# BASES DE DATOS II

## Actividad 4: Estudio teórico/práctico de índices

- Objetivos
  - Realizar una revisión de los conceptos de índices.
  - Ser capaz de recomendar los índices más adecuados para mejorar la eficiencia de consultas simples.
- Descripción de la actividad: De **modo individual** debe realizarse la lectura sobre índices que se adjunta, y la resolución de los ejercicios planteados.
- Presentación de la actividad: Deberán traerse **resueltos los ejercicios en papel**.

### 1. Índices

Un **índice** es una estructura de datos auxiliar que ayuda en la localización de datos bajo una cierta condición de selección. Cada índice posee una **clave de búsqueda** asociada, conjunto de uno o más atributos de un archivo de datos para el que se construye el índice (denominado **archivo indexado**).

El índice está formado por un conjunto de entradas, denominadas **entradas de datos del índice (k*),** que nos permiten localizar (uno o más) registros de datos que tienen un valor de clave de búsqueda ***k*** concreto. Es importante considerar aquí dos aspectos:

- *¿qué se almacena en una entrada de datos de índice?* Existen tres **alternativas** principales:
  [1] ***k****: almacena el registro de datos indexado real con valor de clave *k*. En este caso no hay un archivo de índices separado; se puede pensar como una organización de archivos especial que puede utilizarse en lugar de un *archivo ordenado* o de un *archivo heap.*
  [2] **(*k*, *id*)** donde *id* es el identificador del registro con clave de búsqueda *k*.
  [3] **(*k, lista-id*)** donde *lista-id* es una lista de identificadores de registro con clave de búsqueda *k*. Las entradas para un mismo índice son variables en longitud, dependiendo del número de registros de datos con un cierto valor de clave de búsqueda.

  Las Alternativas [2] y [3] son independientes de la organización del archivo de datos. Las entradas de datos del índice son, en general, mucho más pequeñas que los registros de datos. Por ello, son alternativas mejores que la [1], especialmente si las claves de búsqueda son pequeñas. La Alternativa [3] garantiza una mejor utilización del espacio que la Alternativa [2], pero las entradas de datos son variables en longitud, dependiendo del número de registros de datos con el mismo valor de clave de búsqueda.
  Si se definen varios índices para la misma colección de registros, es recomendable que uno de los índices utilice la Alternativa [1] para reducir el número de almacenamientos de los mismos registros.

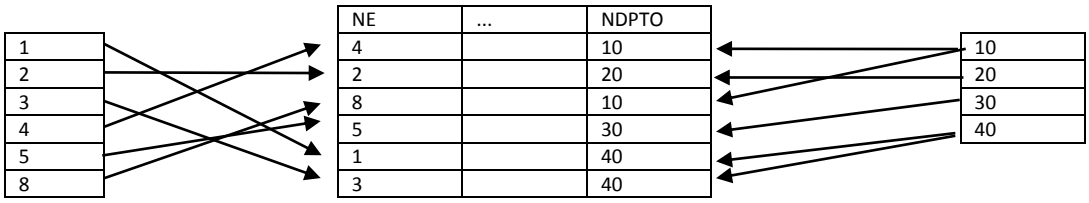
  Ejemplo: Dado un archivo de datos compuesto por registros cuyos campos son NE y NDPTO, entre otros, de la forma:

| NE | ... | NDPTO |
|---|---|---|
| 4 | | 10 |
| 2 | | 20 |
| 8 | | 10 |
| 5 | | 30 |
| 1 | | 40 |
| 3 | | 40 |

  Veamos ejemplos de indexación con las 3 Alternativas:
  Alternativa [1] por el campo NE:

| NE | ... | NDPTO |
|---|---|---|
| 1 | | 40 |
| 2 | | 20 |
| 3 | | 40 |
| 4 | | 10 |
| 5 | | 30 |
| 8 | | 10 |

Alternativa [2] por el campo NE (izquierda) y Alternativa [3] por NDPTO (derecha):




- *¿cómo se organizan las entradas de datos del índice para soportar recuperaciones eficientes?* Es posible hacer búsquedas eficientes sobre un índice para encontrar las entradas de datos deseadas, y después utilizarlas para obtener sus registros de datos. Las técnicas de organización para índices más empleados son las estructuras *basadas en árbol* y las *basadas en hash*, como veremos en el siguiente apartado.

## 1.1. Estructuras de datos de índices

Los tipos de organización de datos más empleados para la implementación de los índices son las estructuras *basadas en árbol* y las *basadas en hash***.** La elección de la estructura de datos puede combinarse con cualquiera de las tres alternativas de entradas de datos vistas en el apartado anterior.

### 1.1.1. Indexación basada en árboles

Las estructuras **basadas en árbol** son un soporte eficiente para las consultas por rango y, excepto en el caso de que los archivos de datos estén ordenados, también son eficientes en inserciones y borrados. Aunque soportan las selecciones de igualdad, no resultan ser estructuras tan adecuadas como las basadas en hash. Dentro de las estructuras basadas en árbol destacan los **árboles B+**.

Una estructura de *árbol-B+* consiste en repartir los valores del índice sobre un bloque (página, unidad mínima de transferencia entre memoria y disco) raíz, unos bloques intermedios y unos bloques hojas. El bloque raíz y los intermedios contienen las direcciones de los otros bloques. Los bloques hoja contienen todos los valores del índice y, para cada valor del índice, el *id* del registro que contiene dicho valor del índice (en caso de que el índice siguiese una Alternativa [3], cada entrada tendría varios *ids* en función del número de entradas de datos que tuviesen el mismo valor para el índice).

La técnica del árbol-B+ está basada en los principios siguientes:

- Un árbol-B+ está siempre equilibrado: hay tantos bloques (intermedios y hojas) a la izquierda como a la derecha del bloque raíz. Estos bloques están generalmente equitativamente rellenos.
- Los bloques hoja están todos al mismo nivel, es decir, a igual profundidad en el árbol. Esto hace que los tiempos de respuesta sean los mismos independientemente del valor del índice.
- Los bloques hoja están colocados en orden creciente (o decreciente, según se especifique en el momento de su creación) de izquierda hacia derecha.
- La búsqueda de un valor comienza por la raíz y se termina a nivel de las hojas.
- Los nodos internos se encargan de dirigir la búsqueda a través de las entradas de datos del índice, siendo los nodos hoja los que contienen las entradas de datos del índice. Los nodos hoja se enlazan entre sí mediante punteros de página. Mediante una lista doblemente enlazada es posible atravesar fácilmente la secuencia de nodos hoja en cualquier dirección.

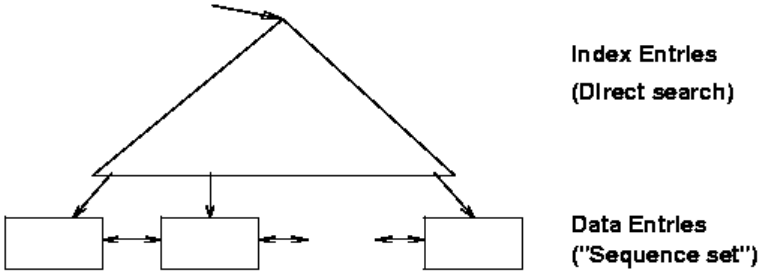



Figura 1.- Estructura de un árbol B+

El número de operaciones de E/S realizadas durante una búsqueda en un índice con esta estructura en árbol B+ es igual a la longitud del camino desde la raíz a una hoja, más el número de páginas hoja con entradas de datos que satisfacen la condición de búsqueda.

Buscar la página hoja correcta en un árbol B+ es más rápido que la búsqueda binaria de las páginas en un archivo ordenado porque cada nodo que no es hoja puede alojar un gran número de punteros a nodos, y la altura del árbol raramente suele ser más de 3 o 4. En la práctica, el número medio de punteros almacenados en cada nodo intermedio es por lo menos 100, lo que quiere decir que un árbol de altura 4 contiene 100 millones de páginas hoja. Por tanto, es posible buscar en un archivo con 100 millones de páginas hoja y encontrar la buscada utilizando 4 operaciones de E/S; una búsqueda binaria del mismo archivo llevaría $log_2\ 100\ 000\ 000$ (más de 25) operaciones de E/S.

### 1.1.2. Indexación basada en Hash

Una estructura **basada en *hash*** está compuesta por un conjunto de *buckets* (cajones). Cada bucket está formado por una **página** (unidad mínima de transferencia entre memoria y disco; es similar al concepto de bloque en Oracle) **primaria**, y 0 o más **páginas de overflow** enlazadas mediante punteros.

Puede determinarse el bucket al que pertenece un registro aplicando una función *hash* a la clave de búsqueda. Dado un número de bucket, una estructura hash permite recuperar la página primaria del bucket en uno o dos accesos E/S en disco.

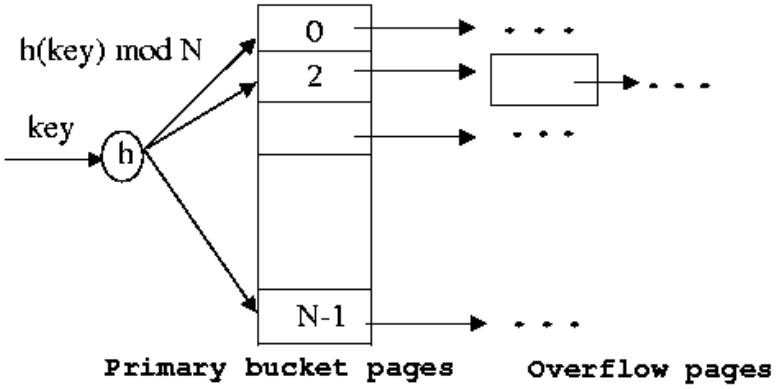



Figura 2.- Estructura basada en hash

Desafortunadamente, las técnicas de indexación basadas en *hash* no soportan las búsquedas por rango. Las técnicas basadas en árbol soportan búsquedas por rango de forma eficiente, y son casi tan eficientes en las selecciones de igualdad que en el caso de indexación basada en *hash*. Por ello muchos sistemas comerciales soportan únicamente índices basados en árbol.

## 1.2. Tipos de índices

### 1.2.1. Índices primarios y secundarios

Un índice sobre un conjunto de campos que incluye la clave primaria se denomina **índice primario**. Se dice que dos entradas de datos están duplicadas si tienen el mismo valor para la clave de búsqueda asociada al índice. Un índice primario no tiene duplicados, pero un índice sobre otro conjunto de campos podría tenerlos. Así, en general, un **índice secundario** contiene duplicados. Si no admite duplicados se denomina **índice único**; es el caso en que la clave de búsqueda está asociada a una clave candidata.

NOTA: Los términos *índice primario* e *índice secundario* se utilizan a veces con un significado diferente: se dice que un índice es primario si utiliza la Alternativa [1], mientras que los que utilizan las Alternativas [2] y [3] se denominan índices secundarios. Sin embargo, aquí consideraremos la definición dada anteriormente.

### 1.2.2. Agrupados vs. No Agrupados

Se dice que un índice está **agrupado** (**clustered**) si los registros de datos del archivo indexado están ordenados del mismo modo que las entradas de datos del índice. Un índice que usa la Alternativa [1] está agrupado, por definición. Los índices que siguen las Alternativas [2] y [3] estarán agrupados si los registros de datos en el archivo indexado están ordenados por la clave de búsqueda.

Un archivo de datos sólo puede agruparse por una clave de búsqueda, lo que significa que sólo puede existir un índice agrupado por archivo. Un índice que no está agrupado se denomina **unclustered.**

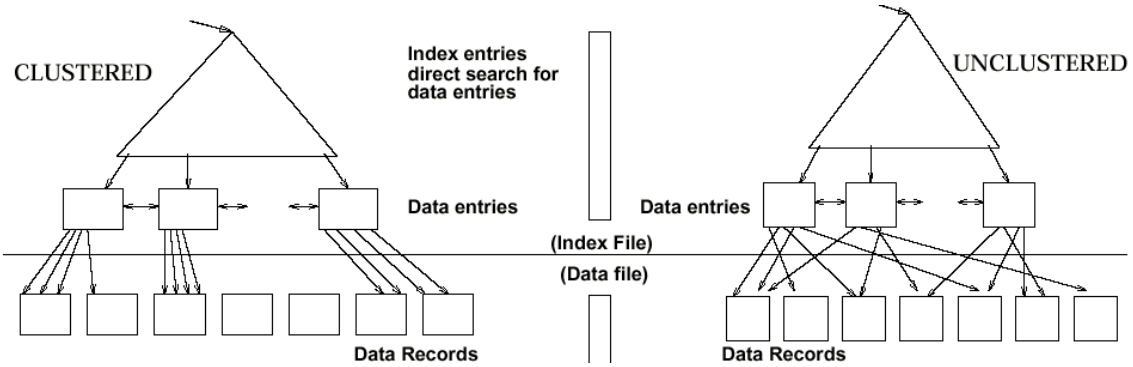



Figura 3.- a) Índice B+ agrupado; b) Índice B+ NO agrupado

El coste en la recuperación de los registros de datos en base a índices varía enormemente en función de si el índice está agrupado o no, sobre todo en el caso de las consultas por rango. Así, si el índice está agrupado, los *ids* apuntan a registros adyacentes, como se muestra en la Figura 3(a), por lo que se recuperan pocas páginas. Si el índice no está agrupado, cada entrada de datos del índice podría contener *ids* que apunten a páginas diferentes (ver Figura 3(b)), sobrecargando la Entrada/Salida.

En la práctica, raramente se crean índices agrupados porque es demasiado caro mantener el orden cuando se actualizan los datos, aunque menos que un archivo completamente ordenado. Así que, habitualmente, un índice agrupado es aquel que utiliza la Alternativa [1], y los índices que utilizan las Alternativas [2] o [3] son no agrupados. De hecho, a veces se usa el término **archivo agrupado** para denotar a un índice bajo la Alternativa [1.

El agrupamiento debe utilizarse pocas veces y solamente cuando esté justificado por consultas frecuentes que se beneficien de él. En particular, generalmente no hay suficientes razones para construir un archivo agrupado utilizando un índice Hash, pues, tal como se ha comentado anteriormente, las consultas por rangos no pueden consultarse utilizando este tipo de índices, exceptuando el caso en el que se realice una igualdad con muchos duplicados.

### 1.2.3. Densos vs. Dispersos (Sparse)

Un índice es **denso** si contiene una entrada de datos del índice **por cada valor de clave de búsqueda** que aparece en un registro del archivo indexado, al menos (porque si se utiliza la Alternativa [2] podría haber duplicados).

Un índice **disperso (sparse)** contiene una entrada de datos del índice **por cada página de registros** en el archivo de datos.

- La Alternativa [1] siempre corresponde a un índice denso.
- La Alternativa [2] puede utilizarse para construir un índice denso o disperso.
- La Alternativa [3] generalmente se utiliza para construir un índice denso.

Ejemplo: En la Figura 4 se muestra un archivo de datos con tres campos (*nombre, edad* y *sal*) y dos índices simples que emplean la Alternativa [2]. El índice de la izquierda es agrupado disperso por *nombre* (el orden de la entradas en el índice y en el archivo de datos coinciden, y hay una entrada de datos del índice por página de registros de datos). El índice de la derecha es no agrupado denso por el campo *edad* (el orden de las entradas del índice es distinto al del archivo de datos, y hay una entrada de datos en el índice por cada registro en el archivo de datos).

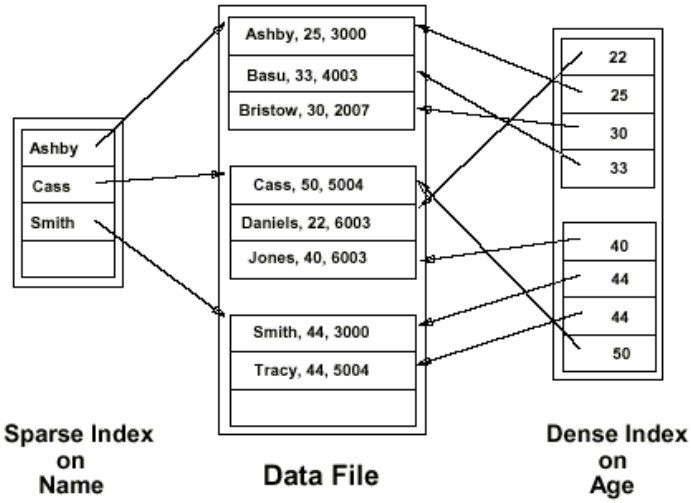



Figura 4.- Ejemplo de índices disperso y denso

No es posible construir un índice disperso que no esté agrupado; esto significa que sólo podremos tener como máximo un índice disperso por archivo de datos. Un índice disperso suele ser mucho más pequeño que uno denso.

# BASES DE DATOS II

### 1.3. Ejercicios de selección de índices

Supongamos las siguientes tablas:

EMP (eno, ename, dno, age, sal, hobby)
DEPT (dno, dname, mgr)

que representan una relación 1-N donde cada departamento se relaciona con los empleados que trabajan en el mismo. Además, para cada departamento se almacena el código del empleado que lo dirige (*mgr*).

Plantear los índices recomendados para cada una de las siguientes consultas, razonando la decisión en cada caso:

SELECT *E.dno*
FROM *EMP E*
WHERE *E.hobby = 'Stamps'*

| Campo índice | Estructura datos | 1º / 2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |

SELECT *E.dno*
FROM *EMP E*
WHERE *E.eno = 552*

| Campo índice | Estructura datos | 1º / 2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |

# BASES DE DATOS II

SELECT *E.dno*
FROM *EMP E*
WHERE *E.age > 40*

| Campo índice | Estructura datos | 1º / 2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |

SELECT *E.dno,* COUNT(*)
FROM *EMP E*
WHERE *E.age > 50*
GROUP BY *E.dno*

| Campo índice | Estructura datos | 1º / 2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |

SELECT *E.ename, D.mgr*
FROM *EMP E, DEPT D*
WHERE *D.dname='Toy'* AND *E.dno=D.dno*

| Campo índice | Estructura datos | 1º/2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |

# BASES DE DATOS II

SELECT *E.ename, D.mgr*
FROM *EMP E, DEPT D*
WHERE *D.dname='Toy'* AND *E.dno=D.dno* ***AND E.age=25***

| Campo índice | Estructura datos | 1º/2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |

SELECT *E.ename, D.dname*
FROM *EMP E, DEPT D*
WHERE *E.sal* BETWEEN 10000 AND 20000
AND *E.hobby='Stamps'* AND *E.dno=D.dno*

| Campo índice | Estructura datos | 1º/2º | Agrupado /No Agrupado | Denso / Disperso |
|---|---|---|---|---|
| | | | | |