7. Agrupamiento (*clustering*)

INMUEBLES y CONTRATOS

I	IA14	En medio, 128		Centro	Castellón	600			
C	10024	Q62	600	Visa	1200	S	1/6/99	31/5/00	12
I	IL94	Riu Ebre, 24		Ronda Sur	Castellón	350			
C	10075	Q76	350	Efectivo	700	N	1/1/00	30/6/00	6
I	IG24	San Francisco, 10			Vinaroz	550			
C	10002	Q56	500	Efectivo	1000	S	1/1/97	30/6/97	6
C	10012	Q74	550	Cheque	1100	S	1/7/99	30/6/00	12

- Inmueble seguido de su grupo de contratos (acceso frecuente a estos datos "juntos").
- Contratos: no hace falta incluir el número del inmueble (hay referencia física) ☺
- Algunos accesos se penalizan ☹
- Es un fichero ordenado ☹

EMPLEADOS por OFICINA

O	O3			
E	EG37	Cubedo	Supervisor	38766623X
E	EG14	Collado	Administrativo	24391223L
E	EG5	Prats	Director	25644309X
O	O5			
E	EL21	Pastor	Director	39432212E
E	EL41	Baeza	Supervisor	39552133T
O	O7			
E	EA9	Renau	Supervisor	39233190F

- Agrupamiento sobre un solo fichero.
- En cada registro se puede eliminar el número de oficina, aunque es necesario incluirlo delante de cada grupo.

8. Índices

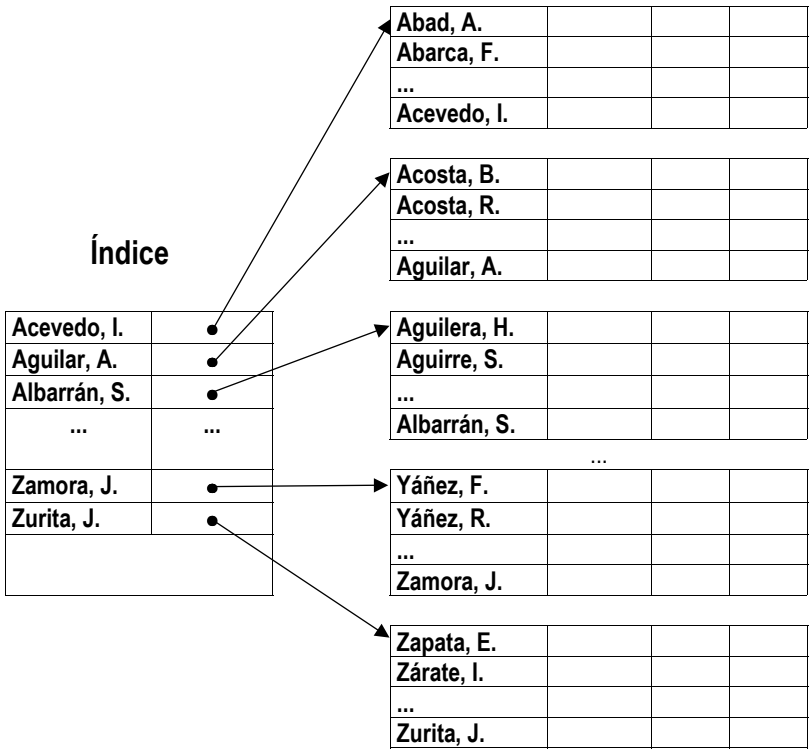
Índices de un solo nivel

- Los índices de un solo nivel son **ficheros ordenados**.
- Sus registros tienen dos campos:
  - **Campo de indexación**: coincide con uno de los campos del fichero de datos.
  - **Dirección del registro** que corresponde al valor del campo de indexación.
- Al ser ficheros ordenados se pueden realizar **búsquedas binarias**.

Tipos de índices de un solo nivel:

- Índices primarios.
- Índices de agrupamiento.
- Índices secundarios.

Índices Primarios



Entradas: registros de longitud fija.

Campo de indexación: campo clave de ordenación del fichero de datos.

Índice no denso.

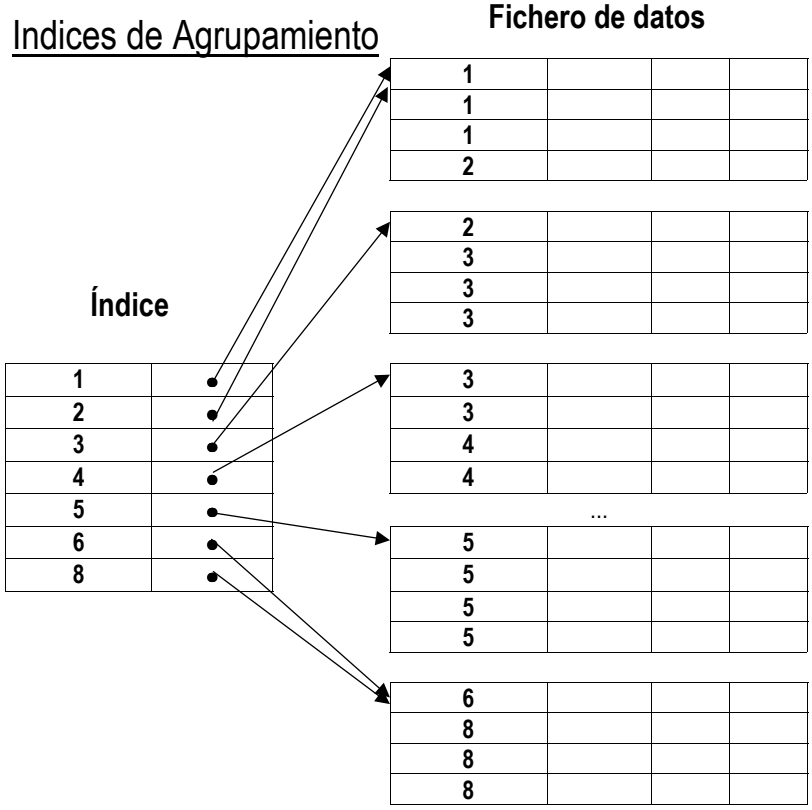
Entrada: valor de la clave del primer/último registro del bloque y puntero a dicho bloque.

Búsqueda binaria sobre el índice: visita menos bloques de disco.

Problema: son ficheros ordenados (opciones: fichero de desbordamiento desordenado; lista enlazada de registros de desbordamiento).

Importante: sobre un fichero ordenado por clave sólo puede definirse un índice primario.

Indices de Agrupamiento



Entradas: registros de longitud fija.

Campo de indexación: campo no clave de ordenación del fichero de datos (campo de agrupamiento).

Índice no denso

Entradas: una por cada valor distinto del campo de agrupamiento. El puntero apunta al primer bloque que contiene un registro con dicho valor.

Búsqueda binaria sobre el índice: visita menos bloques de disco.

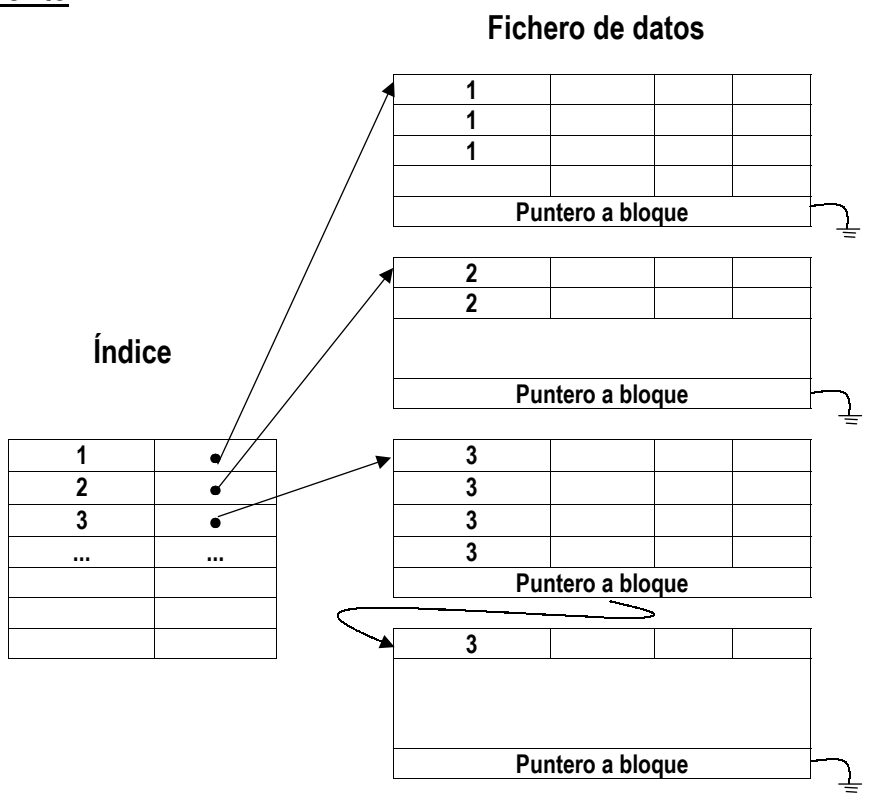
Problema: son ficheros ordenados. (opción: reservar un bloque entero para cada valor distinto del campo de agrupamiento).

Importante: sobre un fichero ordenado por un campo no clave sólo puede definirse un índice de agrupamiento.

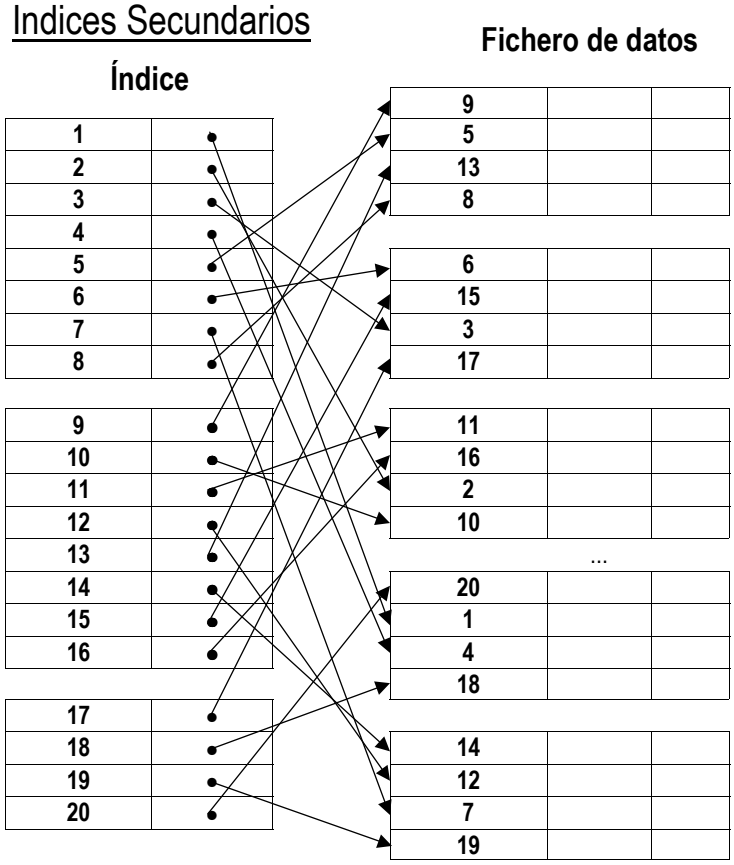
Indices de Agrupamiento

Aquí se ha reservado un bloque para cada valor distinto del campo de agrupamiento.

Se van añadiendo y enlazando bloques conforme sea necesario.



Indices Secundarios



Campo de indexación: cualquier campo que no sea el campo de ordenación.

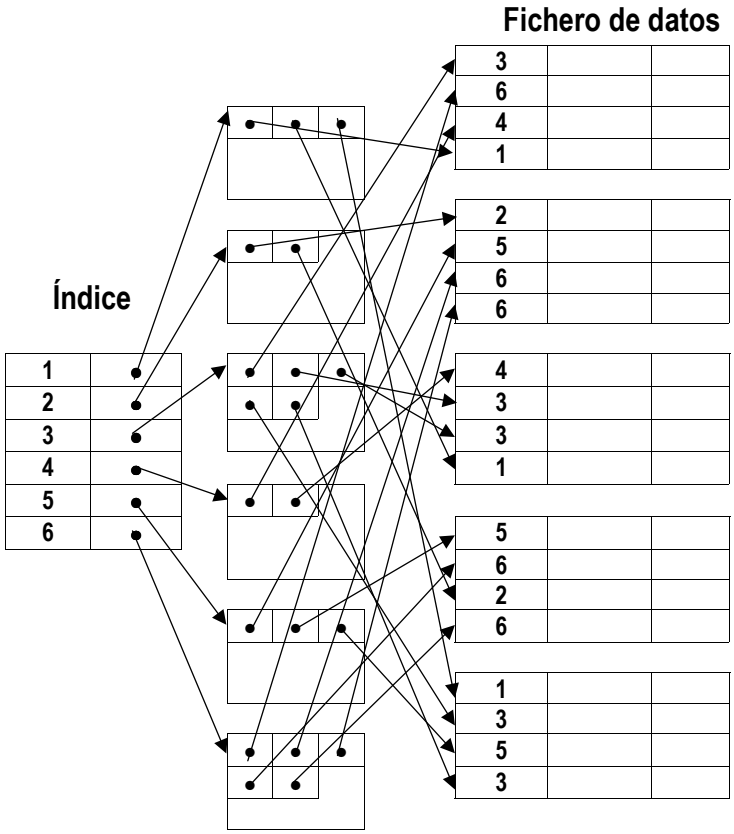
- Si es un **campo clave**: índice denso.

- Si es un **campo no clave**: hay varias opciones.

Importante: pueden definirse varios índices secundarios sobre un mismo fichero.

Los índices densos proporcionan un **ordenamiento lógico** de los registros según el campo de indexación.

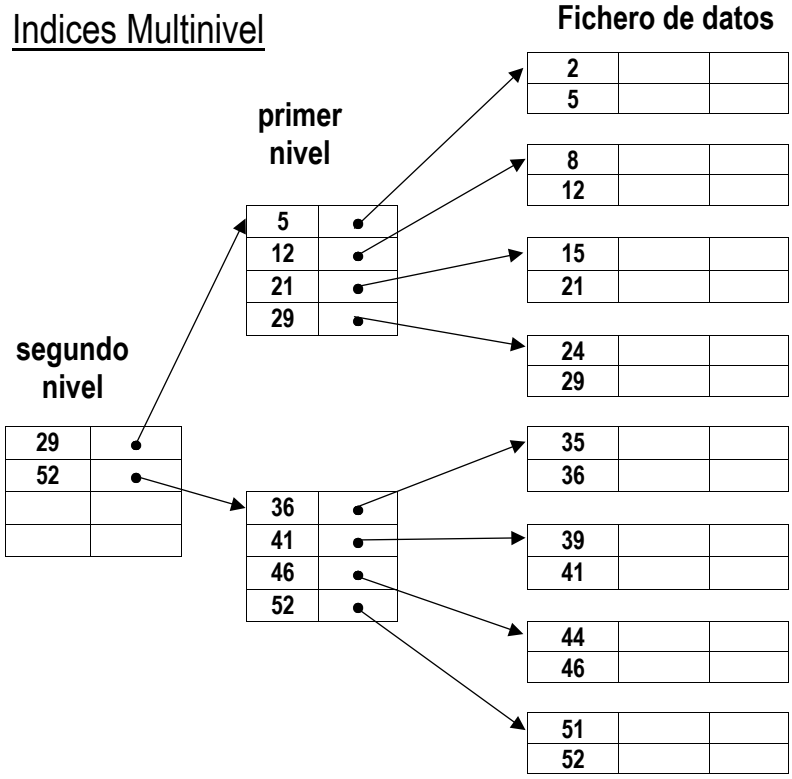
Indices Secundarios



Indice secundario sobre un **campo no clave**:

- Una entrada por cada registro: **índice denso**.
- **Registros de longitud variable**: índice no denso y el campo de la dirección contiene una lista de punteros.
- **Registros de longitud fija**: índice no denso con un nivel extra de indirección para manejar punteros múltiples.

Indices Multinivel



Objetivo: reducir más que con la búsqueda binaria el trozo de índice en donde seguir buscando.

Primer nivel: fichero ordenado con entradas de tamaño fijo y un valor distinto del campo de indexación en cada una.

Siguientes niveles: índices primarios sobre el nivel anterior.

Número de registros por bloque: **r**

- primer nivel i_1 entradas
- segundo nivel $i_2 = \lceil i_1/r \rceil$ entradas
- tercer nivel $i_3 = \lceil i_2/r \rceil$ entradas ...

Se necesita un nivel más si el anterior ocupa más de un bloque.

Un índice multinivel con i_1 **entradas** en el primer nivel tiene $\lceil \log_r i_1 \rceil$ **niveles**.

Reducen el número de accesos a bloque al hacer búsquedas, **pero son ficheros ordenados**.

Ejemplo comparativo de índices

Se tiene un fichero ordenado por campo clave con **30.000 registros** de **100 bytes**.

El tamaño de cada bloque de disco es de **1024 bytes**.

1. ¿Cuántos accesos hay que realizar para encontrar un registro en este fichero a través del campo de ordenación?

Sobre este mismo fichero se define un índice sobre el campo de ordenación para acelerar el tiempo de acceso.

2. ¿Qué tipo de índice será?

3. ¿Cuántos accesos hay que realizar ahora para realizar la misma búsqueda?

Tamaño de las entradas del índice: 9 bytes del campo de indexación + 6 bytes del puntero al bloque que contiene el registro.

4. ¿Cuántos accesos hay que realizar para encontrar un registro en el mismo fichero a través de un campo clave que no es el de ordenación?

5. Si se define un índice secundario sobre este campo para acelerar el tiempo de acceso ¿cuántos accesos hay que realizar ahora para hacer la misma búsqueda?

Tamaño de las entradas del índice: 9 bytes del campo de indexación + 6 bytes del puntero al bloque que contiene el registro.

6. Si este índice secundario se utiliza como primer nivel para un índice multinivel ¿cuántos niveles son necesarios para construirlo?

Ejemplo comparativo de índices

1. Registros por bloque = $\lfloor 1024/100 \rfloor = 10$

Bloques de datos = $\lceil 30000/10 \rceil = 3000$

Búsqueda binaria → $\log_2 3000 = 12$ accesos a bloques

2. Indice primario

3. Entradas por bloque = $\lfloor 1024/15 \rfloor = 68$; bloques de índice = $\lceil 3000/68 \rceil = 45$

Búsqueda binaria → $\log_2 45 = 6$ accesos al índice

Total de accesos a bloques = 6 (índice) + 1 (fichero de datos) = 7

Mediante el índice se ha conseguido ahorrar algo más del 40% en el número de accesos.

4. Búsqueda lineal → 3000/2 = 1500 accesos a bloques

5. Bloques de índice = $\lceil 30000/68 \rceil = 442$

Búsqueda binaria → $\log_2 442 = 9$ accesos al índice

Total de accesos a bloques = 9 (índice) + 1 (fichero de datos) = 10

6. Primer nivel → 442 bloques

Segundo nivel → $\lceil 442/68 \rceil = 7$

Tercer nivel → $\lceil 7/68 \rceil = 1$

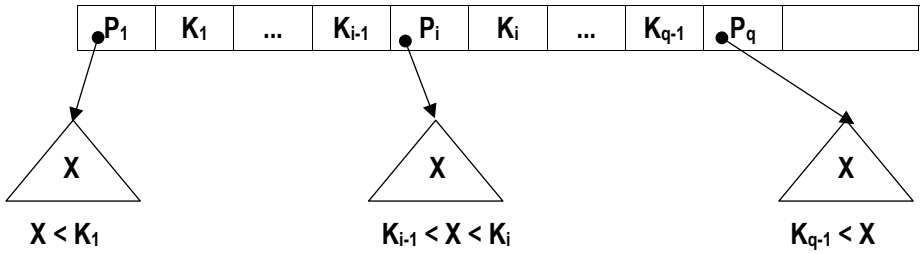
Total de accesos a bloques = 3 (índice) + 1 (fichero de datos) = 4

Arboles B y Arboles B+

Problemas de los índices multinivel: son ficheros ordenados.

Posible solución: reservar espacio en cada bloque para futuras inserciones (índices dinámicos multinievel).

Arboles de búsqueda:

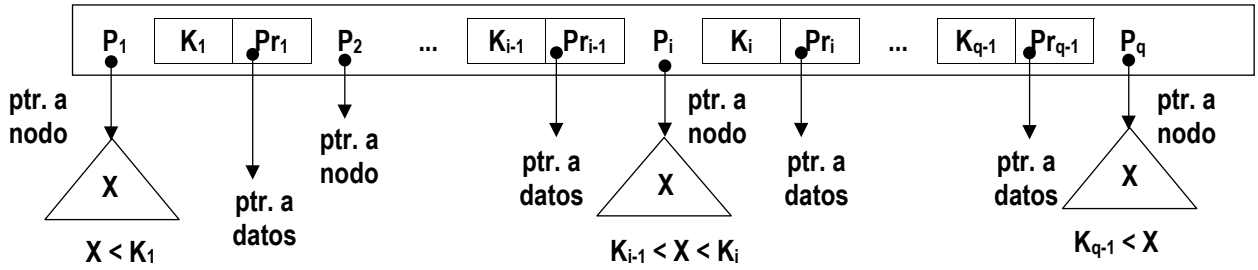


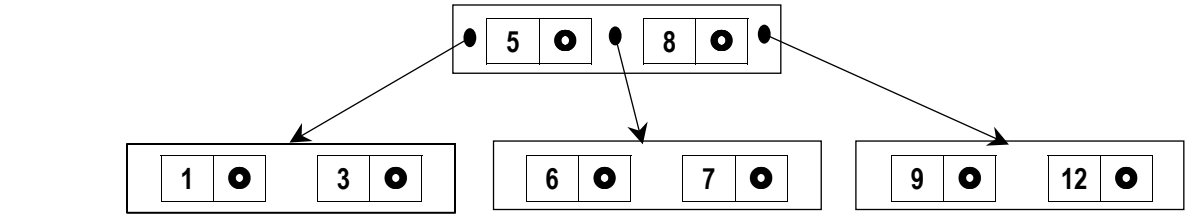
donde $K_1 < K_2 < ... \ K_{q-1}$ dentro de cada nodo.

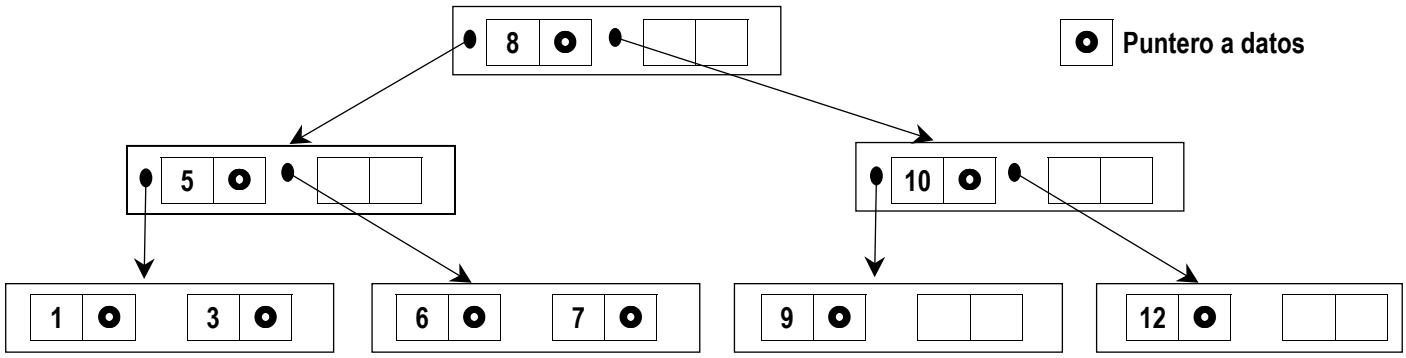
- Los algoritmos que realizan inserciones y borrados no garantizan que el árbol esté equilibrado.
- Las eliminaciones de registros pueden hacer que queden nodos casi vacíos: se desperdicia espacio.
- El que haya nodos casi vacíos también provoca un aumento en el número de niveles.

Arboles B de orden p

- En cada nodo se cumple: $K_1 < K_2 < ... \ K_{q-1}$ con $q \leq p$
- Cada nodo tiene **al menos $\lceil p/2 \rceil$ punteros a nodos** del árbol.
- Todas las hojas están al **mismo nivel**.
- Las **hojas** tienen la misma estructura que los nodos internos, pero los punteros a nodos del árbol son nulos.

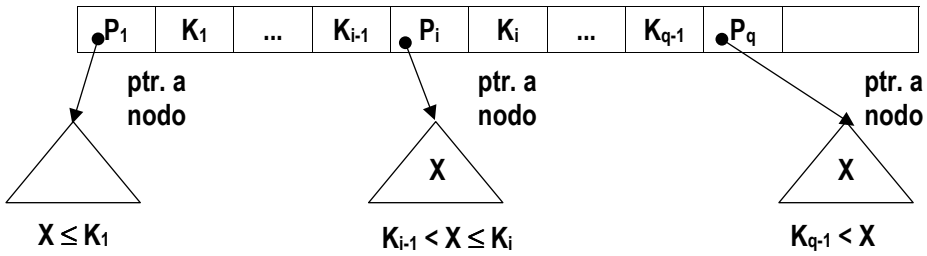


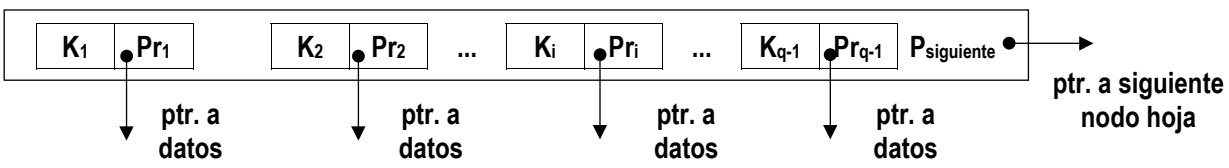




Arboles B+ de orden p

- En cada nodo se cumple: $K_1 < K_2 < \ldots K_{q-1}$ con $q \leq p$
- Cada nodo interno tiene **al menos $\lceil p/2 \rceil$ punteros a nodos** del árbol.
- Todas las hojas están al **mismo nivel**.
- Cada nodo **hoja** tiene al menos $\lceil p/2 \rceil$ valores.





Ejemplo comparativo de árboles B y árboles B+

Dado un fichero con las siguientes características:

- Tamaño campo de indexación V = 9 bytes.
- Tamaño de bloque B = 512 bytes.
- Tamaño de los punteros a registros de datos P_{datos} = 7 bytes.
- Tamaño de los punteros a subárboles $P_{árbol}$ = 6 bytes.

1. ¿Cuál será el orden p de un árbol B?
2. ¿Cuál será el número entradas del árbol B si tiene 3 niveles?

3. ¿Cuál será el orden p de un árbol B+?
4. ¿Cuál será el número entradas del árbol B+ si tiene 3 niveles?

Ejemplo comparativo de árboles B y árboles B+

1. $p * P_{árbol} + (p-1) * (V + P_{datos}) \leq B$

$p \leq 23$

Ya que los nodos están a un 69% de su capacidad :

$p * 0.69 = 16$ punteros

2.

	nodos	entradas	punteros
Raíz	1	15	16
Nivel 1	16	240	256
Nivel 2	256	3840	4096
nivel 3	4096	61440	
		65535	entradas

3. $p * P_{árbol} + (p-1) * V \leq B$

$p \leq 34$

¿p_{hoja} ? → $p_{hoja} * (P_{datos} + V) + P_{árbol} \leq B$

$p_{hoja} \leq 31$

Ya que los nodos están a un 69% de su capacidad :

$p * 0.69 = 23$ punteros

$p_{hoja} * 0.69 = 21$ punteros

4.

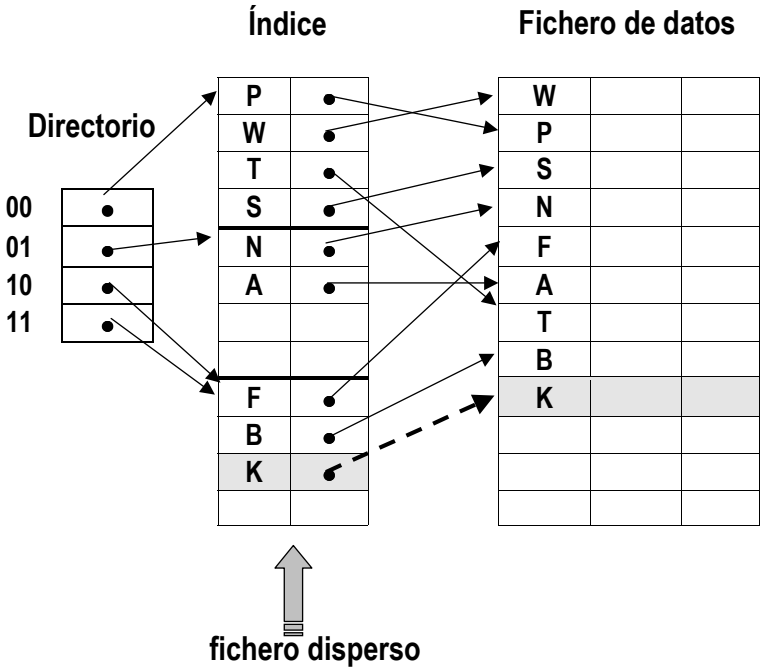
	nodos	entradas	punteros
raíz	1	22	23
nivel 1	23	506	529
nivel 2	529	11638	12167
nivel 3	12167	255507	entradas

Arboles B y Arboles B+

- En los árboles B+, además de tener acceso a los registros mediante el índice, se puede acceder según el orden del campo de indexación de modo secuencial.
- Ya que los nodos internos de un árbol B+ guardan menos punteros, tienen mayor capacidad que los nodos de los árboles B. Esto puede hacer que un árbol B+ tenga menos niveles y por tanto se mejore el tiempo de acceso.
- Ya que los nodos internos y los nodos hoja de un árbol B+ tienen distinta estructura, su orden p puede ser distinto.

Ficheros dispersos como índices

- También se pueden crear estructuras de acceso similares a los índices basándose en la dispersión.
- Las entradas del índice (K, Pr) se pueden organizar como un fichero disperso que va cambiando de tamaño mediante dispersión dinámica, extensible o lineal.
- El algoritmo de búsqueda aplica la función de dispersión sobre K. Una vez se ha encontrado la entrada, el puntero Pr se utiliza para localizar el registro correspondiente en el fichero de datos.



1. Insertar registro K
2. f(K) = 10...
3. Insertar entrada en el índice (fichero disperso)
4. Colocar el puntero al nuevo registro