

Tema I.- Diseño Físico

Ficheros

Índices

Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión. T.Connolly, C. Begg, A. Strachan. Addison-Wesley [cap.17 en 4ª edic.]

Sistemas de gestión de bases de datos. R. Ramakrishnan. McGraw-Hill [cap.9 en 3ª edic.]

Ciclo de vida del desarrollo de una BD

Stage of database system development lifecycle	Examples of data captured	Examples of documentation produced
Database planning	Aims and objectives of database project	Mission statement and objectives of database system
System definition	Description of major user views (includes job roles or business application areas)	Definition of scope and boundary of database application; definition of user views to be supported
Requirements collection and analysis	Requirements for user views; systems specifications, including performance and security requirements	Users' and system requirements specifications
Database design	Users' responses to checking the logical database design; functionality provided by target DBMS	Conceptual/logical database design (includes ER model(s), data dictionary, and relational schema); physical database design
Application design	Users' responses to checking interface design	Application design (includes description of programs and user interface)
DBMS selection	Functionality provided by target DBMS	DBMS evaluation and recommendations
Prototyping	Users' responses to prototype	Modified users' requirements and systems specifications
Implementation	Functionality provided by target DBMS	
Data conversion and loading	Format of current data; data import capabilities of target DBMS	
Testing	Test results	Testing strategies used; analysis of test results
Operational maintenance	Performance testing results; new or changing user and system requirements	User manual; analysis of performance results; modified users' requirements and systems specifications

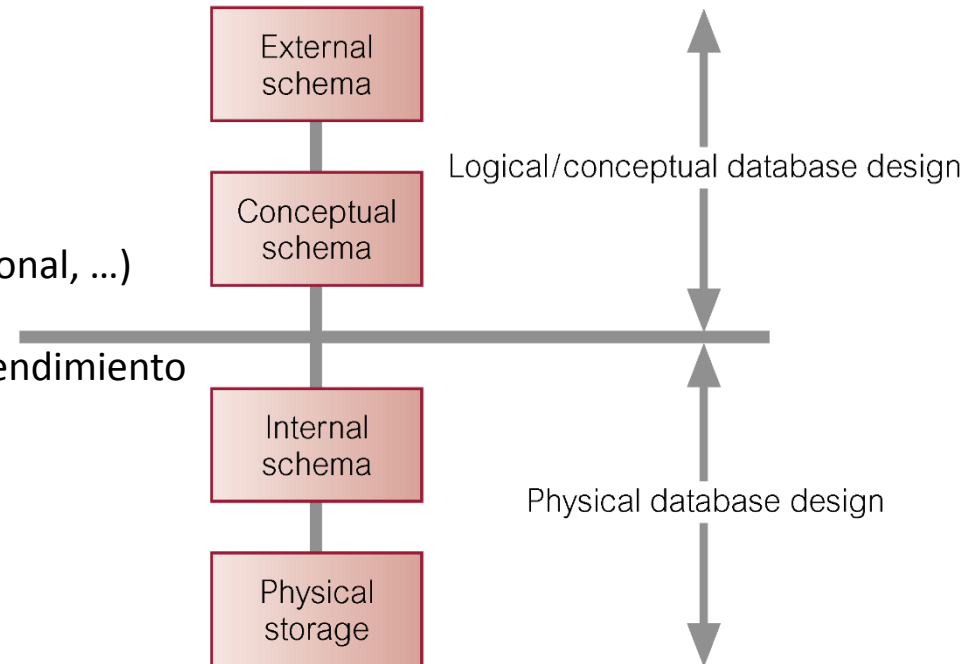
Database Design: Fases de diseño de una BD

- Diseño **conceptual/lógico** (qué)

- Crear el EER (esquema conceptual)
- Definir vistas (esquema externo)
- Crear el modelo lógico (esquema relacional, ...)
- Crear el diccionario de datos
- Refinar los esquemas para mejorar el rendimiento

- Diseño **físico** (cómo)

1. Traducir el modelo lógico al SGBD seleccionado
2. Diseñar la organización de los archivos e índices
 1. Análisis de las transacciones
 2. Elección de la organización de los archivos
 3. Elección de índices
 4. Estimación de los requisitos de espacio de disco
3. Diseñar las vistas de usuario
4. Diseñar los mecanismos de seguridad
5. Considerar la introducción de una cantidad controlada de redundancia
6. Monitorizar y ajustar el sistema final



1. Análisis de las transacciones

- **Objetivo:** Comprender la funcionalidad de las transacciones que se ejecutarán en la BD, y analizar las más importantes.
- Debemos conocer la CARGA DE TRABAJO:
 - Identificar criterios de rendimiento, como:
 - Transacciones que se ejecutan frecuentemente
 - Transacciones críticas para la empresa
 - Picos de carga: momentos (diarios/semanales) en los que la demanda de procesamiento será mayor
 - Criterios de búsqueda utilizados en las consultas
 - Atributos que son actualizados
- Habitualmente no es posible analizar todas las transacciones, pero ...
 - ... *20% de las consultas más activas representan el 80% del acceso total a datos*
- Para identificar las más importantes puede utilizarse:
 - **Matriz cruzada de transacciones/relaciones**
 - **Mapa de uso de transacciones**

Matriz cruzada de transacciones/relaciones

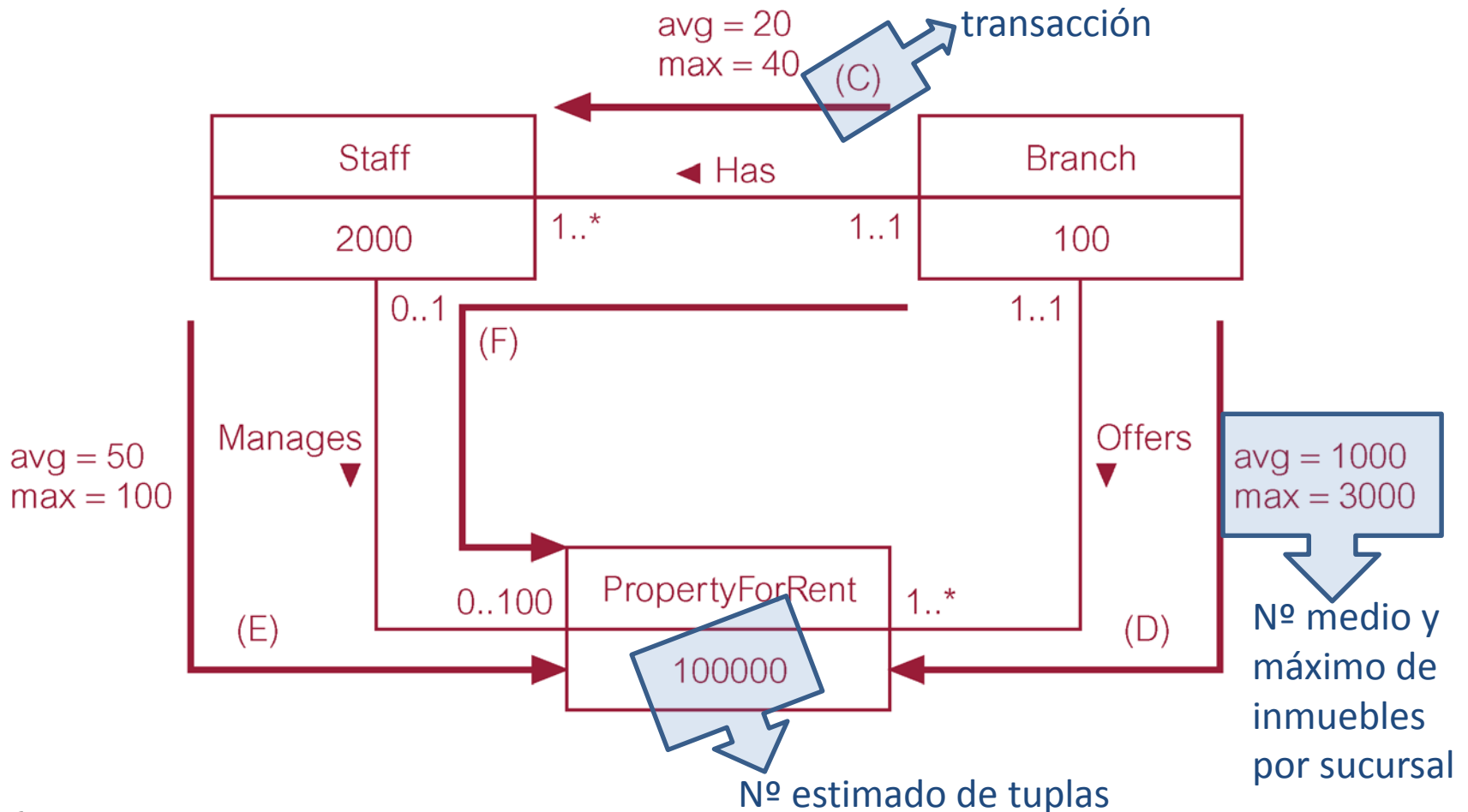
- Muestra las relaciones a las que accede cada transacción

Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch									X				X											
Telephone																								
Staff	X				X				X								X				X			
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X				X	X	X						X				X				X			
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

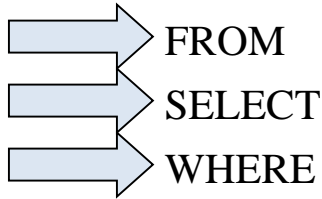
I = Insert; R = Read; U = Update; D = Delete

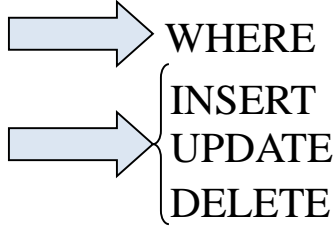
Mapa de uso de transacciones

- Indica cuáles son las relaciones potencialmente más utilizadas



Análisis del uso de los datos

- \forall CONSULTA de transacciones importantes:
 - RELACIONES ?????
 - ATRIBUTOS recuperados ?????
 - ATRIBUTOS involucrados en Joins/Selecciones ?????

FROM
SELECT
WHERE
- \forall ACTUALIZACIÓN de transacciones importantes:
 - ATRIBUTOS involucrados en Joins/Selecciones ?????
 - Candidatos a estructuras de acceso auxiliares
 - TIPO ACTUALIZACIÓN
 - ATRIBUTOS actualizados????
 - Candidatos a evitar estructuras de acceso auxiliares

WHERE
INSERT
UPDATE
DELETE
- Frecuencia esperada de ejecución
 - Ej: *50 veces al día*
- Objetivos de rendimiento
 - Ej: *la transacción debe finalizarse en menos de 1 segundo*
 - Los atributos de las transacciones muy frecuentes o críticas deben tener mayor prioridad a la hora de establecer estructuras de acceso

Formulario de análisis de transacciones

Transaction Analysis Form

1-Sept-2004

Transaction (D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent

Transaction volume

Average: 50 per hour
 Peak: 100 per hour (between 17.00 and 19.00 Monday–Saturday)

→ Frecuencia media de la transacción
 → Pico de carga

```

SELECT propertyNo, p.street, p.postcode, type, rent
FROM Branch b INNER JOIN PropertyForRent p ON
    b.branchNo = p.branchNo
WHERE p.city = 'Glasgow'
ORDER BY rent;
                
```

Predicate: p.city = 'Glasgow'

Join attributes: b.branchNo = p.branchNo

Ordering attribute: rent

Grouping attribute: none

Built-in functions: none

Attributes updated: none

Transaction usage map

1 →

Branch
(100)

2

avg = 1000
max = 3000

↓ Offers

1..1

PropertyForRent
(100000)

1..*

Assume 4 Glasgow offices

Access	Entity	Type of Access	No. of References		
			Per Transaction	Avg Per Hour	Peak Per Hour
1	Branch (entry)	R	100	5000	10000
2	PropertyForRent	R	4000–12000	200000–600000	400000–1200000
Total References			4100–12100	205000–605000	410000–1210000

Mapa de uso de la transacción

Resumen. Para cada relación:

- Tipo de acceso
- Nº tuplas accedidas por ejecución
- Nº tuplas accedidas por hora

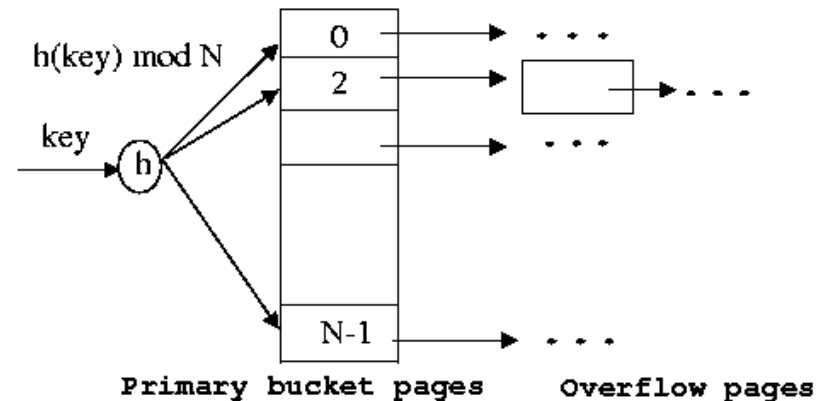
Figure 17.4 Example transaction analysis form.

2.- Elección de la organización de archivos

- **Objetivo:** Determinar una organización de archivo eficiente para cada relación.
 - Los datos se almacenan en archivos de registros
 - Página (bloque): unidad mínima de transferencia entre memoria y disco
 - Cada registro tiene un **identificador** único con la dirección física de su página

Organizaciones de archivos de datos

- HEAP (no ordenada / de montículo)
 - Disposición de registros ALEATORIA
 - Interesante cuando se recuperan todos los registros
- ORDENADO (secuencial)
 - Registros ordenados por CLAVE DE BÚSQUEDA
 - Interesante cuando los registros se recuperan en orden
- HASH
 - { buckets } = { página 1ª [, página overflow] }
 - **$h(\text{clave búsqueda}) = n^{\text{o}} \text{ bucket}$**



3.- Elección de Índices

- Estructura (auxiliar) que ayuda a localizar datos de un archivo bajo una condición
- Se asocia a una CLAVE DE BÚSQUEDA
 - \forall campo puede \in CLAVE BUSQUEDA
 - CLAVE BUSQUEDA \neq Clave
 - Las ENTRADAS DE DATOS DEL INDICE (k^*) localizan los registros de datos con CLAVE BUSQUEDA = k
- ¿Qué se almacena en una ENTRADA DE DATOS DEL INDICE? =>
ALTERNATIVAS:
 - [1] k^* = registro de datos completo
 - [2] $\langle k, \text{identificador del registro con clave de búsqueda } k \rangle$
 - [3] $\langle k, \text{lista-identificadores de registro con clave de búsqueda } k \rangle$
(las entradas de datos del índice son variables en longitud)

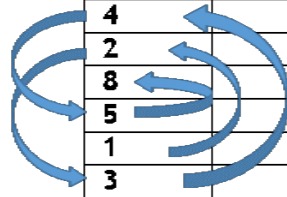
Ejemplo de índices

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

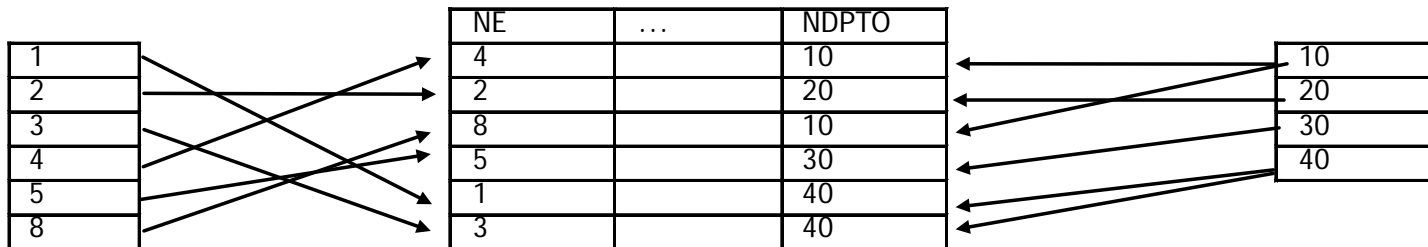
Veamos ejemplos de indexación con las 3 Alternativas:

Alternativa [1] por el campo NE:



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

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



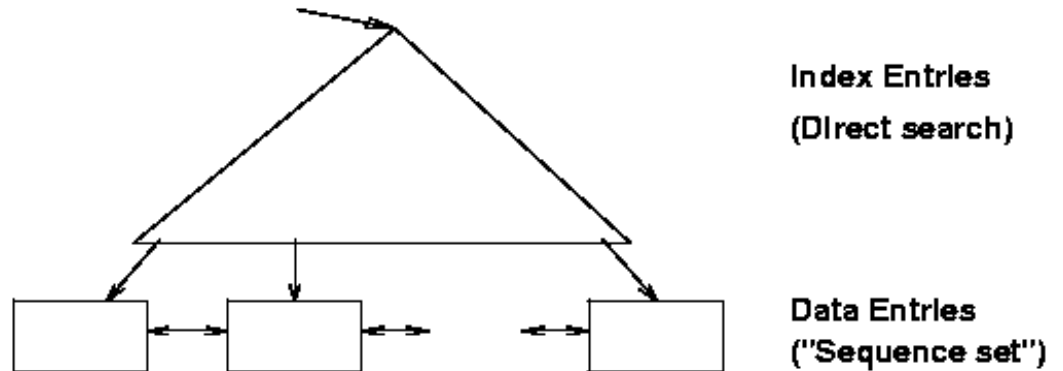
Índices

- Estructura auxiliar que ayuda a localizar datos de un archivo bajo una condición
- Se asocia a una CLAVE DE BÚSQUEDA
 - \forall campo puede \in CLAVE BUSQUEDA
 - CLAVE BUSQUEDA \neq Clave
 - Las ENTRADAS DE DATOS DEL INDICE (k^*) localizan los registros de datos con CLAVE BUSQUEDA = k
- ¿Qué se almacena en una ENTRADA DE DATOS DEL INDICE? =>
ALTERNATIVAS:
 - [1] k^* = registro de datos completo
 - [2] $\langle k, \textit{identificador del registro con clave de búsqueda } k \rangle$
 - [3] $\langle k, \textit{lista-identificadores de registro con clave de búsqueda } k \rangle$
(las entradas de datos del índice son variables en longitud)
- [1] es una organización de archivos especial \cong heap, ordenado, ...
- [1] si registros $\uparrow\uparrow \Rightarrow \uparrow\uparrow$ páginas
- Si nº índices $> 1 \Rightarrow$ 1 índice puede usar [1], resto [2][3]
- [3] +compacta que [2], PERO... tamaño de registro variable

Estructuras de datos de Índices

- **Árbol B+:**

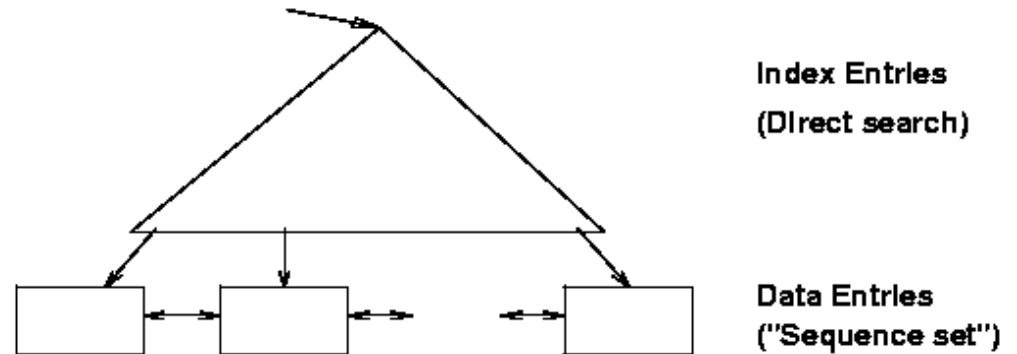
- Balanceado, Ordenado
- E datos del índice son las hojas
- E datos del índice forman una lista doblemente enlazada
- Búsquedas por Rango, Igualdad



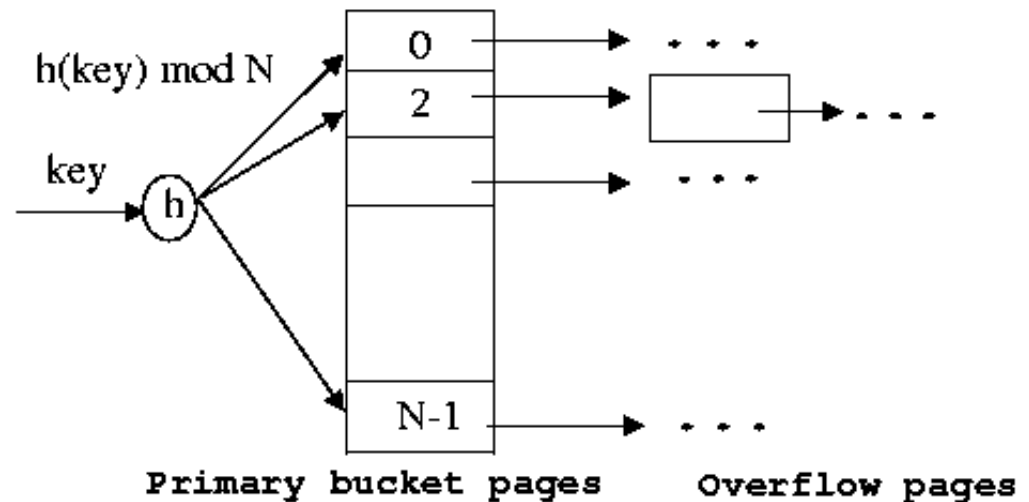
- Nº operaciones de E/S = longitud (raíz a hoja) + nº páginas hoja con E que cumplen la condición
- La búsqueda en árbol B+ es más rápida que en archivo ordenado:
 - nº medio de punteros en cada nodo intermedio > 100:
 - altura del árbol suele ser 3 ó 4
 - ⇒ árbol de altura 4 contiene 100 millones de páginas hoja
 - ⇒ se necesitan 4 operaciones E/S para buscar en un archivo con 100 millones de páginas hoja (una búsqueda binaria llevaría $\log_2 100000000 > 25$ operaciones E/S)

Estructuras de datos de Índices

- **Árbol B+:**
 - Balanceado, Ordenado
 - E datos del índice son las hojas
 - E datos del índice forman una lista doblemente enlazada
 - Búsquedas por Rango, Igualdad



- **Basados en Hash:**
 - {buckets} = {pág. 1ª [, pág. overflow]}
 - h (clave búsqueda)** = nº bucket
 - NO Búsquedas por Rango



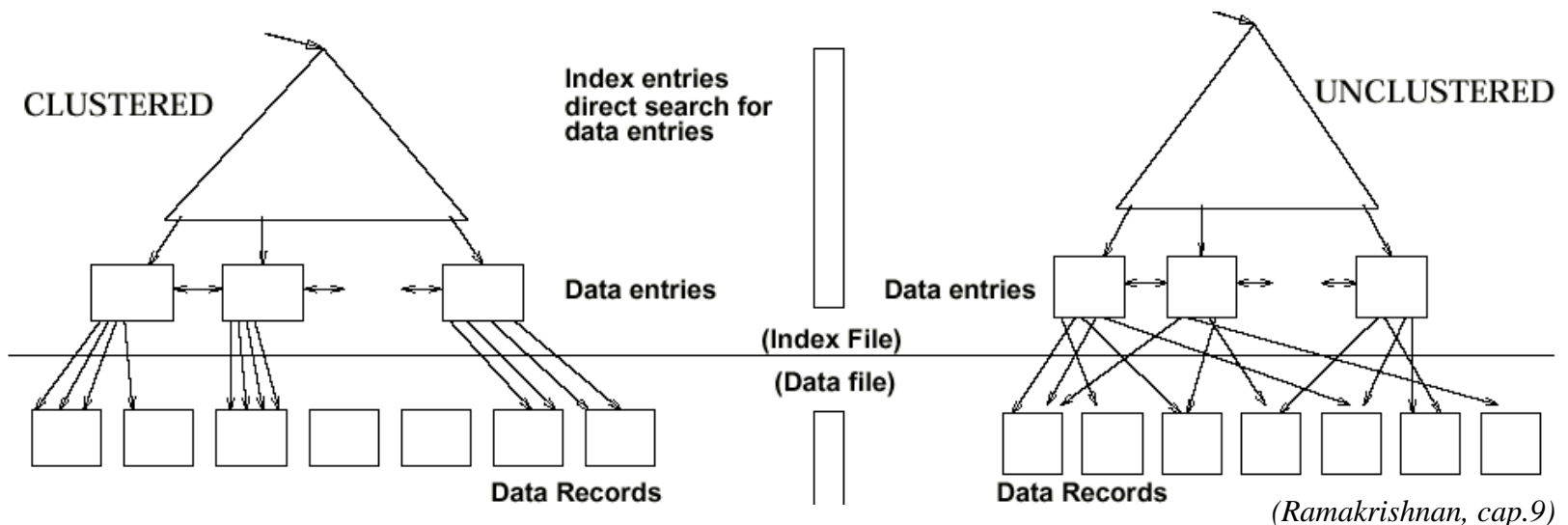
Clasificación de Índices

- Primarios vs. Secundarios

- Primario \Rightarrow incluye la Clave 1ª y garantiza que no contiene duplicados
- Secundario \Rightarrow puede contener duplicados
 - Si NO \exists duplicados \Rightarrow **Índice único** \Rightarrow índice sobre Clave Candidata

- Agrupados (CLUSTERED) vs. No Agrupados (UNCLUSTERED)

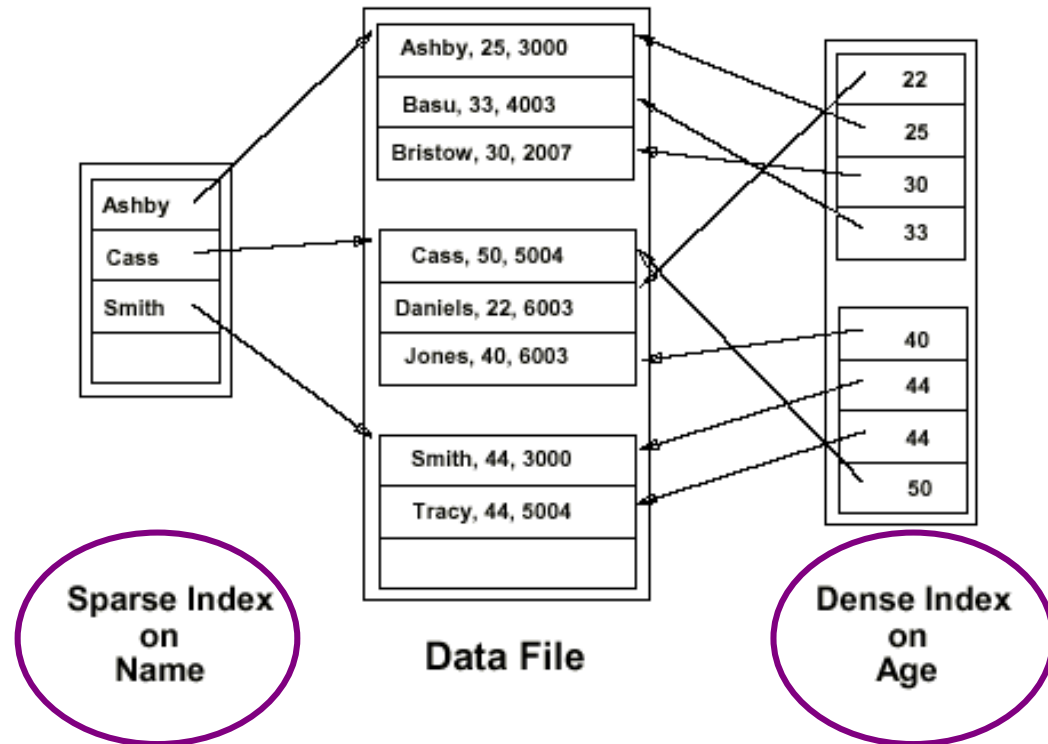
- Agrupado \Rightarrow orden reg. datos = orden entradas índice
- [1] \Rightarrow agrupado (PERO NO " \Leftarrow ", aunque en la práctica suele serlo)
- [2][3] agrupados \Rightarrow reg.datos ordenados por la clave búsqueda
- máximo = 1 índice AGRUPADO por archivo de datos
- Coste de recuperación varía ENORMEMENTE si el índice está agrupado (sobre todo duplicados y rangos)
- Agrupado es muy útil en consultas POR RANGO o con muchos duplicados, PERO ES CARO
- Hash agrupado??? \Rightarrow sólo si hay muchos registros duplicados



Clasificación de Índices

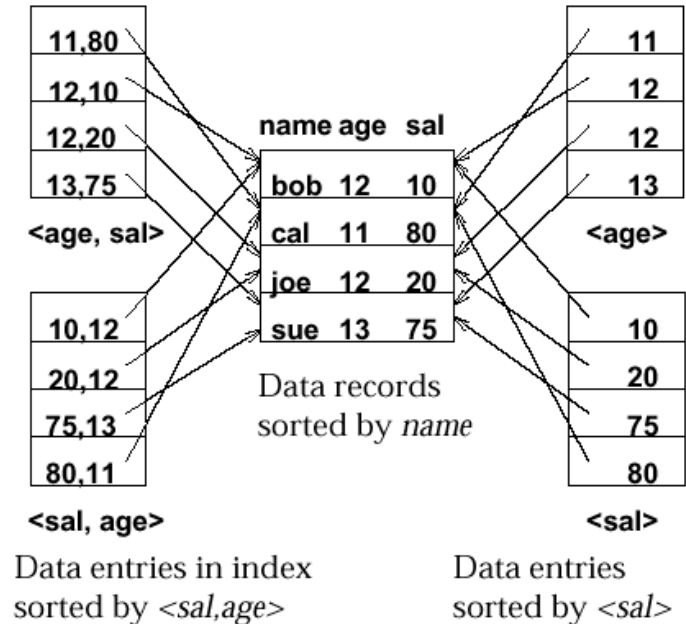
- Densos vs. Dispersos

- Denso $\Rightarrow \forall$ **registro(s)** datos \exists entrada datos índice
- Disperso (sparse) $\Rightarrow \forall$ **página** registro datos \exists entrada datos índice
 - \Rightarrow AGRUPADO \Rightarrow máximo 1 índice DISPERSO por archivo de datos
- [1] \Rightarrow Denso
- [2] \Rightarrow Disperso / Denso
- [3] \Rightarrow Denso (generalmente)



Clasificación de Índices

- Índice Compuesto
 - Formado por varios campos



- Consulta por IGUALDAD
 - ⇒ ej: *age = 20 and sal = 10*
 - ⇒ índice basado en HASH / B+
- Consulta por RANGO
 - (+1) campo NO es constante
 - ⇒ ej: *age < 30 and sal > 40*
 - ⇒ ej: *age = 20*
 - ⇒ índice B+
- El ORDEN de los CAMPOS en el índice es importante
 - ⇒ ej: con la condición (*age = 30 and sal > 40*)
 - <age,sal>* da mejor resultado que *<sal,age>*

Pautas para la selección de Índices

- Construir el índice SÓLO si hay consultas que se beneficien de él. Considerar el impacto en las actualizaciones !!!!!
- Los atributos del WHERE, GROUP BY y ORDER BY son candidatos
 - Condición por igualdad sugiere índice HASH
 - Muy eficientes para joins y consultas exactas
 - Condición por rango exige índice en árbol B+
 - La agrupación es especialmente útil para consultas por rango o con muchos duplicados
- Elegir índices que beneficien a varias consultas
 - Como sólo se puede crear un índice agrupado por relación, elegirlo sobre las consultas más frecuentes y que involucren más tuplas.
- Los índices compuestos deben definirse sólo cuando WHERE / GROUP BY / ORDER BY contienen varias condiciones
 - Si hay selecciones por rango, ordenar los atributos del índice en base a la consulta

Ejemplos de índices simples

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

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

Campo	Estructura	1° / 2°	Agrupado /No Agrupado	Denso / Disperso
E.hobby (si tuplas='Stamps' ↓↓)	Hash (igualdad)	2°	NO Agrupado (↓↓ duplicados, ordenación cara)	Denso (única opción para índices Hash)
E.hobby (si tuplas='Stamps' ↑↑)	Hash (igualdad)	2°	Agrupado (↑↑ duplicados)	Denso (única opción para índices Hash)

Si se recuperan ↑↑ tuplas NO agrupado sería INEFICIENTE

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

Campo	Estructura	1° / 2°	Agrupado /No Agrupado	Denso / Disperso
E.eno	Hash (igualdad)	1°	NO Agrupado (se obtiene sólo 1 tupla)	Denso (única opción)

Ejemplos de índices simples

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

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

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
E.age (si tuplas <i>age</i> >40 ↓↓)	B+ (rango)	2º	Agrupado (∃ duplicados, consulta por rango)	Disperso (+ pequeño)
E.age (si tuplas <i>age</i> >40 ↑↑)	NO INDICE		No Agrupado sería INEFICIENTE Agrupado sería MUY CARO	

El impacto de la agrupación depende del nº tuplas recuperadas:

- 1 tupla (i.e, igualdad sobre clave candidata) ⇒ Agrupado / No Agrupado
 - Pero... Agrupado no aporta ventajas
- Agrupación es interesante con ↑↑ duplicados
- ↑↑ tuplas (> 10%) ⇒ No Agrupado es + caro que recorrido secuencial

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
E.age (si tuplas <i>age</i> >50 ↓↓)	B+ (rango)	2º	Agrupado (∃ duplicados, consulta por rango)	Disperso (+ pequeño)
ó E.dno (si tuplas <i>age</i> >50 ↑↑)	Hash (igualdad) ó B+ (ordenado)	2º	Agrupado (↑↑ duplicados, los empleados se obtienen ordenadamente por departamento)	Denso (única opción para índices Hash) Disperso (+ pequeño)

Ejemplos de índices simples

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

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



Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
D.dname (si tuplas='Toy' ↓↓)	Hash (igualdad)	2º	No Agrupado (↓↓ duplicados, ordenación cara, No consulta por rango)	Denso (única opción para índices Hash)
E.dno	Hash (igualdad)	2º	Agrupado (No clave, ∃ duplicados)	Denso (única opción para índices Hash)

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

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
D.dname (si tuplas='Toy' ↓↓)	Hash (igualdad)	2º	No Agrupado (↓↓ duplicados, ordenación cara, No consulta por rango)	Denso (única opción)
E.age	Hash (igualdad)	2º	No Agrupado (↓↓ duplicados, ordenación cara, No consulta por rango)	Denso (única opción)

Ejemplos de índices simples

EMP (eno, ename, dno, age, sal, hobby)

DEPT (dno, dname, mgr)

```
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	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
D.dno	Hash (igualdad)	1º	NO Agrupado (1 tupla)	Denso (única opción)
E.sal (si tuplas ↓↓)	B+ (rango)	2º	Agrupado (No clave, ∃ duplicados, consulta rango)	Disperso (+ pequeño)
o E.hobby (si tuplas = "Stamps" ↓↓) (si tuplas = "Stamps" ↑↑)	Hash (igualdad)	2º	NO Agrupado (↓↓ duplicados, ordenación cara) Agrupado (↑↑ duplicados)	Denso (única opción) Denso (única opción para índices Hash)

Si se crean *E.sal* y *E.hobby* el optimizador elige el índice más selectivo en cada caso



E.hobby debe ser NO Agrupado
(no puede haber más de 1 índice agrupado por tabla)

Optimización utilizando índices

- El optimizador de consultas del SGBD genera PLANES DE EJECUCIÓN
- Tipos de planes:
 - Utilizando 1 Índice
 - Elige el índice que filtra más tuplas
 - Utilizando Múltiples Índices
 - Aplica \forall índices, recuperando el $\text{IdRegistro} = \text{IdPágina} + \text{Id}$
 - Intersecta los conjuntos de tuplas obtenidos (si AND), o los une (si OR)
 - Ordena el resultado por IdPágina
 - Solo-Índice
 - Si \forall datos están en el índice \Rightarrow NO acceder a los registros de datos
 - Condiciones:
 - Índice DENSO
 - \forall atributos de la SELECT forman parte del índice
 - Suelen ser índices COMPUESTOS
 - Son índices NO AGRUPADOS (NO tiene sentido agrupar !!!!!)

Ejemplos planes solo-índice

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

```
SELECT D.mgr, E.eno  
FROM EMP E, DEPT D  
WHERE E.dno = D.dno
```



Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
E.dno, E.eno	B+ (rango)	2º	NO Agrupado (NO tiene sentido agrupar)	Denso (única opción)

⇒ NO es necesario acceder a la tabla EMP

Además se podría crear un índice por D.dno y D.mgr que aprovecharse también la aproximación solo-índice:

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
D.dno, D.mgr	B+ (para distinguir D.dno en el índice)	1º	No Agrupado (no tiene sentido agrupar)	Denso (única opción)

Ejemplos planes solo-índice

EMP (eno, ename, dno, age, sal, hobby)

DEPT (dno, dname, mgr)

```
SELECT E.dno, COUNT(*)  
FROM EMP E  
GROUP BY E.dno
```

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
E.dno	Hash (igualdad) o B+ (ordenado)	2º	NO Agrupado (NO tiene sentido agrupar)	Denso (única opción)

⇒ Se responde a la consulta utilizando únicamente el índice

Ejemplos planes solo-índice

```
SELECT E.dno, COUNT(*)  
FROM EMP E  
WHERE E.sal = 10000  
GROUP BY E.dno
```

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

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
E.sal, E.dno	B+ (rango)	2º	NO Agrupado (NO tiene sentido agrupar)	Denso (única opción)

PERO si fuese

```
SELECT E.dno, COUNT(*)  
FROM EMP E  
WHERE E.sal > 10000  
GROUP BY E.dno
```

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
E.dno, E.sal	B+ (rango)	2º	NO Agrupado (NO tiene sentido agrupar)	Denso (única opción)

Ejemplos planes solo-índice

```
SELECT E.dno, MIN(E.sal)
FROM EMP E
GROUP BY E.dno
```

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

Campo	Estructura	1° / 2°	Agrupado /No Agrupado	Denso / Disperso
E.dno, E.sal	B+ (rango)	2°	NO Agrupado (NO tiene sentido agrupar)	Denso (única opción)

```
SELECT AVG(E.sal)
FROM EMP E
WHERE E.age = 25
AND E.sal BETWEEN 3000 AND 5000
```

Campo	Estructura	1° / 2°	Agrupado /No Agrupado	Denso / Disperso
E.age, E.sal	B+ (rango)	2°	NO Agrupado (NO tiene sentido agrupar)	Denso (única opción)

Ejercicio

Dados los esquemas: EMP (eid, enombre, direc, sal, edad, deptid)
DEPT (did, dnombre, planta, presupuesto)

Supongamos las siguientes consultas:

- 1.- eid, enombre y direc de empleados con edad dentro de un rango especificado por el usuario
 - 2.- eid, enombre y direc de empleados que trabajen en el departamento cuyo nombre es especificado por el usuario
 - 3.- eid y direc de empleados cuyo nombre sea el especificado por el usuario
 - 4.- Salario medio de empleados
 - 5.- Salario medio de empleados por edad (es decir, para cada edad listar la edad y el salario medio correspondiente)
 - 6.- Listar toda la información de departamentos ordenados por planta
-
- a) Escribir las sentencias SQL correspondientes a cada consulta
 - b) Para cada una de las consultas anteriores, considerándolas de forma individual, ¿qué índices y de qué tipo se deberían crear? Justifica tus elecciones
 - c) Si el SGBD soporta planes sólo-índice ¿cómo cambiaría tu respuesta?
 - d) Repetir los casos b) y c) considerando la existencia de las 6 consultas, siendo todas ellas equivalentes en frecuencia e importancia, y considerando que son más importantes que las actualizaciones.

Clustering

Un cluster contiene datos de 1 o más tablas
Implementa relaciones 1:N

1000		
Dupont	calle...	París
10	05/01/91	
30	07/01/91	
50	28/01/91	
90	02/02/91	

1001		
Simon	París
20	15/11/90	
40	26/12/88	
60	02/02/91	
100	02/02/91	

1002		
Toto	...	Lyon
70	15/11/90	
80	28/01/91	

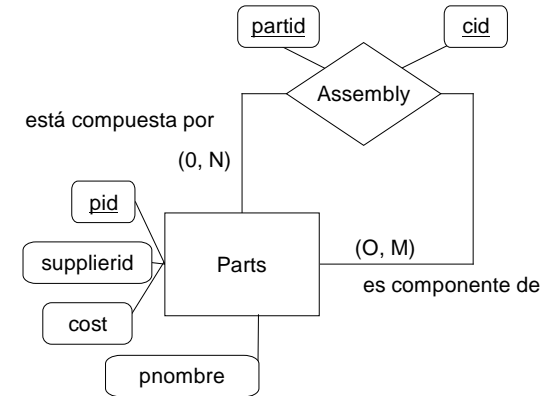
→ clave de cluster

- **Ventajas**
 - Mejora el rendimiento si los registros se consultan de forma conjunta
- **Desventajas**
 - Actualización más costosa
 - Consultas a una tabla sola son más costosas
 - Si la clave de cluster cambia \Rightarrow la línea debe “recolocarse”
- La CLAVE DE CLUSTER no debería actualizarse con mucha frecuencia

Clustering

Ejemplo: Componentes (inmediatos) de cada pieza,
ordenando las piezas por código

```
SELECT P.pid, P.pnombre, A.cid
FROM PARTS P, ASSEMBLY A
WHERE P.pid = A.partid
ORDER BY P.pid
```

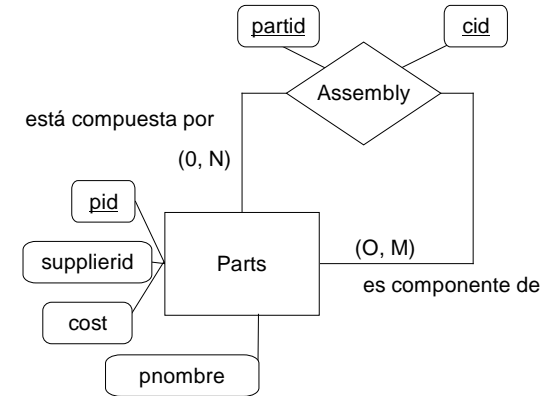


OPCIÓN 1 (INDICES):

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
P.pid	B+ (ordenado)	1º	Agrupado (se ordenan por pieza)	Disperso (+ pequeño)
A.partid	Hash (igualdad)	2º	Agrupado (No clave, ∃ duplicados)	Denso (única opción para índices Hash)

OPCIÓN 2 (CLUSTERING) = 1 reg. PIEZA + { COMPONENTES / P.pid = A.partid }

Clustering



Ejemplo: Componentes (inmediatos) de las piezas suministradas por “X”

```

SELECT P.pid, P.pnombre, A.cid
FROM PARTS P, ASSEMBLY A
WHERE P.pid=A.partid AND P.supplierid='X'
    
```

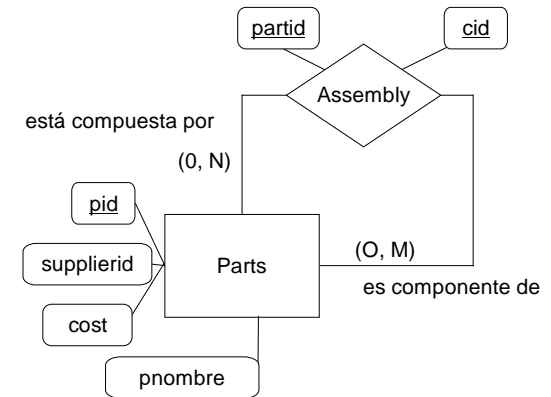
OPCIÓN 1 (INDICES):

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
P.supplierid (si tuplas='X' ↓↓)	Hash (igualdad)	2º	No Agrupado (↓↓ duplicados, ordenación cara)	Denso (única opción para índices Hash)
A.partid	Hash (igualdad)	2º	Agrupado (No clave, ∃ duplicados)	Denso (única opción para índices Hash)

OPCIÓN 2 (CLUSTERING) = 1 reg. PIEZA + { COMPONENTES / P.pid = A.partid } + índice *P.supplierid*
 ⇒ índice **A.partid** NO se necesita

Clustering

```
SELECT P.pid, A.cid
FROM PARTS P, ASSEMBLY A
WHERE P.pid=A.partid AND P.cost=10
```



OPCIÓN 1 (INDICES):

Campo	Estructura	1º / 2º	Agrupado /No Agrupado	Denso / Disperso
P.cost (si tuplas = 10 ↑↑)	Hash (igualdad)	2º	Agrupado (No clave, ↑↑ duplicados)	Denso (única opción para índices Hash)
A.partid	Hash (igualdad)	2º	Agrupado (No clave, ∃ duplicados)	Denso (única opción para índices Hash)

OPCIÓN 2 (CLUSTERING): 1 PIEZA + {COMPONENTES / P.pid = A.partid} + **índice P.cost**

⇒ índice **A.partid** NO se necesita

RESUMEN:

El Clustering en relaciones

- Acelera los joins (1:N)
- Recuperación secuencial + lenta
- Actualizaciones que alteran las longitudes de los registros más lentas

Bibliografía

- ***Sistemas de gestión de bases de datos.*** R. Ramakrishnan. McGraw-Hill [cap.9 en 3ª edic.]
- *Fundamentos de Sistemas de Bases de Datos.* Ramez A. Elmasri, Shamkant B. Navathe.
- *Fundamentos de Bases de Datos.* A. Silberschatz. McGraw-Hill
- *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión.* T.Connolly, C. Begg, A. Strachan. Addison-Wesley