TEMA 5: NIVEL INTERNO

FUNDAMENTOS DE BASES DE DATOS

Curso 2015/2016

ORGANIZACIÓN Y METODOS DE ACCESO: SECUENCIAL

Curso 2015/2016

OBJETIVO

- Minimizar el número de accesos a disco

- Trabajaremos a dos niveles:
  - Organización de registros de datos a nivel de almacenamiento
  - Adición de estructuras complementarias para acelerar el acceso a dichos registros

MÉTODOS DE ORG. Y ACCESOS A LOS DATOS

- Asumimos que:
  - los datos no caben en memoria principal
  - varios usuarios
  - un procesador
  - un controlador
  - un disco

CRITERIOS DE EVALUACIÓN

- Memoria total ocupada en relación a ala que realmente necesitan los datos

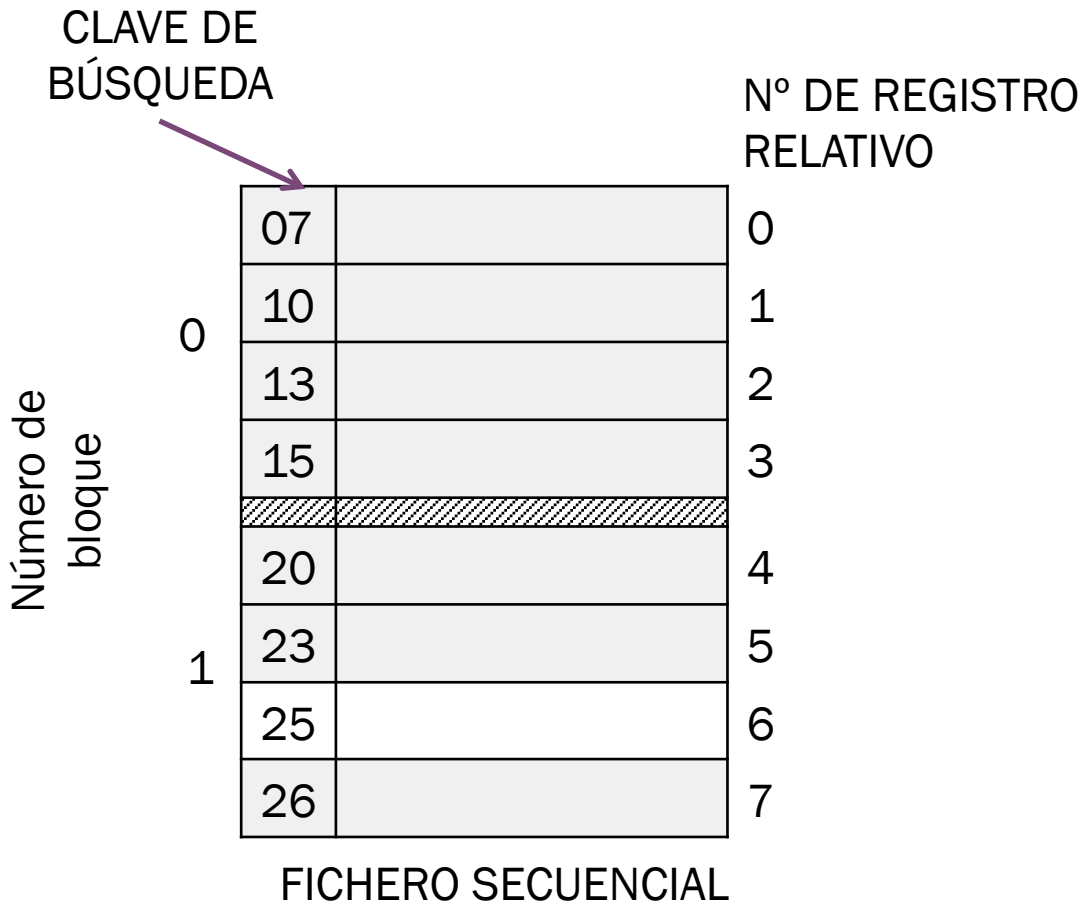
- Tiempo de acceso a los datos, medido en número de registros accedidos bajo diferentes condiciones:
  - Por clave de búsqueda (C1)
  - Por intervalo en la clave de búsqueda (C2)

- **Número de accesos a disco en las operaciones de inserción y de borrado de datos**

ORGANIZACIÓN SECUENCIAL

- Un fichero de acceso secuencial es aquél donde los registros están almacenados consecutivamente en un dispositivo de almacenamiento
- Para acceder a un registro determinado debemos pasar obligatoriamente por los registros que le preceden
- Los registros están ordenados por una clave (**clave física**)

ORGANIZACIÓN SECUENCIAL



ORGANIZACIÓN SECUENCIAL

- Ejemplo: Mostrar la relación completa de departamentos
- La consulta se resolvería rápidamente si los departamentos están almacenados conjuntamente en bloques contiguos de un fichero
- Pero… ¿qué pasa si quereos plantear consultas por valor de clave o por rango de valores?

ORGANIZACIÓN SECUENCIAL

- El primer caso implica:
  - Recorrer uno tras otro cada uno de los registros
  - En el peor caso (no encontrarse o ser el último) la búsqueda sería de O(N)

- El segundo caso implica:
  - Realizar la búsqueda por valor de clave de la cota inferior del intervalo y continuar hasta alcanzar la cota superior

ORGANIZACIÓN SECUENCIAL

- Inserción de un nuevo registro:
  - Buscar el bloque que le corresponde y…
    - Si hay sitio, se inserta el nuevo registro
    - En caso contrario, o bien se opta por crear un nuevo bloque o bien se crea un bloque de desbordamiento
  - Es recomendable dejar espacio vacío en los bloques para evitar los problemas de reorganización

ORGANIZACIÓN SECUENCIAL

- Operación de borrado de un registro:
  - Buscar el registro
  - Puede implicar una reorganización local de los registros de un bloque

ORGANIZACIÓN SECUENCIAL

- Esta forma de organizar los registros no está exenta de inconvenientes, pero puede subsanarse mediante el uso de estructuras adicionales que nos permitan:
  - Acelerar la localización de los datos
  - Disminuir el número de bloques de disco transferidos

- Entre las técnicas más populares se encuentra:
  - Índices
  - Acceso directo

INDEXACIÓN

- Tiene por objetivo disminuir el tiempo de acceso a los datos por una clave de búsqueda
- Es similar a la idea de un índice en un libro

INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

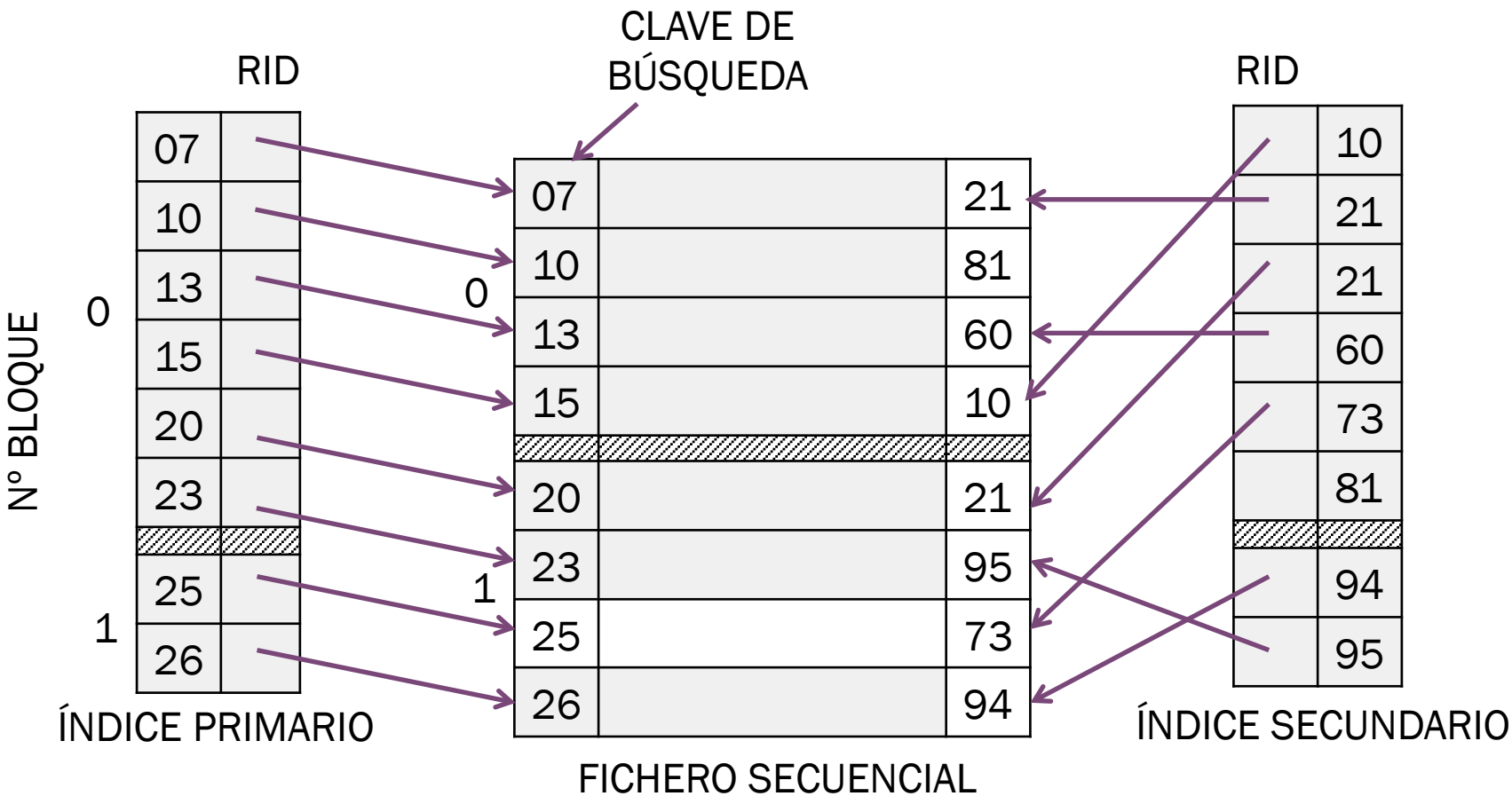
- Partimos de un fichero secuencial
- Disponemos de una estructura adicional: ***fichero índice:***
  - campo clave: la clave de búsqueda
  - campo de referencia: RIDs de registos
- Son más pequeños que los del fichero de datos, aunque el número de ellos es el mismo en ambos ficheros

INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

- Índice primario
  - la clave de búsqueda es el mismo campo clave por el que se está ordenando el fichero de datos

- Índices secundarios
  - construidos sobre otros campos que no sean la clave física del fichero de datos

ÍNDICE SECUNDARIO



INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

- Consulta por un valor de la clave
  - Sobre el índice localizamos la clave (recorrido secuencial)
  - Obtenemos el RID del registro requerido
  - Vamos a disco para recuperar el bloque de datos donde se encuentra el registro señalado por el RID
  - *La búsqueda en el índice es más rápida*

INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

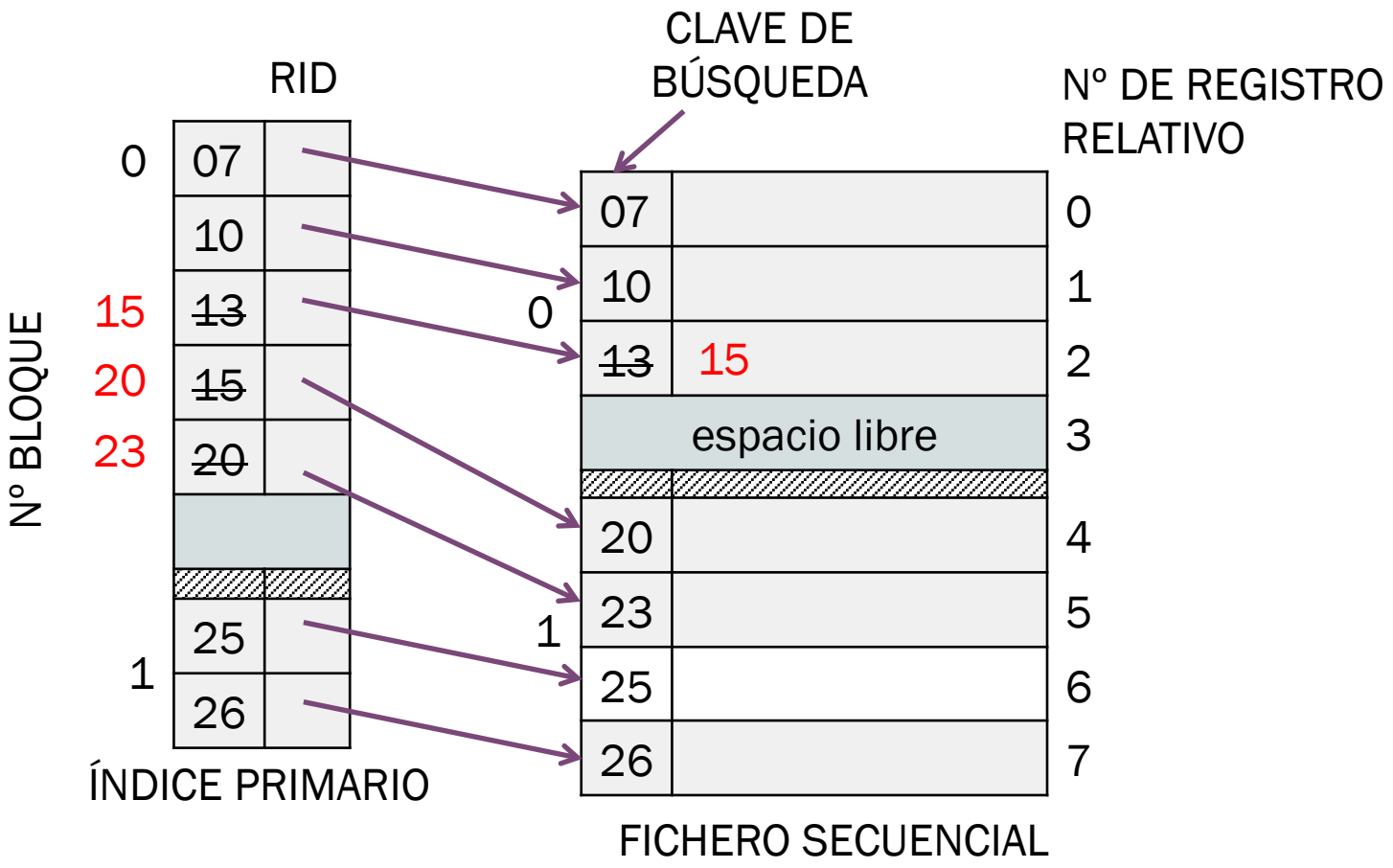
- Consulta por rango de valores
  - Búsqueda en el índice por valor de clave de la cota inferior
  - Recorrido de las entradas del índice que están en el intervalo, recuperando los registros correspondientes gracias a su RID

INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

- Inserción de un nuevo registro
  - Las mismas operaciones que en el fichero secuencias
  - Hay que actualizar también el índice

- Borrado de un nuevo registro
  - Borrado de un registro en el fichero de datos
  - Borrado de la entrada en el índice

EJEMPLO DE BORRADO (CLAVE=13)



INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

- Se puede montar un índice sobre más de un campo de un registro
  - La clave del índice será la concatenación de los campos indicados

- Pero… cuidado con… (ejemplo)
  - Un índice sobre nombre y DNI
  - Es útil para consultas que involucran el nombre o bien el nombre y el DNI
  - No es útil para consultas sobre el DNI

INDEXACIÓN: FICHERO SECUENCIAL INDEXADO

- Los índices:
  - Aceleran el acceso a los datos
  - Ralentizan las otras operaciones ya que hay que mantener el índice actualizado
- Hay que considerar la conveniencia de crear cada índice:
  - Frecuencia de las consultas
  - Frecuencia de las operaciones de mantenimiento de los datos

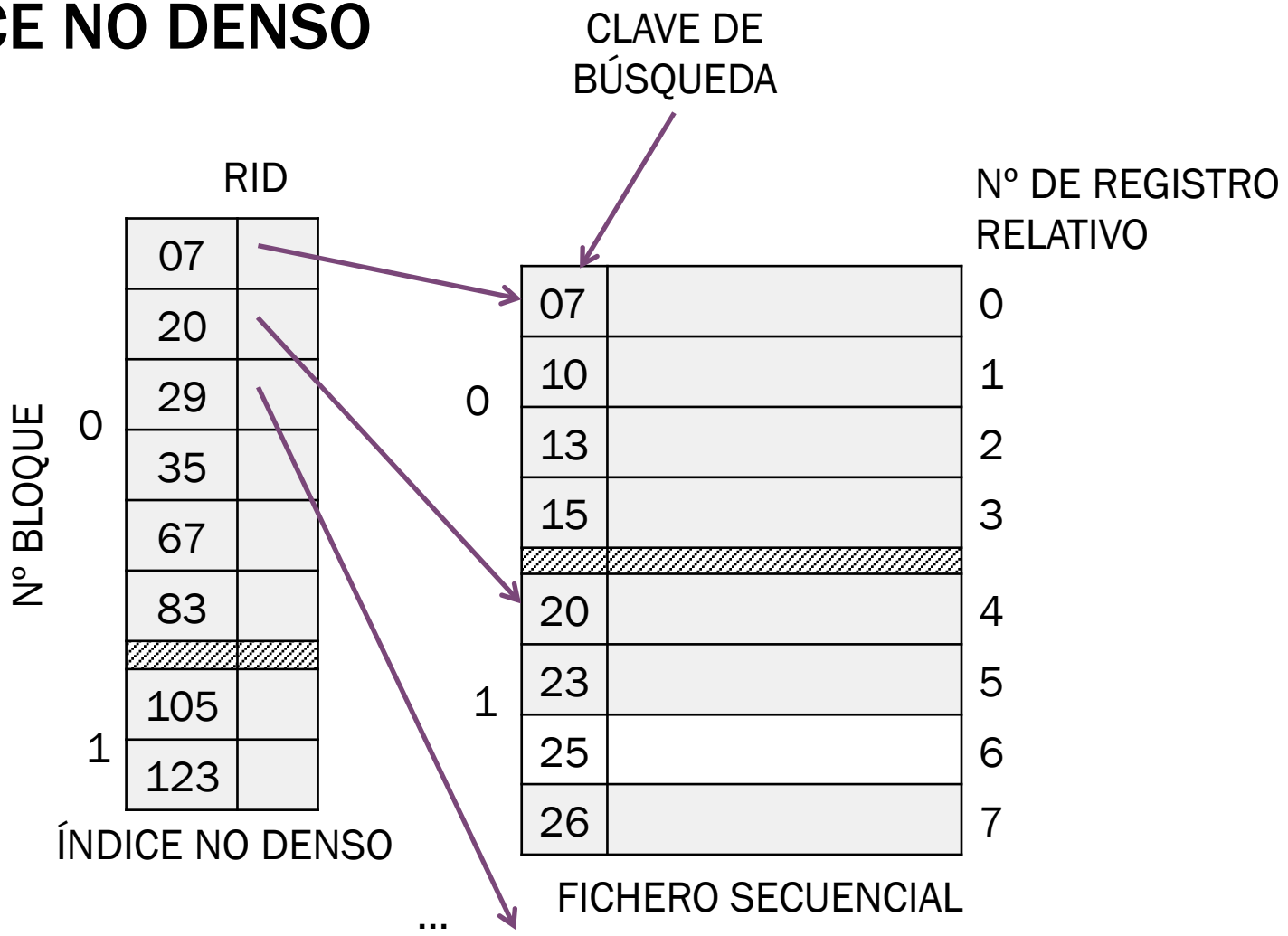
INDEXACIÓN

- Índice denso
  - Mismo número de registros que en el fichero de datos que indexa
  - Ideal: mantener el índice en memoria principal
  - Realidad: los índices siguen siendo muy grandes

INDEXACIÓN

- Índice no denso
  - Registros compuestos por la clave de búsqueda y la dirección de comienzo del bloque donde puede encontrarse el registro consultado por valor de clave
  - El número de registros en el índice se reduce al número de bloques del fichero de datos
  - El acceso secuencial al índice no denso se acelera

ÍNDICE NO DENSO



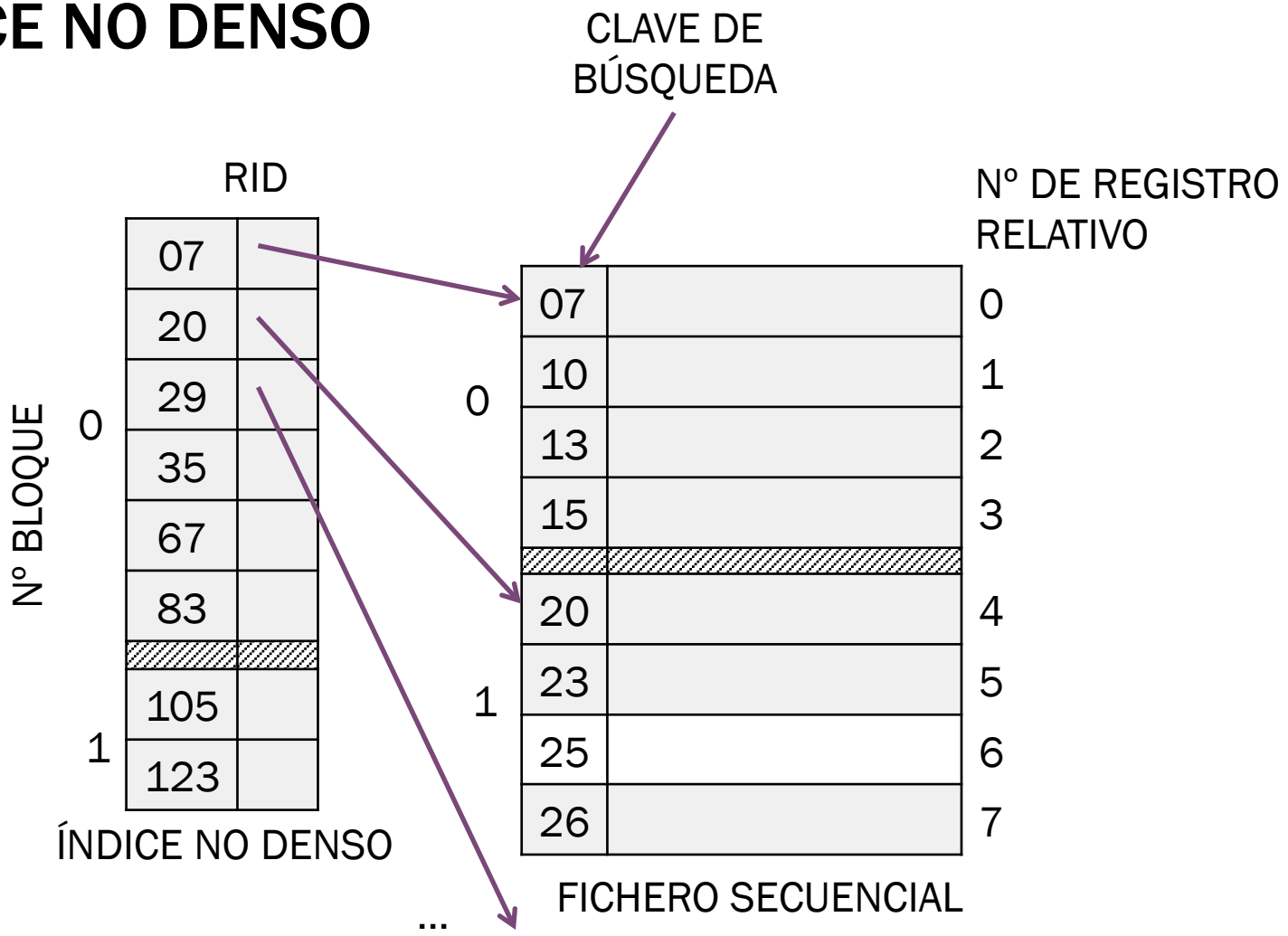
ÍNDICE NO DENSO

- Diferencias en el proceso de búsqueda
  - Una vez encontrado el bloque donde podría encontrarse el registro:
    - Hay que cargarlo en memoria
    - Hay que hacer una búsqueda secuencial
    - No tiene costes en términos de acceso a disco
- No se tiene garantía alguna de encontrar el registro deseado hasta consultar el bloque de datos leído

ÍNDICE NO DENSO

- Los índices no densos sólo se pueden definir sobre la clave física
- El mantenimiento de un índice no denso es menos costoso:
  - Inserción y borrado menos frecuentes
  - Solo ocurren cuando la operación afecta al valor representativo del bloque

ÍNDICE NO DENSO



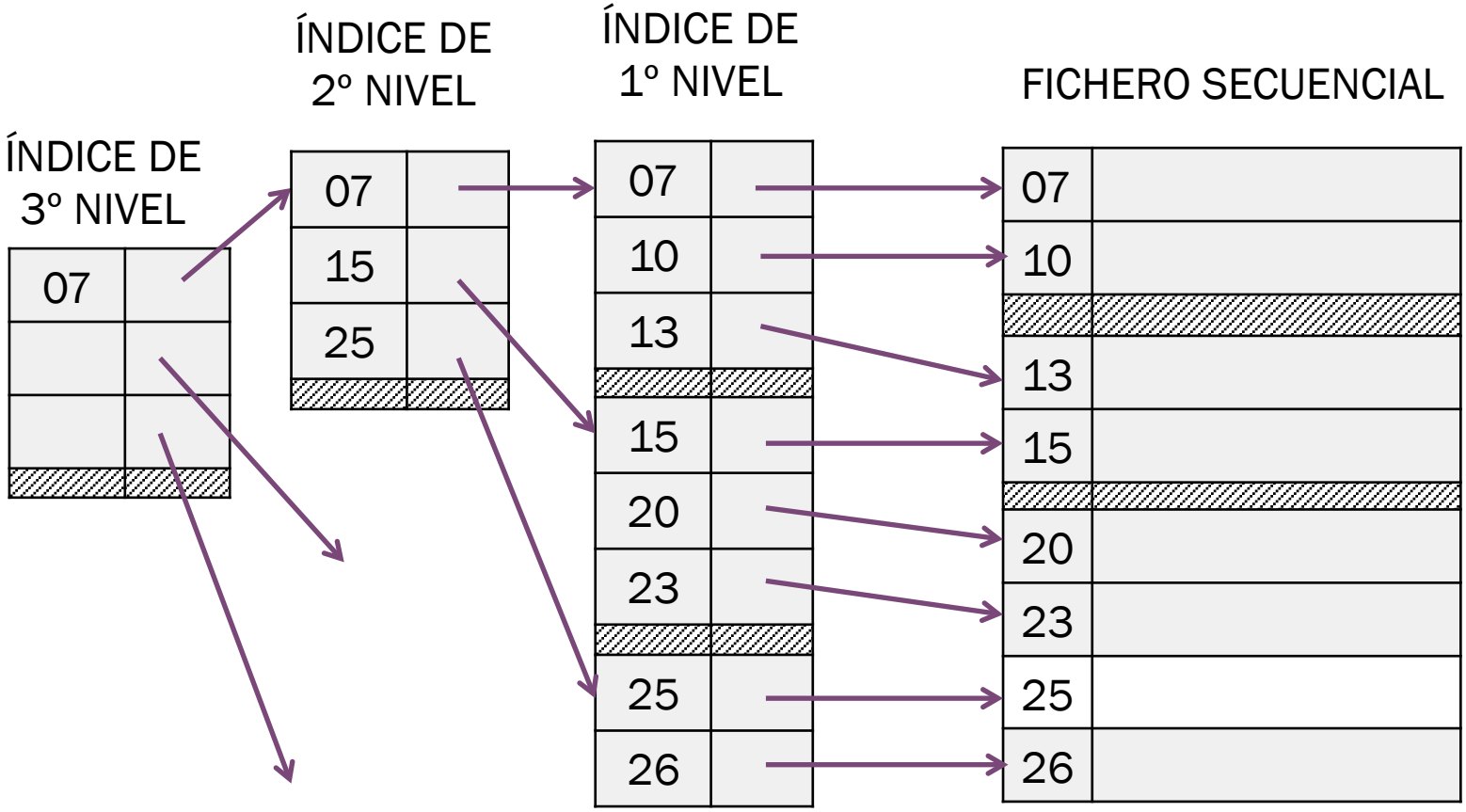
ÍNDICES JERÁRQUICOS

- Volvemos al objetivo de disminuir el tiempo necesario para recorrer el índice en busca de un registro
  - Idea: crear índices sobre índices
  - Varios niveles en el acceso a los datos
- Un índice multinivel está formado por:
  - Un índice de primer nivel sobre le fichero de datos (puede ser denso o no dependiendo de la clave)
  - Otros índices, no densos, construidos sucesivamente unos sobre otros

ÍNDICES JERÁRQUICOS

- El tamaño de los bloques se establece con la idea de optimizar cada una de las operaciones de acceso el disco físico
- Se reduce el número de accesos a disco para localizar un registro (en el peor caso tantos como niveles)
- Se complica el mantenimiento del índice

ÍNDICE JERÁRQUICO O MULTINIVEL



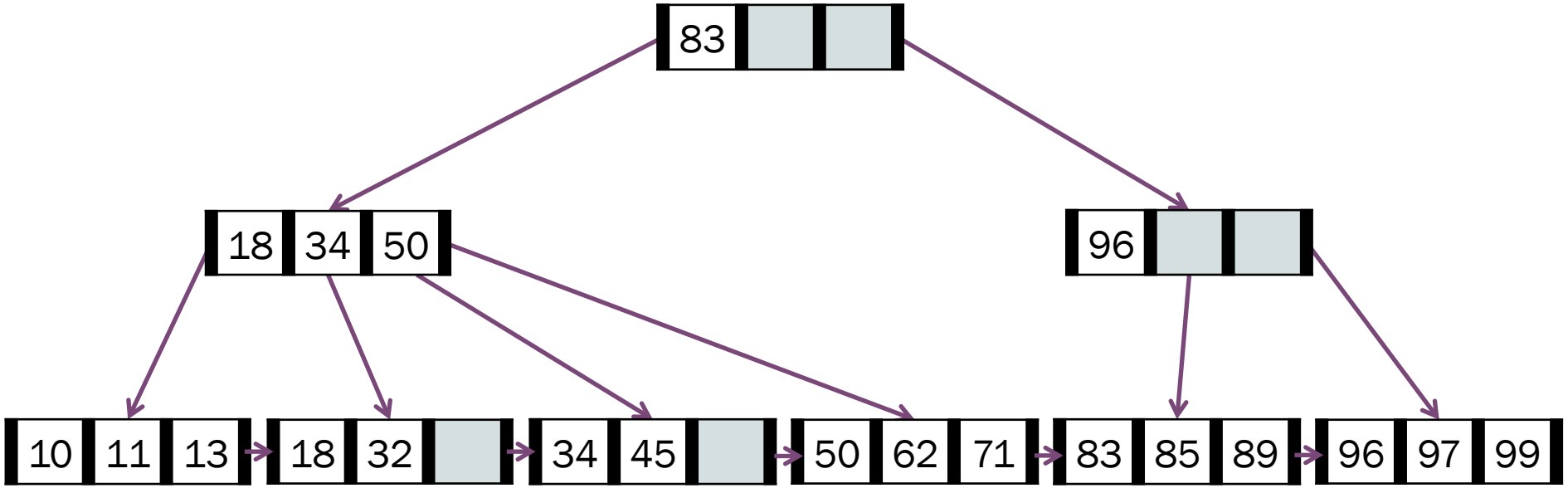
INDEXACIÓN POR ARBOLES-B

- Los árboles-B (B de balanceado) pueden verse como una generalización de un índice multinivel
- Proporcionan un mecanismo ideal para construir índices equilibrados

INDEXACIÓN POR ARBOLES-B

- Estructura:
  - Bloque de nivel superior: raíz del árbol
  - Bloque del índice de nivel inferior: hojas
  - Cada nodo del árbol coincide con un bloque de memoria
  - Las claves contenidas en cada nodo nos guiarán hasta el siguiente bloque (nodo) del nivel inmediatamente inferior
  - Orden n (como mucho n-1 valores de clave en el nodo y n punteros p_i que apuntan a un nodo hijo)

INDEXACIÓN POR ARBOLES-B



INDEXACIÓN POR ARBOLES-B

- Restricciones dentro de los nodos
  - Los valores de clave C_i están ordenados dentro del nodo
  - Los nodos están rellenos como mínimo hasta la mitad
    $n/2 \leq i = \text{número de punteros} \leq n$
  - Los valores x del subárbol apuntado por P_i cumplen:
    - $C_{i-1} \leq x < C_i$
    - Excepto para:
      - i = 1, donde $x < C_1$
      - i = m, donde $x \geq C_m$

INDEXACIÓN POR ARBOLES-B

- Los nodos hoja tienen una estructura diferente
  - Parejas (clave, RID)
  - Punteros al siguiente nodo hoja
- La lista concatenada de nodos hoja tienen gran utilidad a la hora de hacer consulta por intervalos

INDEXACIÓN POR ARBOLES-B

- Restricciones dentro de los nodos hoja
  - Los nodos están ordenadas por clave
  - Todas las claves ha de ser menores que las del siguiente nodo hoja
  - Los nodos han de estar como mínimo rellenos hasta la mitad
  - Todos los nodos hoja se encuentran en el mismo nivel
    - Árbol equilibrado
    - Todos los caminos desde la raía a un nodo hoja tienen la misma longitud

INDEXACIÓN POR ARBOLES-B

- Proceso de consulta (clave)
  - Navegamos desde la raíz bajando niveles
  - Buscamos el registro en el nodo hoja y recuperamos el registro del fichero de datos gracias al RID

- Proceso de consulta (rango)
  - Se localiza el nodo hoja que contiene el valor inferior
  - Se recorren los nodos hoja hasta alcanzar el superior, recuperando los registros pertinentes del fichero de datos

INDEXACIÓN POR ARBOLES-B

- Proceso de inserción y borrado
  - Se utilizan algoritmos que garantizan que el árbol resultante sea equilibrado