Asignatura: Bases de Datos I

TEMA 4: Nivel Físico de las Bases de Datos

Departamento de Estadística, Informática y Matemáticas

Edurne Barrenechea

Índice

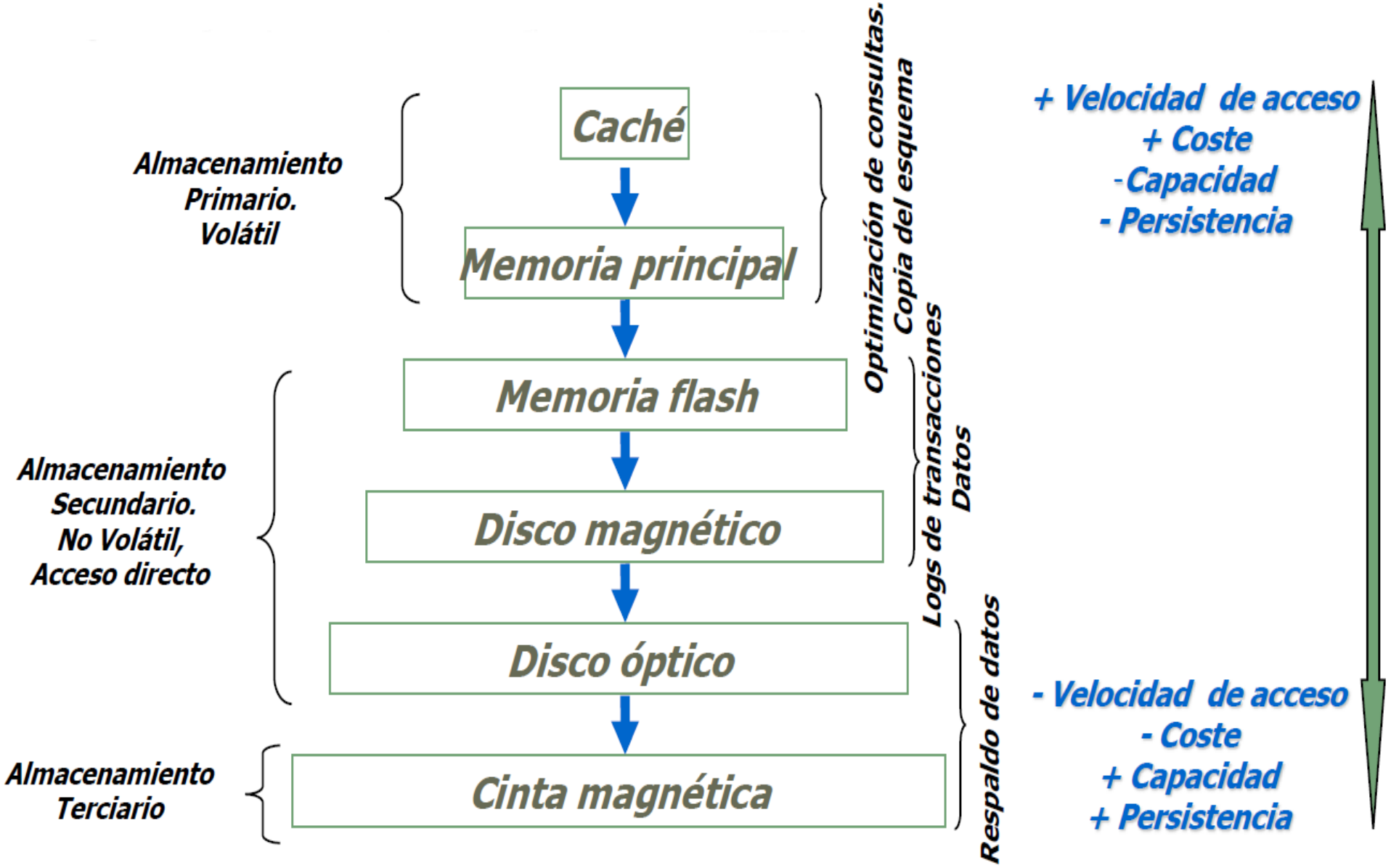
1. Introducción

- La colección de datos que constituye una BBDD debe almacenarse físicamente en un medio de almacenamiento de un ordenador.

- El SGBD puede recuperar, actualizar y procesar esos datos cuando los necesite.

- Los medios de almacenamiento forman una *Jerarquía de almacenamiento* según la **velocidad de acceso a los datos**, el **coste** y la **fiabilidad**:
  - ✓ Almacenamiento **Primario**,
  - ✓ Almacenamiento **Secundario**,
  - ✓ Almacenamiento **Terciario**.

1. Introducción: Jerarquías de memoria y dispositivos de almacenamiento



1. Introducción: Jerarquías de memoria y dispositivos de almacenamiento

Primario

- *Cache:* **es la memoria más rápida.**
- *Memoria principal:*
  - **Es rápida pero de tamaño pequeño para una BBDD.**
  - **No es segura ante fallos eléctricos.**

Secundario

- *Memoria "flash" o estado sólido:* Discos **SSD**, pendrives, tarjetas de memoria, etc.
  - **Es rápida y segura ante fallos.**
  - **Puede ser útil para pequeños volúmenes de datos**
  - **No volátiles.**

- *Discos magnéticos:* Discos duros (**HDD**), cintas magnéticas, diskette
  - **Es de acceso directo.**
  - **Soporta fallos, es totalmente fiable.**
  - **Se guarda en disco toda la BBDD.**

- *Almacenamiento óptico (CDROM):* CDs, DVDs, Blu-Ray, etc.
  - **Son grabables una vez y legibles múltiples veces.**
  - **Se utilizan en BBDD no volátiles (bibliográficas, documentales etc..).**

Terciario

- *Cinta magnética:*
  - **Es barata y de gran tamaño; pero de acceso secuencial.**
  - **Se utiliza para copias de seguridad.**

1. Introducción: Jerarquías de memoria y dispositivos de almacenamiento

HDD vs SSD: principales diferencias

Características	SSD (Solid-State Drive)	HDD (Hard Disk Drive)
Capacidad	En general entre 256 GB y 4 TB	**En general entre 1 y 10 TB**
Consumo	Menor consumo	Mayor consumo
Coste	**Más caros (1TB >= 200€)**	**Más económicos (1TB <= 50€)**
Ruido	Más silencioso por no tener partes móviles	Algo más ruidoso por tener partes móviles
Vibraciones	No vibra por no tener partes móviles	El giro de sus discos puede provocar leves vibracioness
Fragmentación	No tiene	Puede darse
Durabilidad	**Sus celdas pueden reescribirse un número limitado de veces**	Con partes mecánicas que pueden dañarse con movimientos
Tiempo de arranque de SO	7 segundos	16 segundos
Transferencia de datos	**En general, entre 200 y 550 MB/s**	En general entre 50 y 150 MB/s
Afectado por el magnetismo	No	El magnetismo puede eliminar datos

https://www.xataka.com/basics/hdd-vs-ssd
http://www.eldiario.es/cv/amigoinformatico/Disco-duro-HDD-SSD_6_601999796.html

1. Introducción: almacenamiento de bases de datos

- Los datos de una BBDD están almacenados en un **medio de almacenamiento** en el ordenador:
  - Almacenamiento secundario de **Disco magnético.**
- **Técnicas de almacenamiento en disco:**
  - Formas de **organizar ficheros de datos** en el disco para **acceder** a BBDD con un **rendimiento aceptable.**
- Cada técnica tiene ventajas/inconvenientes, conocidas por
  - Diseñadores y ABDs
  - Implementadores de SGBD
- Una aplicación, en cierto momento, sólo necesita **acceder a una parte de la BBDD**. Deberá...
  1. Localizarla en disco,
  2. Copiarla a la memoria principal,
  3. Procesarla,
  4. Reescribirla en disco (si se actualiza).

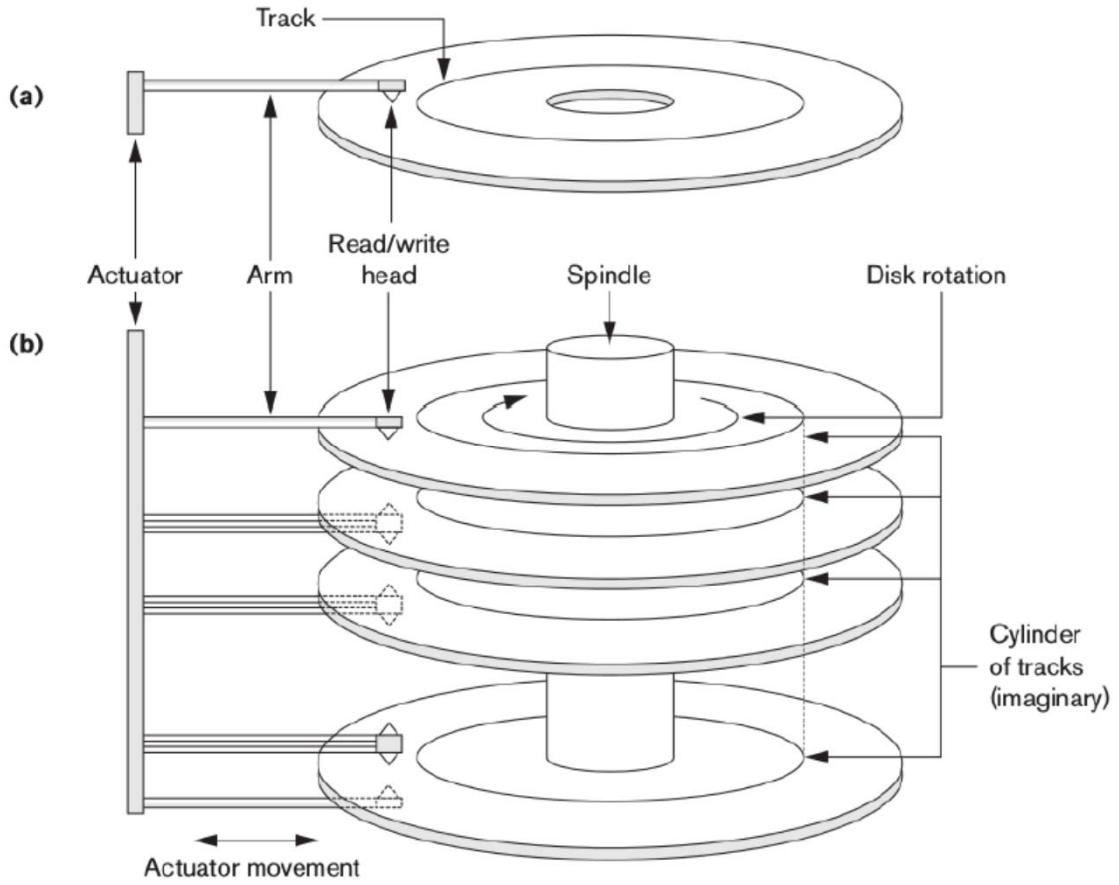
Índice

2. Dispositivos de almacenamiento secundario

- Almacenan gran cantidad de datos (12TB donde 1TB = 1 billón de bytes , ... en continuo aumento el tamaño y reducción de precio).
- **Discos duros (Hard Drive Disk)**: conjunto de discos magnéticos con un eje de rotación común. El paquete de discos **gira continuamente** a una velocidad que suele estar entre 5.400 y 7.200 rpm (10.000 rpm)
- Cada **cara** (**superficie**) del disco una **cabeza lectura/escritura** con movimiento transversal.
- La cara se divide en **Pistas** (2 a 1500). Las pistas de las distintas superficies que tienen el mismo diámetro se llaman **Cilindros**.
- Las pistas están divididas en **Sectores**.
- **Bloque**: secuencia continua de sectores que se trasladan "de una vez" a la memoria para ser procesados.
  - Los **bloques** (**páginas**) son divisiones creadas por el SO al formatear el disco (no cambia dinámicamente).
  - Todos los bloques del disco tienen el mismo tamaño (entre 512 y 4.096 bytes).
  - Los bloques están separados por **espacios interbloque** con información de control (grabada en el formateo).
  - ***Dirección de hardware de un bloque: cilindro + superficie + nº bloque (dentro de la pista).***

2. Dispositivos de almacenamiento secundario

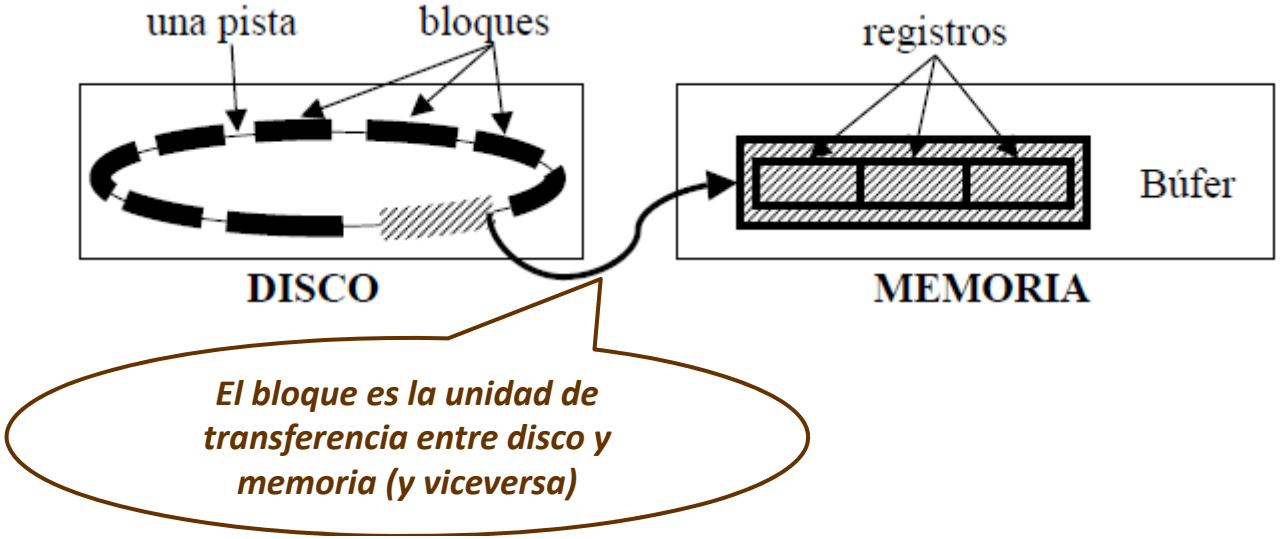
Figura 13.1 del libro Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe



(a) Un disco de una cara con hardware de lectura/escritura,
(b) Un paquete de discos con hardware de lectura/escritura.

2. Dispositivos de almacenamiento secundario

- La transferencia de información entre memoria principal y disco tiene lugar en **unidades de bloque**.
- Cuando se produce una orden de lectura, se copia uno o varios bloques en el llamado ***buffer de la memoria*** (área reservada de la memoria principal), si se necesita efectuar una escritura se copia el bloque correspondiente al bloque del disco.
- Los discos son dispositivos de **acceso aleatorio:** se accede a cualquier bloque de información solamente conociendo su dirección física en el disco, sin necesidad de recorrerlos todos.



2. Dispositivos de almacenamiento secundario

Tiempo de acceso a los datos del disco

- **Tiempo de búsqueda *(tb)***
  - Para colocar la cabeza sobre la pista (mecánico)
  - Depende de la distancia a recorrer
  - Es el principal retardo: 6-12 mseg
- **Retardo rotacional *(rr)*** (o latencia)
  - Esperar a que el principio del bloque requerido pase bajo la cabeza
  - Caso mejor: acceso inmediato
  - Caso peor: una vuelta entera
- **Tiempo de transferencia de bloque *(ttb)***
  - Depende del tamaño del bloque, de las pistas y rpm (125-250 Kb / seg, 1-5 Mb / seg)
- **Ejemplos**: o Tiempo medio para encontrar y transferir **1 bloque**:

  (***tb*** + ***rr*** +***ttb***) mseg

  - Al transferir **varios bloques** del **mismo cilindro** se reduce el tiempo:
    - ✓ **k** bloques **NO** contiguos:

      tb + **k** * (***rr*** + ***ttb***) mseg.
    - ✓ **k** bloques contiguos:

      tb + ***rr*** + **k** * ***ttb*** mseg.
- Las organizaciones de ficheros tratan de **minimizar el nº de transferencias** de disco a memoria ya que el **Tº medio es bastante alto** en comparación con lo que tarda la CPU en procesar sus registros (cuando están en el búfer). La localización de datos en disco es un **cuello de botella** importante en las aplicaciones de BBDD.

Índice

3. Ubicación de los registros de fichero en disco

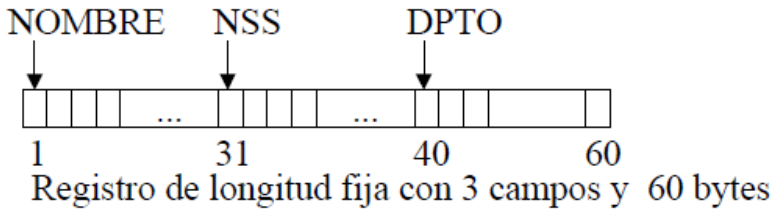
- **Datos** en disco organizados en **ficheros de registros**
- **Registro**
  - Colección de valores de datos relacionados entre sí
  - Cada valor (1 o más bytes) corresponde a un **campo**
  - Cada campo tiene asociado un **tipo de datos**
  - Definición de **Tipo de Registro**: { (nombre-campo,tipo-datos) }

```
NOMBRE DEL TIPO DE REGISTRO   NOMBRE DEL CAMPO    TIPO DE DATOS DEL CAMPO (valores que puede aceptar)

type EMPLEADO = record        NOMBRE              : packed array [1..30] of character;
                              NSS                 : packed array [1..9] of character;
                              SALARIO             : integer;
                              CÓDIGO_PUESTO       : integer;
                              DEPARTAMENTO        : packed array [1..20] of character;
                              FECHA_CONTRATO      : packed array [1..4] of character;
                      end;
```

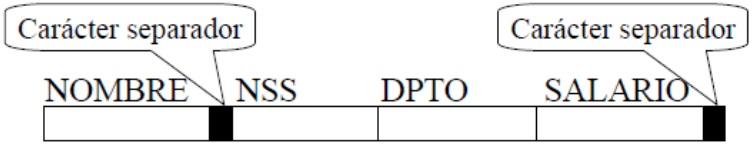
3. Ubicación de los registros de fichero en disco

- **Fichero**: secuencia de registros
- **Tipos de Registros**:
  - **Longitud fija**: todos los registros del fichero tienen exactamente el mismo tamaño (en bytes).



Registro de longitud fija con 3 campos y 60 bytes

  - **Longitud variable**: los registros del fichero pueden tener tamaños diferentes.
    - Todos los registros del mismo tipo pero uno o más campos:
      - ✓ son de tamaño variable. ***Campos de longitud variable***.
      - ✓ pueden tener valores en algunos de los registros y en otros no. ***Campos opcionales*****.**
    - El fichero contiene registros de diferentes tipos y por tanto tamaño variable (fichero mixto).



Registro con 2 campos de longitud fija (NSS y DPTO)
y 2 de longitud variable(NOMBRE y SALARIO)

Ambas figuras son la Figura 13.5 libro Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe

3. Ubicación de los registros de fichero en disco

Grabación de registros de longitud fija en bloques

Bloque = *unidad de transferencia de datos* entre el disco y la memoria

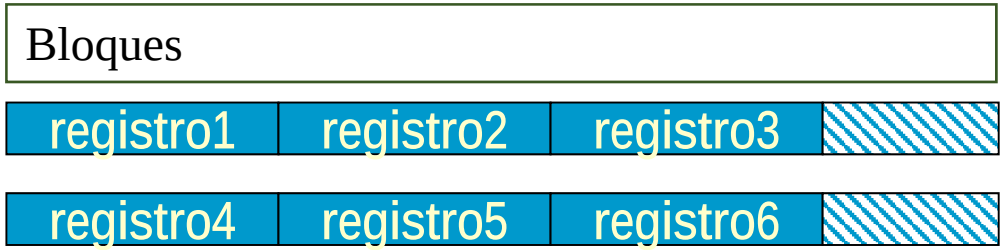
B = nº de bytes de un bloque

B>=R

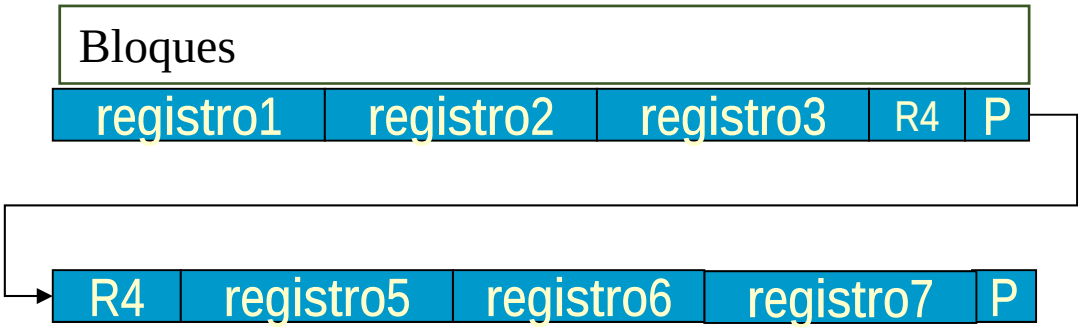
R = nº de bytes de cada registro *(longitud fija)*

B - (fbl * R) bytes libres por bloque

fbl = $\lfloor B/R \rfloor$ nº de registros por bloque *factor de bloques*



Organización no extendida



Si R>=B

Organización extendida

3. Ubicación de los registros de fichero en disco

Grabación de registros de longitud variable en bloques

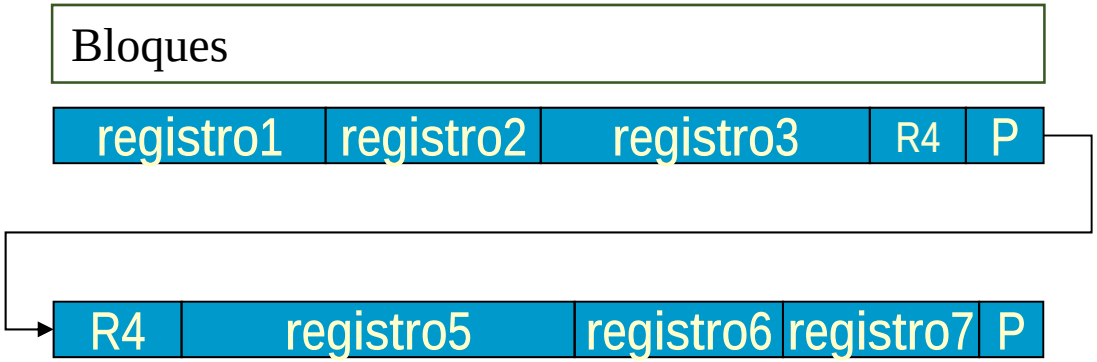
B = nº de bytes de un bloque

r = nº de registros *(longitud variable)*

fbl nº medio de registros por bloque *factor de bloques*

$b = \lceil r / fbl \rceil$ bloques

Organización no extendida



Organización extendida

3. Ubicación de los registros de fichero en disco

Borrado de registros longitud fija

registro 0	Navacerrada	C-102	8.000,00
registro 1	Collado	C-305	67.789,00
registro 2	Becerril	C-215	34.789,00
registro 3	Centro	C- 101	38.798,00
registro 4	Moralzarzal	C-222	148.987,00
registro 5	Navacerrada	C-201	348.765,00
registro 6	Galapagar	C-217	49.786,00
registro 7	Centro	C-410	51.678,00
registro 8	Navacerrada	C-218	2.698,00

registro 0	Navacerrada	C-102	8.000,00
registro 1	Collado	C-305	67.789,00
registro 2			
registro 3	Centro	C- 101	38.798,00
registro 4	Moralzarzal	C-222	148.987,00
registro 5	Navacerrada	C-201	348.765,00
registro 6	Galapagar	C-217	49.786,00
registro 7	Centro	C-410	51.678,00
registro 8	Navacerrada	C-218	2.698,00

Solución 1 desplazar registros

registro 0	Navacerrada	C-102	8.000,00
registro 1	Collado	C-305	67.789,00
registro 2	Centro	C- 101	38.798,00
registro 3	Moralzarzal	C-222	148.987,00
registro 4	Navacerrada	C-201	348.765,00
registro 5	Galapagar	C-217	49.786,00
registro 6	Centro	C-410	51.678,00
registro 7	Navacerrada	C-218	2.698,00

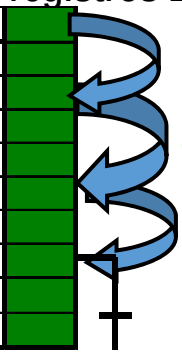
Solución 2 desplazar el último registro

registro 0	Navacerrada	C-102	8.000,00
registro 1	Collado	C-305	67.789,00
registro 2	Navacerrada	C-218	2.698,00
registro 3	Centro	C- 101	38.798,00
registro 4	Moralzarzal	C-222	148.987,00
registro 5	Navacerrada	C-201	348.765,00
registro 6	Galapagar	C-217	49.786,00
registro 7	Centro	C-410	51.678,00

Solución 3 control de registros borrados

Situación del fichero después de borrar los registros 1,4 y 6

cabecera			
registro 0	Navacerrada	C-102	8.000,00
registro 1			
registro 2	Becerril	C-215	34.789,00
registro 3	Centro	C- 101	38.798,00
registro 4			
registro 5	Navacerrada	C-201	348.765,00
registro 6			
registro 7	Centro	C-410	51.678,00
registro 8	Navacerrada	C-218	2.698,00



Lista libre : lista enlazada con los registros borrados

Cuando se produzca una inserción se "gestiona" la lista libre.

Sencillo de implementar ya que longitud registro borrado= longitud registro insertado

3. Ubicación de los registros de fichero en disco

Técnicas de asignación de los bloques de un fichero en disco

- **Contigua:** bloques consecutivos de disco. Agiliza lectura de todo el fichero pero dificulta la expansión del fichero.
- **Enlazada:** cada bloque tiene un apuntador al siguiente bloque. Facilita la expansión y dificulta la lectura.
- **Grupos de bloques:** combinación de las anteriores. Un grupo de bloques consecutivos y luego enlaza los grupos.
- **Indexada:** bloques que contienen sólo índices. Los índices son apuntadores a los bloques que realmente contienen los datos.

3. Ubicación de los registros de fichero en disco

Descriptor o cabecera de un fichero

- Contiene Información sobre el fichero que es necesaria para los programas que tienen acceso a los registros de dicho fichero:
  - Direcciones en el disco de los bloques del fichero
  - Descripciones de formato de los registros como:
    - Longitud y orden de campos en ***registros de longitud fija***
    - Códigos de tipo de campo, separadores, códigos de tipo de registro, en ***registros de longitud variable***

- **Búsqueda de registros en el disco**
  - Copia de uno (o más) bloques en los buffers de memoria principal.
  - Los programas buscan en los buffers el registro ***utilizando la cabecera***.
  - Si se desconoce dirección del bloque que contiene los registros deseados, implica búsqueda secuencial a través de los bloques.

- **Objetivo de una organización**
  - **Mínima transferencia de bloques para la localización de los registros deseados.**

Índice

4. Organización de ficheros

- ***Organizaciones de ficheros***, determinan la ***colocación*** física de los registros del fichero en el disco y por tanto, ***cómo se puede acceder*** a ellos:
  - ✓ Fichero heap (o desordenado)
  - ✓ Fichero ordenado (o secuencial)
  - ✓ Fichero hash (o disperso)
  - ✓ **Árboles B**
- ***Métodos de acceso, (sección 5)*** permite un acceso eficaz a los registros de un fichero basándose en campos alternativos a los utilizados para la organización principal del fichero, la mayoría son índices (un solo nivel, multinivel, **árboles B+).**

- Operaciones sobre archivos: open, find, read, findnext, delete, modify, insert, close…

- Hay ***métodos de acceso*** que NO se pueden aplicar a algunas ***organizaciones de fichero.***
- En otros casos es posible aplicar varios ***métodos de acceso*** a una misma ***organización de fichero***. Interesa encontrar la organización idónea en cada caso y será la que permita realizar de manera más eficiente las operaciones más frecuentes.

Índice

4.1 Ficheros *heap* (1)

Ficheros *heap* (desordenado o montón)

- Ficheros de registros NO ordenados (**secuenciales**)
- Organización más simple:
  - Registros en el orden en el que se insertan.
  - Inserciones siempre al final del fichero.

- Puede utilizar organización extendida o no extendida y registros de longitud fija o longitud variable.

Operaciones

- **Inserción**
  1. Poner en el búfer el último bloque del fichero.
  2. Añadir al búfer el registro a insertar.
  3. Reescribir el búfer en el disco.
  4. Guardar la dirección del bloque en el descriptor.
- **Búsqueda**
  - Lineal: bloque a bloque hasta que un registro cumpla la condición (costoso)
  - Sea **b** nº bloques del fichero: - **Costo medio**: acceso a **b**/2 bloques;
    - **Costo caso peor**: **b** (ningún registro cumple la condición)

4.1 Ficheros *heap* (2)

Ficheros *heap* (desordenado o montón)

- **Eliminación**
  1. Buscar el registro a borrar
  2. Transferir el bloque al búfer
  3. Eliminar el registro del búfer
  4. Reescribir el búfer en el disco

Problema: deja espacios libres en el disco (pueden ser muchos, si se borran muchos registros).
Sol1: reorganización periódica del fichero (para recuperar el espacio libre)
- Recorrer secuencialmente los bloques
- Recolocar los registros quitando los huecos

Sol2: intentar insertar sobre huecos y si no se puede insertar al final
- Cálculos extra para encontrar bloques con huecos

- **Modificación**
  1. Buscar el registro a modificar
  2. Transferir el bloque al búfer
  3. Modificar el registro en el búfer
  4. Reescribir el búfer en el disco

Problema: registros de tamaño variable
Sol:
- Eliminar el registro a modificar
- Insertar el registro modificado

4.1 Ficheros *heap* (3)

Caso especial de ficheros *heap*: ficheros relativos o directos

- **NO ordenados**
- **Registros de longitud fija**
- **Bloques NO extendidos** y de asignación continua.
- En este caso es muy sencillo tener acceso a cualquier registro por la **posición** que ocupa en el fichero: sean ***r*** registros con factor de bloque ***fbl***
  - Los registros se numeran como ***0, 1, 2, 3, ..., r-1***
  - Los registros en cada bloque se numeran como ***0, 1, ..., fbl-1***
- Así el registro *i*-ésimo estará en:
  - Bloque $\lfloor i/fbl \rfloor + 1$
  - Registro ***(i mod fbl)*** del bloque *(mod es el resto de la división)*

- **NO** ayudan a localizar un registro según una condición de búsqueda pero facilita la construcción de rutas de acceso en el fichero, como los índices que estudiaremos posteriormente.

Índice

4.2 Ficheros ordenados (1)

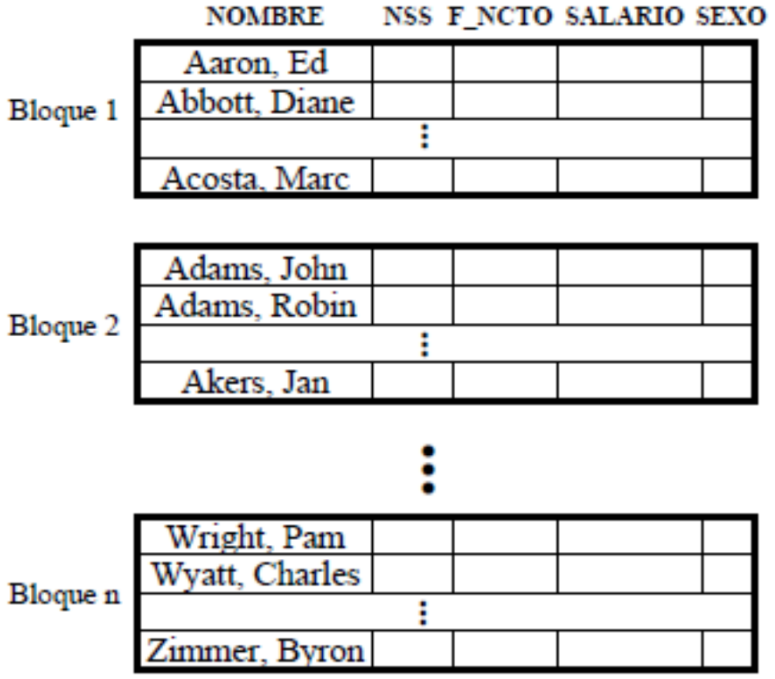
Ficheros ordenados (secuenciales)

NO es un fichero ***Heap*** recién ordenado

- Secuencial o secuencial ordenado
- Registros se mantienen físicamente ordenados según los valores de un campo: el **Campo de Ordenación**

- Si el campo de ordenación es clave, entonces se llama **Clave de Ordenación**
- Se requiere reorganizar el fichero de vez en cuando para restaurar el orden secuencial según el **Campo de Ordenación.**

Algunos bloques del fichero ordenado de registros EMPLEADO siendo el campo NOMBRE el Campo de Ordenación
(Figura 13.7 Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe)



4.2 Ficheros ordenados (2)

Ficheros ordenados (secuenciales): Ventajas

- Lectura eficiente en orden del **Campo de Ordenación.**
- Acceso eficiente al siguiente registro en el orden del **Campo de Ordenación**:
  - Estará en el búfer (si quedan registros por leer),
  - Si no, trae el siguiente bloque (si hay).
- Posibilidad de **Búsqueda binaria**:
  - Mucho más rápida,
  - Sólo **si la condición de búsqueda se basa en un valor del campo clave de ordenación**
- **Búsqueda binaria** (dicotómica): se efectúa sobre bloques en lugar de sobre registros.
- **Ejemplo**: tenemos los bloques 1, 2, …, b y sus direcciones están la cabecera del fichero.

- Se desea encontrar un registro con valor *K* en su campo clave de ordenación
- Costo medio con búsqueda binaria: acceso a $\log_2(b)$ bloques
- Costo medio con búsqueda lineal: b/2 (si está el bloque) o b (si no está)
- Se puede buscar sobre el **Campo de Ordenación** por =, y también por <, >, <=, >=.
- Estos ficheros **NO ofrecen ventajas** para búsquedas **por campos distintos al de ordenación.**

4.2 Ficheros ordenados (3)

Ficheros ordenados (secuenciales): Operaciones

- Se debe mantener el orden de los registros.
- Inserción y eliminación: operaciones costosas.
- **Inserción**
  1. Buscar el bloque donde insertar basándonos en el valor del **Campo de Ordenación**;
  2. Obtener espacio, desplazando el resto de registros del fichero;
  3. Media: lectura y reescritura de b/2 bloques **!!;**
  4. Insertar el nuevo registro y reescribir el bloque.

- Alternativa:
  - Crear un fichero temporal desordenado, **fichero de desbordamiento (*fd*),**
    - ✓ Los nuevos registros se insertan al final del ***fd***;
    - ✓ Periódicamente: reconstruir el fichero ordenado, colocando la posición que le corresponda a los registros del ***fd*** ;
    - ✓ Inserción más eficiente ;
    - ✓ Búsqueda más ineficiente.

4.2 Ficheros ordenados (4)

Ficheros ordenados (secuenciales): Operaciones

- **Eliminación**
  - Similar a la inserción para la eliminación física
  - Más sencillo si se usan marcadores de borrado y se reorganiza periódicamente
- **Modificación**
  - Según el caso podrá realizarse búsqueda binaria
  - Si se modifica el **Campo de Ordenación** puede precisar cambiar de posición el registro. Eso supone hacer una eliminación y una inserción.
  - Si los registros son de longitud fija se podrá reescribir el bloque en la misma posición.

Uso de ficheros ordenados en BD

- Estos ficheros se usan poco en aplicaciones de BD
- Se usan más con un índice primario: ficheros secuenciales indexados

Índice

4.3 Ficheros *hash*

Ficheros hash (directos o dispersos)

- Se elige un campo, llamado ***Campo de Dispersión***, *y a*l valor de ese campo se le aplica una función llamada ***Función de dispersión (o aleatorización)*** que, tomando como entrada dicho valor devuelve un número que será la dirección del bloque de disco en que se almacenará el registro.
- Aceleran el acceso a los registros cuando se busca un único registro según el ***Campo de Dispersión***.
- La condición de búsqueda es solo la **igualdad** sobre el ***Campo de Dispersión.***
- Normalmente el ***Campo de Dispersión*** suele ser el campo clave.
- La búsqueda del registro dentro del bloque se puede hacer en el búfer de memoria principal. Suele bastar con un acceso al bloque para obtener un registro.
- Existen numerosas ***Funciones de Dispersión*** y su eficacia radica en que distribuyan los registros lo más posible minimizando el número de colisiones (producida cuando dos valores distintos del campo de dispersión producen el mismo valor de dispersión). Para estas situaciones se deben usar una segunda función, buscar la siguiente posición vacía dentro del bloque o usar técnicas de encadenamiento de registros.

Índice

5. Métodos de acceso: Índices

- **Asumimos que existe un fichero con algunas de las organizaciones previas: *heap, ordenado o hash.***

- Existen estructuras de acceso auxiliares denominadas ***ÍNDICES***, que se utilizan para **acelerar la recuperación** de registros en respuesta a ciertas condiciones de búsqueda.

- ***Son similares a los índices analíticos de cualquier libro de texto, es decir, son como una lista en orden alfabético de los términos de un libro incluyendo la página o páginas en que se encuentra.***

5. Métodos de acceso: Índices

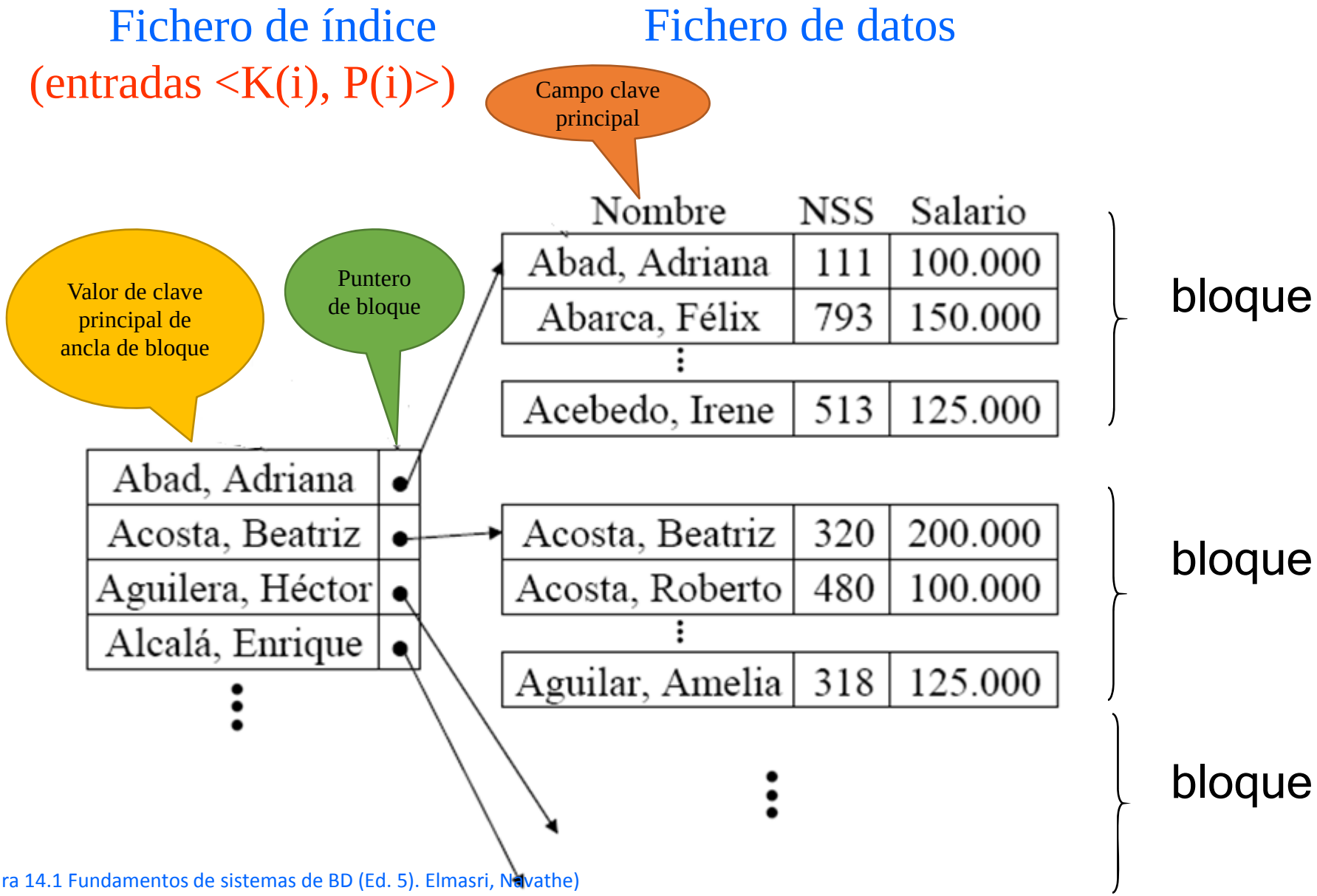
- Es posible utilizar *cualquier campo* del fichero para crear un ***índice*** y se denota como ***Campo de Indexación.***
- Para encontrar uno o varios registros del fichero basándose en ciertos criterios de selección de un ***Campo Indexado: primero se accede al índice que apunta a uno o más bloques del fichero donde están almacenados los registros requeridos.***
- Existen varios tipos de ***índices*** y cada uno de ellos utiliza una **estructura de datos** particular para acelerar la búsqueda:
  - **5.1 Ficheros ordenados (*índices de un nivel*)**: Primario, Agrupación, Secundario
  - **5.2 Árboles (*índices multinivel*): Árbol B^+ , B, B^***
- Se pueden construir ***varios índices*** con diferentes campos en el mismo fichero, pero **SOLO** puede tener **uno Primario o uno de Agrupación**, ya que físicamente el fichero de datos tiene como máximo un campo de ordenación física; el resto serán **Secundarios**.

Índice

5.1 Ficheros ordenados (índices de un nivel)

- El fichero de índice contiene un **conjunto de entradas.**
- Cada **entrada** incluye dos campos, para almacenar...
  - Un **valor** de los almacenados en el ***Campo de Indexación.***
  - Un **puntero** al **registro** que contiene dicho valor **o** al **bloque** que lo contiene.
- Las **entradas** están **ordenadas!!!**
  - Según valores extraídos del ***Campo de Indexación*** del fichero de datos,
  - Por tanto, es posible realizar **búsquedas binarias** sobre el ***índice.***
- Índices **densos** y **no densos**
  - Denso si contiene **una entrada por cada registro del fichero**
  - NO denso (o disperso), **una entrada por cada bloque de disco del fichero de datos**
- **Operaciones** sobre ficheros con índices:
  - Buscar implica búsqueda binaria **sobre el índice**
  - Insertar y Eliminar **pueden** provocar **modificación del índice**
    - **Menos accesos a bloques**, pero ∃ **coste de mantener el índice**

5.1.1 Ficheros ordenados (índices de un nivel): ÍNDICE PRIMARIO



(Figura 14.1 Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe)

5.1.1 Ficheros ordenados (índices de un nivel): ÍNDICE PRIMARIO

- Sobre **ficheros** con registros **ordenados** según los valores de un **campo clave.**
- El ***campo de indexación*** es la **clave principal** (campo ordenación=clave principal)

El **índice** es un fichero **ordenado** con...

- Una entrada por cada **bloque** del fichero de datos (índice **NO denso**)
- Cada entrada, *i*, tiene **longitud fija y 2 campos**:
  - *K(i)*: valor del ***campo de indexación*** de un registro **ancla de bloque** **(ancla=valor clave principal del primer registro del bloque)**
  - *P(i)*:Puntero al comienzo de cada **bloque**

- **Operaciones** sobre ficheros con índices primarios:
  - **Insertar** y **Eliminar pueden** provocar la **modificación del índice si afectan a los registros ancla.**

5.1.1 Ficheros ordenados (índices de un nivel): ÍNDICE PRIMARIO

- Nº bloques índice << Nºbloques fichero:
  - Nºentradas < Nºregistros fichero
  - Tamaño entrada < tamaño registro (suele ser)
- Búsqueda binaria:
  - Nº accesos a bloques del índice << nº accesos a bloques de datos
- Búsqueda de la clave con valor ***K***:
  - Búsqueda binaria en índice
  - Encontrar entrada ***i*** de índice : obtener el puntero ***P(i)***
  - Lectura del bloque ***i*** del fichero de datos
  - Búsqueda binaria en bloque de datos ***i***

5.1.1 Ficheros ordenados (índices de un nivel): ÍNDICE PRIMARIO

Fichero sin índice:

Ejemplo

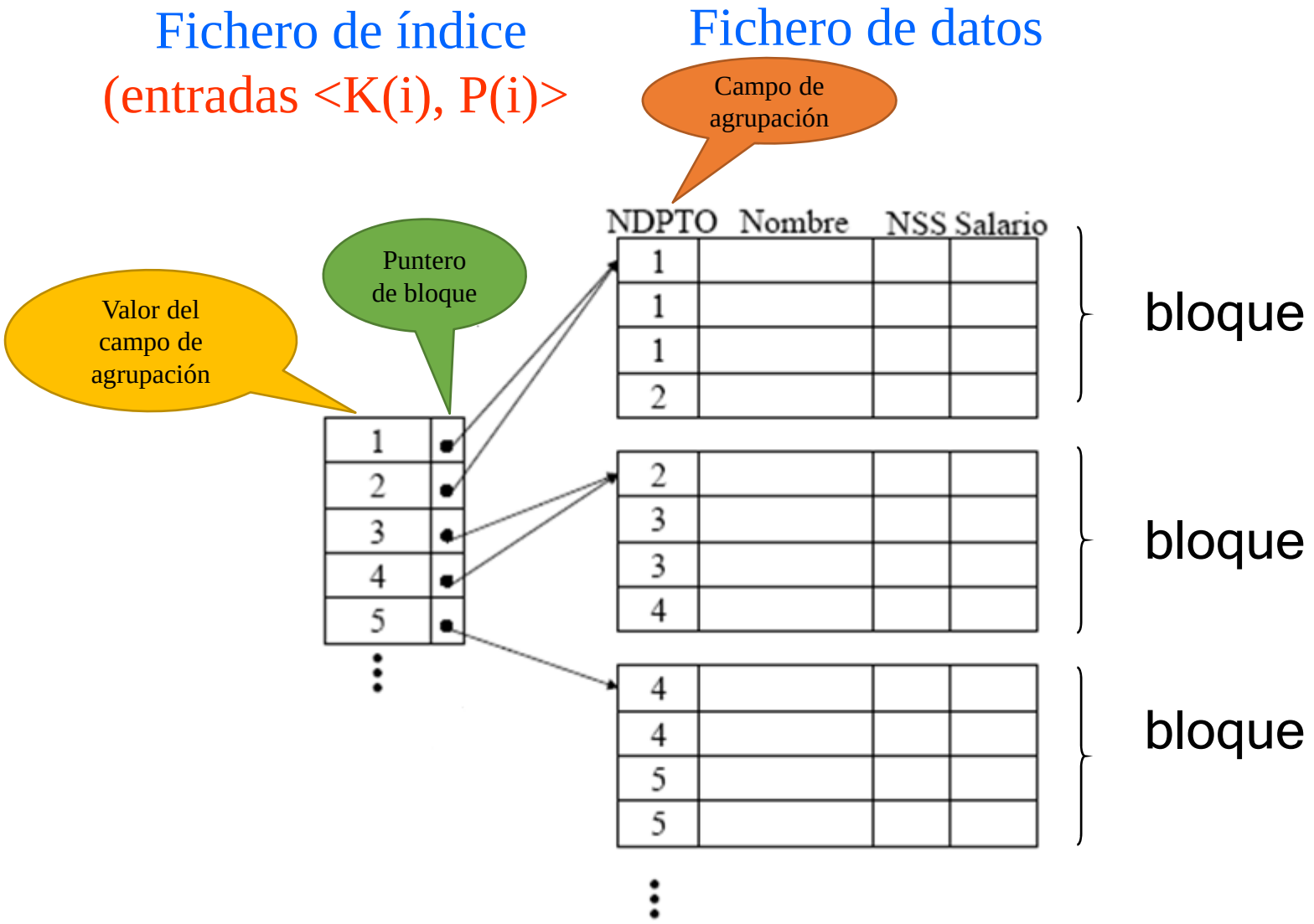
- Fichero ordenado de datos con r=30.000 registros, almacenados en disco con tamaño bloque B=1.024 bytes
- R=100 bytes/registro *(longitud registro fija y Org. no extendida)*
- fbl=$\lfloor B/R \rfloor$=$\lfloor \mathbf{1.024/100} \rfloor$=10 registros/bloque *(factor de bloque)*
- b=$\lceil r/fbl \rceil$=$\lceil 30.000/\mathbf{10} \rceil$=3.000 bloques *(requeridos para almacenar el fichero de datos)*
- Búsqueda binaria en el fichero de datos (está ordenado según el campo búsqueda): $\log_2$ b= $\log_2$ 3.000= 12 accesos a bloque

Fichero con índice Primario:

- Tamaño Entradas índice: R_i=(9+6)=15 bytes/registro
- fbl_i= $\lfloor B/R_i \rfloor$=$\lfloor 1.024/\mathbf{15} \rfloor$=68 entradas/bloque *(factor de bloque)*
- r_i=3.000 entradas=nº bloques del fichero de datos!
- b_i=$\lceil r_i/fbl_i \rceil$=$\lceil \mathbf{3.000}/68 \rceil$= 45 bloques *(requeridos para almacenar el fichero índice)*
- Búsqueda binaria en índice:$\log_2$ b_i= $\log_2$ 45= 6 accesos a bloque
- Hace falta 1 acceso extra a fichero para acceder al registro de datos:

  total = 7 accesos a bloque ➔ 7 (con índice primario) < 12 (sin índice)

5.1.2 Ficheros ordenados (índices de un nivel): ÍNDICE de AGRUPACIÓN



Índice agrupado en el campo no clave de ordenación NDPTO de un fichero EMPLEADO

(Figura 14.2 Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe)

5.1.2 Ficheros ordenados (índices de un nivel): ÍNDICE de AGRUPACIÓN

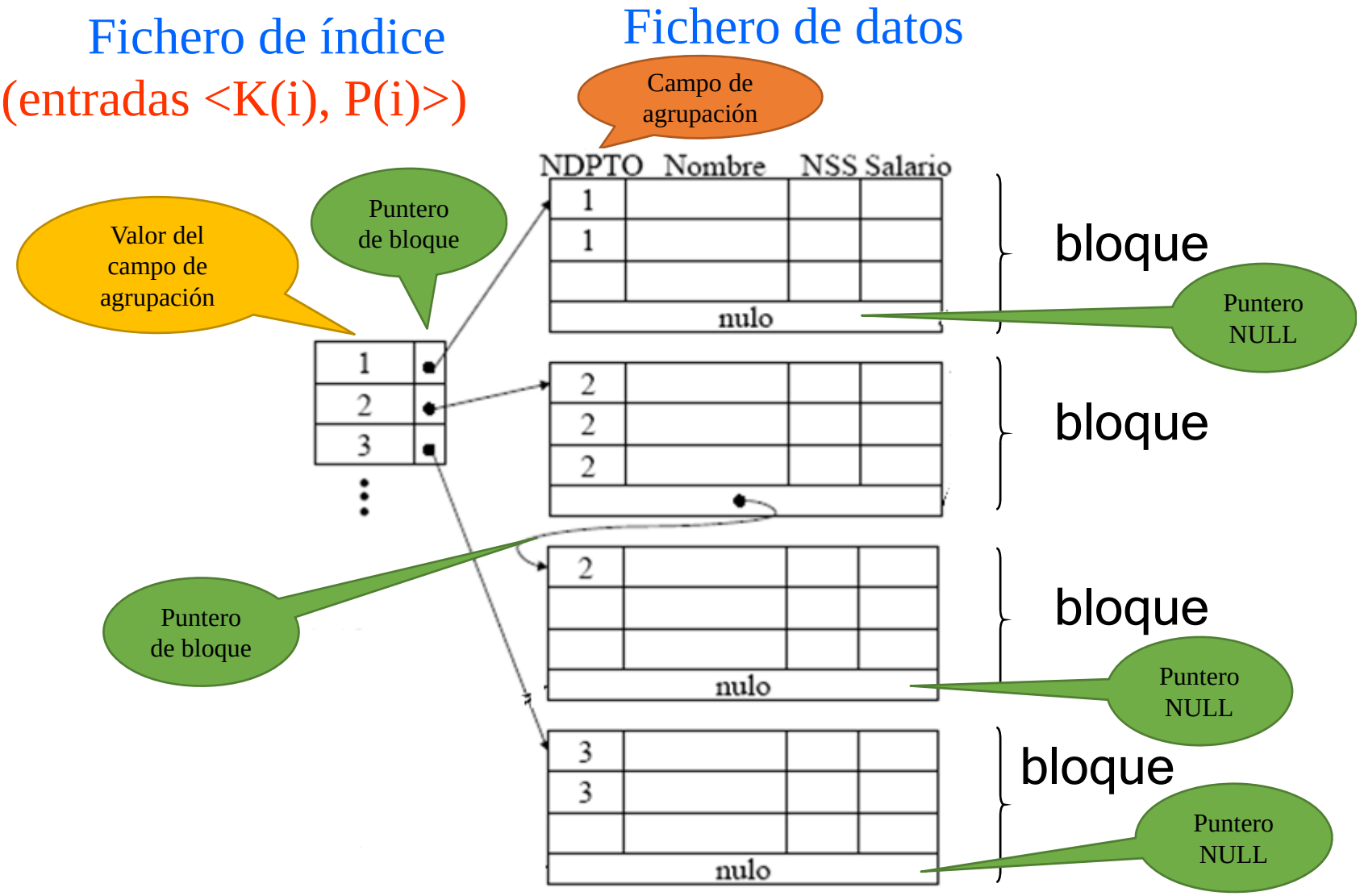
- Sobre **ficheros** con registros **ordenados** según los valores de un **campo no clave,** denotado como **campo de agrupamiento (**c.a.), (puede haber más de un registro con el mismo valor en este campo).
- El ***campo de indexación*** es dicho **campo de agrupamiento**

El **índice** es un fichero **ordenado** con...

- Una entrada por cada **valor distinto** del campo de agrupamiento del fichero de datos (índice **no denso**)
- Cada entrada tiene **longitud fija y 2 campos**:
  - Valor **distinto** del campo de indexación (= de agrupamiento)
  - Puntero al primer **bloque** en que aparece dicho valor

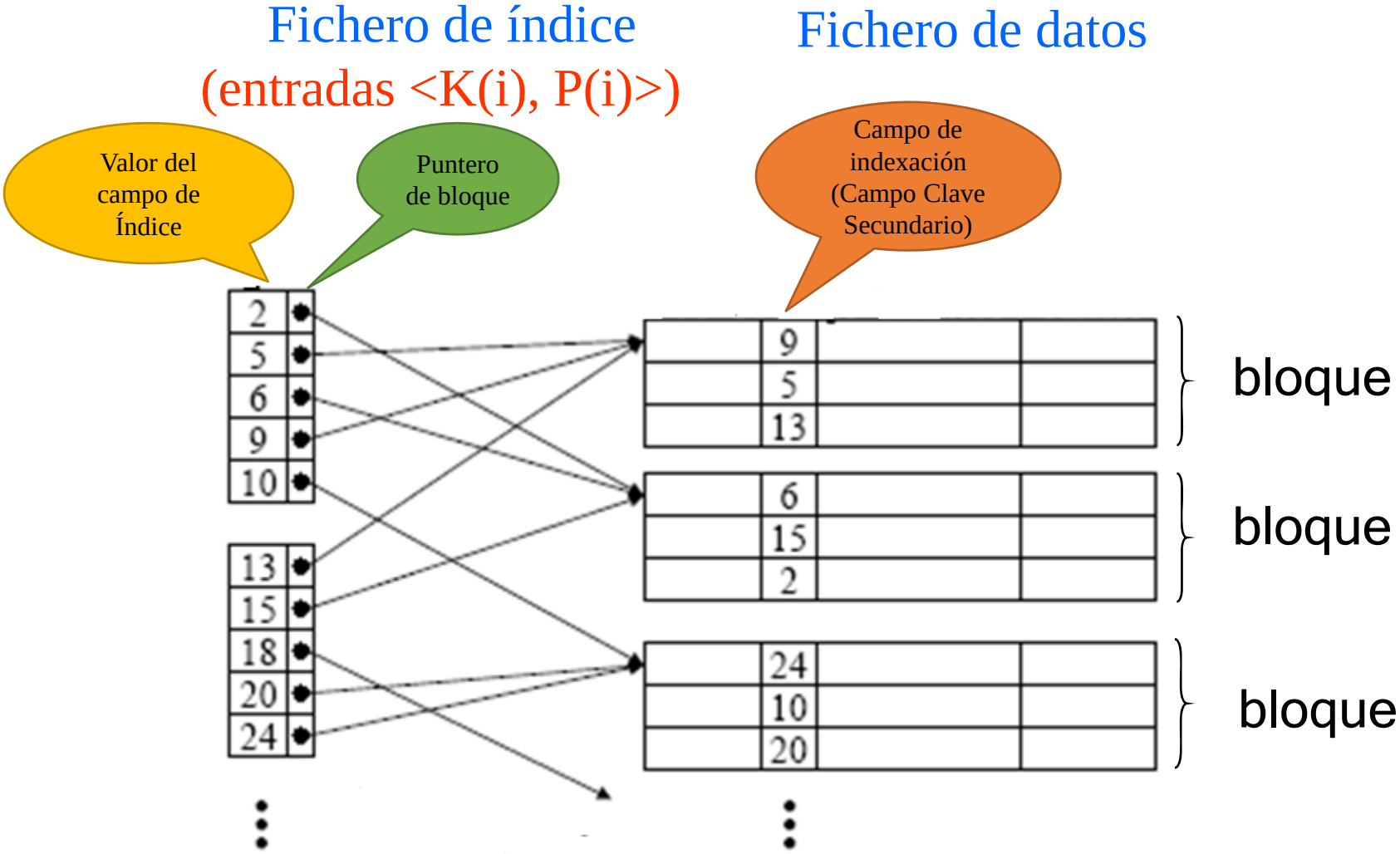
- **Operaciones** sobre ficheros con índices de agrupamiento:
  - **Insertar** y **Eliminar costosas** y **provocan modificación del índice**
    - Alivio si se reserva un bloque (del fichero de datos) para cada valor distinto del campo de agrupación (o varios bloques enlazados)
    - acelera la recuperación de registros que tienen el mismo valor en el campo de agrupamiento

5.1.2 Ficheros ordenados (índices de un nivel): ÍNDICE de AGRUPACIÓN



Índice agrupado con bloques individuales para cada grupo de registros que tienen el mismo valor de campo de agrupación (Figura 14.3 Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe)

5.1.3 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO CLAVE*



Índice secundario denso (con punteros de bloque) en un campo clave desordenado de un fichero (Figura 14.4 Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe)

5.1.3 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO CLAVE*

- ***Índice secundario*** es un fichero ordenado con 2 campos:
  - Campo de indexación ≠ campo de ordenación archivo de datos
  - **Apuntador a bloque o a registro**
- **Pueden definirse varios índices secundarios sobre el mismo fichero.**

- **Sobre un campo clave (por tanto valor ≠ para cada registro):**
  - **Hay una entrada de índice por *cada registro* del fichero de datos (índice denso). La entrada de índice i será <*K(i),P(i)*>**
  - **Las entradas de índice ordenadas por *K(i)*: búsqueda binaria**
  - **Los registro del fichero de datos *no están ordenados físicamente* según los valores del campo clave secundaria, por eso se crea una entrada de índice por cada registro del fichero de datos y no por cada bloque.**
  - **Una vez transferido el bloque a memoria: busca el registro**
  - **Consume más espacio y tiempo que uno primario: mayor nº de entradas.**

5.1.3 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO CLAVE*

- Sobre un campo que es **clave** del fichero de datos, y que **no** marca la **ordenación** de los registros de datos.
- El campo de indexación es dicho campo: **clave secundaria**

El **índice** es un fichero **ordenado** con...
- Una entrada por cada **registro** del fichero de datos (índice **denso**)
- Cada entrada tiene **longitud fija y 2 campos**:
  - Valor del campo de indexación (= clave secundaria)
  - Puntero al **registro** o al **bloque** en que aparece dicho valor

- **Operaciones** sobre ficheros con índices de este tipo:
  - **Insertar** y **Eliminar provocan la modificación del índice**
- Tiene **más entradas** que un índice primario
  - Ocupa más espacio. Mayor tiempo de búsqueda
  - Pero *mejoría* en el tº búsqueda de un registro arbitrario, ya que búsqueda lineal en el fichero de datos si no existe el ***Índice Secundario***.

5.1.3 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO CLAVE*

Ejemplo

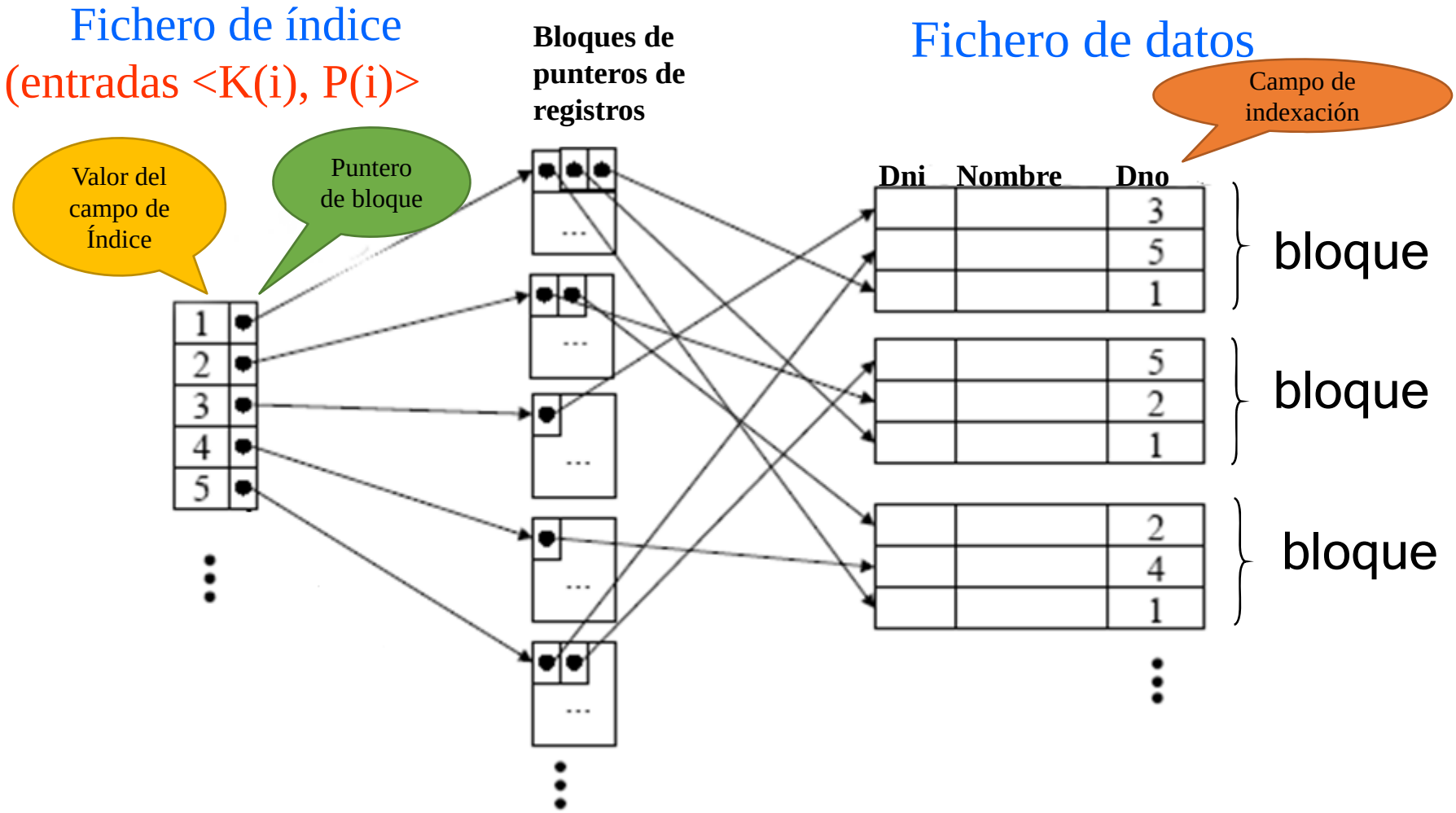
Fichero sin Índice Secundario:

- r=30.000 registros, B=1.024 bytes/bloque
- R=100 bytes/registro, b=3.000 bloques
- **Búsqueda lineal en fichero de datos** (no está ordenado según el campo de búsqueda): b/2=3.000/2= 1.500 accesos a bloque

Fichero con Índice Secundario:

- Entradas R_i= (9+6) = 15 bytes
- $fbl_i=\lfloor B/R_i \rfloor=\lfloor 1.024/\mathbf{15} \rfloor$=68 entradas/bloque ***(factor de bloque)***
- r_i=30.000 entradas=nº registros
- $b_i=\lceil r_i/fbl_i \rceil=\lceil 30.\mathbf{000}/68 \rceil$=442 bloques ***(requeridos para almacenar el IS, el IP sólo 45)***
- Búsqueda binaria en índice: $\log_2 b_i=\log_2 442$=9 accesos a bloque
- Hace falta 1 acceso extra a fichero para acceder al registro de datos:
- total = 10 accesos a bloque → 10 < 1.500 (sin índice)
- *IS* proporciona *ordenación lógica* de los registros según campo de indexación.

5.1.4 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO NO CLAVE*



IS sobre campo no clave implementado con un nivel de indirección, entradas de índice de long. fija y valores únicos en los campos)

(Figura 14.5 Fundamentos de sistemas de BD (Ed. 5). Elmasri, Navathe)

5.1.4 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO NO CLAVE*

- Sobre un campo que **no** es **clave** del fichero de datos, y que **no** marca la **ordenación** de los registros de datos (por tanto varios registros del fichero de datos pueden tener igual valor en el campo de indexación.)
- El campo de indexación es dicho campo.
- **Opciones** para crear el **índice**, fichero **ordenado** con...

1. Una entrada por cada **registro** del fichero de datos (índice **denso**)
   - entradas de **longitud fija** y 2 campos: valor y puntero a **registro**
   - **valores repetidos** del campo de indexación

Por tanto, Varias entradas de índice con el mismo valor *K(i)*, una por registro.

2. Una entrada por **valor distinto** del campo de indexación (**no denso**)
   - entradas de **longitud variable**: valor y varios punteros a **bloque**

Por tanto, entradas de índice sean registros de longitud variable donde cada *K(i)* tiene asociado <*P(i,1)*,…,*P(i,n)*,>, donde *P(i,x)* es el puntero a cada bloque que contiene un registro con valor del campo de indexación *K(i)*.

5.1.4 Ficheros ordenados (índices de un nivel): ÍNDICE SECUNDARIO *SOBRE CAMPO NO CLAVE*

- Sobre un campo que **no** es **clave** del fichero de datos, y que **no** marca la **ordenación** de los registros de datos (por tanto varios registros del fichero de datos pueden tener igual valor en el campo de indexación.)
- El campo de indexación es dicho campo.
- **Opciones** para crear el **índice**, fichero **ordenado** con...

3. Una entrada por **valor distinto** del campo de indexación, y un **nivel adicional de indirección** (índice **no denso**)
   – entradas de **longitud fija** y 2 campos:
     - Valor del campo de indexación
     - Puntero a un **bloque de punteros a registro**

✓ Nivel adicional de indirección: cada entrada apunta a un bloque de punteros a registro.

✓ Si un valor se repite tanto que desborda el bloque de punteros: lista enlazada de bloques

✓ Algoritmos búsqueda e inserción: sencillos.

5.1 Ficheros ordenados (índices de un nivel): RESUMEN

Tipos de índices basados en las propiedades del campo de indexación

	Campo de índice utilizado para ordenar el fichero	Campo de índice NO usado para ordenar el fichero
El campo de indexación es una clave	Índice Primario	Índice Secundario (Clave)
El campo de indexación NO es una clave	Índice de Agrupación	Índice Secundario (NO Clave)

Propiedades de los tipos de índice

Tipo de índice	Nº de entradas de índice (primer nivel)	Denso/NO Denso	Bloque de anclaje en el fichero de datos
Primario	Nº de bloques en el fichero de datos	NO Denso	Sí
Agrupación	Nº de valores del campo de índice distintos	NO Denso	Sí/No[1]
Secundario (Clave)	Nº de registros del fichero de datos	Denso	No
Secundario (NO Clave)	Nº de registros[2] o Nº de valores del campo de índice distintos[3]	Denso o NO Denso	No

[1] Sí, si cada valor distinto del campo de ordenación inicia un nuevo bloque ; no, en cualquier otro caso.
[2] Para la opción 1 de este tipo de índices.
[3] Para las opciones 2 y 3 de este tipo de índices.