BASES DE DATOS
TEMA 3

El nivel interno.

3.1 Introducción

Un SGBD debe almacenar de forma permanente grandes cantidades de datos, y gestionar de forma eficiente los datos y su almacenamiento.

- *Organización física. Incluye:*
  - *Sistemas software*
  - *Sistema operativos*
  - *Sistemas de gestión de ficheros*
  - *Controladores de dispositivos*
  - *Hardware*
- *Nivel Interno:*

Expresa operaciones sobre los datos a nivel de unidades mínimas de almacenamiento (páginas de BBDD). Implementado en el SGBD provee al DBA de mecanismos para optimizar el almacenamiento y el acceso a los datos.

- *Nivel Físico:*

Capa de abstracción sobre el hardware, implementada en el SO. El nivel interno realiza el acceso a los medios de almacenamiento mediante llamadas al nivel físico.

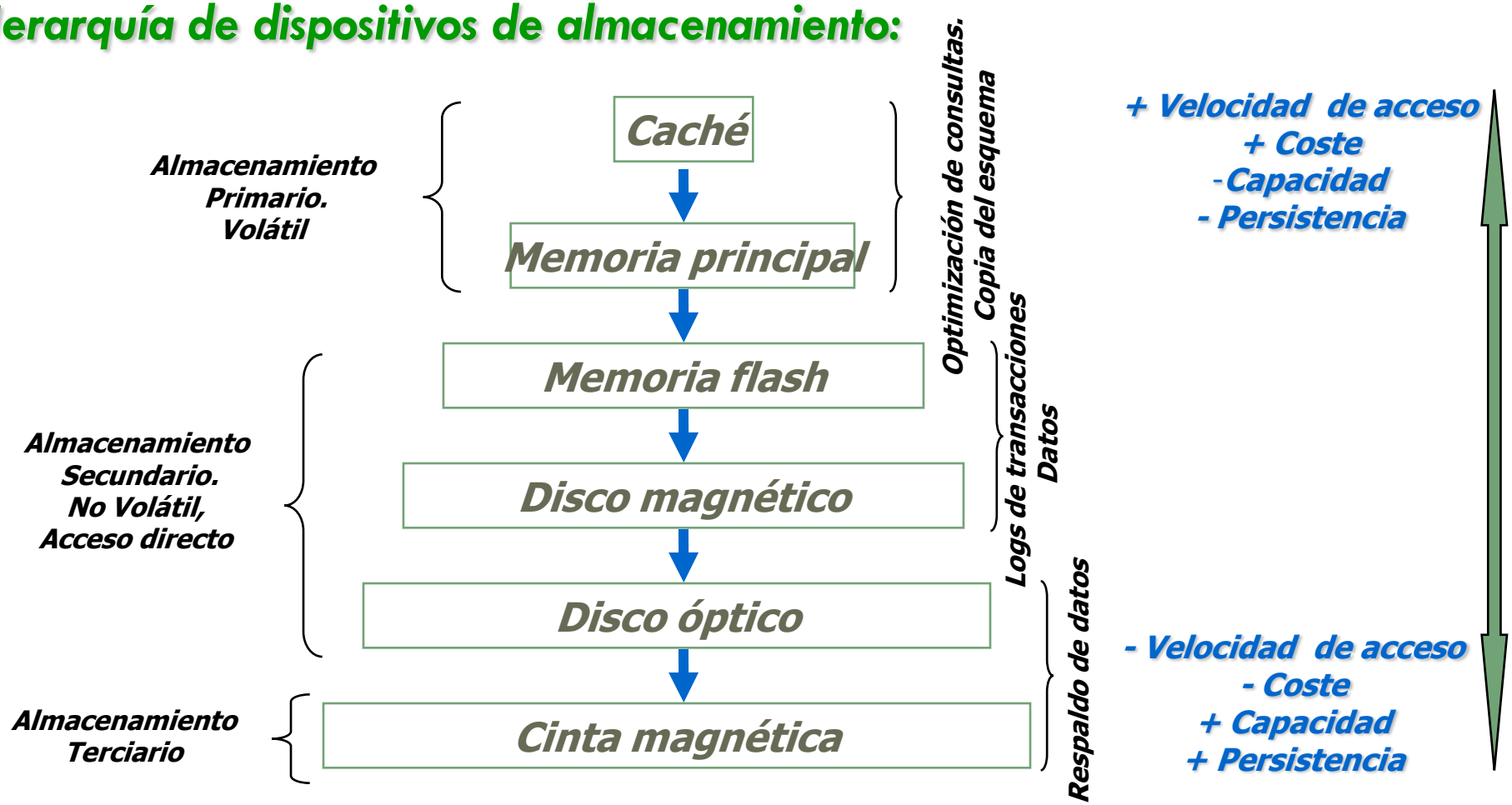
3.1. Introducción.

Los medios físicos de almacenamiento:

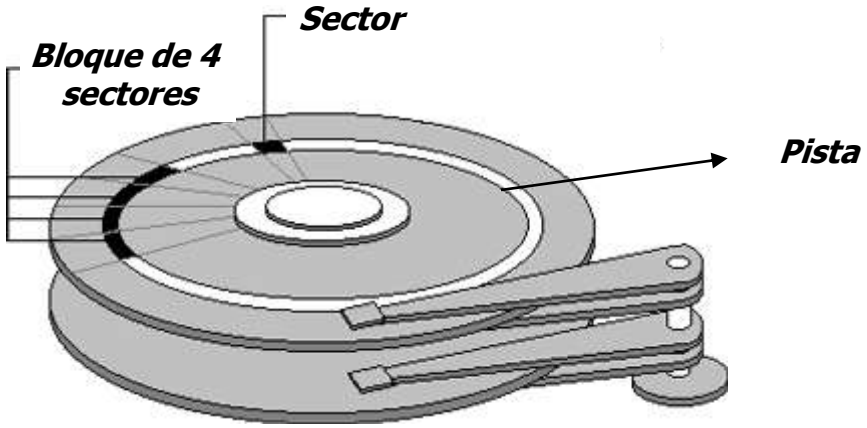
- *Cache:* es la memoria más rápida
- *Memoria principal:*
  - es rápida pero de pequeño tamaño para una base de datos.
  - No es segura ante fallos eléctricos
- *Memoria "flash" (EPROM):*
  - es rápida y segura ante fallos.
  - Puede ser útil para pequeños volúmenes de datos no volátiles
- *Discos magnéticos:*
  - Es de acceso directo
  - Soporta fallos y es totalmente fiable
  - Se guarda en disco toda la base de datos
- *Almacenamiento óptico:(CDROM):*
  - Son grabables una vez y legibles múltiples veces. Permiten cambiar discos
  - Se utilizan en bases de datos no volátiles (bibliográficas, documentales etc..)
- *Cinta magnética:*
  - Es barata y de gran tamaño ; pero de acceso secuencial
  - Se utiliza para copias de seguridad

3.2 Dispositivos de almacenamiento.

Jerarquía de dispositivos de almacenamiento:



3.2.1. Estructura lógica de los discos duros.



Estructura física:

- Conjunto de discos magnéticos con un eje de rotación común.
- En cada cara una cabeza lectura/escritura con movimiento transversal.
- Cara dividida en Pistas (2 a 1500), también denominado cilindro.
- Sectores, mínima unidad de almacenamiento (512 bytes).
- Bloque: secuencia continua de sectores que se trasladan "de una vez" a la memoria para ser procesados. Se asocia a cada fichero.
- Dirección de un bloque: cilindro+superficie+sector de comienzo
- Suelen poseer memoria cache intermedia entre el disco y el buffer de E/S.

3.2.1. Estructura lógica de los discos duros.

- *Medidas de rendimiento:*
  - *Tiempo medio de acceso (ta): tiempo medio transcurrido entre una instrucción y la obtención de la información (milisegundos)*
  - *Tiempo medio de búsqueda (tb): tiempo medio de posicionamiento en pista*
  - *Tiempo de latencia rotacional (tl): tiempo medio de posicionamiento en sector*

    $ta = tb + tl$
  - *Velocidad sostenida de Lectura/escritura (MB/s) y velocidad de rotación (rpm).*
  - *Fiabilidad: Tiempo medio entre fallos*

Accesos de lectura/escritura de datos:

- *Latencia (15-45 ms) = Búsqueda transversal (10-40 ms) + Rotación (5ms)*
- *Transferencia: Se envían los datos (ms por bloque).*
- *Tiempo de escritura = Tiempo de Lectura*
- *Tiempo modificación = 2 x Tiempo de Lectura*

3.2.2. Memoria Principal.

- *Constituye el dispositivo de almacenamiento primario de los ordenadores.*
- *De carácter Volátil, en ella se almacenan los datos que precisan los programas para su ejecución.*

El SGBD la usa para los siguientes propósitos:

- *Como memoria caché que almacena los datos accedidos más recientemente.*
- *Para almacenar parámetros de estado.*
- *Para almacenar código ejecutable de las sentencias de datos usadas.*

Desde el punto de vista del nivel interno:

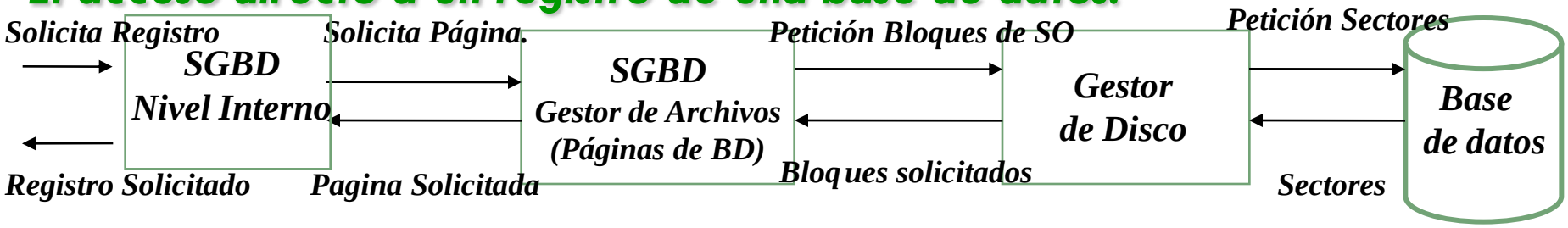
- *Memoria temporal de almacenamiento intermedio, donde se almacenarán los datos afectados por las operaciones del SGBD.*
- *Es del orden de 1000 veces más rápidas que la memoria secundaria.*
- *La optimización de su uso proporciona un mayor rendimiento en el procesamiento de los datos.*

3.3. Métodos de acceso a la BD.

Define como se transforma un registro almacenado en una repres. física a nivel de almac. secundario. Difiere de un SGBD a otro.

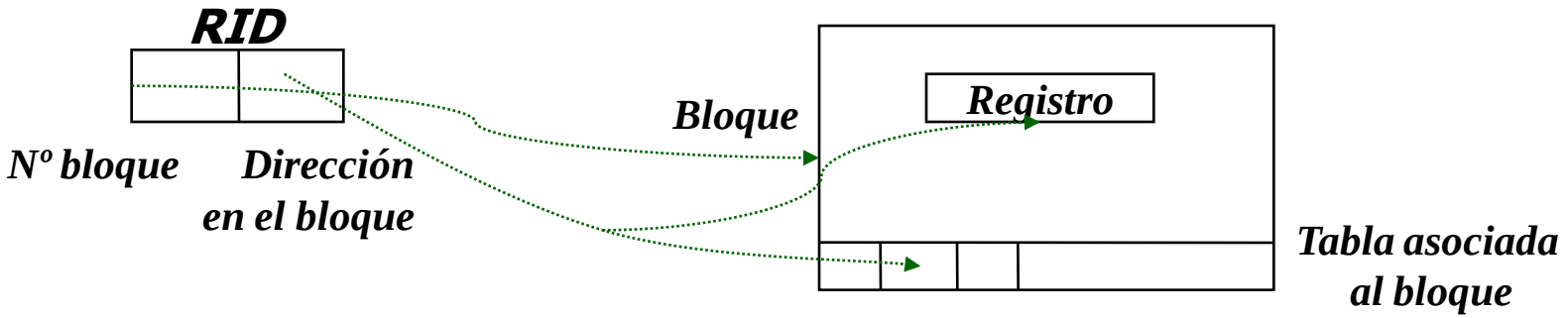
Acceso Secuencial o directo.

El acceso directo a un registro de una base de datos:



Problema: ¿ Como localizar un registro en un bloque?

Solución: El RID (Record identification) que actúa como dirección del registro



3.3.1. El Gestor de disco del S.O.

Organiza la información en archivos a nivel de SO, cada uno formado por páginas.

BD	⟺	*Cjto. Archivos SO*
Archivo SO	⟺	*Cjto. Bloques*
Bloque	⟺	*Cjto. Sectores*

- *Implementado en el SO.*
- *Lo usan la mayoría de SGBD. Excepciones aportan eficiencia pero restan portabilidad.*
- *Gestiona el espacio libre. Bloques libres.*
- *Ofrece funciones al adm. de almacenamiento del SGBD:*
  - *Crear nuevo archivo de SO.*
  - *Obtener bloque b del archivo a.*
  - *Reemplazar bloque b dentro del archivo a.*
  - *Añadir bloque b nuevo al archivo a.*
  - *Eliminar bloque b del archivo a.*
  - *Eliminar un archivo.*

3.3.2. El administrador de almacenamiento del SGBD.

- *También denominado gestor de archivos del SGBD.*
- *Correspondencia entre campos, registros y archivos con bloques y archivos (cjto. de bloques) del gestor de disco del SO.*
- *Se reserva espacio para reducir la fragmentación y satisfacer necesidades de almacenamiento.*
- *Optimizados para un acceso flexible y eficiente a los datos.*
- *Debe permitirse el acceso secuencial y directo (RID) a los registros.*
- *Funciones básicas:*
  - *Crear un nuevo archivo almacenado a.*
  - *Eliminar archivo almacenado a. Deja libre páginas de BD.*
  - *Recuperar registro r del archivo almacenado a. RID->Página*
  - *Añadir registro al archivo almacenado a. Devuelve el RID.*
  - *Eliminar registro r del archivo almacenado a.*
  - *Actualizar registro r en el archivo almacenado a.*

3.4. Representación de la BD en el nivel interno.

Misión n. interno: Traducir estructuras del n. conceptual a intermedias cercanas al n. físico optimizando operaciones de entrada/salida a disco.

Optimización del acceso a un disco.

Problema: ***Mejorar la velocidad de acceso a los bloques de un determinado disco***

Técnicas de mejora "físicas" (dependientes del sistema operativo):

- *Planificación:* ***el controlador del disco se encarga de organizar las peticiones de acceso de forma que la cabeza oscile lo menos posible***
- *Utilización de memoria intermedia no volátil:* ***las actualizaciones se graban en una memoria más rápida y se van procesando "off-line"***
  - ***Memoria RAM***
  - ***Disco secuencial***

Técnicas de mejora "lógicas" (asociadas a la organización de ficheros)

- *Acceso secuencial en bloque contiguos*
- *Agrupamientos:* ***Técnicas de agrupamiento de registros en las páginas de nivel interno.***
  - ***Intra-Archivo: Mismo tipo misma página.***
  - ***Inter-Archivo: Diferentes archivos misma página. Debe existir relación entre los registros de distintos tipos.***

3.4. Representación de la BD en el nivel interno.

El modelo de datos físico.

Elementos constituyentes:

- Campo
- Registro
- Archivo

Tipos de registros

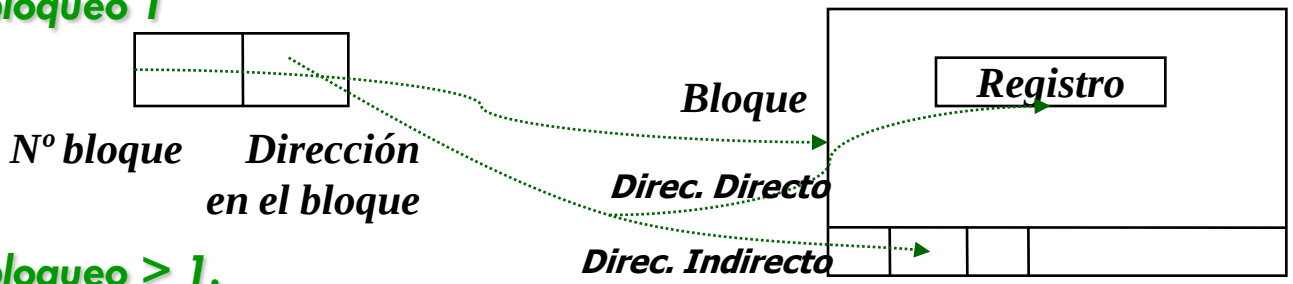
- Longitud fija: número fijo de campos de tamaño fijo (archivos planos)
- Longitud variable:
  - Campos de longitud variable (archivos de texto, representación en cadenas de caracteres...)
  - Numero variable de campos de longitud fija (archivos no planos)

  Se pueden representar como de longitud fija:
  - Reservando espacio máximo
  - Mediante listas lineales

3.4. Representación de la BD en el nivel interno.

Representación con factor de bloqueo 1

- ✓ Cuadricula completa.
- ✓ Uso de punteros.
- ✓ Detalle de la página 0.



Representación con factor de bloqueo > 1.

- RID: Número página + Direccionamiento indirecto.
- Vector de acceso a registros.
  - ✓ Ofrece flexibilidad. Cambios de posicionamiento en la página no provocan cambios de RID solo cambios en la ref. del vector.
  - ✓ Se emplea en estructuras asociadas: índices, tablas hash, árboles de búsqueda,…
  - ✓ Cada SGBD tiene su propia organización. Oracle: tablespaces, segmentos, extensiones y bloques.

Conclusiones:

- No existe relación directa fichero almacenado/físico. Es sw del n. interno absorbe estas diferencias.
- Barrido secuencial muy costoso -> Técnicas de direccionamiento.

3.5. Métodos de organización y acceso a los datos.

Organización de los registros en los ficheros:

Criterio o mecanismo para situar un registro dentro de un fichero. Está fuertemente ligada a los problemas de acceso a un fichero.

Tipos básicos de organización

- **Organización "en pila":**

  Los registros se almacenan conforme van llegando. En sucesivas etapas se reorganizan para compactar el fichero pero sin ningún criterio específico.

- **Organización secuencial:**

  Los registros se almacenan ordenados según el valor de un determinado "campo clave" que normalmente los identifica unívocamente.

- **Organización en acceso directo**

  Los registros se ordenan en el fichero según la transformación del valor de un "campo clave" en una "dirección", mediante una función adecuada.

3.5. Métodos de organización y acceso a los datos.

Problema Básico:

Dada "información lógica" acerca de un determinado registro (el valor de un cierto campo), ¿Cómo localizar dicho registro dentro de un fichero (acceder a él)?.

Enfoques para la solución:

- *Secuencial*
- *Indexación:*
  - *Organización secuencial indexada. Índices densos*
  - *Índices no densos.*
  - *Índices jerárquicos o multinivel.*
  - *Arboles-B*
- *Direccionamiento directo:*
  - *Simple.*
  - *Hashing básico.*
- *Hashing extendido.*
- *Estructuras Multilistas.*

3.5.1. La organización secuencial

- *Estructura de los ficheros:*

  Lista lineal de registros ordenados según el valor de un "campo clave" (clave física). Para acceder se recorre secuencialmente la lista.

  Un fichero sólo puede estar ordenado por una clave fisíca.

- *Problemas:*
  - ***Búsqueda por clave*** → ***Barrido Secuencial. O(N)***
  - ***Búsqueda por rango*** → ***Búsqueda por clave 'cota inferior' + búsqueda secuencial a partir de esta. O(N)***
  - ***Inserción*** → ***Búsqueda secuencial +posible nuevo bloque***

    La inserción obliga a reordenar la lista pág. desbordamiento o nuevo bloque
  - ***Borrado*** → ***Búsqueda secuencial***

    El borrado obliga a reordenar la lista

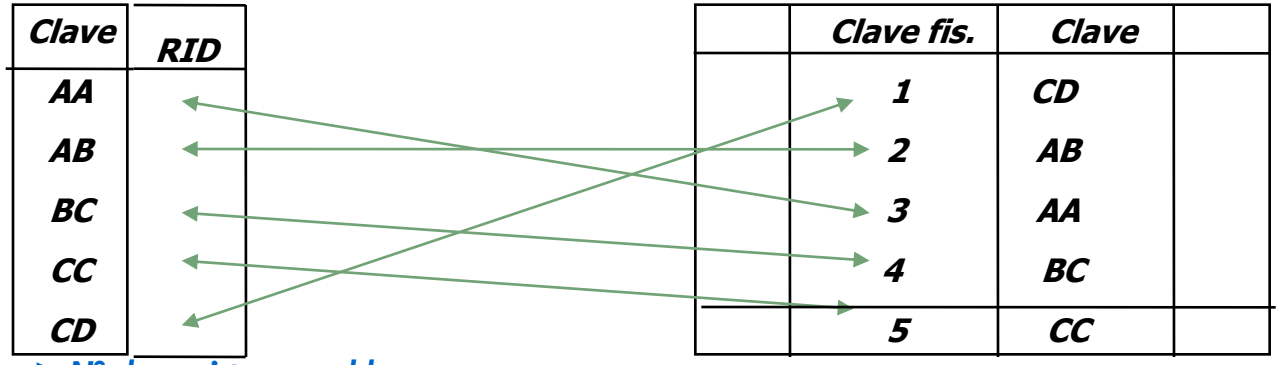
  La organización secuencial "pura" obliga a repetidas reorganizaciones de los ficheros

- *Solución*

  Estructuras adicionales que aceleren la localización y transferencia de datos.

3.5.2. La indexación. Fichero Secuencial Indexado (Índices densos).

El objetivo es disminuir el tiempo de acceso mediante clave. Para ello se utiliza una tabla índice donde se almacena la clave y el RID del registro ordenada por el campo clave.



- *Nº de índices por bloque > Nº de registros por bloque*
- *Dos tipos de índices: Primarios (por campo clave) y secundarios (por otro, los valores se pueden repetir).*
- *Recorrido de registros por índice primario siempre será más rápido que por índice secundario.*
- *Índices simples (un solo campo) e índices compuestos (varios campos).*

Operaciones

- *Búsqueda por rango* → *Barrido Secuencial del índice + Acceso a disco para obtener el bloque que contiene el registro. Seguiremos hasta obtener el valor superior del intervalo.*
- *Inserción* → *Inserción del registro (igual que en el caso secuencial)+ inserción del índice.*
- *Búsqueda por clave* → *Barrido Secuencial del índice + Acceso a disco para obtener el bloque que contiene el registro.*
- *Borrado* → *Borrado del registro + borrado del índice.*

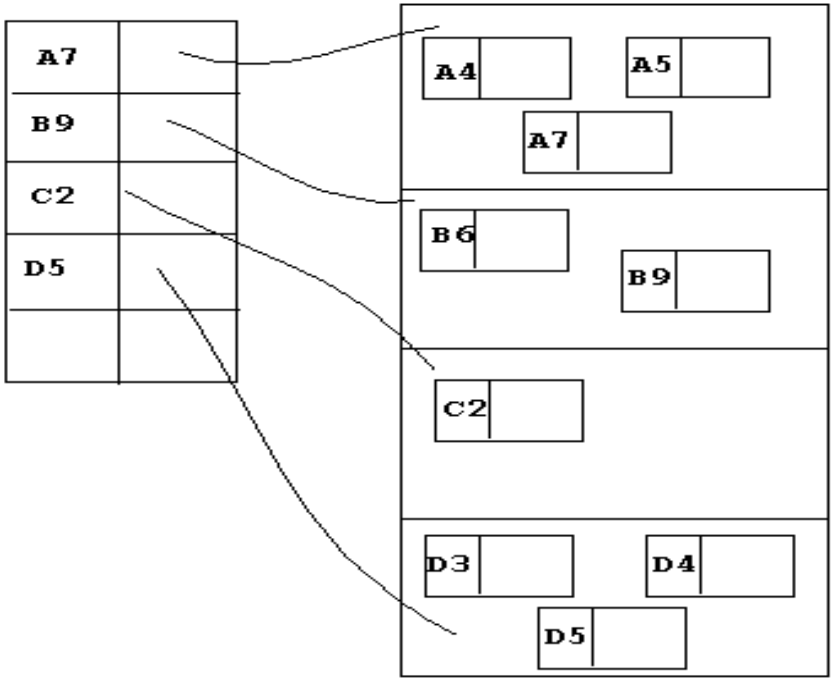
Conclusión: Los índices aceleran el acceso a los datos pero penalizan el borrado y la inserción. Hay que crear un nº de índices por fichero adecuado.

3.5.2. La indexación. Índices no densos.

Una tabla índice es a su vez un fichero con organización secuencial

Problemas con el uso de tablas índices:

- ***Reducir el tamaño de los índices.***
- ***Actualización del fichero implica actualización de los índices.***



Indices no densos

3.5.2. La indexación. Índices no densos.

- *Formados por pares: clave de búsqueda, dirección de comienzo del bloque donde puede estar el registro consultado.*
- *Los índices no densos sólo se pueden construir sobre la clave física.*
- *Se carga el bloque, y se realiza búsqueda secuencial en mem. principal.*
- *Búsquedas existenciales necesitan al menos una lectura de disco.*
- *Las modificaciones sobre la tabla índice son menos frecuentes. Solo cuando afectan al valor menor dentro de un bloque de datos.*

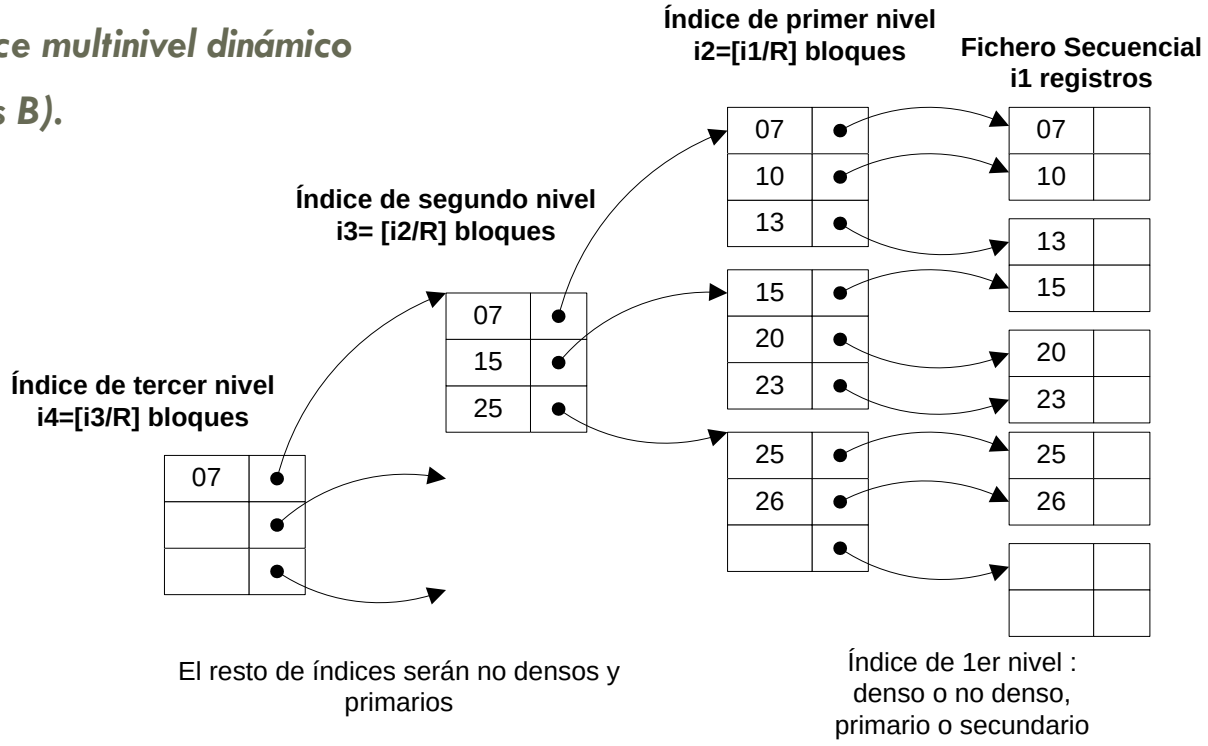
Problema: Búsquedas en tablas grandes sobre valores no indexados.

- ***Técnicas de Búsqueda***
- ***Indices multinivel***
- ***Arboles B***

3.5.2. La indexación. Índices jerárquicos o multinivel.

Idea Básica: ***Tratar la tabla índice como un fichero e indexar de forma no densa.***

- ✓ ***Se optimiza la búsqueda por clave: con N niveles se harán N accesos a disco en el peor caso.***
- ✓ ***Al ser todos los ficheros ordenados se mantienen los problemas en inserciones y borrados.***
- ✓ ***Para resolverlos se utiliza una técnica que deja espacio en cada uno de los bloques: Índice multinivel dinámico (implementado mediante árboles B).***



R = registros índices por bloque

3.5.2. La indexación. Indexación por árboles B.

Árbol equilibrado cuyos nodos son páginas.

Árbol-B = Índices multinivel + estructura de árbol.

Al igual que en el caso de multinivel, buscamos restringir el proceso de búsqueda y minimizar el nº de bloques consultados en disco.

Árbol-B de orden N: n-1 valores clave y n punteros por nodo.

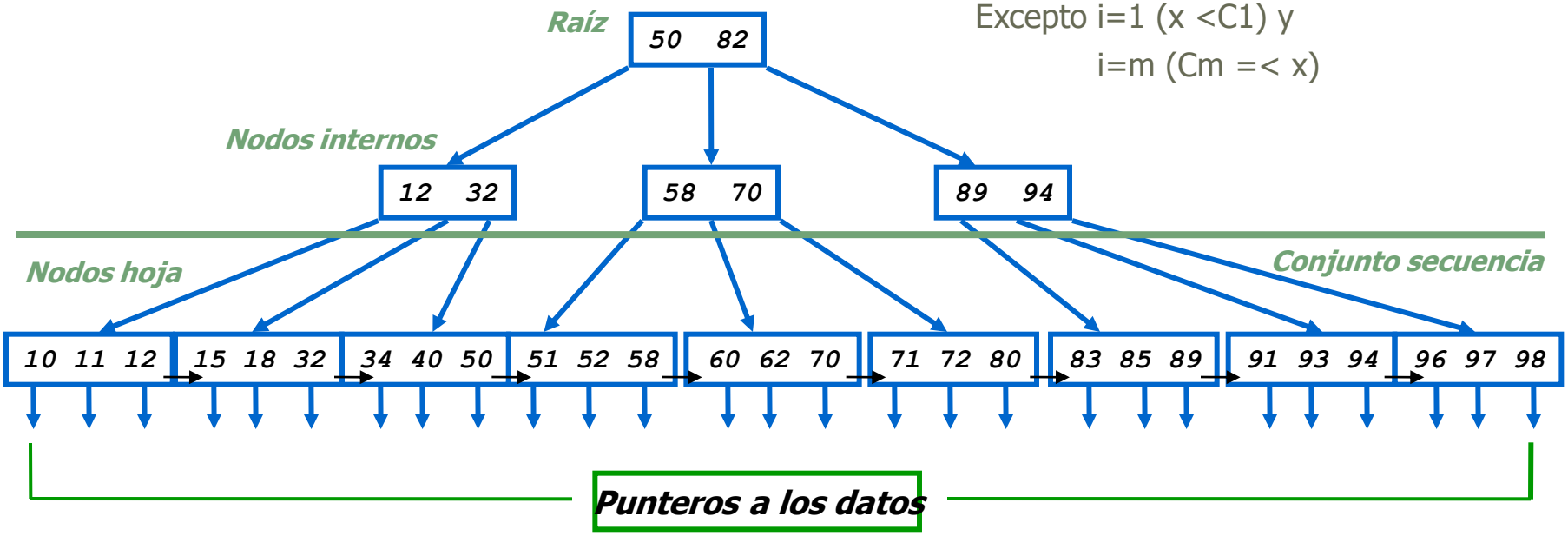
Restricciones:

- Los valores están ordenados dentro del nodo.
- Al menos la mitad de punteros usados por nodo.
- Para todo valor x del árbol apuntado por Pi se cumple

Ci-1 =< x < Ci.

Excepto i=1 (x <C1) y

i=m (Cm =< x)



3.5.2. La indexación. Indexación por árboles B.

Algunas consideraciones

- ***Se puede montar el árbol B no sobre el índice sino sobre el fichero de datos directamente***

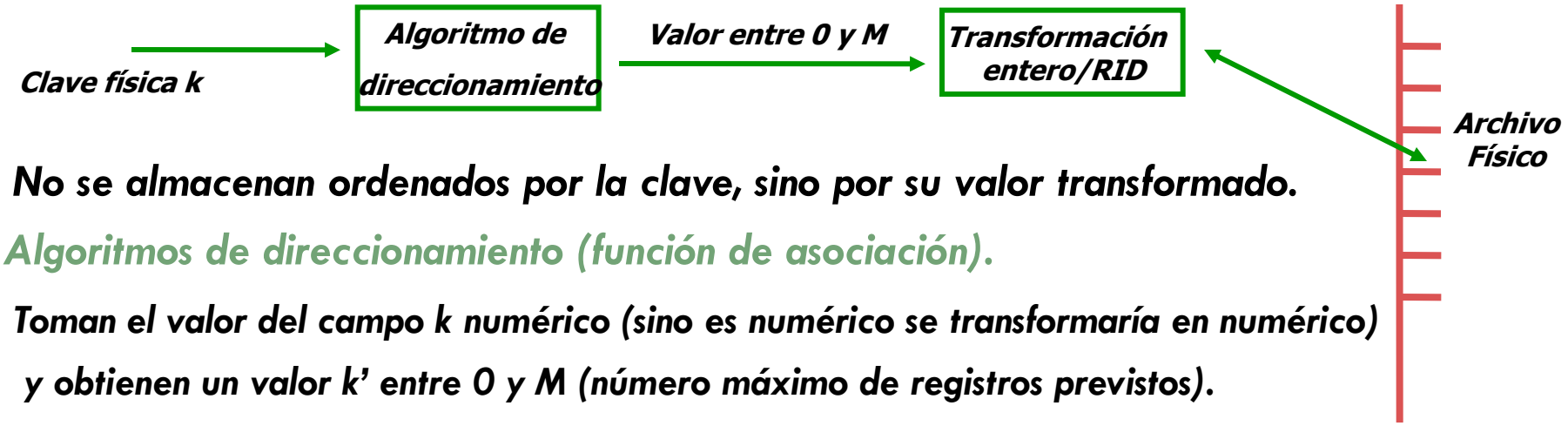
  → *La clave ha de ser física*

- ***Usualmente no todas las páginas han de estar llenas. Al menos la mitad de cada nodo debe estar llena.***
- ***Existen algoritmos para la búsqueda, inserción y borrado de un dato, que garantizan que el árbol resultante sea equilibrado.***
- ***Existen muchas variantes y mejoras de la estructura de árbol B que tratan de adaptarse a la forma del fichero.***
- ***El árbol B es la idea clave de la organización VSAM (IBM 1975, 76..)***

3.5.3. El direccionamiento directo.

- *Idea básica:*

  Se calcula la dirección de un registro, a partir de un campo de este (campo clave), por medio de un algoritmo ó función.



  No se almacenan ordenados por la clave, sino por su valor transformado.

- *Algoritmos de direccionamiento (función de asociación).*

  Toman el valor del campo k numérico (sino es numérico se transformaría en numérico) y obtienen un valor k' entre 0 y M (número máximo de registros previstos).

- *Algunos algoritmos.*
  - ***Cuadrados Centrales. Se eleva la clave al cuadrado y se varios tantos digitos centrales.***
  - ***Congruencias. Se divide la clave por M y se toma el resto.***
  - ***Desplazamientos. Para alfanuméricos. Se superponen los digitos binarios de la clave y se suman.***
  - ***Conversión de base. Cambio de base y se suprimen algunos dígitos del resultado.***

3.5.3. El direccionamiento directo

- *Problemas con el direccionamiento directo*
- *Problema Básico:*

  Los valores del campo clave no están uniformente distribuidos y por tanto los valores de k' tampoco lo están.

  ↓

  Los registros se "acumulan" en determinadas zonas del fichero, no están uniformemente distribuidos.

  - *Existencia de huecos.*
  - *Existencia de "colisiones" o sinónimos.* → *Zona de desbordamiento.*
- *Solución: Hashing*

  El fichero se "desmenuza" en "cubos" (páginas) y se direcciona a cubos no a registro.

  Si un cubo está lleno → *"Cubos de overflow"*

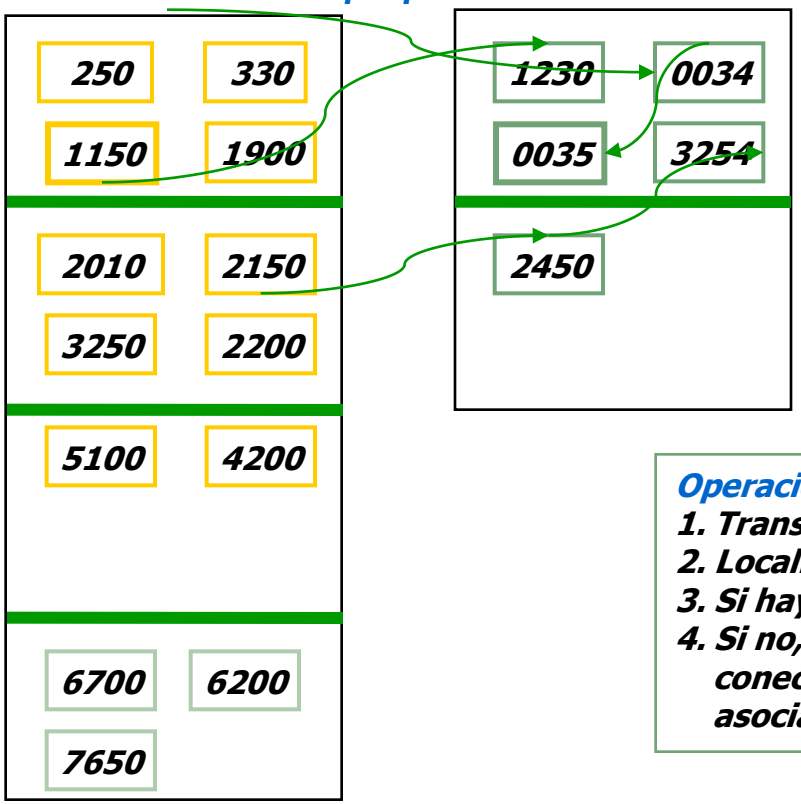
3.5.4. El direccionamiento directo. Hashing básico o estático.

Se pretende asignar más espacio a los rangos de valores donde aparecen más valores de la clave.

Elementos del Hashing: *Número de cubos* *Número de cubos de overflow* *Dígitos de la clave*

Número de registros por cubo *Algoritmo de direccionamiento*

Ejemplo:



Elementos del Ejemplo:

- ✓ ***4 cubos***
- ✓ ***2 cubos de overflow***
- ✓ ***Clave K de cuatro dígitos [0-7999]***
- ✓ ***4 registros por cubo***
- ✓ ***Algoritmo de direccionamiento:***
  k'=(k div 2000) mod 4

Operación de inserción:

1. *Transformar la clave.*
2. *Localizar cubo asociado*
3. *Si hay sitio se inserta*
4. *Si no, al cubo de overflow conectándolo con el asociado*

Operación de búsqueda:

1. *Transformar la clave.*
2. *Localizar cubo asociado*
3. *Búsqueda secuencial en el cubo*
4. *Si no se encuentra se busca en los cubos de overflow mediante los punteros.*

3.5.4. El direccionamiento directo. Hashing básico o estático.

Problemas con el "hashing":

- Es necesario conocer la distribución previa de las claves.
- Es difícil mantener la secuencialidad.
- Sigue existiendo el riesgo de colisiones.
- Al aumentar el número de registros aumenta el tiempo de acceso a través de listas lineales.
- Se hace necesarias las reorganizaciones.

Hashing Dinámico

3.5.4. El direccionamiento directo. Hashing extendido o dinámico.

Ideas básicas:

- ***Se direcciona a una tabla índice auxiliar que mantiene los punteros a las páginas.***
- ***Esta tabla suele estar en memoria. Debemos diseñar la estructura para que sea posible.***
- ***La asignación de páginas a un rango de valores de clave se hace dinámicamente generando páginas conforme se va necesitando.***

3.5.4. El direccionamiento directo. Hashing extendido o dinámico.

Datos de partida:

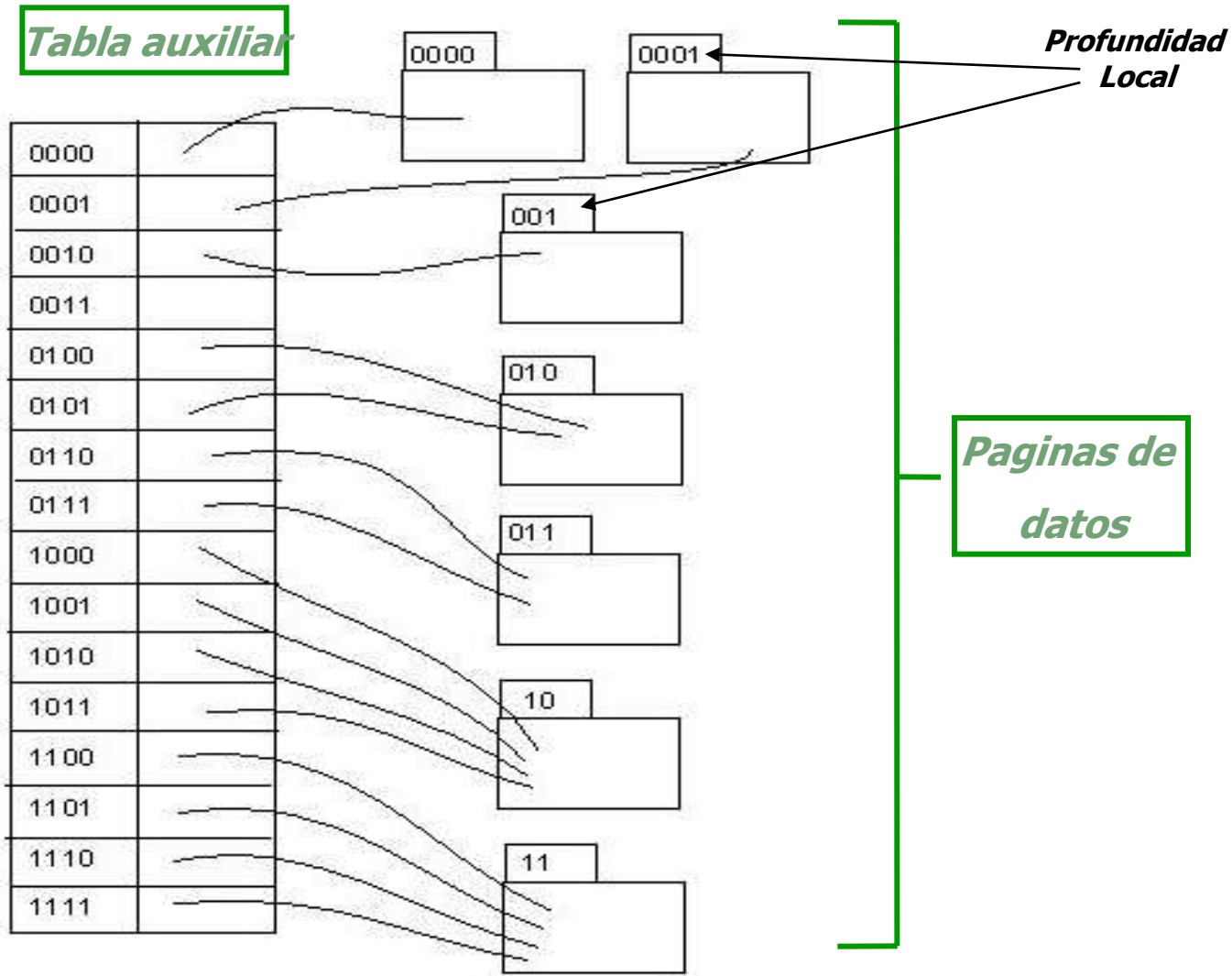
- ✓ *k= clave física para direccionar*
- ✓ *k'= h(k) valor entero entre 0 y M*
- ✓ *n= número de bits que tiene k' en binario*
- ✓ *d <=n, los d primeros dígitos de k' seleccionan la página donde está el registro y se llaman pseudollave.*
- ✓ *b<d<=n, inicialmente el archivo tiene 2^b cubos distintos , como máximo tendrá 2^d*

Algoritmo

- ❑ ***Se considera una tabla índice en memoria con 2^d filas , en la primera columna de esta tabla, (valores de campo llave) se sitúan todas las posibles sucesiones de d dígitos binarios. d es la "profundidad" de la tabla***
- ❑ ***En principio, todas las entradas cuyos b primeros dígitos son iguales apuntan al mismo cubo. Allí se almacenan los registros cuyo valor de k' tiene esos b primeros dígitos. Todos los cubos tienen en principio "profundidad local" igual a b.***
- ❑ ***Cuando se llena un cubo se divide en 2, poniendo en uno de ellos los registros con el dígito b+1 de k' a 0 y en otro los que lo tienen igual a 1. La profundidad local de estos cubos aumenta una unidad.***

3.5.4. El direccionamiento directo. Hashing extendido o dinámico.

Ejemplo:



3.5.5. Estructuras "Multilista".

Ideas básicas:

- *Se conectan mediante una lista cerrada todos los registros que:*
  1. *O tienen un mismo valor en un determinado campo,*
  2. *O que se relacionan con un determinado registro de otro fichero.*
- *En cualquier caso se consideran dos ficheros conectados por un conjunto de listas cerradas.*
- *El fichero "básico" (hijo, miembro etc..) se puede estructurar a su vez con alguna organización conocida.*
- *El fichero de valores de búsqueda (caso 1) , o el fichero que sirve para organizar la lista (caso 2) se denomina fichero padre, propietario etc.. Y se puede estructurar también con alguna organización adicional.*
- *Se pueden contemplar varios "padres" para un mismo fichero básico o "hijo".*

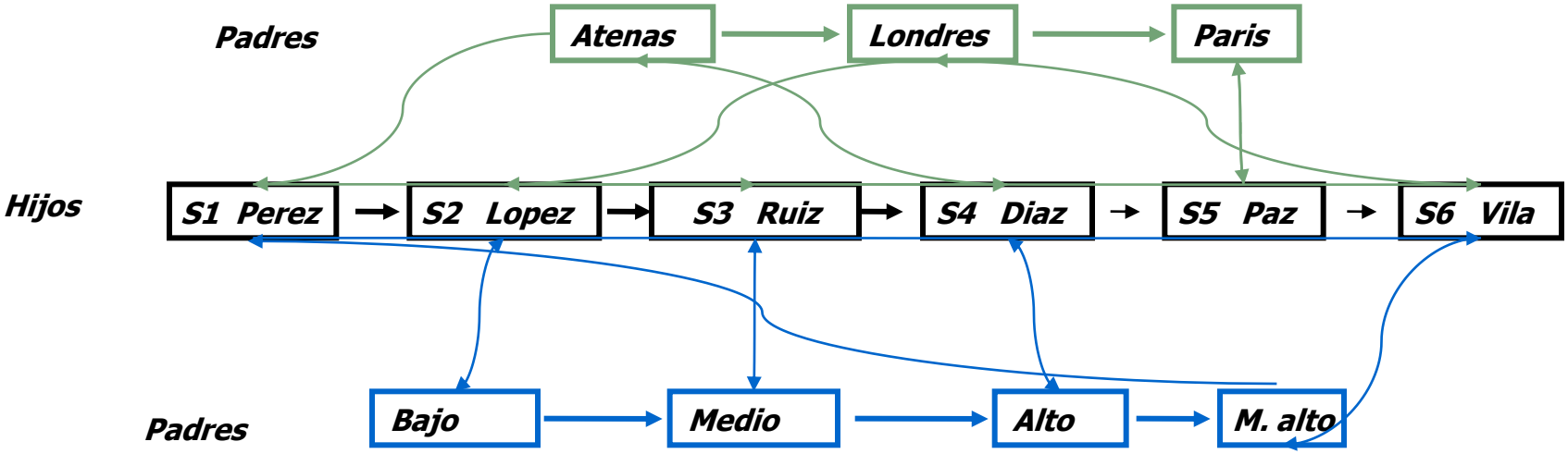
3.5.5. Estructuras "Multilista".

Estructuras "Multilista":

Ejemplo (caso 1):

Fichero proveedores

Código	Nombre	Nivel	Ciudad
S1	Perez	Muy Alto	Atenas
S2	Lopez	Bajo	Londres
S3	Ruiz	Medio	Atenas
S4	Diaz	Alto	Atenas
S5	Paz	Muy Alto	Paris
S6	Vila	Muy Alto	Londres



3.5.5. Estructuras "Multilista".

Ventajas del sistema:

- *Facilidad para ciertos accesos (padre → hijo).*
- *Economía de espacio, al eliminar redundancia.*
- *Verdadera estructura de base de datos.*

Inconvenientes:

- *Problemas con otros accesos (hijo → padre).*
- *Problemas de mantenimiento.*
- *Siempre hay que diseñar un acceso alternativo de los ficheros implicados.*

Soluciones alternativas

- *Suponer todos los punteros dobles.*
- *Duplicar información, almacenando el registro completo en el fichero hijo o base.*

3.5.5. Otras organizaciones en bases de datos

Bases de datos con estructura de grafo

- *Utilización de estructuras multilistas para conectar "físicamente" los ficheros.*
- *Utilización de acceso sofisticados para la llave física.*

Bases de datos relacionales

- *Las conexiones entre ficheros se mantienen utilizando el modelo de datos.*
- *El acceso a los ficheros se hace mediante tablas índices densos organizadas cómo B-árboles.*
- *No existe representación secuencial por llave física.*
- *En ocasiones se utiliza en el agrupamiento.*