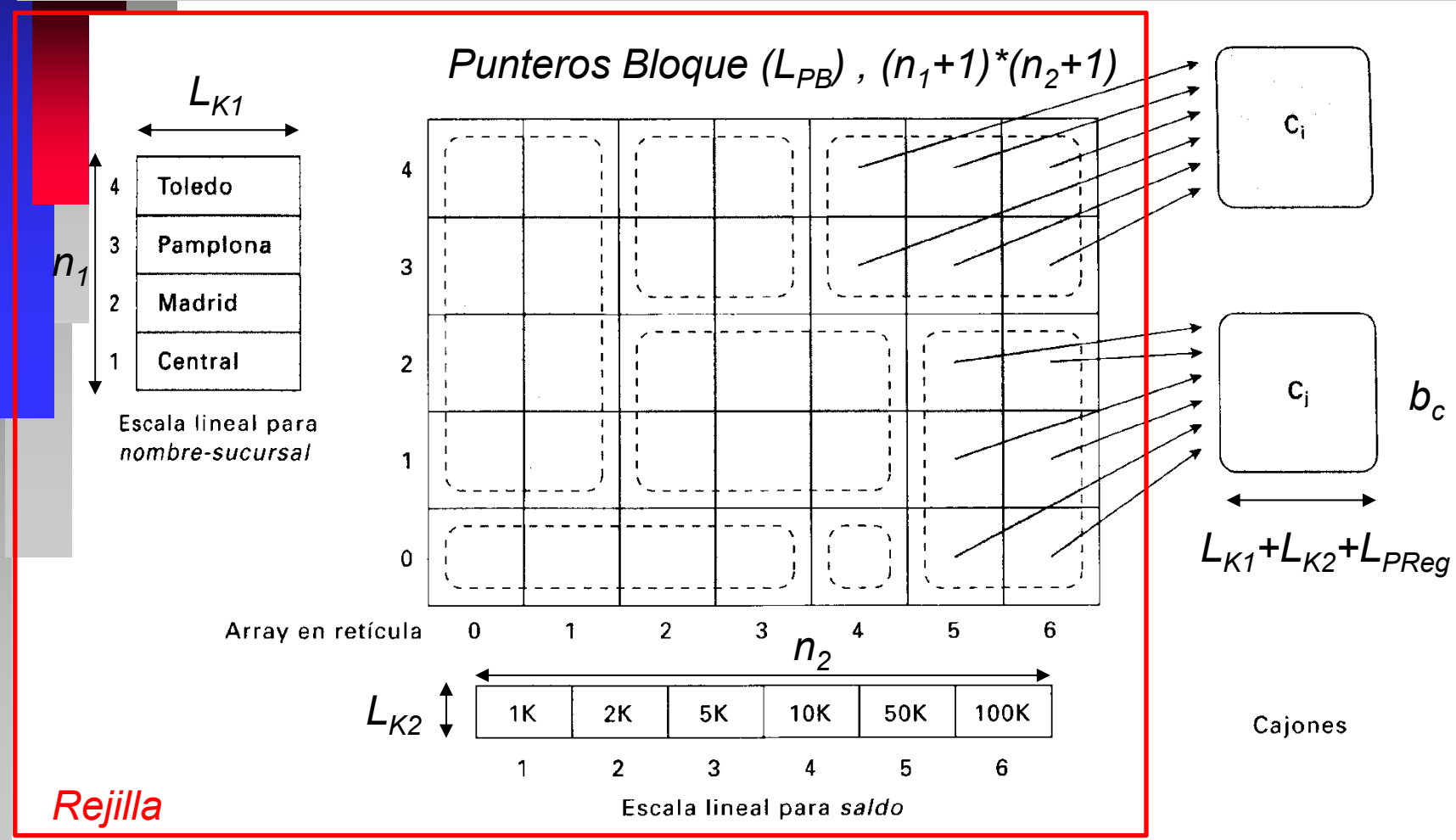
- *Para este tipo de casos se puede:*

  - *Usar primero el índice Campo1 y luego el Campo2*
  - *Usar primero el Campo2 y luego el Campo1*
  - *Hacer la intersección de ambos índices con las condiciones dadas.*


# *Archivos en Retícula*

- *Son archivos que intrínsecamente sitúan la información para su uso con múltiples índices.*
- *Se basan en la utilización de escalas lineales.*
- *Las celdas de los archivos en retícula apuntan a los cajones de datos.*
- *Si se desean utilizar n índices en la retícula, se crearán arrays de retícula n-dimensionales.*

# Archivos en Retícula




$C_{indice} = C_{Leer\ Rejilla} + C_{Leer\ Cajones}$

$L_{Rejilla} = n_1*L_{K1}+n_2*L_{K2}+(n_1+1)*(n_2+1)*L_{PB}$ bytes

$B_{Rejilla} = \lceil L_{Rejilla} / B \rceil$ bloques


# *Índices en Retícula: Ejemplo*

- *Ejemplo de índice en retícula:*

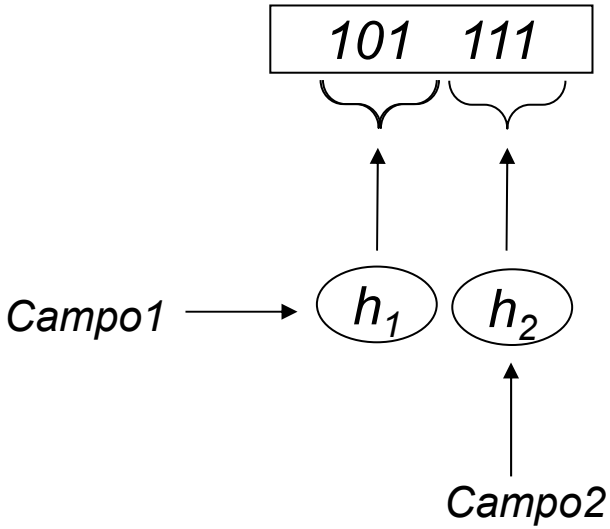
*ejemplos_indices_rejilla.pptx*


# *Asociación dividida*

- *Es un tipo de asociación donde se asignan valores de la función hash a las claves compuestas, y a partir de ellas se lleva a cabo la indexación.*
- *La clave hash se divide en tantos segmentos como claves individuales se utilicen para generar la clave principal.*

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

$C_{indice} = C_{leer\ cajones}$




# *Asociación dividida: Ejemplo*

- *Ejemplo asociación dividida:*

*ejemplos_asociacion_dividida.ppt*

# Índices de Mapas de Bits

- *Índices especializados para la consulta sencilla sobre varias claves.*
- *Registros numerados secuencialmente empezando de 0*

| número de registro | nombre | sexo | dirección | nivel-ingresos |
|---|---|---|---|---|
| 0 | Juan | m | Pamplona | L1 |
| 1 | Diana | f | Barcelona | L2 |
| 2 | María | f | Jaén | L1 |
| 3 | Pedro | m | Barcelona | L4 |
| 4 | Katzalin | f | Pamplona | L3 |

Mapas de bits para *sexo*

| | |
|---|---|
| m | 10010 |
| f | 01101 |

Mapas de bits para *nivel-ingresos*

| | |
|---|---|
| L1 | 10100 |
| L2 | 01000 |
| L3 | 00001 |
| L4 | 00010 |
| L5 | 00000 |

- *Número de mapas de cada campo → V(campo)*
- *Cada mapa tiene $n_R$ bits.*
- *Leer secuencialmente los registros*
- *Para contar tuplas*

$L_{mapa} = n_R$ *bits*

$b_{mapa} = \lceil (L_{mapa}/8) / B \rceil$

*Número mapas = V(campo)*

$C_{indice} = C_{leer\ mapas\ bits}$


# Índice Mapa de Bits: Ejemplo

- Ejemplo mapa de bits:

S03-U1-ejemplos_indices_bitmap.ppt