Bases de Datos

Indexación y Asociación

Contenidos

- Conceptos básicos
- Indices Ordenados
- Árboles B+
- Arboles B
- Asociación estática

Conceptos básicos

- Los Indices se utilizan parta aumentar la velocidad de acceso a los datos
- Clave de búsqueda: atributo o conj. de atributos que se utilizan para buscar en un archivo.
- Un fichero Indice está formado por registros de la forma

Clave de búsqueda	Puntero

- Dos tipos de indices:
  - Indices ordenados: los valores están ordenados
  - Indices asociados: las claves de búsqueda están distribuidas uniformemente a los largo de cajones utilizando una función de asociación.

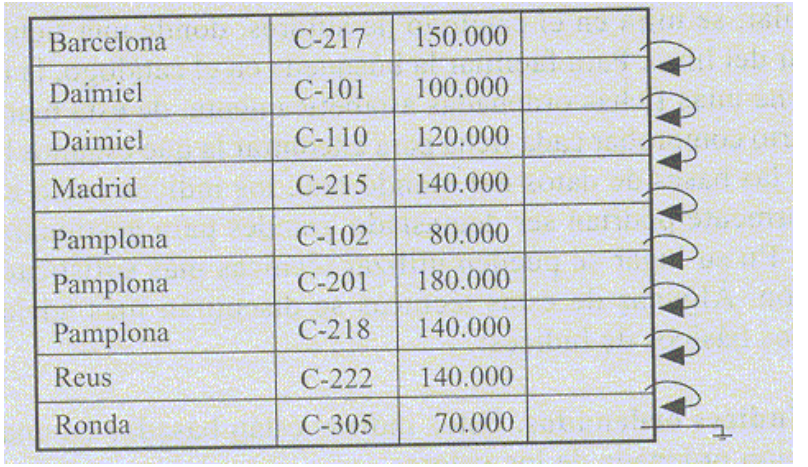
Criterios de evaluación de los Índices

- Tipos de acceso que se soportan eficazmente, p.ej.:
  - registros con un valor concreto de atributo
  - registros con un atributo entre un rango de valores
- Tiempo de acceso
- Tiempo de insercción
- Tiempo de borrado
- Espacio adicional requerido

Indices Ordenados

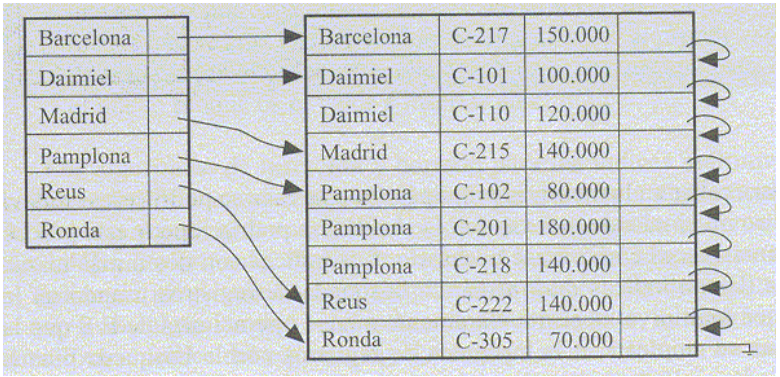
- Los registros indices se almacenan ordenados por el valor de la clave de búsqueda.
- Indices primarios: en un archivo ordenado secuencialmente, es el indice cuya clave de búsqueda especifica el orden secuencial del archivo.
  - También se llama índice con agrupación.
  - La clave de búsqueda de un índice primario suele ser la clave primaria, aunque no necesariamente.
- Indice secundario: es un índice cuya clave de búsqueda especifica un orden distinto del orden secuencial del archivo.
- Archivo Secuencial Indexado: archivos ordenados secuencialmente con índice primario.

Archivo Secuencial Indexado (ejemplo)



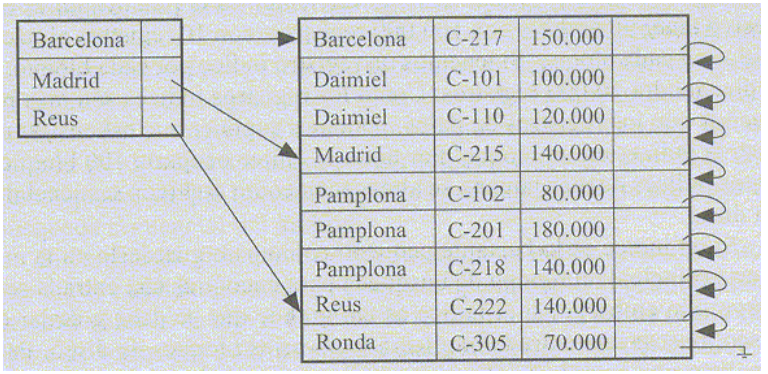
Indice Denso

Aparece un registro indice para cada valor de la clave de búsqueda en el archivo



Indice Disperso

Sólo se crea un registro índice para algunos de los valores de la clave de búsqueda.



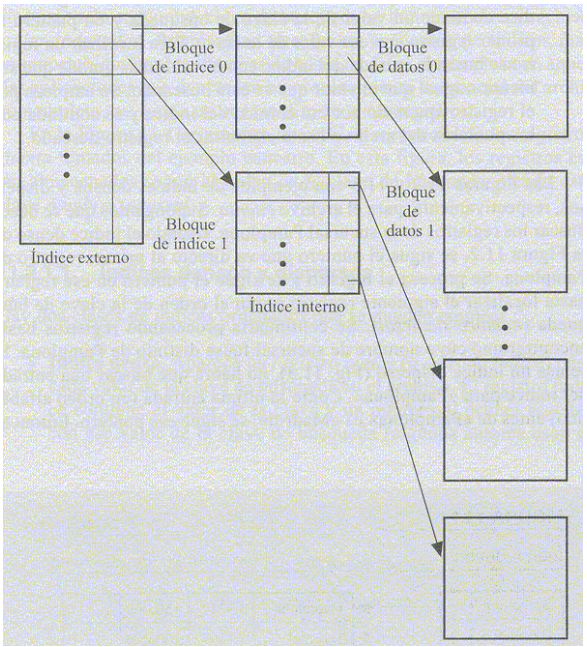
Indice Denso vs. Disperso

- Generalmente más rápido localizar un registro con Indice Denso que con Disperso.
- Los indices dispersos utilizan menos espacio, y tienen un mantenimiento menor para las insercciones y borrados.
- Un buen compromiso entre tiempo de acceso y espacio adicional requerido es tener un índice disperso con una entrada del índice por cada bloque.

Indices Multinivel (1/2)

- Si el índice primario no cabe en memoria, el acceso se hace costoso.
- Para reducir el número de accesos de disco, se trata el índice como si fuera un archivo secuencial y se construye un índice disperso sobre él.
  - Indice externo: un índice disperso del índice primario
  - Indice interno: el índice primario
- Si incluso el índice externo es demasiado grande para caber en memoria, se podría crear otro nivel de indexación.
- En las insercciones y borrados hay que actualizar los índices a todos los niveles.

Indices Multinivel (2/2)



Actualización del Indice: Borrado

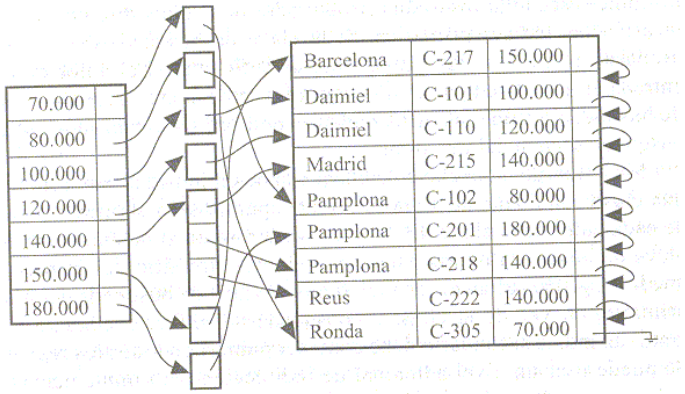
- Si el registro borrado era el único registro en el archivo con ese valor de clave de búsqueda, la clave de búsqueda se borra del índice también.
- Borrado en un indice de un solo nivel:
  - Indice denso: el borrado de la clave de búsqueda es similar al borrado de un registro.
  - Indice disperso: si una entrada para la clave de búsqueda existe en el índice, se borra reemplazando la entrada en el índice con la siguiente clave de búsqueda (en orden). Si la siguiente clave de búsqueda ya tiene una entrada, se borra sin más sin reemplazarla.

Actualización del Indice: Inserción

- Insercción en un Indice de un solo nivel:
  - Primero se realiza una búsqueda utilizando la clave de búsqueda del registro a insertar.
  - Indices densos: si el valor de la clave de búsqueda no aparece en el índice, el valor se inserta en el índice.
  - Indices dispersos: si almacena una entrada por cada bloque, no es necesario cambiar el índice, a menos que se cree un nuevo bloque. En este caso, el primer valor de la clave (en orden) que aparezca en el nuevo bloque es el valor a insertar en el índice.
- Los algorimos de insercción y borrado multinivel son simples extensiones de los algoritmos de un único nivel.

Indices Secundarios

- Es como un indice primario, excepto en que los registros apuntados por el indice no están almacenados sucesivamente.



Indices primarios y secundarios

- Los índices secundarios tienen que ser densos.
- Los índices ofrecen sustanciales beneficios cuando se utilizan para buscar registros.
- Cuando se modifica un archivo, se debe actualizar cada índice del archivo.
- La actualización de los índices imponen un tiempo adicional en la modificación de la Base de Datos.

Archivos de Indices de árbol B^+

- Desventajas de los archivos secuenciales indexados: el rendimiento se degrada según crece el archivo. Esta degradación se resuelve reorganizando el archivo.
- Ventajas de árboles B^+: automaticamente se reorganiza con cambios pequeños y locales en las insercciones y borrados. No se requiere la reorganización total del archivo.
- Desventajas de árboles B^+: una degradación al insertar y borrar, y espacio extra.
- Las ventajas de los árboles B^+ son mayores que sus desventajas y se usan ampliamente, siendo una alternativa a los archivos secuenciales indexados..

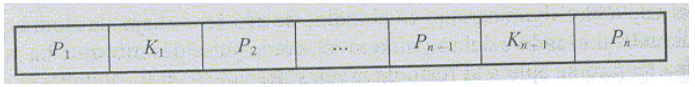
Archivos de Indice de árbol B^+

Un árbol B^+ satisface estas propiedades:

- Todos los caminos de la raíz a las hojas tienen la misma longitud.
- Cada nodo que no es raíz ni hoja tiene entre $\lceil n/2 \rceil$ y n hijos, donde n está fijo para cada árbol en particular.
- Un nodo hoja tiene entre $\lceil (n-1)/2 \rceil$ y (n-1) valores.
- Casos especiales:
  - si la raíz no es una hoja, tiene como mínimo 2 hijos.
  - Si la raíz es una hoja, puede tener entre 0 y (n-1) valores.

Estructura de un nodo de árbol B^+

- Nodo típico



  - K_i son los valores de la clave de búsqueda
  - P_i son
    - punteros a hijos (para nodos que no son hojas), o
    - punteros a cajones (para nodos hoja).
- Los valores de la clave de búsqueda están ordenados:

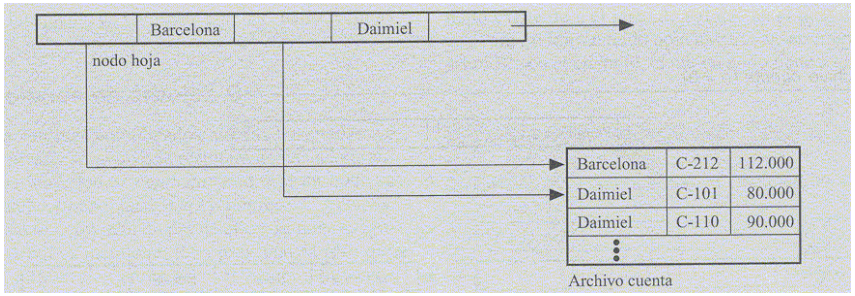
$$K_1 < K_2 < K_3 < \ldots < K_{n-1}$$

Nodos hoja en árboles B^+

Propiedades de un nodo hoja:

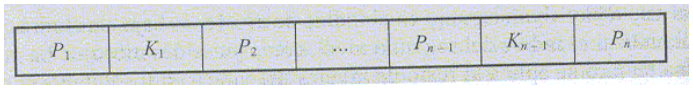
- Para i=1, 2, ..., n-1, el puntero P_i apunta o bien a un registro del archivo con valor de la clave de búsqueda K_i, o bien a un cajón de punteros, cada uno de los cuales apunta a un registro del archivo con valor de la clave de búsqueda K_i.
- Si L_i, L_j son nodos hojas y $i < j$, entonces cada valor de la clave de búsqueda en L_i es menor que cada valor de la clave en L_j.
- P_n apunta al siguiente nodo hoja en orden de la clave de búsqueda.

Nodos hoja en árboles B^+

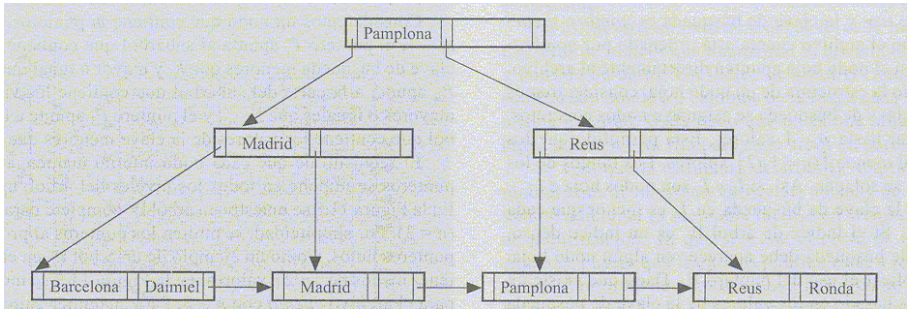


Nodos internos en árboles B+

- Los nodos internos del árbol B+ forman un índice multinivel disperso sobre los nodos hoja. Para un nodo interno con m punteros:
  - P_1 apunta a la parte del subárbol que contiene los valores de la clave de búsqueda menores que K_1
  - Para i=2, 3, ..., m-1, P_i apunta al subárbol que contiene los valores de la clave menores que K_i y mayor o igual que K_{i-1}
  - P_m apunta a la parte del subárbol que contiene los valores de la clave mayores o iguales a K_{m-1}

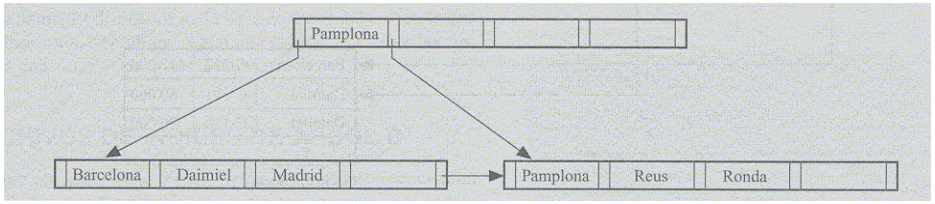


Ejemplo de árbol B+



Árbol B+ para el archivo cuenta (n=3)

Ejemplo de árbol B+



Árbol B+ para el archivo cuenta (n=5)

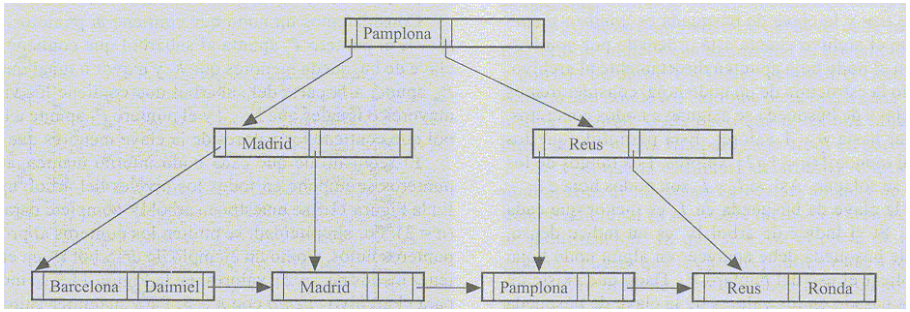
- Los nodos hoja deben tener entre 2 y 4 valores ($\lceil (n-1)/2 \rceil$ y n-1, con n=5)
- Los nodos internos distintos de la raíz deben tener entre 3 y 5 hijos ($\lceil n/2 \rceil$ y n, con n=5)
- La raíz debe tener como mínimo 2 hijos

Observaciones sobre árboles B+

- Como las conexiones entre nodos se hace a través de puntero, no hay ninguna suposición sobre que los nodos cercanos "lógicamente" lo sean "fisicamente".
- Los niveles de nodos internos forman una jerarquía de indices dispersos.
- El árbol B+ contiene un número relativamente pequeño de niveles (logarítmico en el tamaño del archivo principal), por lo que las búsquedas se pueden realizar eficientemente.
- Las insercciones y borrados también son eficientes, ya que el índice se reestructura en tiempo logarítmico.

Consultas en árboles B+ (1/2)

- Hay que seguirlo por el orden de la clave de búsqueda



Consultas en árboles B+ (2/2)

- El camino que se recorre en el procesamiento de una consulta (de la raíz a la hoja) no es mayor de $\log_{(n/2)}K$, siendo K los valores de la clave de búsqueda del archivo.
- Un nodo es generalmente del mismo tamaño que un bloque de disco, tipicamente 4kB, y n es alrededor de 100 (40 bytes por registro de índice).
- Con un millón de valores de la clave de búsqueda y n=100, tenemos que como mucho $\log_{(50)}(1.000.000)$ = 4 nodos se acceden en una búsqueda.
- Si tuvieramos un árbol binario equilibrado con un millón de valores, accederíamos a alrededor de 20 nodos para una busqueda. (acceso a E/S ≅ 30 ms)

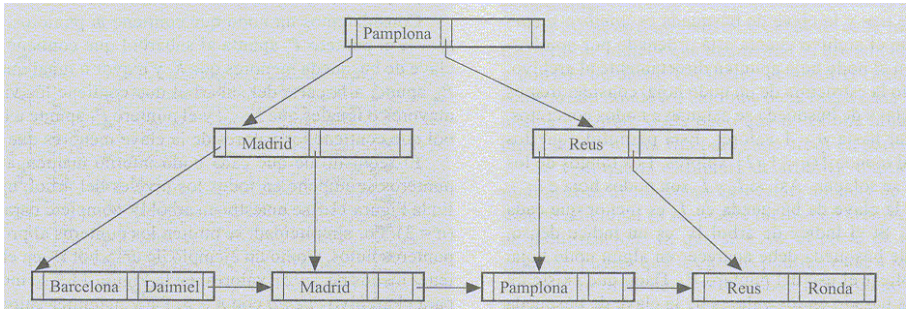
Insercción en árboles B+ (1/3)

- Encontrar el nodo hoja en donde aparece el valor de la clave de búsqueda.
- Si el valor de la clave ya está en el nodo hoja, se añade el registro al fichero.
- Si el valor de la clave no está allí, entonces añadimos el registro al archivo, y después:
  - Si hay espacio en el nodo hoja, insertamos la pareja (k,p) en el sitio adecuado.
  - Si no hay espacio, dividimos el nodo hoja e insertamos la pareja (k,p) de acuerdo a lo sgte.

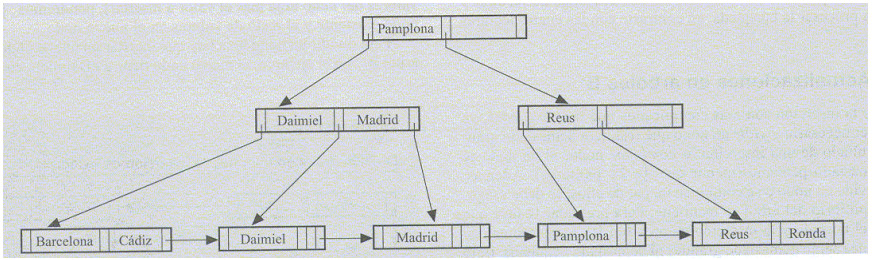
Insercción en árboles B+ (2/3)

- Divisón de un nodo:
  - cogemos los n pares (k,p) (incluído el nuevo a insertar) ordenados. Colocamos los primeros $\lceil n/2 \rceil$ en el nodo original, y el resto en un nuevo nodo.
  - Sea p el puntero al nuevo nodo, y sea k la menor clave en ese nodo. Insertamos (k,p) en el padre del nodo que estamos dividiendo. Si el padre está completo, lo dividimos y propagamos hacia arriba esa división.
- La división de nodos se haría hacia arriba hasta que se encontrara un nodo que no estuviese completo.

Inserción en árboles B+ (3/3)



Inserción de "Cádiz" en el árbol B+



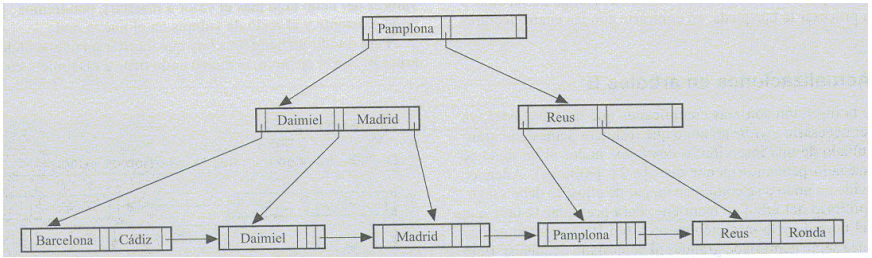
Eliminación en árboles B+ (1/2)

- Encuentra el registro a ser borrado, y elimínalo del archivo o del cajón.
- Eliminamos el par (k,p) del nodo hoja (si el cajón queda vacío).
- Si el nodo después de la elminación ha quedado con pocos elementos, y junto con la de un hermano caben en un solo nodo:
  - Insertamos todos los pares de los dos nodos en el nodo de la izquierda y eliminamos el de la derecha.
  - Eliminamos el par (k_{i-1}, P_i), donde P_i es el puntero al nodo eliminado, del nodo padre, recursivamente utilizando el procedimiento anterior.

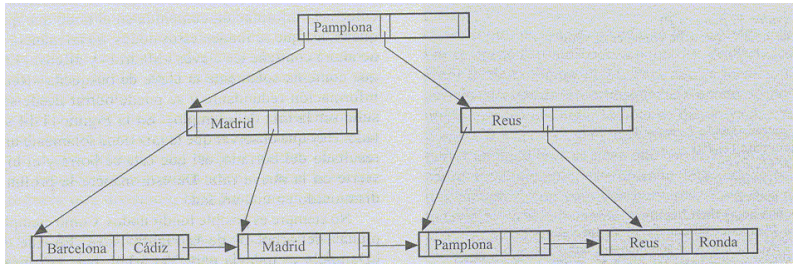
Eliminación en árboles B+ (2/2)

- En otro caso, si el nodo tiene menos elementos de los necesarios, y junto los de su hermano no caben en un sólo nodo:
  - Redistribuír los elementos entre los dos nodos tal que los dos tengan más del mínimo necesario.
  - Actualizar los correpondientes valores de la clave de búsqueda en los padres de los nodos.
- Las eliminaciones de nodos pueden propagarse hacia arriba. Si la raíz quedase sólo con un puntero, se borraría y el único hijo sería ahora la raíz.

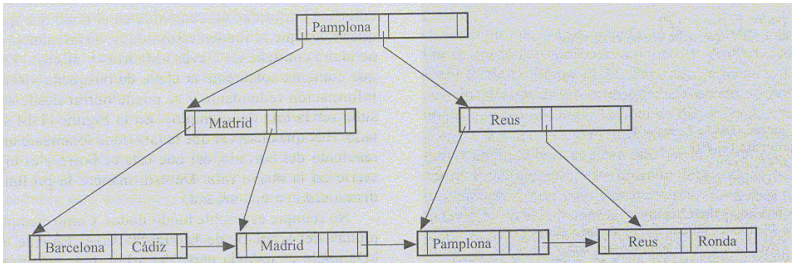
Ejemplos de borrado en árboles B+ (1/3)



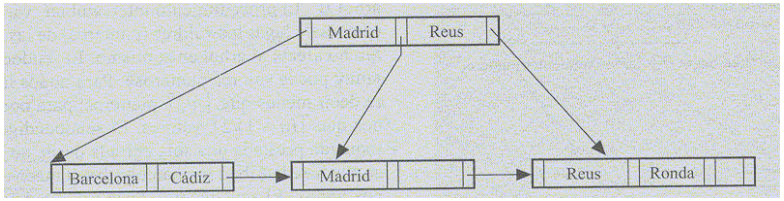
Borrado de "Daimiel"



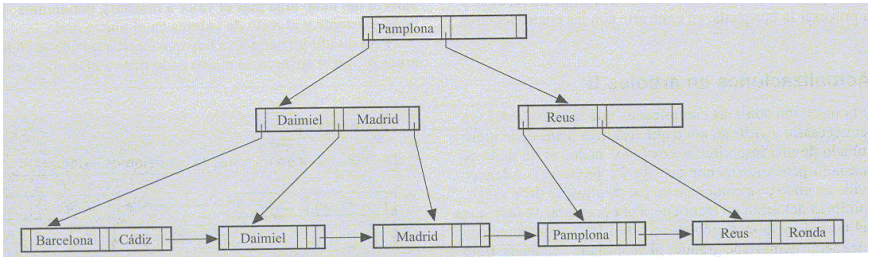
Ejemplos de borrado en árboles B+ (2/3)



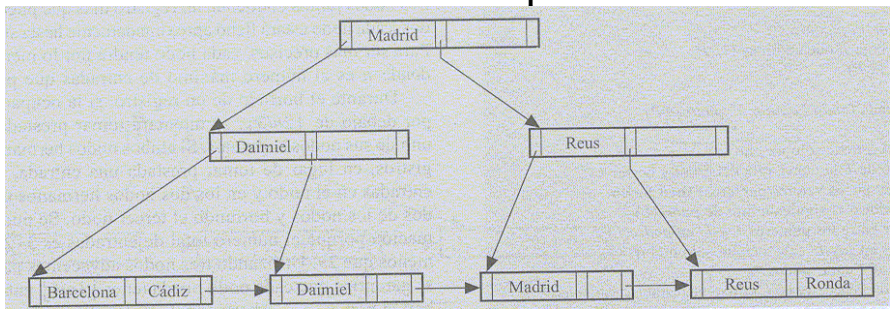
Borrado de “Pamplona”



Ejemplos de borrado en árboles B+ (3/3)



Borrado de “Pamplona”

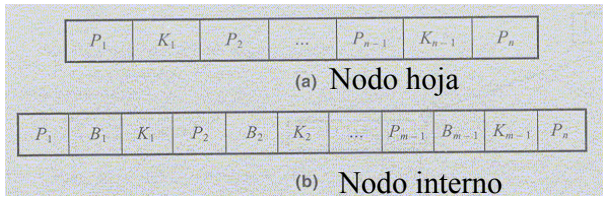


Organización de archivos con árboles B+

- La degradación de los archivos indexados se resuelve mediante los índices en árbol B+
- La degradación de los archivos de datos se resuelve utilizando una organización de archivos en árbol B+
- Los nodos hojas guardan registros, en vez de punteros.
- Puesto que los registros son más grandes que los punteros, el número máximo de registros que se pueden guardar en un nodo hoja es menor que el número de punteros en un nodo interno.
- Los nodos hojas siguen manteniendose medio llenos
- La insercción y borrado se manejan de la misma forma que en los índices de árbol B+

Archivos de Indices de árbol B (1/2)

- Son similares a los árboles B+: los árboles B sólo permiten una única aparición de las claves de búsqueda, eliminando la redundancia en su almacenamiento.
- Las claves en un nodo interno no vuelven a aparecer en el árbol B, por lo que necesitamos incluír un puntero adicional.



Archivos de Indices de árbol B (2/2)

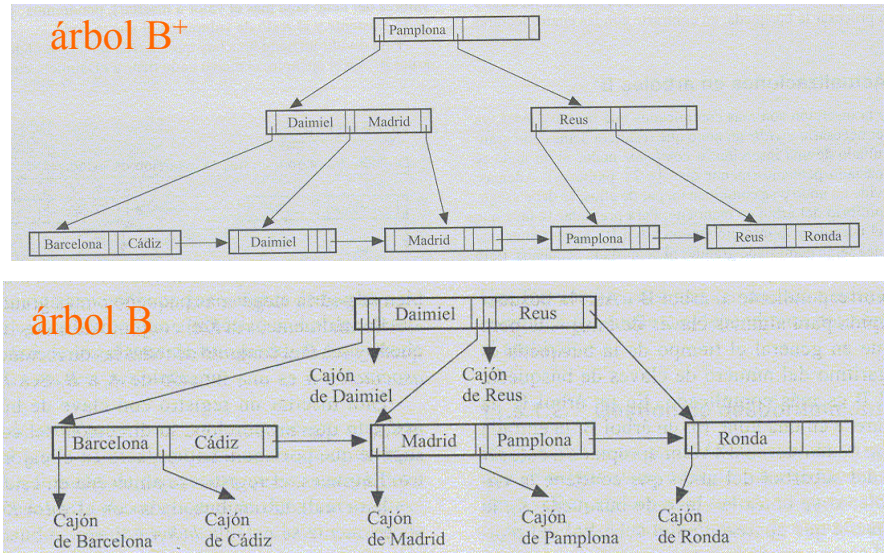
Ventajas:

- Utiliza menos nodos que un árbol B^+
- Algunas veces se encuentran los valores antes de alcanzar los nodos hojas

Desventajas:

- Sólo una pequeña parte de las claves se encuentran antes.
- Al ser los nodos internos más grandes, disminuye su grado de salida, por lo que el árbol ha de ser más profundo que su correspondiente B^+
- La insercción y borrado son más complicados que B^+
- La implementación es más complicada que B^+

Ejemplo de árbol B



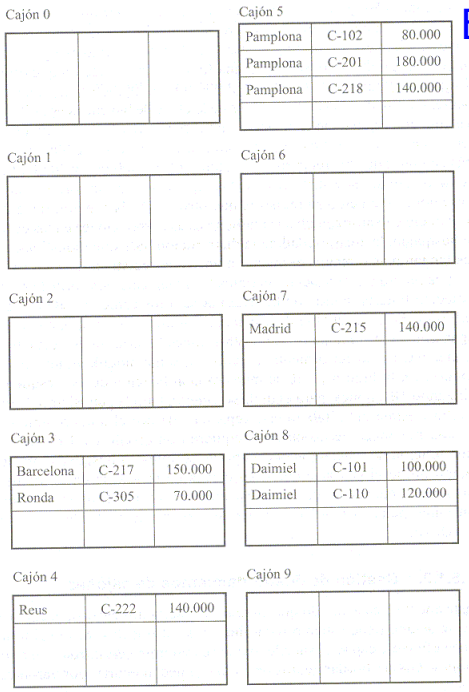
Asociación estática

- Un cajón (bucket) es una unidad de almacenamiento que contiene uno o más registros.
- En un archivo organizado por asociación, el cajón de un registro se obtiene directamente de su clave de búsqueda utilzando la función de asociación.
- La función de asociación h es una función desde el conj. K de todos los valores posibles de clave de búsqueda al conj. B de direcciones de cajones.
- La función de asociación se utiliza para localizar registros para acceso, insercción y borrado.
- Registros con distinta clave de búsqueda pueden estar en el mismo cajón; por lo que habrá que buscar secuencialmente todo el cajón para localizar el registro.

Funciones de Asociación

- La peor función de asociación asigna todas las claves de búsqueda al mismo cajón; esto haría el tiempo de acceso proporcional al número de claves en el fichero.
- Una función de asociación ideal es uniforme, i.e. se asigna a cada cajón el mismo número de claves de búsqueda.
- También es aleatoria, y así cada cajón tendrá el mismo número de registros asignados independientemente de la distribución actual de claves de búsqueda.
- Las funciones típicas de asociación realizan el cálculo sobre la representación binaria interna de la clave de búsqueda.

Ejemplo de Organización Asociativa (1/2)



Ejemplo de Organización Asociativa (2/2)

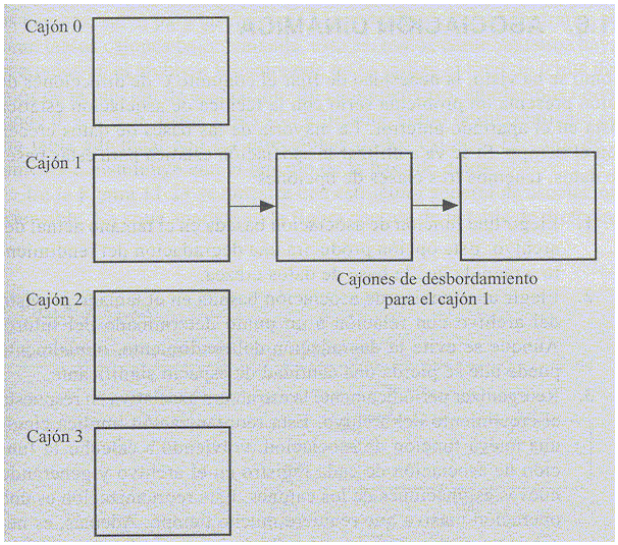
Organización asociativa del archivo cuenta, utilizando nombre-sucursal como clave.

- Hay 10 cajones
- Representamos la letra *i*-ésima por el entero *i*
- La función de asociación devuelve la suma de la representación de las letras *módulo* 10.

Manejo del desbordamiento de cajones

- El desbordamiento de los cajones puede ocurrir a causa de:
  - Cajones insuficientes
  - Atasco en la distribución de cajones:
    - Varios registros tienen la misma clave de búsqueda
    - La función de asociación elegida puede producir una distribucción irregular de las claves de búsqueda
- La probabilidad de desbordamiento se puede reducir, pero no eliminar: cajones de desbordamiento.

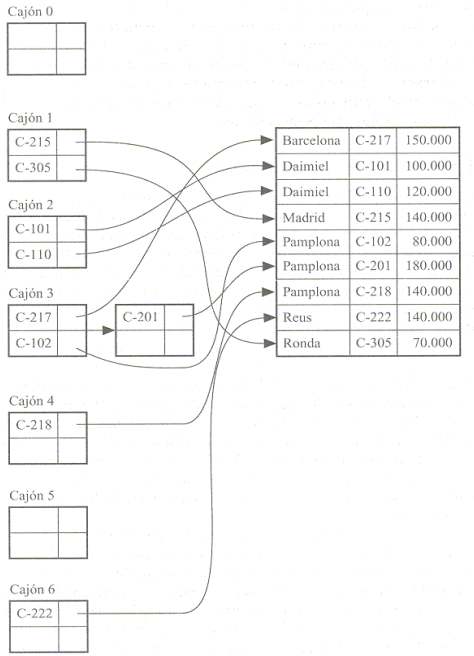
Ejemplo de cajones de desbordamiento



Indices asociativos

- La asociación se puede emplear también para estructuras de índices.
- Los índices asociativos organizan las claves junto a sus punteros en un fichero asociativo.
- Los índices asociativos son siempre secundarios

Ejemplo de Indice asociativo



Indice asociativo de la clave de búsqueda número-cuenta del archivo cuenta.