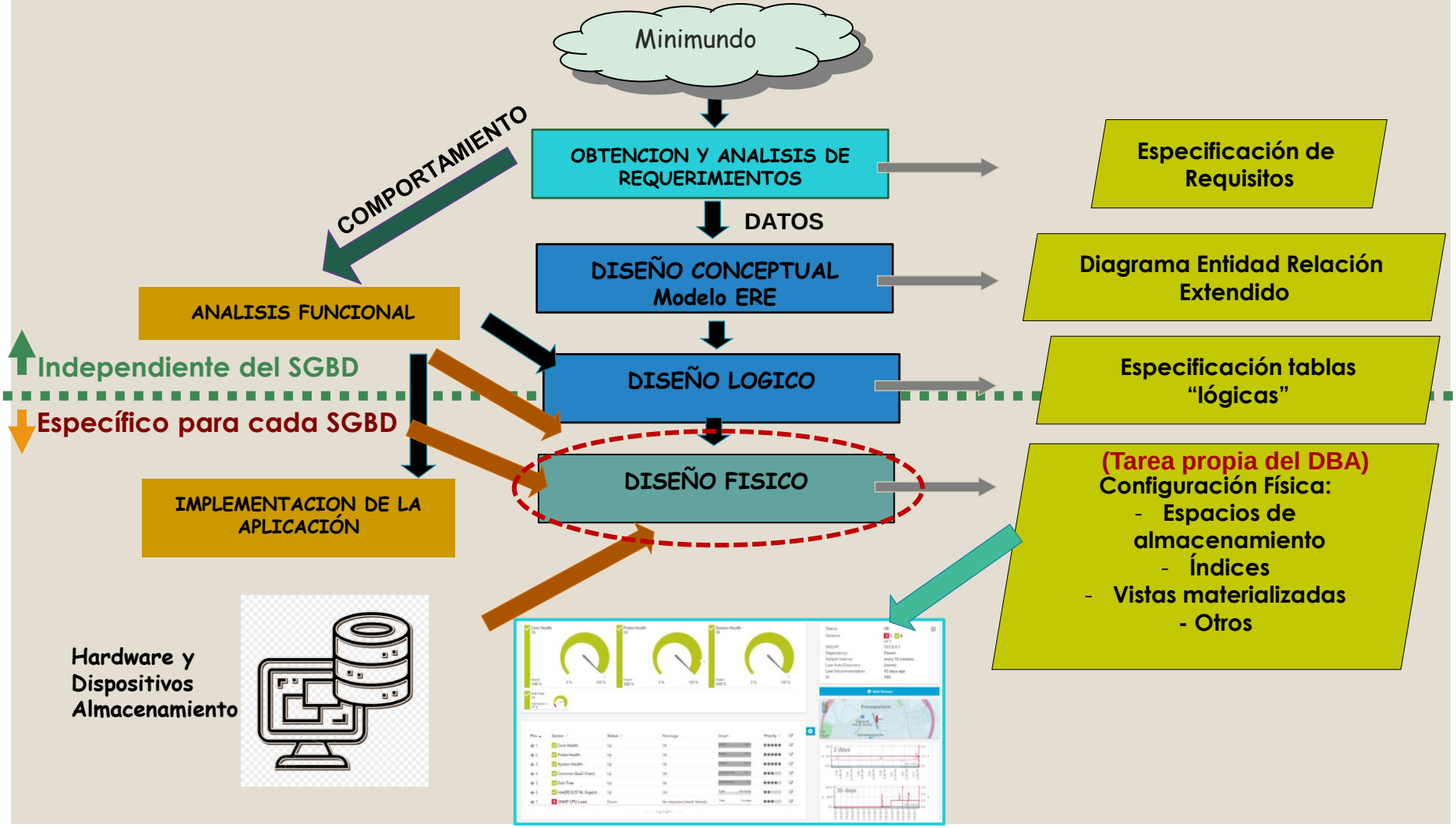
UNIDAD

Diseño Físico
de una Base de Datos

Proceso de Construcción de una BD
DISEÑO FÍSICO



DISEÑO FÍSICO DE UNA BASE DE DATOS

OBJETIVO:

Garantizar que el SGBD permita gestionar

la/s bases de datos

con una **PERFORMANCE APROPIADA**

(**tiempos de respuesta**)

Diseño Físico
Performance del SGBD

Factor clave en relación a los tiempos de respuesta:

Cantidad de Operaciones E/S

- ***Operaciones de Transferencia entre MP (buffers) y Disco*** –

Organización de Archivos/Ficheros
(Organización Primaria)

Forma en la que se estructurar los archivos en el disco

Estructuras de Acceso
(Organización Secundaria)

Proporcionan un **medio para llegar a los datos almacenados** (archivos)

Métodos de Acceso

Proporcionan el acceso a los datos haciendo uso de las estructuras de acceso, si existen

Minimice la cantidad de transferencias de páginas

necesarias para encontrar el dato que se necesita

Requisitos para realizar el Diseño Físico (DBA)

Conocimiento **profundo** del SGBD donde se implemente la BD:

- Los diferentes tipos de organización de archivos/ficheros
- Tipos de índices disponibles (estructuras de acceso auxiliares)
- Tipos de datos ofrecidos
- Soporte de integridad referencial (Clave Foráneas)
- Parámetros de configuración: Ej. Tamaño de página (en PostgreSQL, el tamaño por defecto es 8kbytes)
- Cláusulas SQL disponibles para el diseño físico no presentes en el estándar

- ❖ MAXIMICEN LA EFICIENCIA DE LAS OPERACIONES (CONSULTAS, ETC.) MÁS FRECUENTES
- ❖ EQUILIBREN REQUISITOS CONTRAPUESTOS

"Profundicemos" en los elementos relativos al Diseño Físico

A. Organización Primaria

Determina la forma en que los registros del fichero/archivo se colocan físicamente en el disco y cómo se puede acceder a ellos:

1. Ficheros no ordenados (de montículo)
2. Ficheros ordenados (secuenciales)
3. Ficheros mixtos
4. Ficheros de direccionamiento calculado

B. Estructuras de Acceso - Organización Secundaria (Índices)

Proporcionan caminos de acceso alternativos a los ficheros/archivos primarios (archivos de datos propiamente dichos).

Existen muchos tipos de índices (algunos):

1. Ordenados de un nivel:
   - Primario,
   - De Agrupamiento
   - Secundario
2. De múltiples niveles: Árbol B, B+, B*

A. Organización primaria

1. Ficheros No Ordenados (Montículo)

- Registros almacenados al final del fichero, en orden de inserción
  - **Inserción muy eficiente**
  - Búsqueda lineal
  - Eliminación física o por marca:
    - Reorganización para recuperar espacio desocupado
    - O aprovechar huecos para ubicar nuevos registros
  - Modificación de un registro:
    - Si la longitud es variable puede provocar una inserción y una eliminación
  - Lectura ordenada supone una copia ordenada del fichero
    - Suele utilizarse una técnica de ordenación externa

A. Organización primaria

2. Ficheros Ordenados (secuencial) 1/2

- Registros almacenados en orden según algún campo:
  - Campo de ordenación
  - Si el campo es clave, se llama clave de ordenación

- Ventajas respecto a los ficheros no ordenados:
  - **Lectura ordenada muy eficiente** si el orden es según el campo de ordenación: lectura secuencial
  - **Búsqueda del siguiente** (en el orden del campo de ordenación) no suele necesitar ningún otro acceso a bloque
  - **Búsqueda de un registro dado su valor** del campo de ordenación es rápido: búsqueda binaria
  - Si la lectura ordenada o la búsqueda no están basados en el campo de ordenación, todas las ventajas desaparecen

A. Organización primaria

2. Ficheros Ordenados (secuencial) 2/2

- Inserción:
  - Encontrar posición correcta para el registro
  - Abrir espacio mediante desplazamiento de registros **(costoso!!!)**

Para **aumentar la eficiencia de la inserción**:

  - Dejar espacio libre en cada bloque para nuevos registros
  - Fichero de desbordamiento auxiliar, no ordenado, que periódicamente se ordena y se fusiona con el fichero principal
    - La búsqueda se complica: binaria en principal-secuencial en auxiliar

- Eliminación física o por marca:
  - Con problema de eficiencia similares a la inserción

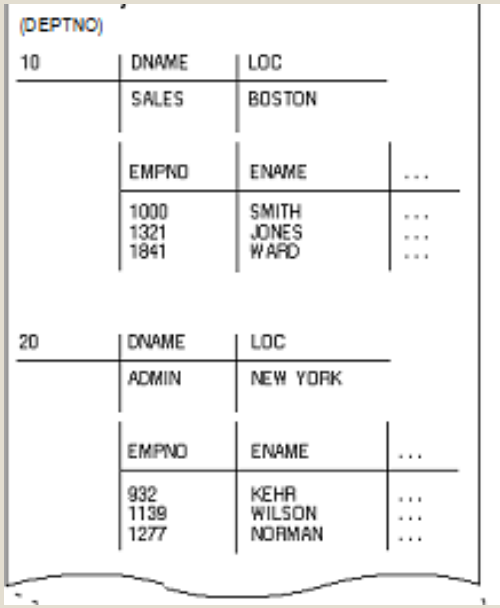
- Modificación:
  - Si la **condición de búsqueda se basa en el campo de ordenación, búsqueda binaria**. De lo contrario, búsqueda lineal
  - Si el campo modificado es el de ordenación, el nuevo valor puede provocar su cambio de ubicación: eliminación + inserción

A. Organización primaria

3. Ficheros Mixtos 1/2

- Los registros provienen de distintas tablas
  - Están relacionadas mediante una o varias columnas
  - Las operaciones más frecuentes son las consultas y no es habitual modificar las columnas comunes
  - Solicitudes frecuentes de filas relacionadas de ambas tablas (JOIN de las tablas)
- En el fichero:
  - Los registros relacionados están físicamente adyacentes
  - Clave del fichero mixto: campos <comunes>de los registros
- Ventajas
  - **Mayor eficiencia de las operaciones con JOIN** entre tablas
  - **Valores de las claves del fichero mixto almacenados sólo una vez**
- Desventaja
  - Muy ineficiente para acceder a los registros de la tabla subordinada (Empleados)

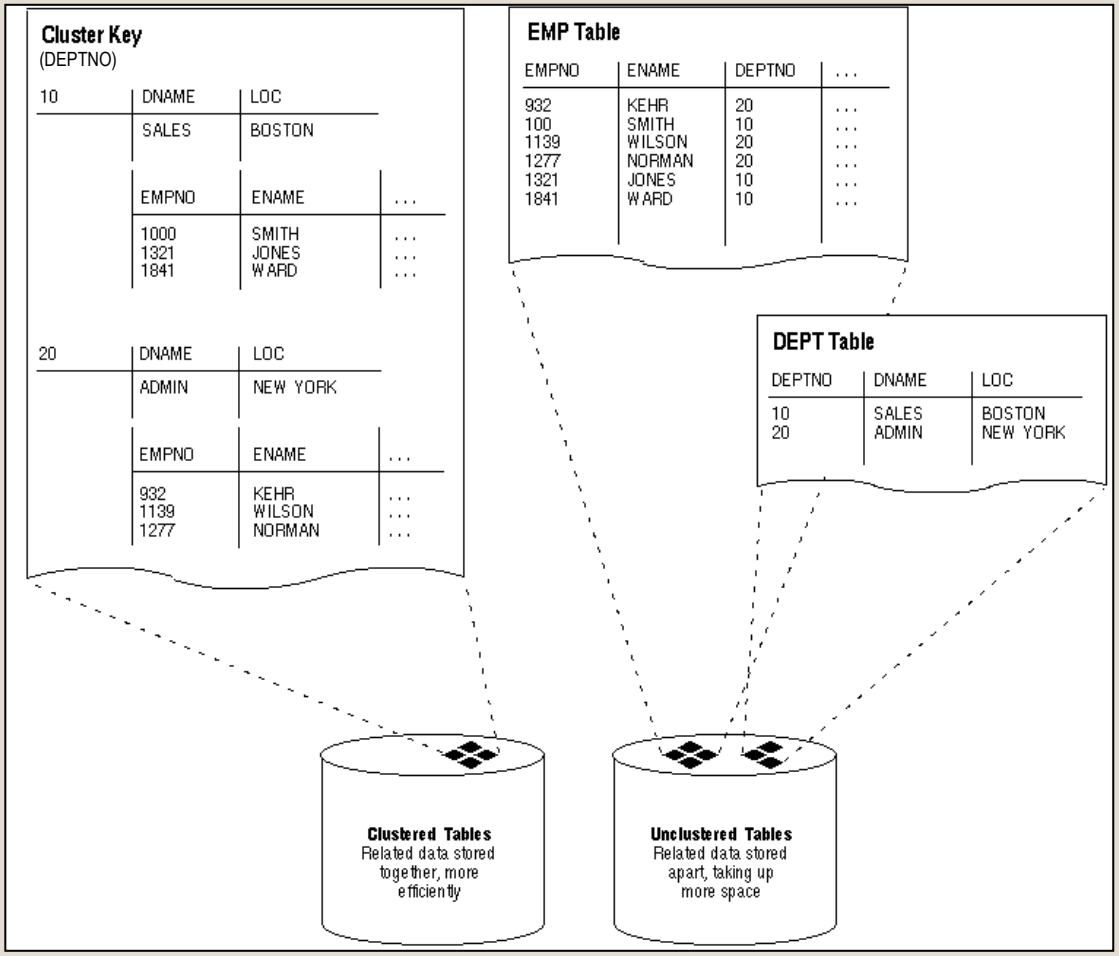
Departamentos-Empleados



A. Organización primaria

3. Ficheros Mixtos 2/2

Ejemplo de uso de un fichero mixto vs. dos ficheros individuales



A. Organización primaria

4. Ficheros direccionamiento calculado (Hashing) 1/2

- Permite acceso rápido a registros según una condición de búsqueda de igualdad sobre un sólo campo:
  - Campo de dispersión
  - Si es clave del fichero, se llama clave de dispersión

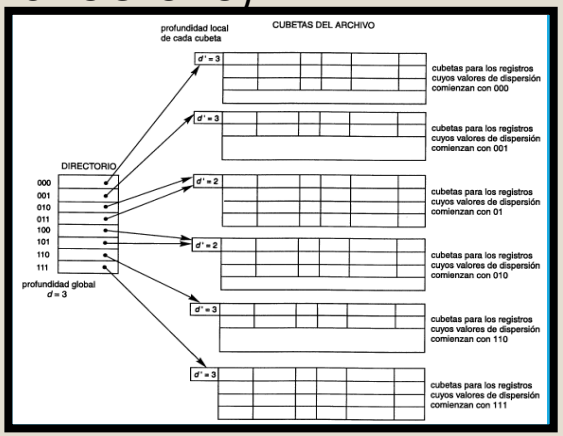
- Función de dispersión h
  - Se aplica al valor k del campo de dispersión de un registro
  - Produce la dirección del bloque de disco en el que está el registro
    - La localización del registro dentro del bloque se hace en el buffer

A. Organización primaria

4. Ficheros direccionamiento calculado 2/2

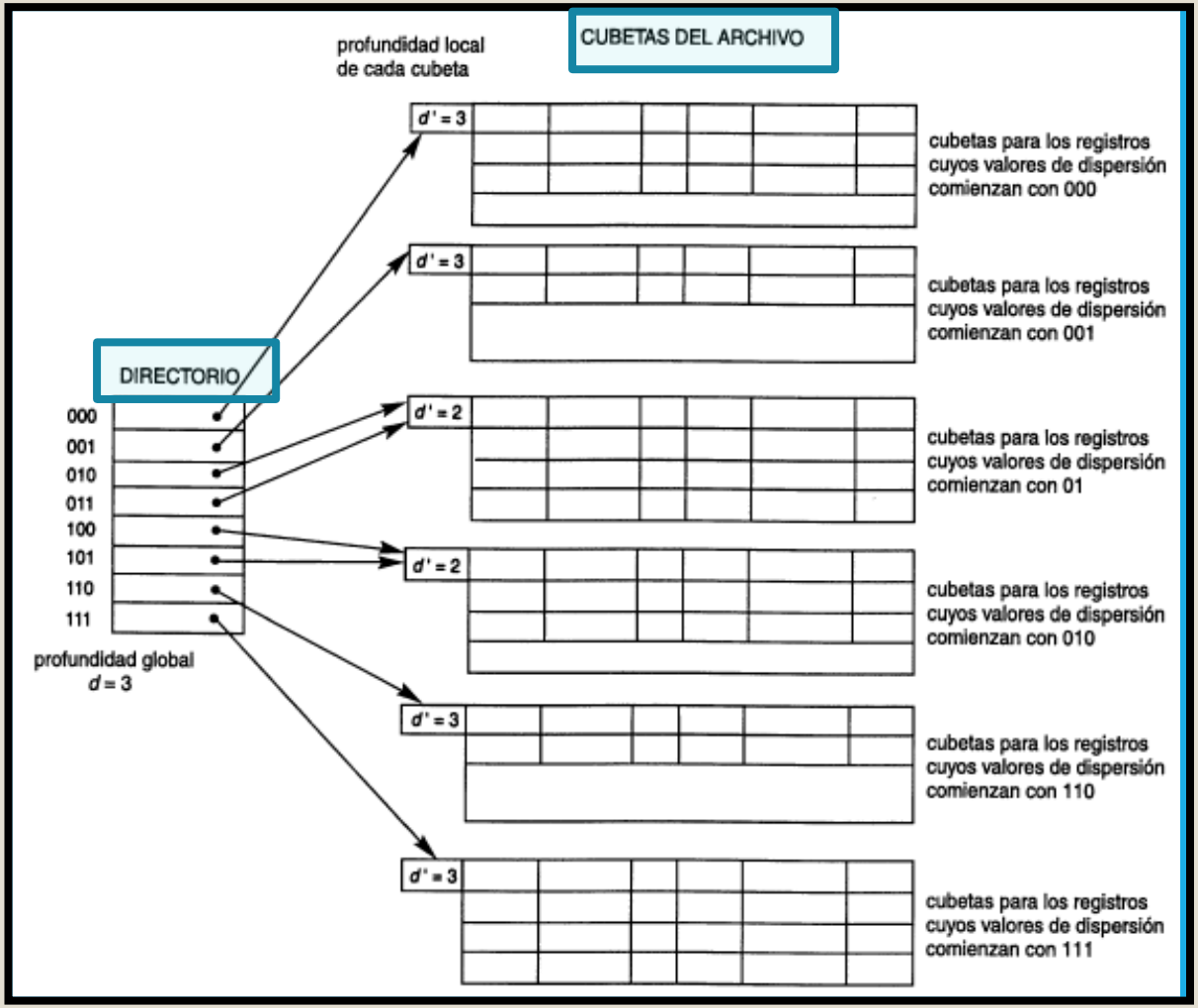
- Desventajas del direccionamiento calculado:
  - No se garantiza que valores distintos de k produzcan direcciones diferentes (**colisiones!!!**)

- Dentro de las técnicas de dispersión:
  - Interna
  - Externa (archivos en disco – cantidad cubetas fijas - colisiones)
  - Dinámica (cantidad de cubetas dinámica sin directorio)
  - **Dinámica extensible:**
    - Fichero + Directorio
      - Directorio: Array de 2^d direcciones de cubeta
      - D: Profundidad global del directorio



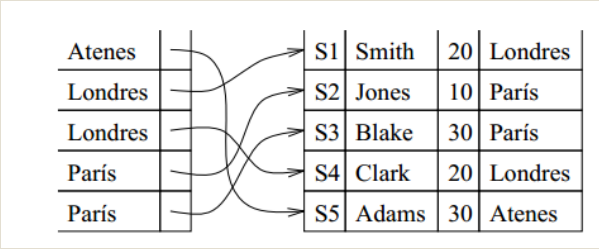
A. Organización primaria

4. Ficheros direccionamiento calculado (Hashing Extensible)



B. Organización Secundaria - Estructuras de Acceso

Índices

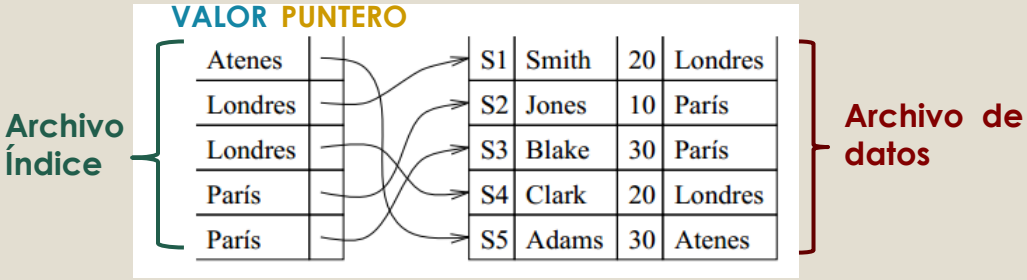


- Un índice es una estructura de fichero adicional:
  - Almacenado en disco
  - Utilizado junto con el fichero de datos (fichero principal)
  - Fichero principal organizado según una organización primaria
- Agiliza la obtención de registros según valores de cierto campo del fichero (campo de indexación): Primero se accede al índice que aporta el puntero del "lugar" donde está almacenado el registro
- Es posible crear:
  - Un índice sobre cualquier campo de un fichero
  - Varios índices sobre un mismo fichero (dependiendo del tipo de índice)
- Un índice puede ser de uno de estos tipos:
  1. Ordenado de un nivel
  2. De múltiples niveles: Arbol B, Arbol B^+

B. Organización secundaria

1. Índices de un nivel 1/2

- Cada registro del índice contiene 2 campos:
  - **Valor del campo de indexación**
  - **Puntero al registro que contiene dicho valor o al bloque que lo contiene**



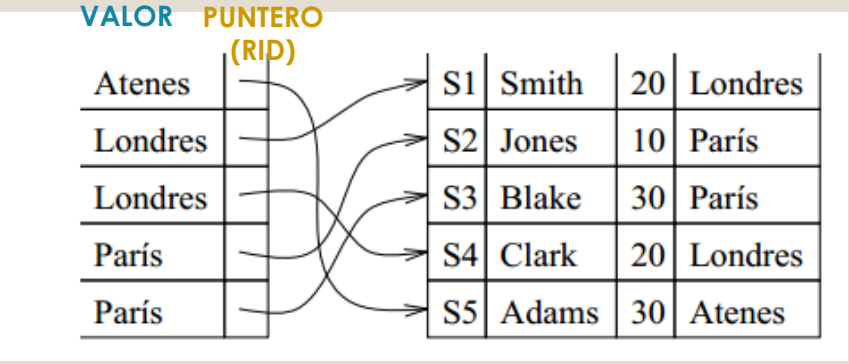
- Las entradas (registros)están ordenadas según valor del campo de indexación, es posible realizar búsquedas sobre el índice
- Operaciones sobre ficheros con índices:
  - Buscar implica búsqueda sobre el índice
  - Insertar y Eliminar pueden provocar modificación del índice
  - **Menos accesos a bloques**, pero existe coste de mantener el índice

B. Organización secundaria

1. Índices de un nivel 2/2

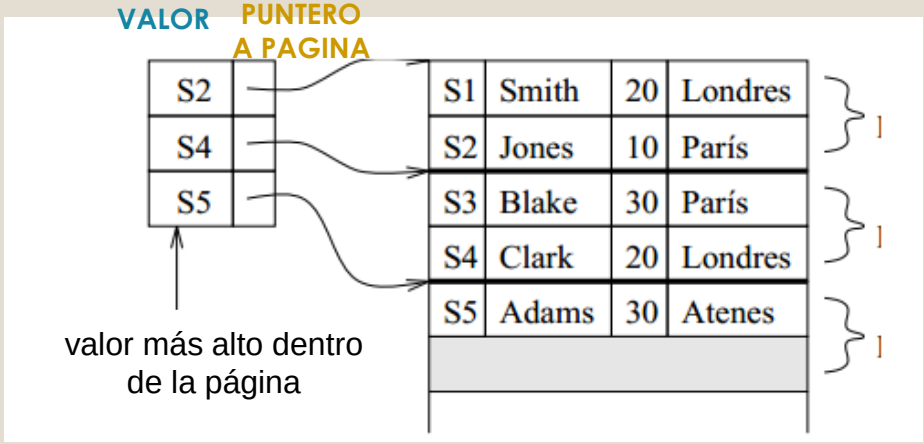
- Índices densos y no densos
  - **Denso** si contiene una entrada **por cada registro** del fichero
  - **En otro caso, es no denso** (o disperso)

Denso



Archivo de datos no ordenado por campo de indexación

No Denso

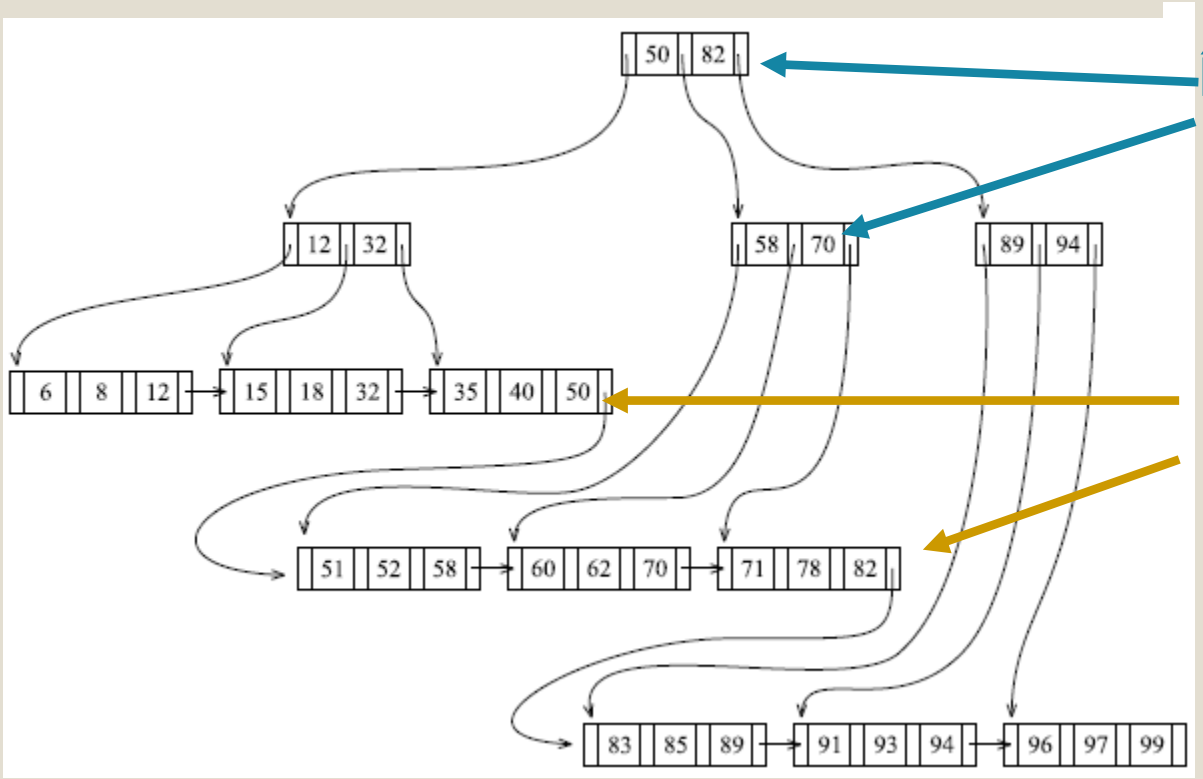


Archivo de datos DEBE estar ordenado por campo de indexación

B. Organización secundaria

2. Índice de múltiples niveles

Árbol B: Agiliza la búsqueda en el archivo índice



¿Cuándo se crea (automáticamente)/debiera crear (manualmente) un índice?

- Conviene **crear un índice** (manualmente) sobre cierto campo **si** se va a acceder con **frecuencia** al fichero según una **condición (SELECCIÓN o JOIN) u ordenamiento** sobre dicho campo

  + El índice incrementará la velocidad de esas consultas o accesos

  - Cada índice sobre una columna hace que la inserción, borrado y actualización sean más complejas y consuman más tiempo

- Normalmente, todos SGBD creará un **índice automáticamente por la clave primaria de cada tabla**

Diseño Físico en PostgreSQL

"Algunos elementos"

Elementos de PostgreSQL relativos al diseño físico

1. Parámetros (existen muchos parámetros que pueden setearse)
   - Tamaño de Página: Puede modificarse al instalar (o recompilar) el SGBD mediante el parámetro **block size**
     - Si se quiere visualizar el valor actual:

       SELECT current_setting('block_size');

Elementos de PostgreSQL relativos al diseño físico

2. Espacios de tablas (tablespaces)

- Los espacios de tablas son lugares dentro del dispositivo de almacenamiento no volátil donde se almacenará la base de datos (carpeta/directorio).
- Sirven para facilitar la gestión de los archivos que constituyen la base de datos.
- Cuando se instala PostgreSQL se crean automáticamente dos tablespaces:
  - **pg_global** se utiliza para catálogos de sistemas compartidos
  - **pg_default** es el espacio de tablas predeterminado a menos que se anule mediante una cláusula TABLESPACE.
- Para visualizar los tablespaces que existen se puede consultar el catálogo:
  - SELECT spcname FROM pg_tablespace;

Elementos de PostgreSQL relativos al diseño físico

2. Espacios de tablas (tablespaces)
   - Definición y uso de espacio de tablas
     - CREATE TABLESPACE [nombre_ts]
       LOCATION '[directorio]';

       Ejemplos:
       - *Creación:*
         ***CREATE TABLESPACE** prueba **LOCATION** 'D:\BANCO';*
       - *Uso explícito:*

         ```
         CREATE TABLE cuenta
         ( nro INTEGER primary key,
           fecha_creacion DATE,
           dni_cliente INTEGER,
           saldo MONEY) TABLESPACE prueba;
         ```

Elementos de PostgreSQL relativos al diseño físico

3. Creación de Índices
   - CREATE [UNIQUE] INDEX [CONCURRENTLY] [nombre_indice] ON nombre_tabla
     [USING metodo] ({column_name} [ASC | DESC] [,...])
     [TABLESPACE tablespace_name];

   - UNIQUE: Permite indicar que la columna indexada no permite valores duplicados.
   - CONCURRENTLY: Permite no bloquear la tabla indexada mientras se crea el índice.
   - Método: Los valores posibles son btree, hash, gist, spgist y gin. El valor por defecto es btree.

     Ejemplos:
     - ***CREATE INDEX*** *indice1* ***ON*** *cuenta (dni_cliente);*
     - ***CREATE INDEX*** *indice2* ***ON*** *cuenta (dni_cliente)* ***DESC;***

Elementos de PostgreSQL relativos al diseño físico

3. Creación de Índices Agrupados/Cluster (Ordenamiento Físico de Archivos)
   - No existe en PostgreSQL la posibilidad dentro de la creación de un índice, indicar que los registros de la table indexada se almacenen según el orden del campo/s de indexación.
   - Para poder obtener esa funcionalidad, luego de crear el índice, se hace uso de la instrucción CLUSTER.

     CLUSTER [nombre_cluster] USING [nombre_indice];

     Ejemplos:

     - ***CREATE INDEX*** *indice1* ***ON*** *cuenta (dni_cliente);* *---- ya lo habíamos creado, pero para recordarlo*
     - ***CLUSTER*** *cluster1* ***ON USING*** *indice1;*

Descansemos

unos minutos…

Después, viene

lo mejor…