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, 4ª edic.]

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

Índice

- Ciclo de vida del desarrollo de sistemas de BD
- Diseño físico de una BD
- Análisis de las transacciones
- Índices
  - Estructuras de datos de índices
  - Tipos de índices
  - Pautas para la selección de índices
  - Optimización utilizando índices

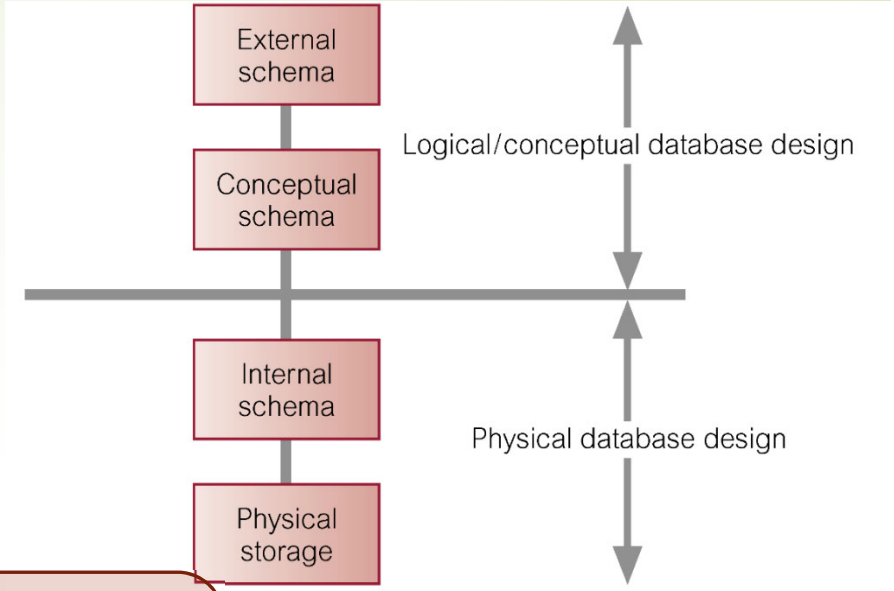
Ciclo de vida del desarrollo de una BD

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

(Connolly & Begg cap.17)

Database Design: Fases de diseño de una BD

- Diseño **conceptual/lógico** *(qué)*
  - Crear el EER (esquema conceptual)
  - 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
     a) Análisis de las transacciones
     b) Elección de la organización de los archivos
     c) Elección de índices
     d) 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

(Connolly & Begg cap.17)

1. Análisis de las transacciones

- **Objetivo**: Comprender las transacciones que se ejecutarán en la BD, y analizar las más importantes.
  - Transacción = {operaciones de acceso a la BD (inserción, consulta, modificación, eliminación) que forman una unidad atómica}

- 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 que la demanda de procesamiento es mayor
  - Criterios de búsqueda utilizados en las consultas
  - Atributos que son actualizados

- Habitualmente no es posible analizar todas las transacciones, pero …
  - *Regla 80/20: 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

(Connolly & Begg cap.17)

Matriz cruzada de transacciones/relaciones

- Muestra las relaciones a las que accede cada transacción

Table 17.1 Cross-referencing transactions and relations.

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								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

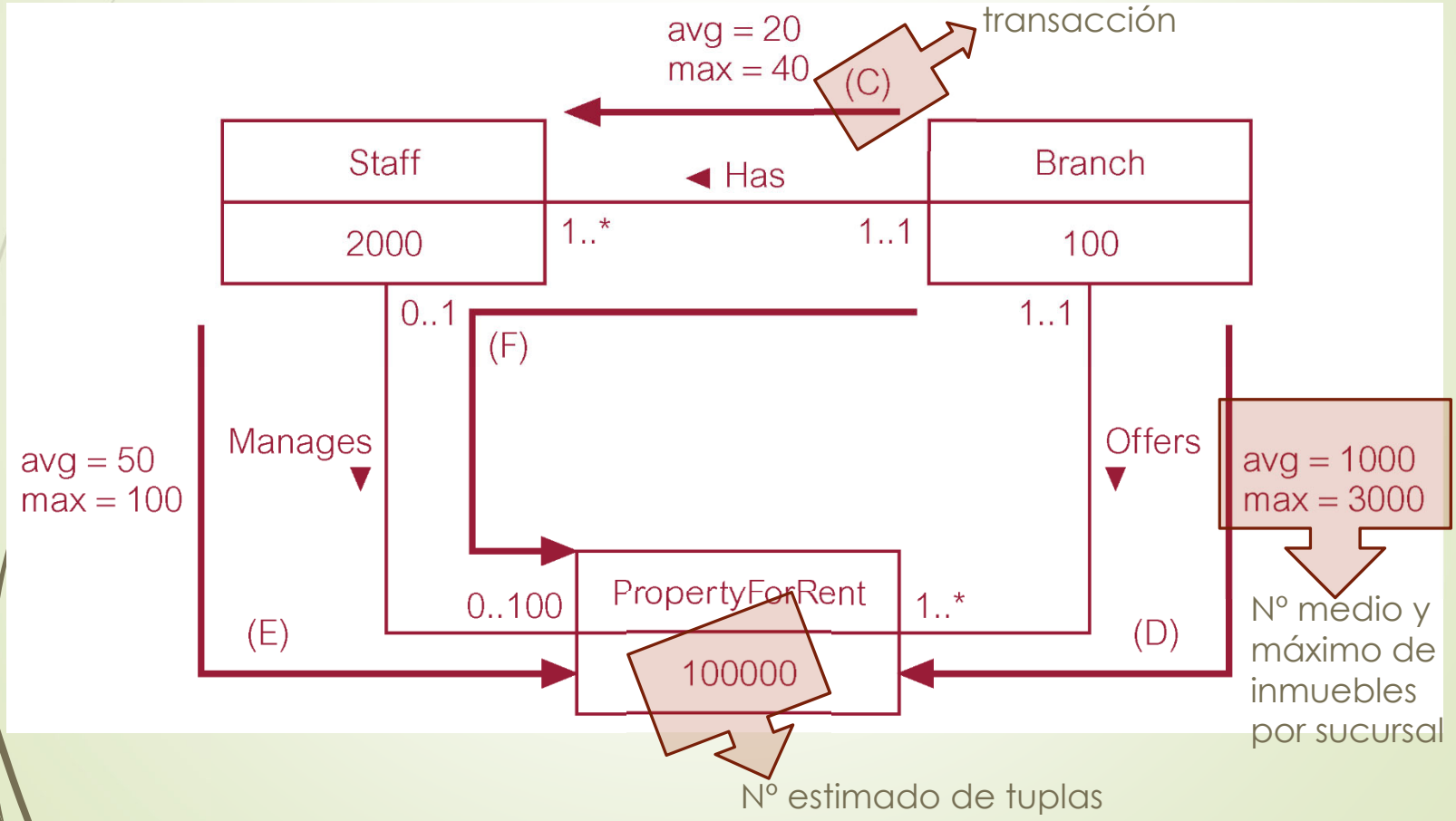
transacción

Nº accesos hora/día/…

(Connolly & Begg cap.17)

Mapa de uso de transacciones

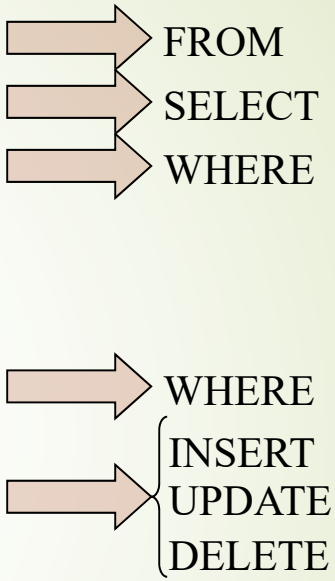
- Indica cuáles son las relaciones potencialmente más utilizadas



(Connolly & Begg cap.17)

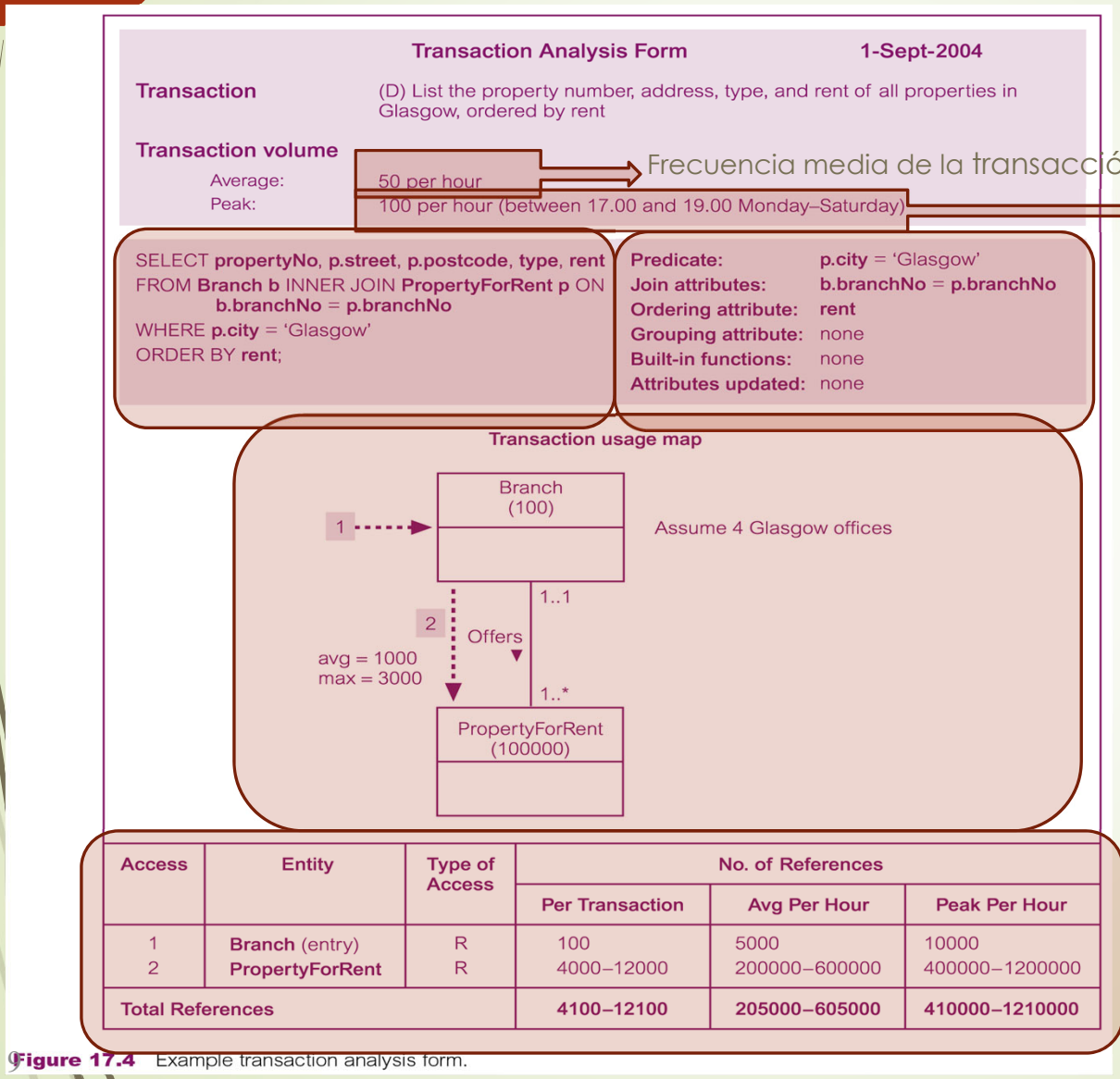
Análisis del uso de los datos

- ∀ CONSULTA de transacciones importantes:
  - RELACIONES ?????
  - ATRIBUTOS recuperados ?????
  - ATRIBUTOS involucrados en Joins/Selecciones ?????
    - Candidatos a estructuras de acceso auxiliares
- ∀ 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
- Frecuencia esperada de ejecución
  - Ej: *50 veces al día*
- Objetivos de rendimiento
  - Ej: *la transacción debe finalizarse en menos de 1 segundo*
  - Atributos de transacciones muy frecuentes o críticas son candidatos a estructuras de acceso auxiliares



(Connolly & Begg cap.17)

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

Mapa de uso de la transacción

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**

Resumen. Para cada relación:
- Tipo de acceso
- Nº tuplas accedidas por ejecución
- Nº tuplas accedidas por hora

Figure 17.4 Example transaction analysis form.

(Connolly & Begg cap.17)

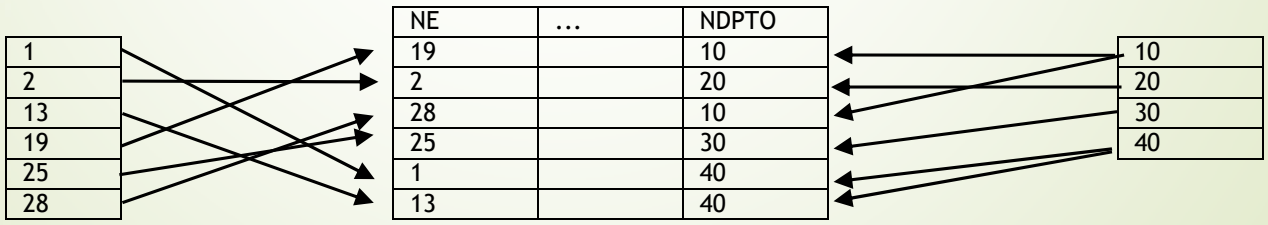
2.- Elección de Índices

- Estructura auxiliar que ayuda a localizar datos de un archivo bajo una condición
- Se asocia a una CLAVE DE BÚSQUEDA
  - ∀ campo puede ∈ CLAVE BÚSQUEDA
  - CLAVE BÚSQUEDA ≠ Clave
- Está compuesto por ENTRADAS DE DATOS DEL INDICE
  - La entrada *k** localiza los registros de datos con CLAVE BUSQUEDA = *k*

    <k, rowid** de registro con clave de búsqueda **k>

    <k, lista-rowid** de registros con clave de búsqueda **k>

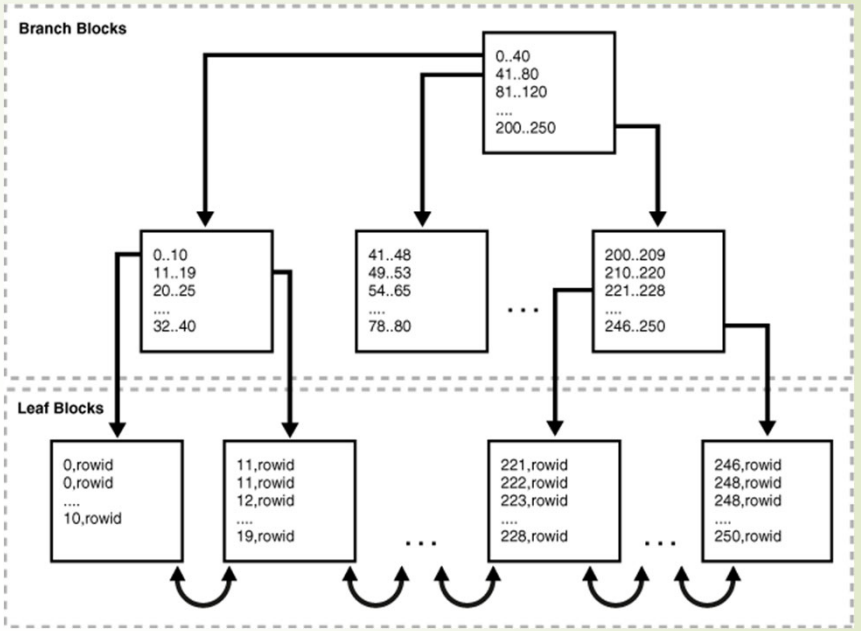
    (las entradas de datos del índice son variables en longitud)



(Ramakrishnan, cap.9)

Estructuras de datos de índices

- Árbol B+:
  - Balanceado, Ordenado
  - Las entradas de datos del índice son las hojas
  - Las entradas de 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 entradas 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 $\log_2$ 100.000.000 > 25 operaciones E/S)

(Ramakrishnan, cap.9)

Estructuras de datos de Índices

- Índice bitmap
  - Almacena un bitmap por cada valor de clave
  - Cada entrada de datos del índice almacena punteros a varios registros de datos
  - Cada bit en el bitmap se mapea a un rowid
    - Si el bit es 1, el registro contiene el valor clave

Ej:

```
SELECT ID, LAST_NAME, MARITAL_STATUS, GENDER
FROM   CUSTOMERS
ORDER BY id;
```

ID	LAST_NAME	MARITAL_STATUS	GENDER
1	Kessel		M
2	Koch		F
3	Emmerson		M
4	Hardy		M
5	Gowen		M
6	Charles	single	F
7	Ingram	single	F

Un índice bitmap para la columna Gender sería de la forma:

Value	Row1	Row2	Row3	Row4	Row5	Row6	Row7
M	1	0	1	1	1	0	0
F	0	1	0	0	0	1	1

(Ramakrishnan, cap.9)

Estructuras de datos de Índices

- Índice bitmap
  - Se utiliza cuando:
    - Existe un nº elevado de registros de datos
    - El nº de valores distintos es menos del 1% del nº registros de datos (p.ej. género, estado civil,…)
    - El fichero es de solo lectura o estático
    - Las consultas son por igualdad con AND, OR y NOT

Ej: Supongamos una tabla con registros como se muestran a continuación:

CUSTOMER #	MARITAL_ STATUS	REGION	GENDER	INCOME_ LEVEL
101	single	east	male	bracket_1
102	married	central	female	bracket_4
103	married	west	female	bracket_2
104	divorced	west	male	bracket_4
105	single	central	female	bracket_2
106	married	central	female	bracket_3

Supongamos que existen índices bitmap para las columnas MARITAL_STATUS y REGION. Entonces, dada la consulta:

```
SELECT COUNT(*) FROM CUSTOMER
WHERE MARITAL_STATUS = 'married' AND REGION IN ('central', 'west');
```

Los índices bitmap pueden procesar esta consulta con gran eficiencia contando el número de unos en el bitmap resultante, como se muestra a continuación:

status = 'married'		region = 'central'		region = 'west'						
0		0		0		0		0		0
1		1		0		1		1		1
1	AND	0	OR	1	=	1	AND	1	=	1
0		0		1		0		1		0
0		1		0		0		1		0
1		1		0		1		1		1

Para localizar los clientes que cumplen la condición, bastará con utilizar el bitmap final para acceder a la tabla.

(Ramakrishnan, cap.9)

Estructuras de datos de Índices

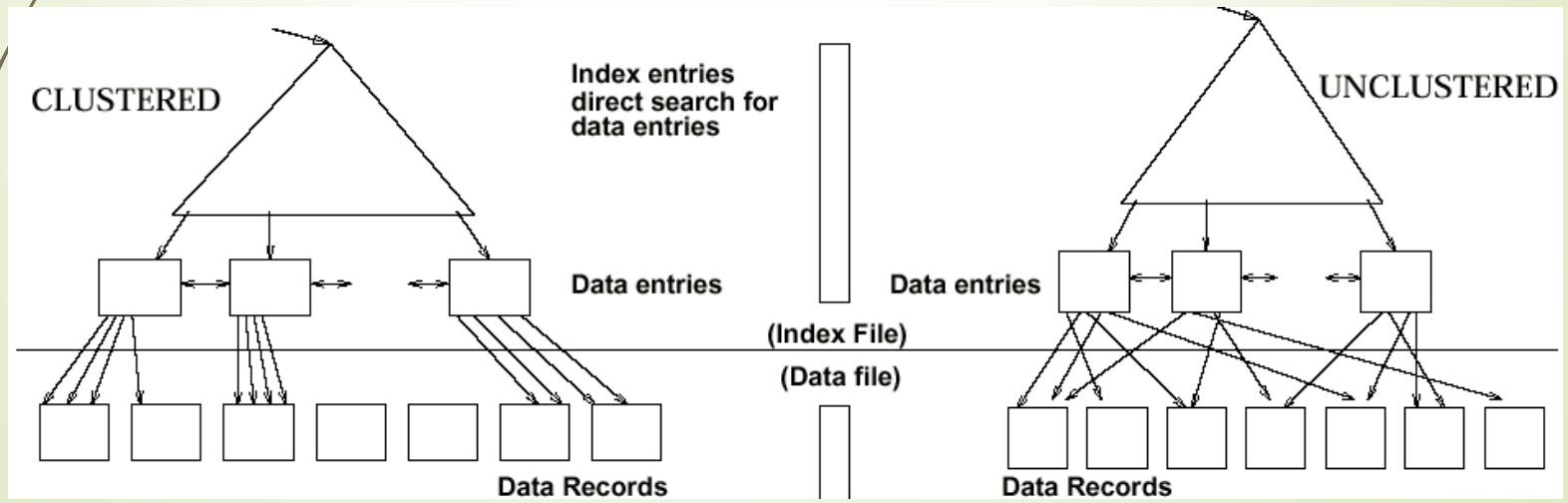
- Árboles B+ y bitmap
  - Los mapas de bits se pueden emplear como mecanismo de almacenamiento comprimido en los nodos hoja de los árboles B+ de los valores clave que aparecen muy frecuentemente.
  - *<k, lista de rowids de registros con clave de búsqueda k>*

se sustituye por un mapa de bits
si k aparece en muchos registros

(Ramakrishnan, cap.9)

Clasificación de Índices

- Únicos vs. No Únicos
  - Único ⇒ garantiza que no contiene duplicados
  - No único ⇒ puede contener duplicados
- Agrupados (CLUSTERED) vs. No Agrupados (UNCLUSTERED)
  - Agrupado ⇒ orden reg. datos = orden entradas índice
  - máximo = 1 índice AGRUPADO por archivo de datos
  - Coste de recuperación varía ENORMEMENTE si el índice está agrupado (en duplicados y rangos)
  - Agrupado es muy útil en consultas POR RANGO o con muchos duplicados, PERO ES CARO

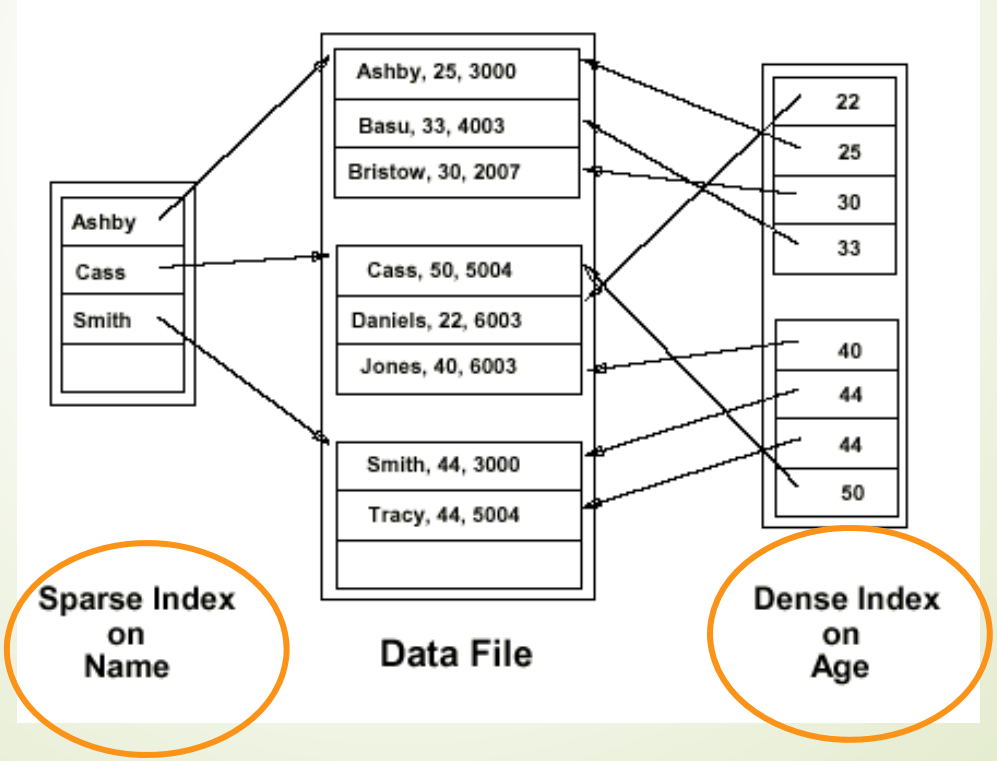


(Ramakrishnan, cap.9)

Clasificación de Índices

- Densos vs. Dispersos
  - Denso ⇒ Existe una entrada de datos del índice por cada **registro** de datos
  - Disperso (spare) ⇒ Existe una entrada de datos del índice por cada **página** de datos

    ⇒ AGRUPADO ⇒ máximo 1 índice DISPERSO por archivo de datos



(Ramakrishnan, cap.9)

Ejemplos de índices simples

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

EMP (<u>eno</u>, ename, dno, age, sal, hobby)
DEPT (<u>dno</u>, dname, mgr)

Campo	Agrupado /No Agrupado	Denso / Disperso
E.hobby (si tuplas='Stamps' ↓↓)	NO Agrupado (↓↓ duplicados, ordenación cara, NO consulta por rango)	Denso (única opción)
E.hobby (si tuplas='Stamps' ↑↑)	No se plantea índice porque sería ineficiente. Es más eficiente recuperar todas las tuplas y filtrar	

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

Campo	Agrupado /No Agrupado	Denso / Disperso
E.eno	NO Agrupado (se obtiene sólo 1 tupla)	Denso (única opción)

(Ramakrishnan, cap.9)

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	Agrupado /No Agrupado	Denso / Disperso
E.age (si tuplas *age* > 40 ↓↓)	Agrupado (∃ duplicados, consulta por rango)	Disperso (+ pequeño)
E.age (si tuplas *age* > 40 ↑↑)	No se plantea índice porque sería ineficiente	

El impacto de la agrupación depende del nº tuplas recuperadas:

- 1 tupla (i.e, igualdad sobre clave candidata) ⇒ No Agrupado
- Agrupación es interesante con ↑↑ duplicados
- ↑↑ tuplas ⇒ No Agrupado es + caro que recorrido secuencial

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

Campo	Agrupado /No Agrupado	Denso / Disperso
E.age (si tuplas *age* > 50 ↓↓)	Agrupado (∃ duplicados, consulta por rango)	Disperso (+ pequeño)
ó		
E.dno (si tuplas *age* > 50 ↑↑)	Agrupado (↑↑ duplicados, los empleados se obtienen ordenadamente por departamento)	Disperso (+ pequeño)

(Ramakrishnan, cap.9)

Ejemplo de pasos de ejecución

Describir los pasos que sigue el SGBD para resolver la siguiente consulta

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

considerando los siguientes supuestos:

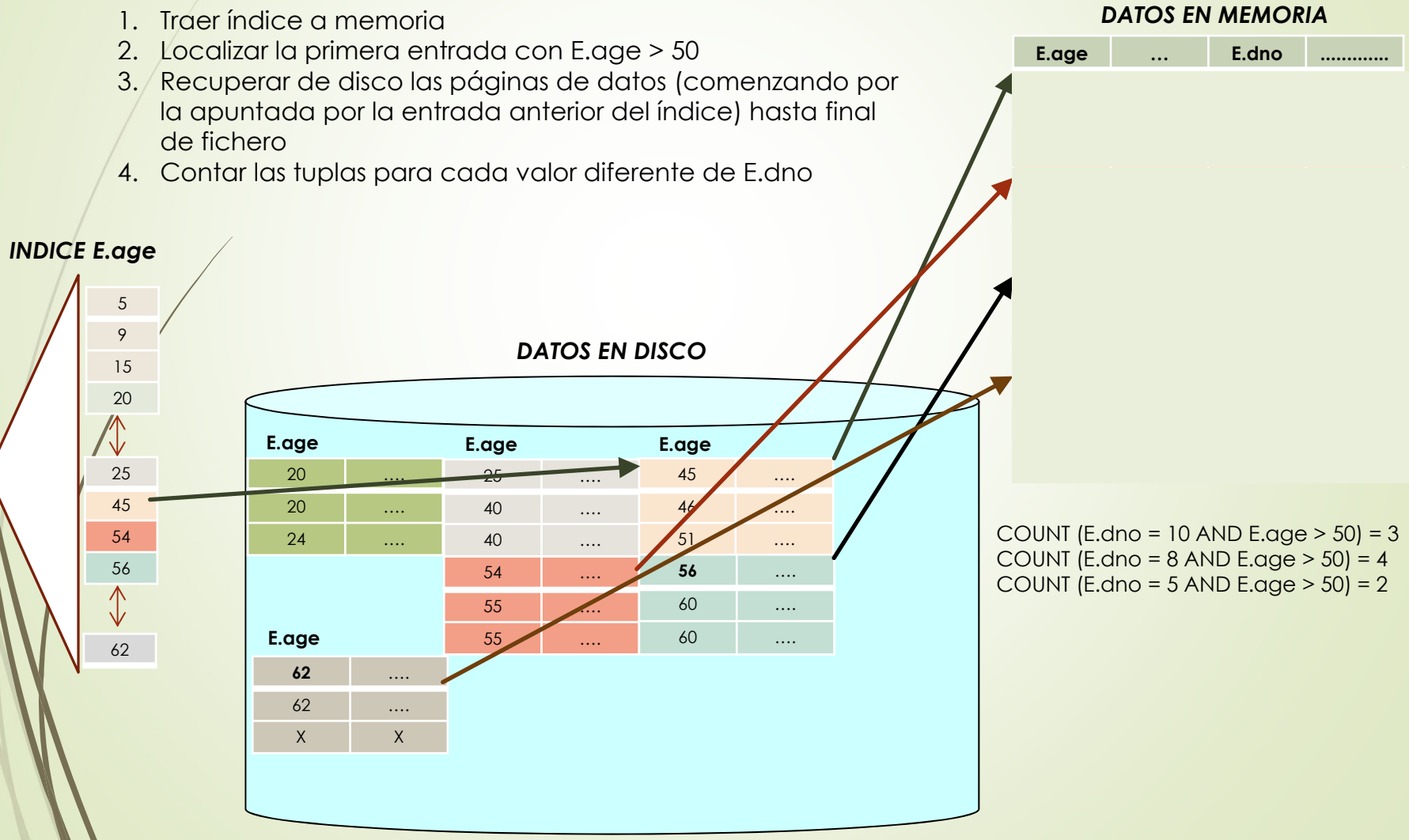
a) Existen pocos empleados con edad mayor de 50, por lo que se ha definido un índice <***E.age***> B+ agrupado disperso

b) Existen muchos empleados con edad mayor de 50, por lo que se ha definido un índice <***E.dno***> B+ agrupado disperso

Índice <E.age> B+ agrupado disperso

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

1. Traer índice a memoria
2. Localizar la primera entrada con E.age > 50
3. Recuperar de disco las páginas de datos (comenzando por la apuntada por la entrada anterior del índice) hasta final de fichero
4. Contar las tuplas para cada valor diferente de E.dno

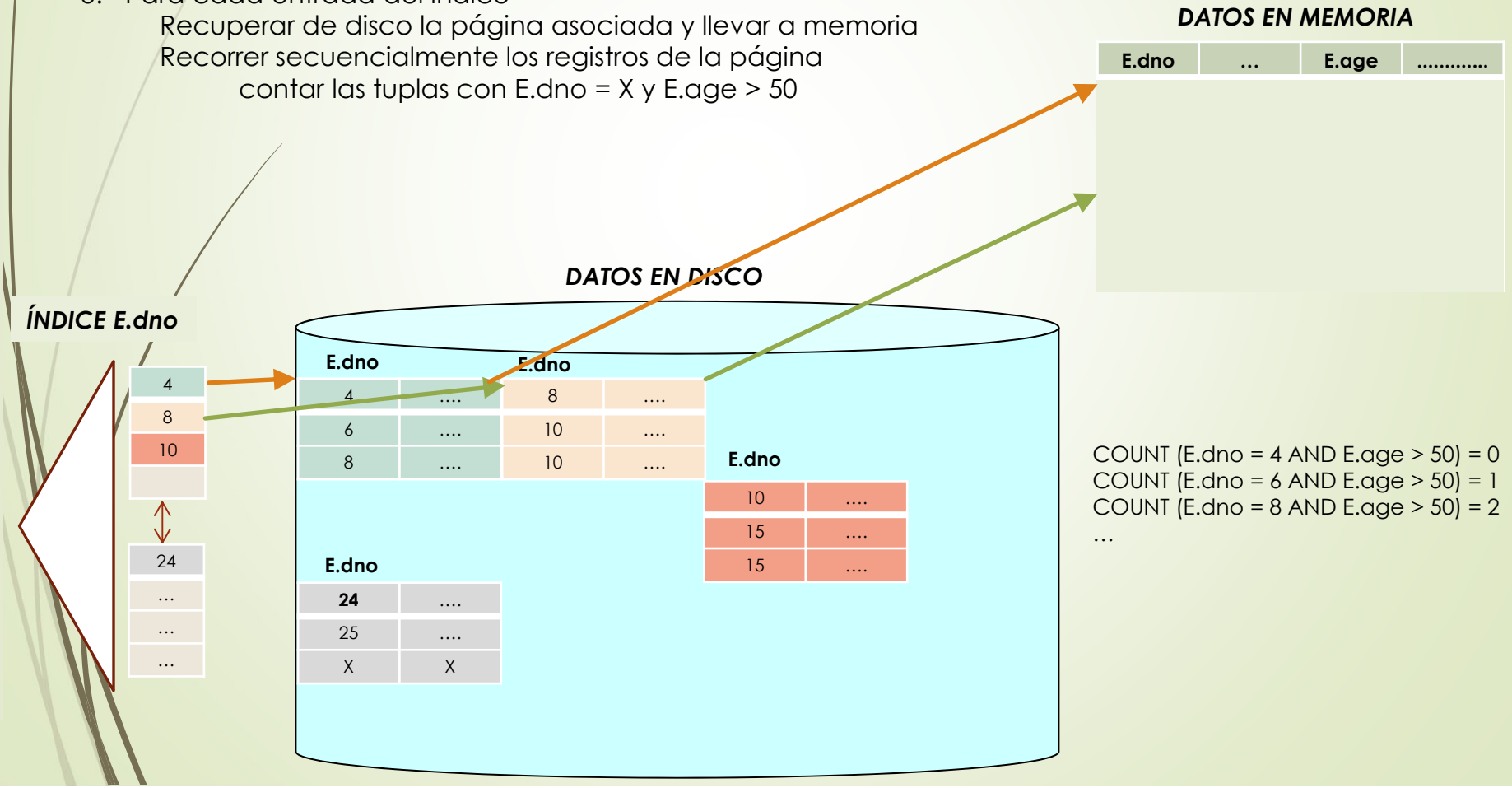


COUNT (E.dno = 10 AND E.age > 50) = 3
COUNT (E.dno = 8 AND E.age > 50) = 4
COUNT (E.dno = 5 AND E.age > 50) = 2

Indice <E.dno> B+ agrupado disperso

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

1. Traer índice a memoria
2. Recorrer secuencialmente las entradas del índice
3. Para cada entrada del índice
   Recuperar de disco la página asociada y llevar a memoria
   Recorrer secuencialmente los registros de la página
   contar las tuplas con E.dno = X y E.age > 50



Ejemplos de índices simples

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

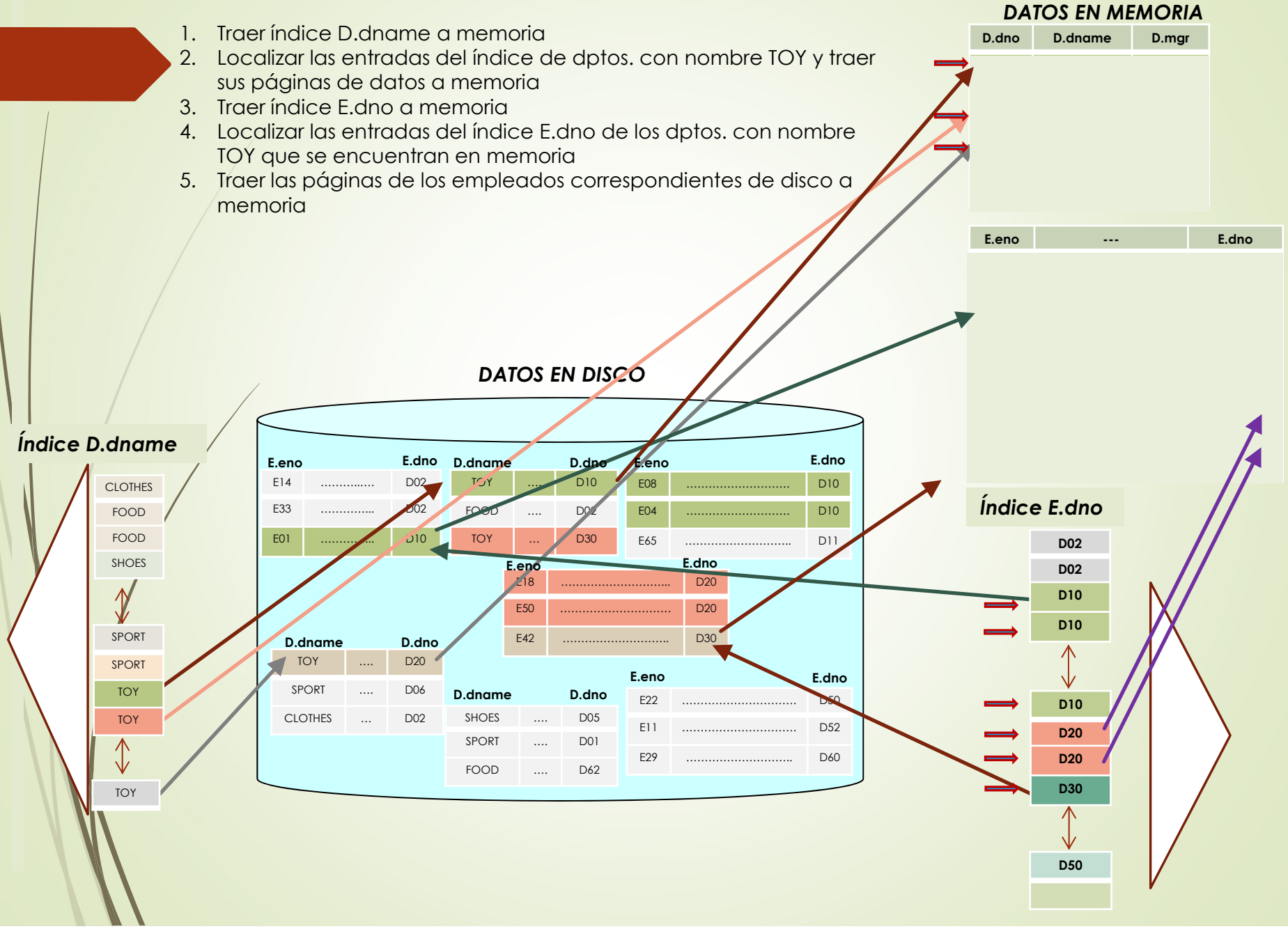
EMP (eno, ename, dno, age, sal, hobby)
DEPT (dno, dname, mgr)

Campo	Agrupado /No Agrupado	Denso / Disperso
D.dname (si tuplas='Toy' ↓↓)	No Agrupado (↓↓ duplicados, ordenación cara, No consulta por rango)	Denso (única opción)
E.dno	Agrupado (No clave, ∃ duplicados)	Disperso (+ pequeño)

(Ramakrishnan, cap.9)

Indices <D.dname> No agrupado y <E.dno> agrupado

1. Traer índice D.dname a memoria
2. Localizar las entradas del índice de dptos. con nombre TOY y traer sus páginas de datos a memoria
3. Traer índice E.dno a memoria
4. Localizar las entradas del índice E.dno de los dptos. con nombre TOY que se encuentran en memoria
5. Traer las páginas de los empleados correspondientes de disco a memoria



Ejemplos de índices simples

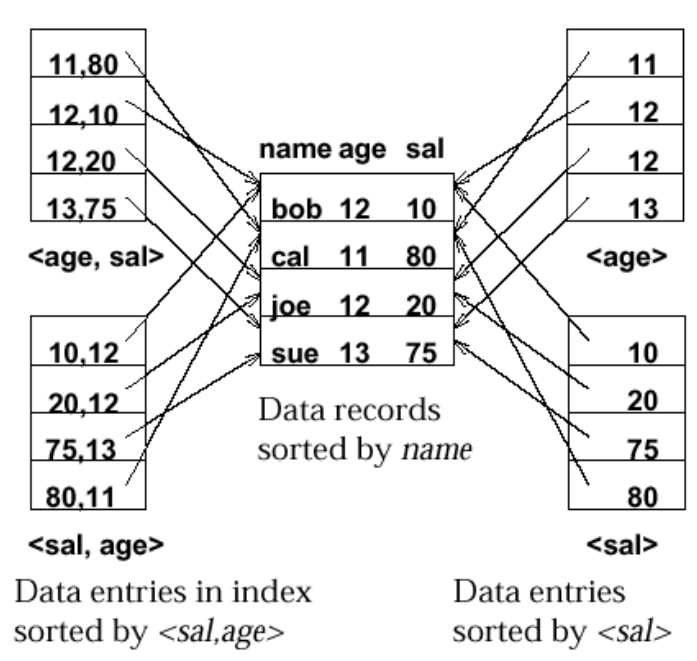
```
SELECT E.ename, D.mgr
FROM EMP E, DEPT D
WHERE D.dname='Toy' AND E.dno=D.dno
      AND E.age = 25
```

Campo	Agrupado /No Agrupado	Denso / Disperso
D.dname (si tuplas='Toy' ↓↓)	No Agrupado (↓↓ duplicados, ordenación cara, No consulta por rango)	Denso (única opción)
E.age (si tuplas = 25 ↓↓)	No Agrupado (↓↓ duplicados, ordenación cara, No consulta por rango)	Denso (única opción)

(Ramakrishnan, cap.9)

Clasificación de Índices

- Índice Compuesto
  - Formado por varios campos



- Consulta por IGUALDAD ⇒ ej: *age = 20 and sal = 10*
- Consulta por RANGO
  (+1) campo NO es constante ⇒ ej: *age = 30 and sal > 40*
  ⇒ ej: *age = 20*
- 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*>

(Ramakrishnan, cap.9)

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, HAVING y ORDER BY son candidatos
  - Condición por igualdad o rango sugiere índice en árbol B+
    - Muy eficientes para joins y consultas exactas
    - La agrupación es especialmente útil para consultas por rango
- 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 son interesantes cuando WHERE / GROUP BY / ORDER BY contienen varias condiciones
  - Si hay selecciones por rango, ordenar los atributos del índice en base a la consulta

(Ramakrishnan, cap.9)

Utilización de índices

- Cuando existen varios índices, el SGBD puede optar por
  1. Utilizar 1 Índice
     - Elige el índice que filtra más tuplas
  2. Utilizar Múltiples Índices
     - Aplica ∀ índices, recuperando el rowid = IdPágina + Id
     - Intersecta los conjuntos de tuplas obtenidos (si AND), o los une (si OR)
     - Ordena el resultado por IdPágina

(Ramakrishnan, cap.9)

Utilizando múltiples índices

```
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
```

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

Campo	Agrupado /No Agrupado	Denso / Disperso
D.dno	NO Agrupado (suponiendo pocas tuplas en la tabla)	Denso (única opción)
E.sal	Agrupado (No clave, ∃ duplicados, consulta rango)	Disperso (+ pequeño)
y		
E.hobby	NO Agrupado (sólo puede haber más de 1 índice agrupado por tabla)	Denso (única opción)

Si se crean *E.sal* **y** *E.hobby* el optimizador puede: 1) elegir el índice más selectivo, o 2) intersectar los identificadores de ambos índices

E.hobby debe ser NO Agrupado
(no puede haber más de 1 índice agrupado por tabla y se prioriza el rango)

(Ramakrishnan, cap.9)

Planes solo-índice

Plan solo-índice

- Si ∀ datos están en el índice ⇒ NO es necesario acceder a los registros de datos
- Condiciones:
  - Índice DENSO
  - ∀ atributos de la parte SELECT forman parte del índice
- Suelen ser COMPUESTOS
- Son NO AGRUPADOS (NO tiene sentido agrupar porque no se accede a la tabla !!!!!)

(Ramakrishnan, cap.9)

Ejemplos planes solo-índice

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

EMP (<u>eno</u>, ename, dno, age, sal, hobby)
DEPT (<u>dno</u>, dname, mgr)

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.dno, E.eno>	**NO Agrupado** (NO se accede a la tabla)	**Denso** (única opción)

⇒ NO es necesario acceder a la tabla EMP

Además se podría crear un índice compuesto <D.dno, D.mgr> que aprovechase también la aproximación solo-índice:

Campo	Agrupado /No Agrupado	Denso / Disperso
<D.dno, D.mgr>	**NO Agrupado** (NO se accede a la tabla)	**Denso** (única opción)

(Ramakrishnan, cap.9)

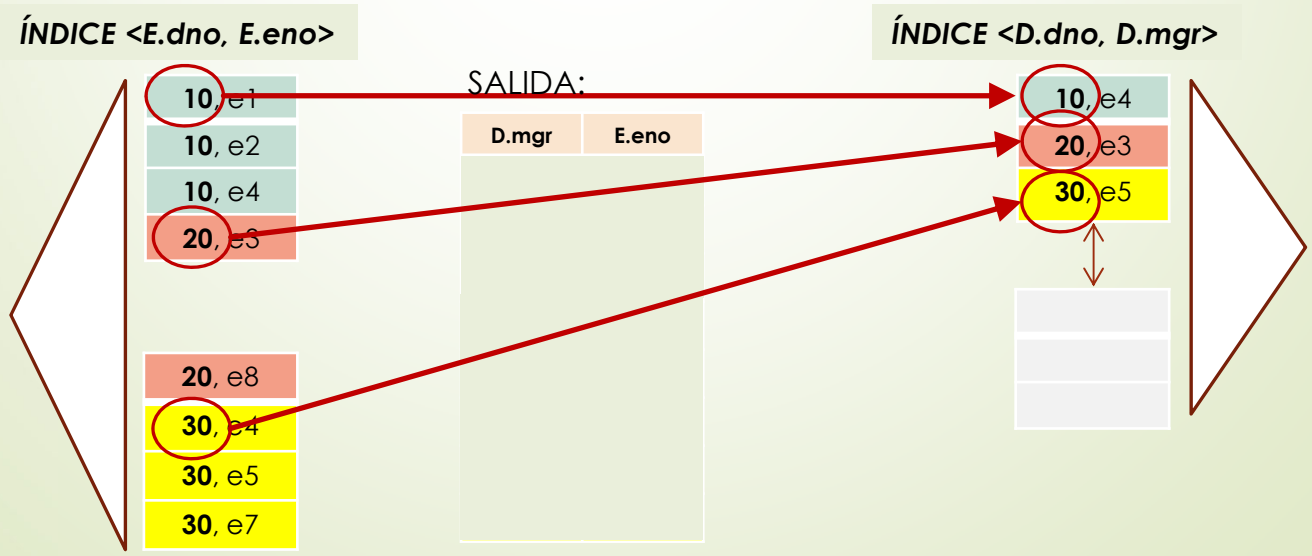
Ejemplos planes solo-índice

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	Agrupado /No Agrupado	Denso / Disperso
<E.dno, E.eno>	**NO Agrupado** (NO se accede a la tabla)	**Denso** (única opción)
<D.dno, D.mgr>	**NO Agrupado** (NO se accede a la tabla)	**Denso** (única opción)

1. Traer índices a memoria
2. Recorrer secuencialmente las entradas del índice <E.dno, E.eno>
3. Para cada E.dno = X diferente
   Localizar la entrada con D.dno = X en el índice <D.dno, D.mgr>
   extraer todos los E.eno con E.dno = X



Ejemplos planes solo-índice

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

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

OPCIÓN BÁSICA

Campo	Agrupado /No Agrupado	Denso / Disperso
E.dno	Agrupado (↑↑ duplicados, los empleados se obtienen ordenadamente por departamento)	Disperso (+ pequeño)

OPCIÓN SOLO-ÍNDICE

Campo	Agrupado /No Agrupado	Denso / Disperso
E.dno	NO Agrupado (NO se accede a la tabla)	Denso (única opción)

⇒ Se responde a la consulta utilizando únicamente el índice

PASOS

1. Traer índice a memoria
2. Recorrer secuencialmente las hojas
3. Contar las entradas con el mismo valor de E.dno

(Ramakrishnan, cap.9)

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)

OPCIÓN BÁSICA

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.sal>	Agrupado (No clave, ∃ duplicados)	Disperso (+ pequeño)

OPCIÓN SOLO-ÍNDICE

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.sal, E.dno>	NO Agrupado (NO se accede a la tabla)	Denso (única opción)

PERO si fuese ….

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

OPCIÓN BÁSICA

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.sal>	Agrupado (consulta por rango)	Disperso (+ pequeño)

OPCIÓN SOLO-ÍNDICE

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.dno, E.sal>	NO Agrupado (NO se accede a la tabla)	Denso (única opción)

(Ramakrishnan, cap.9)

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)

OPCIÓN BÁSICA

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.dno, E.sal>	NO Agrupado (sólo se accede a empleados con salario mínimo por departamento)	Denso (única opción)

OPCIÓN SOLO-ÍNDICE

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.dno, E.sal>	NO Agrupado (NO se accede a la tabla)	Denso (única opción)

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

OPCIÓN BÁSICA

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.age, E.sal>	NO Agrupado (se accede a pocas tuplas tras 2 filtrados)	Denso (única opción)

OPCIÓN SOLO-ÍNDICE

Campo	Agrupado /No Agrupado	Denso / Disperso
<E.age, E.sal>	NO Agrupado (NO se accede a la tabla)	Denso (única opción)