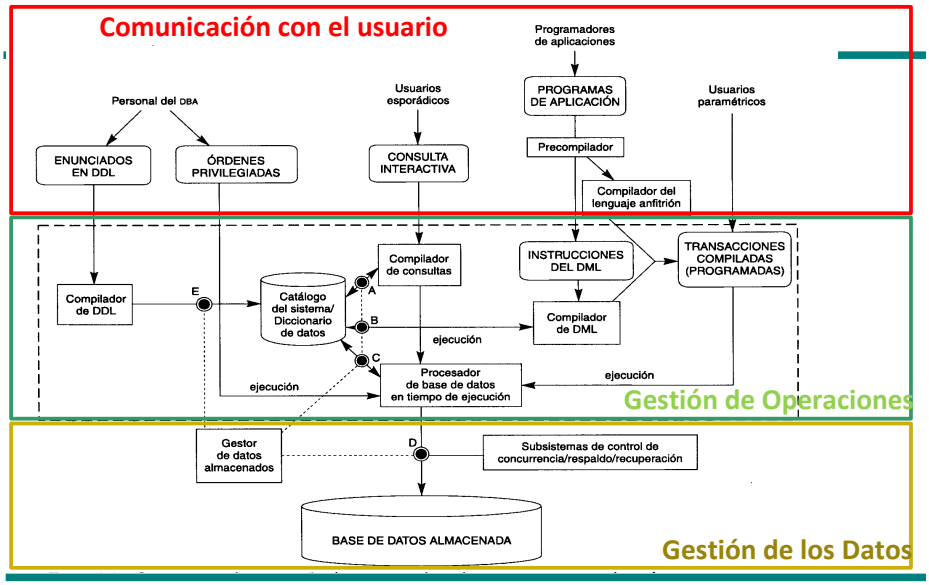
Fundamentos de Bases de Datos

Introducción a las Estrategias de un RDBMS

Arquitectura de un DBMS



Algunos problemas que nos interesan

- Cómo se procesan las consultas?
- Cómo buscar soluciones más eficientes?
- Cómo se resuelve el acceso concurrente?
- Y la recuperación en caso de fallo?

Procesamiento y Optimización de Consultas

- Organización y acceso a los datos
  - Qué estructuras de datos suelen implementar los manejadores y cómo se utilizan.
- Procesamiento de Consultas.
  - Diferentes algoritmos para implementar los operadores del álgebra relacional
- Optimización de Consultas
  - Estrategias que utiliza un manejador para modificar las consultas y para elegir los algoritmos a ejecutar.

Control de concurrencia

- Transacciones
  - Operaciones complejas que se ven como una sola
- Concurrencia
  - Qué debe considerar el manejador para que varias transacciones se ejecuten simultáneamente y produzcan resultados correctos en la base?.

Recuperación

- Estrategias de recuperación
  - Mecanismos que suelen implementar los manejadores para recuperar los datos frente a fallas del sistema.

Organización y Acceso a Datos

Referencias: capítulos 13 y 14 de Fundamentals of Database Systems 4ª edición

Procesamiento y Optimización de Consultas

- Organización y Acceso a los Datos
  - Estructuras de datos que facilitan su acceso.
- Procesamiento de Consultas.
  - Como se implementan los operadores del Algebra Relacional en función de las estructuras anteriores.
- Optimización de Consultas.
  - Dada una consulta, qué técnicas se pueden utilizar para mejorar su performance.

Organización y Acceso a los Datos

- Cómo se deben organizar físicamente los datos para tener la flexibilidad que proporciona un manejador?
- Qué hay a nivel físico?
  - Discos:
    - Conjunto de bloques de tamaño fijo que son direccionables a través de alguna forma de dirección por alguna instrucción específica. (Read)
  - Partición:
    - Subconjunto contiguo de los bloques de un disco.

Organización y Acceso a los Datos

- Qué hay a nivel físico?
  - File Systems (FS)
    - Estructura de datos que facilita el acceso del S.O. a los distintos bloques de la partición. Típicamente se construye con comandos como Format, Newfs, mkfs, etc. dependiendo del sistema operativo.
  - Archivo
    - Estructura lógica que se construye sobre el FS. Es la que usualmente utilizan los programas de aplicación. Típicamente se ven como una secuencia de caracteres o como una secuencia de registros delimitados por algún carácter especial o de tamaño fijo. Típicamente, el acceso siempre es secuencial.

Cómo utilizan los DBMSs esto?

- Típicamente toman una partición del disco o un gran archivo y usan dicha área como su "disco".
- Re implementan :
  - estructuras de datos y sus algoritmos de manipulación
  - algoritmos de ordenación
  - mecanismos de buffering y paginado
- Cuando se habla de estructuras se debe pensar en cómo organizar los datos para una (o más) tablas.

Se reutiliza el conocimiento del área de *sistemas operativos*

Formas de acceder a los registros

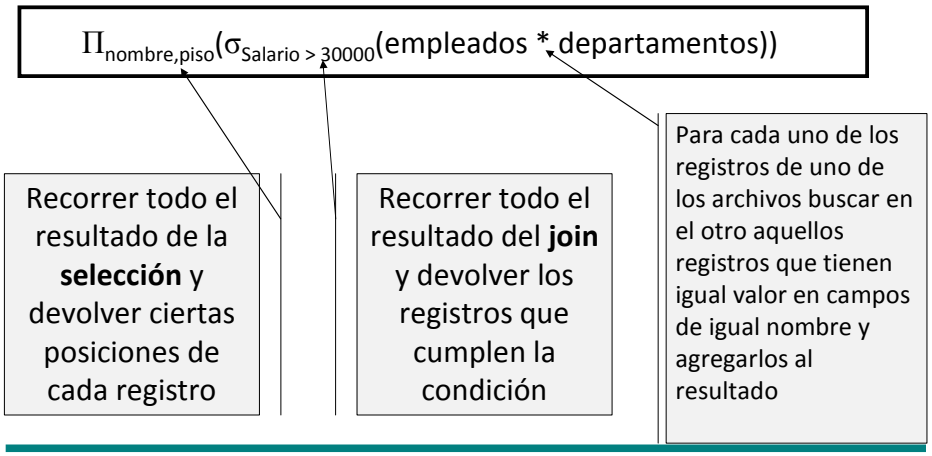
- El acceso secuencial a archivos es bueno para un DBMS?
- Consideremos las siguientes tablas y la siguiente consulta:
  - empleados(nombre,edad,salario,depto)
  - departamentos(nroD,nombreD,piso,gerente)

```
select  e.nombre, d.piso
from    empleados e, departamentos d
where   e.depto = d.nroD
        and e.salario > 30000
```

- Como se ejecutaría esta consulta si sólo contáramos con acceso secuencial a los archivos?

Formas de acceder a los registros (2)

- Supongamos que hay un archivo de empleados y otro de departamentos



Costos de acceso

- En el JOIN, si M es el tamaño de empleados, y N es el tamaño de departamentos → se necesitan M*N operaciones de acceso a disco

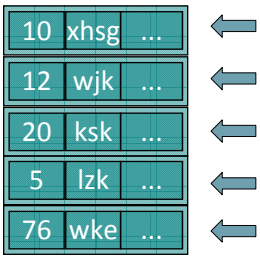
Son muy costosas!!

- Cómo se puede mejorar?
  - Usando otras estructuras de datos sobre disco que faciliten el acceso.

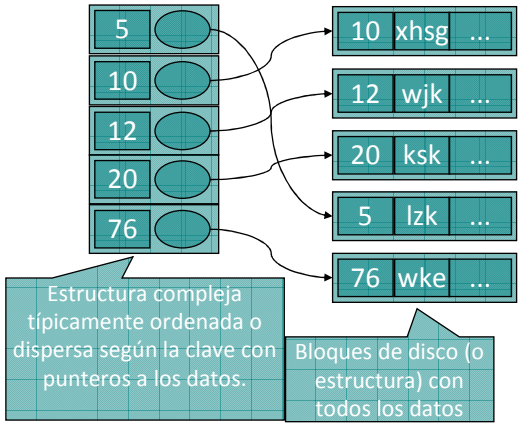
INDICES

Índices

Archivo Secuencial



Archivo Indexado (clave el 1er atributo)



Tipos de Índices.

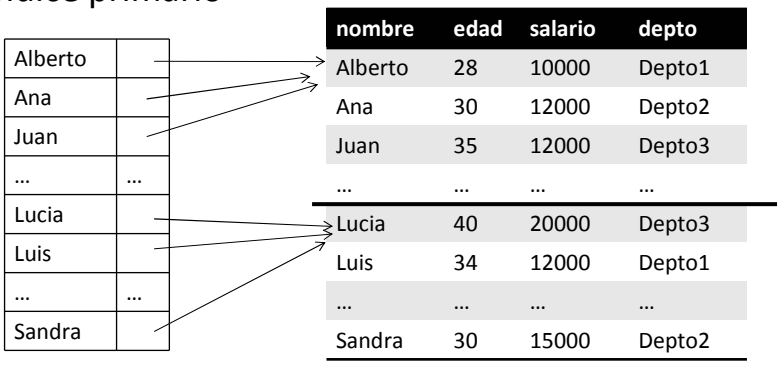
- Físicos vs. Lógicos
  - Referencias a ubicación en disco o referencias a índices físicos
  - Consideraremos sólo los físicos
- Ordenados vs. No ordenados
  - Requieren o no del ordenamiento de los datos por el atributo a indexar.
- Densos o no densos
- Simples vs. Multi-nivel

Índices ordenados

- Índice primario:
  - sobre clave primaria
  - por cada clave: dirección en el disco (bloque o bloque + offset)
- De agrupamiento (cluster index):
  - sobre un atributo que **NO ES CLAVE PRIMARIA**
  - Por cada **valor distinto:** dirección en el disco al primer registro del grupo (bloque o bloque + offset)

Índices con datos ordenados (1)

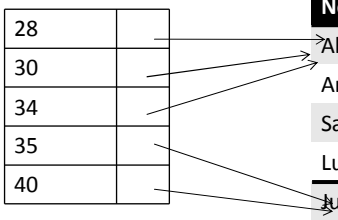
- Índice primario



nombre	edad	salario	depto
Alberto	28	10000	Depto1
Ana	30	12000	Depto2
Juan	35	12000	Depto3
…	…	…	…
Lucia	40	20000	Depto3
Luis	34	12000	Depto1
…	…	…	…
Sandra	30	15000	Depto2

Índices con datos ordenados (2)

- Índice cluster por edad



Nombre	Edad	salario	depto
Alberto	28	10000	Depto1
Ana	30	12000	Depto2
Sandra	30	15000	Depto2
Luis	34	12000	Depto1
Juan	35	12000	Depto3
Lucia	40	20000	Depto3
…	…	…	…
…	…	…	…

Índices no ordenados

- Índices secundarios:
  - Se crea una estructura auxiliar ordenada por el campo a indexar
  - Por cada valor un puntero al bloque donde se encuentra el valor
    - Si el valor no se repite: un solo bloque
    - Si no, una lista. Tengo tantos punteros como bloques contengan al valor.

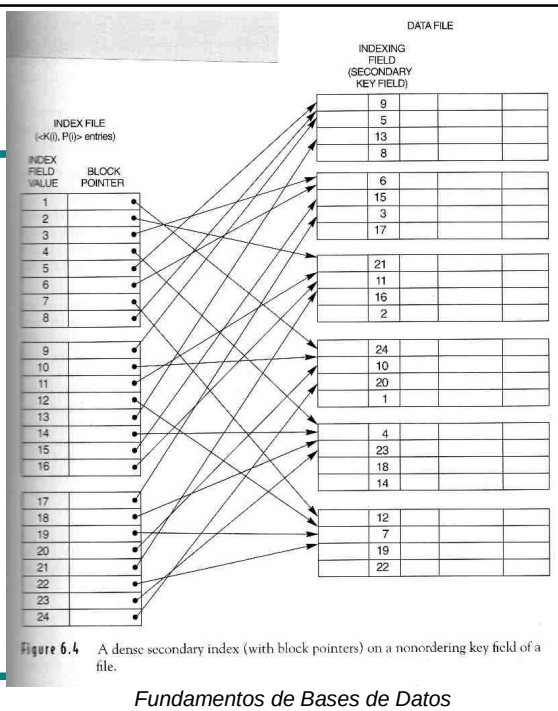


Figure 6.4 A dense secondary index (with block pointers) on a nonordering key field of a file.

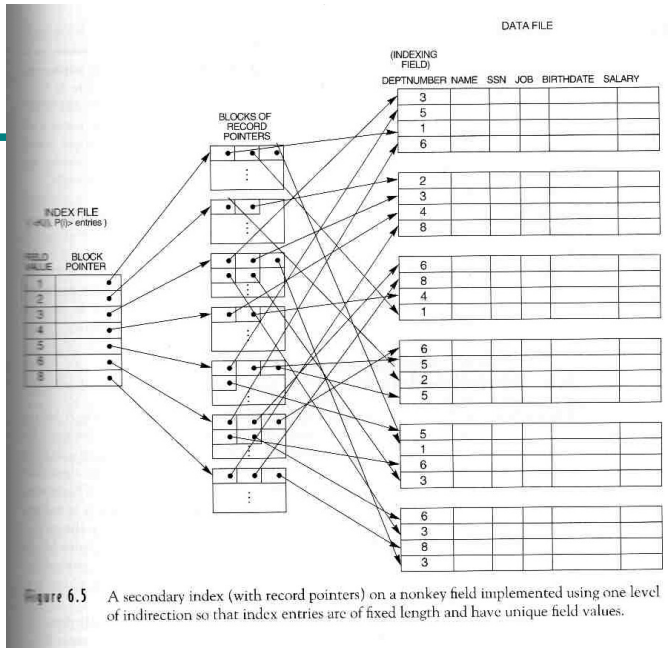


Figure 6.5 A secondary index (with record pointers) on a nonkey field implemented using one level of indirection so that index entries are of fixed length and have unique field values.

Índices: otras estructuras

- Hash
  - Muy buen comportamiento en la inserción y en la recuperación por condiciones de igualdad.
  - No funciona bien para condiciones con relaciones de orden
- Árboles B y B+
  - Buen comportamiento en recuperación tanto por condiciones de igualdad como de orden. Buen comportamiento en la inserción
  - Ocupa más disco.

Índices: Arbol B Vs. B+

- Arboles B

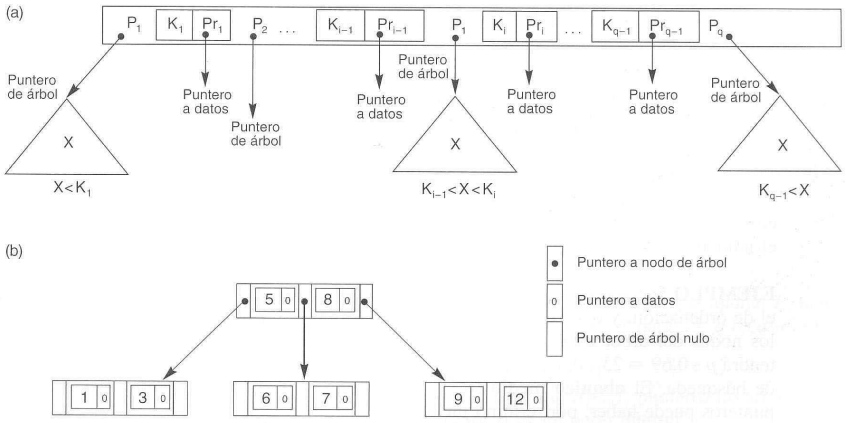


Figura 6.10. Estructuras de árbol B. (a) Nodo de árbol B con $q - 1$ valores de búsqueda. (b) Árbol B de orden $p = 3$. Los valores se insertaron en el orden 8, 5, 1, 7, 3, 12, 9, 6.

Índices: Arbol B Vs. B+

- Arboles B+

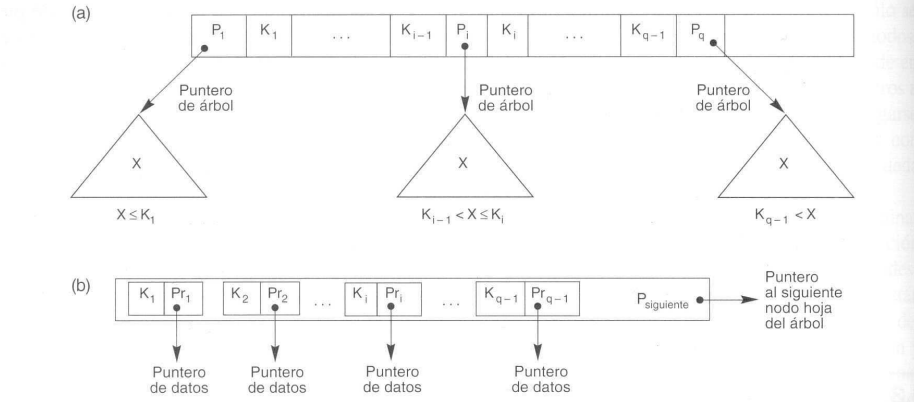


Figura 6.11. Nodos de un árbol B^+. (a) Nodo interno de un árbol B^+ con $q - 1$ valores de búsqueda. (b) Nodo hoja de un árbol B^+ con $q - 1$ valores de búsqueda y $q - 1$ punteros a datos.

Parámetros para la Comparación de Estructuras.

- En base a cantidad de operaciones básicas para realizar una operación en la estructura (búsqueda, recorrida, inserción, etc.)
  - No por operaciones en memoria, sino por operaciones de I/O (acceso al disco). Recordar los órdenes de las operaciones sobre las estructuras.
- Elementos a considerar a la hora de contabilizar accesos al disco:
  - Un bloque del disco, puede almacenar varios nodos de una estructura dada
  - El acceso se hace, habitualmente, a través de buffers que leen simultáneamente más de un bloque. Esto usualmente permite mejoras en los algoritmos.

Resumen sobre Organización y Acceso a los Datos

- Los DMBSs implementan diferentes estrategias para organizar los registros de una tabla:
  - Registros desordenados con acceso secuencial.
  - Registros ordenados con acceso secuencial.
  - Registros indexados por la clave primaria con hash.
  - Registros indexados por la clave primaria con árbol B+.
  - Registros indexados por otro atributo con un índice cluster.