

Motivació Emmgatzematge 1

```

drop table bulto;
Create table bulto(a integer primary key, b char(60), c integer);

-- insertar 4000 tuples
CREATE OR REPLACE FUNCTION ins_4000() RETURNS void
AS $$
DECLARE
i INTEGER;
BEGIN
  FOR i in 1 .. 4000 LOOP
    insert into bulto values(i,i,i);
  END LOOP;
END;
$$LANGUAGE plpgsql;

select * from ins_4000();

```

```

select count(*)
from bulto b1, bulto b2
where b1.c>10*b2.c

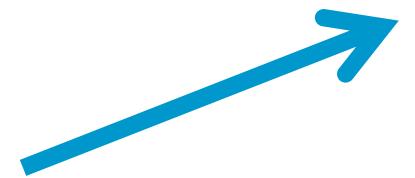
```

```

drop index prova;
create unique index prova on bulto(c) with (fillfactor= 80);

select count(*)
from bulto b1, bulto b2
where b1.c>10*b2.c

```



Subfinestra de sortida

Sortida de dades	Explain	Missatges	Historial
------------------	---------	------------------	-----------

Temps total d'execució de la consulta:4591 ms.
1 registre trobat.



Subfinestra de sortida

Sortida de dades	Explain	Missatges	Historial
------------------	---------	------------------	-----------

Temps total d'execució de la consulta:170 ms.
1 registre trobat.

Motivació Emmgatzematge 2

```

drop table bulto;
Create table bulto(a integer primary key, b char(60), c integer);

-- insertar 400000 tuples
CREATE OR REPLACE FUNCTION ins_400000() RETURNS void
AS $$
DECLARE
i INTEGER;
BEGIN
  FOR i in 1 .. 400000 LOOP
    insert into bulto values(i,i,i);
  END LOOP;
END;
$$LANGUAGE plpgsql;

select * from ins_400000();

```

Explain
 Select * from bulto where c=240000;

```

drop index prova;
create unique index prova on bulto(c) with (fillfactor=80);

```

Explain
 Select * from bulto where c=240000;



QUERY PLAN	
	text
1	Seq Scan on bulto (cost=0.00..9939.00 rows=1 width=69)
2	Filter: (c = 240000)



QUERY PLAN	
	text
1	Index Scan using indx on bulto (cost=0.00..8.76 rows=1 width=69)
2	Index Cond: (c = 240000)

9. Emmagatzemament i mètodes d'accés a les BD

- Memòria externa
- L'arquitectura dels components d'emmagatzematge
- Implementació de mètodes d'accés
- Annex: Càlcul de costos pels diferents mètodes d'accés

Memòria externa: Necessitats

	mem. interna	mem. externa
preu	alt	baix
temp d'obtenció	constant (10^{-6} s a 10^{-9} s) micros a nanos	variable (10^{-3} s) milis
capacitat	poca	il.limitada
volatilitat	si	no volàtil

- La memòria externa és necessària perquè és **permanent** (no volàtil) i proporciona una gran **capacitat**.
- La memòria externa té el problema del **temps** d'obtenció de les dades.

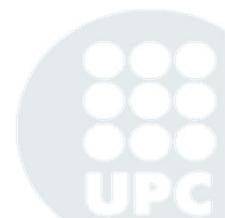
L'arquitectura dels components d'emmagatzematge

- Introducció i objectius
- Arquitectura dels components d'emmagatzematge
- El nivell físic
 - La pàgina
 - La fila
 - El camp
 - Gestió de la pàgina
 - Altres tipus de pàgines
 - Extensió i fitxer
 - Entrades/sortides en SGBD
- El nivell virtual
 - Justificació, definició
 - Estructura de l'espai virtual
 - Adreçament en un SGBD
 - Record IDentifier (RID)
 - Tipus d'espais virtuals
- Altres aspectes



Arquitectura dels components d'emmagatzematge: Introducció i objectius

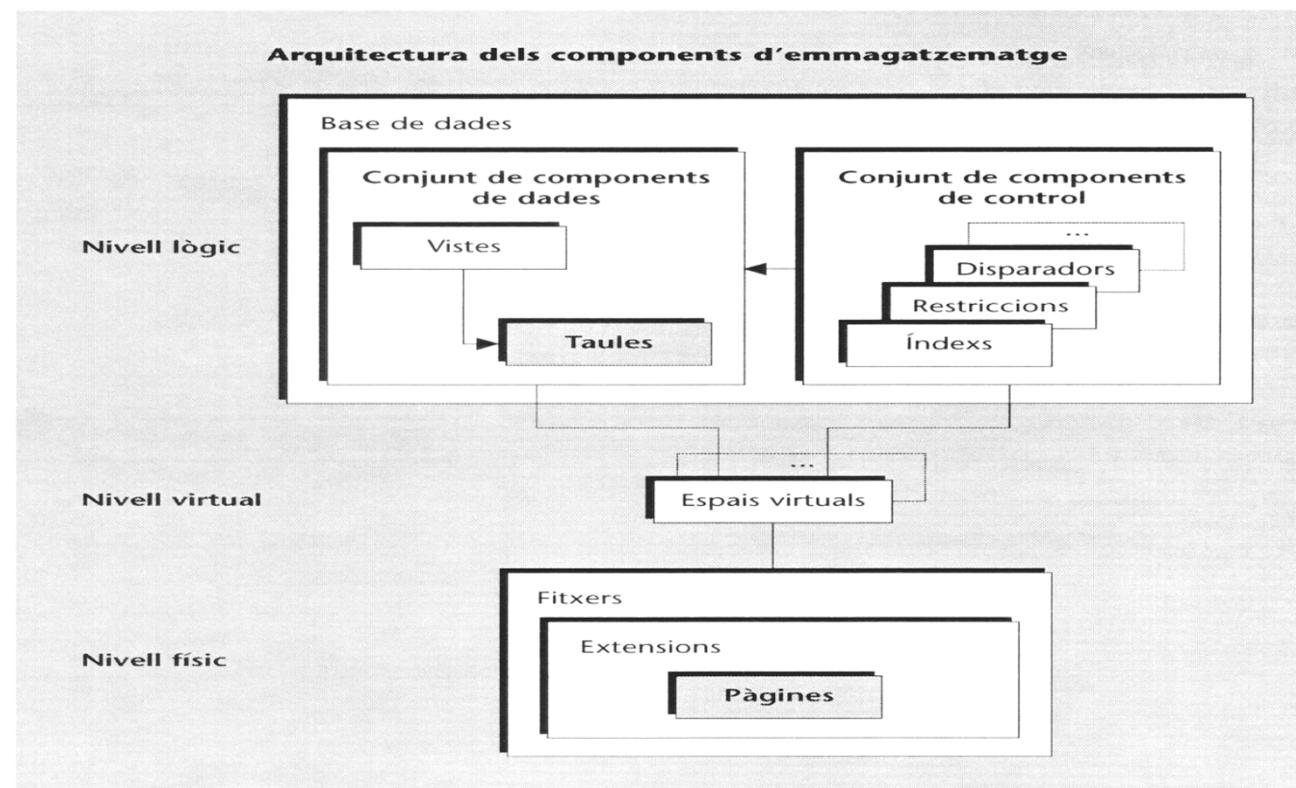
- Els components lògics s'han d'emmagatzemar en un suport no volàtil -> cal garantir-ne l'accessibilitat.
- Cada sistema segueix un model diferent (no hi ha estàndard per a les qüestions físiques) però existeixen uns patrons similars si deixem de banda els detalls.
 - “màxim comú divisor” dels components usats en els sistemes comercials
- **Objectius General**
 - Tot i que les dades d'una base de dades és guarden en fitxers, aquests segueixen uns patrons més complexos que els dels fitxers simples. L'objectiu és conèixer aquests patrons, per tal de poder fer un disseny físic d'una base dades més acurat.



Arquitectura dels components d'emmagatzematge

- BD: Conjunt de dades persistents!
 - No han de desaparèixer entre execucions
 - Emmagatzemades en un medi no volàtil (disc magnètic)
- L'arquitectura de tres nivells (lògic, virtual, físic) ens serveix per comprendre la relació entre les dades tal i com les veu el programador o usuari final i tal com estan emmagatzemades:

Figura 1



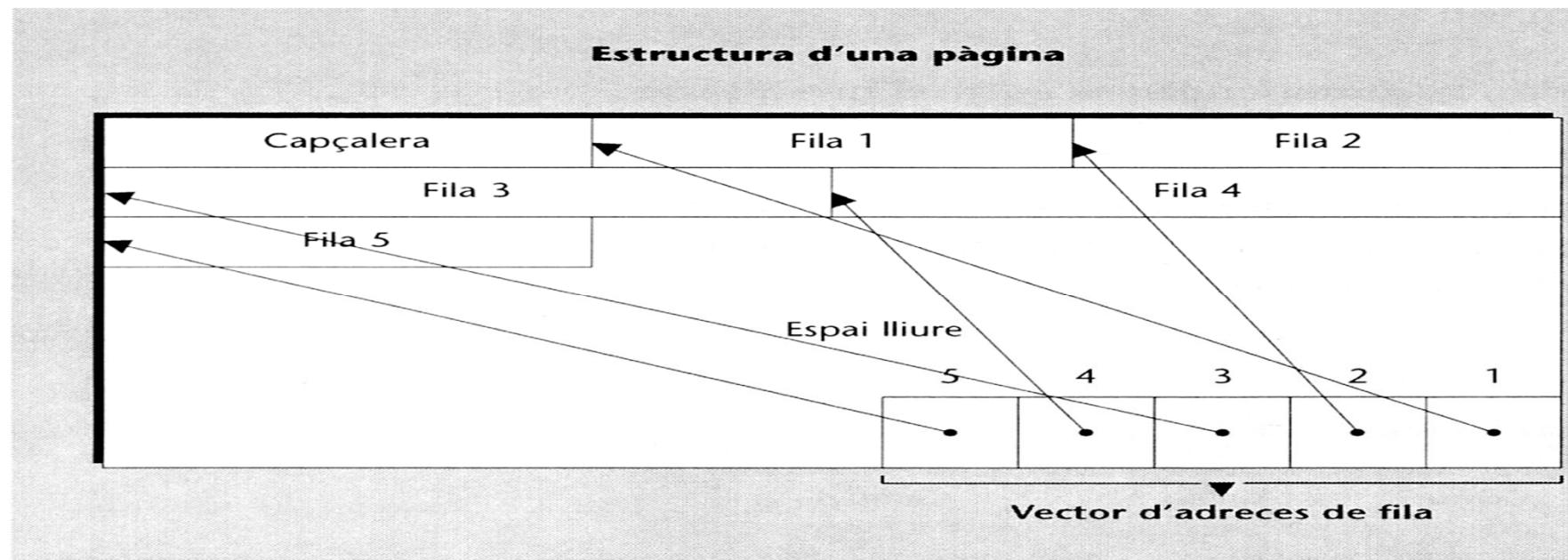
El nivell físic

- Les dades s'emmagatzemen a discos magnètics controlats pels SO, que és qui realment efectua les lectures i escriptures. Ara bé, és l'SGBD el que decideix quan fer aquestes operacions i el que coneix com estan físicament estructurades les dades i les pot interpretar.
- Hi ha tres components essencials:
 - Fitxer: és la unitat global a partir de la qual el sistema operatiu gestiona les dades en els discos magnètics
 - Extensió (extent): és la unitat d'adquisició d'espai per a cada fitxer. L'extensió és un múltiple enter de la pàgina.
 - Pàgina (page): és el component físic més petit. Conté i emmagatzema les dades del nivell lògic



La pàgina

- El concepte de pàgina en BD es pot veure de dos punts de vista:
 - **Unitat discreta de transport de dades (d'E/S)** entre la memòria externa (disc) i memòria interna. En SO normalment *bloc*.
 - **Unitat d'organització de les dades emmagatzemades.** L'espai del disc s'estructura en un nombre múltiple de pàgines, i cada pàgina es pot adreçar individualment.
- És pot accedir a menys d'una pàgina? I a més?
- **Estructura física :** Longitud fixa (2K,4K, 8K).



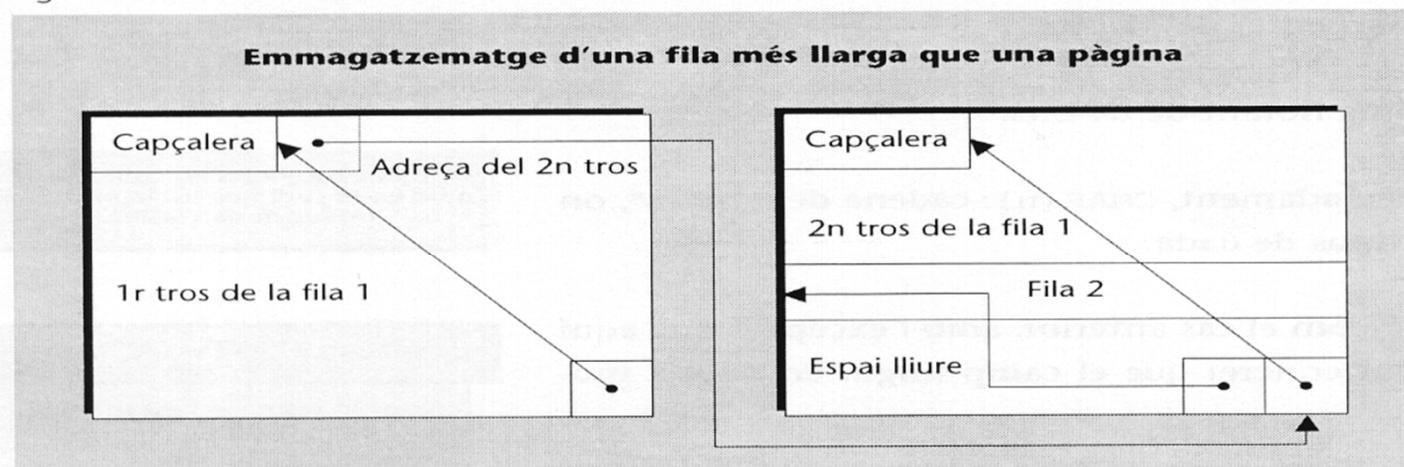
La fila

- Estructura fila:



- La capçalera conté informació com: longitud total de la fila, identificador de la taula a la qual pertany.
- Veiem com seria l'emmagatzematge d'una fila més llarga que una pàgina, tot i que no és aconsellable tenir aquest tipus de files.

Figura 4



El camp

- Estructura física:

Capçalera	Contingut
-----------	-----------

- A la capçalera del camp:

- Ens diu si el camp és Null / not null (si el camp admet valors nuls)
 - Longitud del camp
- Si el camp és de long fixa i not null no té capçalera

- Contingut. Formats més usuals:

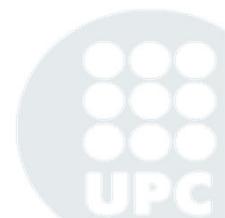
- Smallint: binari enter 2 bytes
- Integer: binari enter 4 bytes
- Float: coma flotant 8 bytes
- Char(n): n bytes en ascii
- Char varying (varchar): n bytes en ascii però n és la longitud real del valor
- Decimal(p,s): $\lceil (p+1)/2 \rceil$ (és interessant comparar-lo amb float)
- Date: aaa | mm | dd 4 bytes

- Exemple

```
Create table T ( camp1 varchar(10),
                 camp2 varchar(10),
                 camp3 char(10) not null)
Insert into T values ("Joan", NULL, "123")
```

Camp 1 6 bytes	Camp 2 1 bytes	Camp 3 10 bytes
1	4	Joan

1	4	Joan	0	123
---	---	------	---	-----



Motivació Emmagatzematge 3

```
drop table bulto;
Create table bulto(a integer primary key, b varchar(60), c integer);
```

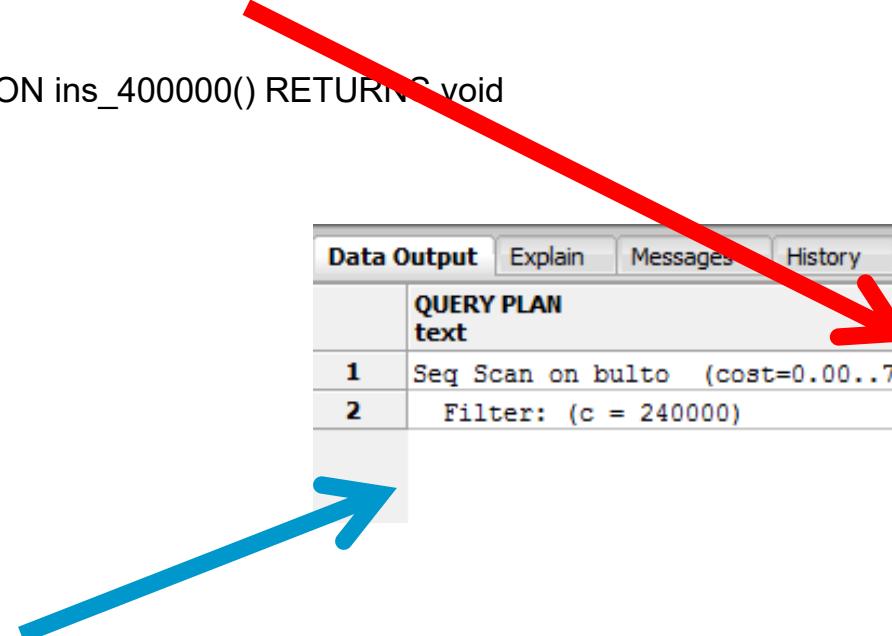
```
-- insertar 400000 tuples
CREATE OR REPLACE FUNCTION ins_400000() RETURNS void
AS $$
DECLARE
i INTEGER;
BEGIN
FOR i in 1 .. 400000 LOOP
    insert into bulto values(i,i,i);
END LOOP;
END;
$$LANGUAGE plpgsql;
```

```
select * from ins_400000();
```

Explain
 Select * from bulto where c=240000;

```
drop index prova;
create unique index prova on bulto(c) with (fillfactor=80);
```

Explain
 Select * from bulto where c=240000;



QUERY PLAN text	
1	Seq Scan on bulto (cost=0.00..7163.00 rows=1 width=14)
2	Filter: (c = 240000)

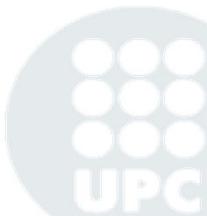


QUERY PLAN text	
1	Index Scan using indx on bulto (cost=0.00..8.76 rows=1 width=69)
2	Index Cond: (c = 240000)

Gestió de la pàgina

El SGBD utilitza una sèrie de procediments de gestió de les pàgines i del seu contingut:

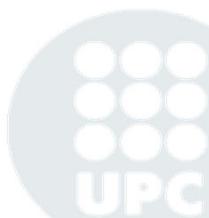
- Formatació
- Càrrega inicial de files
- Alta posterior d'una nova fila
- Baixa d'una fila
- Adaptació al canvi de longitud d'una fila



Altres tipus de pàgines

- Pàgines d'altres components: vistes, disparadors, procediments
 - Totes les definicions al catàleg → com les taules
- Pàgines d'índexs
 - S'explica la seva estructura en la segona part d'aquest tema → Mètodes d'accés
- Pàgines d'objecte gran
 - La representació física obliga a emmagatzemar una quantitat de bytes molt superior a la de les dades tradicionals:

• Nom clients	varchar(60)	60 bytes màxim
• Powerpoint	transparències	6 MB
 - La representació tradicional no és adequada, així el tractament ha de ser diferent → es dóna una idea de com és més endavant



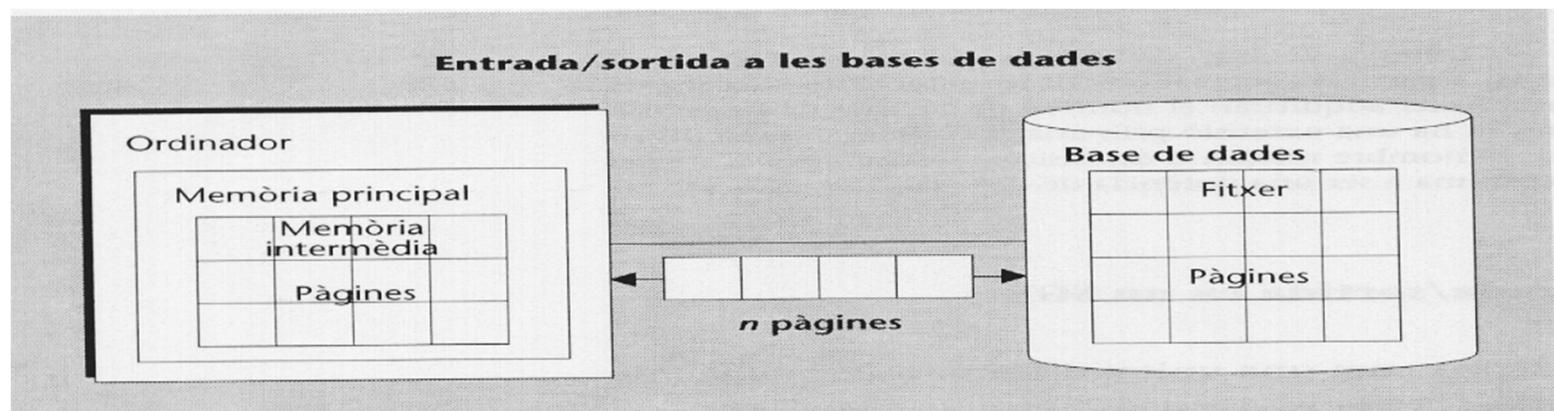
Extensió i Fitxer

- Una extensió és un nombre enter de pàgines consecutives que el SO adquireix a petició de l'SGBD quan aquest detecta que necessita més espai per a un fitxer determinat. Automàtica.
- Un fitxer és un conjunt d'extensions.
- L'SGBD interacciona amb el SO i no amb els fitxers. Pot costar trobar aquest terme de manera explícita.
A alguns SGBD petits tota la BD és un fitxer !!
- Per què han de ser consecutives les pàgines?
 - Eficiència dels tractaments seqüencials !!!
 - Disc: seek=12 msg; latència= 3 msg; Transfer=1 msg (4k)
18Gb 9801 cilindres Cilindre 2Mb pista 150K
 - N pàgines no consecutives → cost = $N(s+r+t)$
» $100000 (12 + 3 + 1) = 26'6$ minuts
 - N pàgines consecutives → aprox cost = $(s+r+ N*t)$
» $(12 + 3 + 100000) = 1,6$ minuts

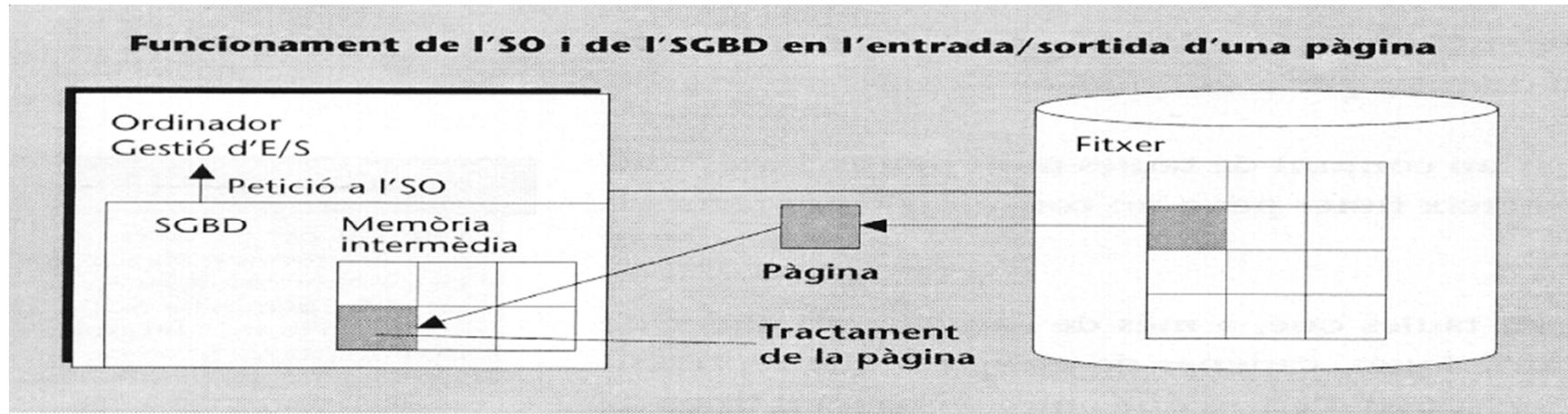


Entrades/sortides en SGBD

- En BD la longitud de la pàgina és fixa per a tots els fitxers.
Senzillesa: Rendiment
- Optimització E/S
 - Doble buffering per a la lectura de pàgines
 - Retenir pàgines a memòria
 - Portar a memòria més d'una pàgina consecutiva (no seek, no latency)



Entrades/sortides en SGBD



- Hi ha diversos motius pels quals la gestió de les pàgines no es pot deixar en mans del sistema operatiu:
 - Degut al concepte de transacció, que només coneix l'SGBD, algunes pàgines s'han de gravar al disc en cert moments del temps: en el moment de commit, en el moment d'abort, etc
 - Optimització

EL nivell virtual: Justificació i definició

- Justificació:

- Si Taula \Leftrightarrow Fitxer, no cal un nivell intermedi que faci corresponde components lògics amb físics, però
 - Si taules molt grans: hi ha fragments en dispositius diferents
 - Si taules molt petites: s'han d'agrupar en un fitxer
 - Si hi ha objectes grans
 - Si hi ha índexs, disparadors ...

- Definició:

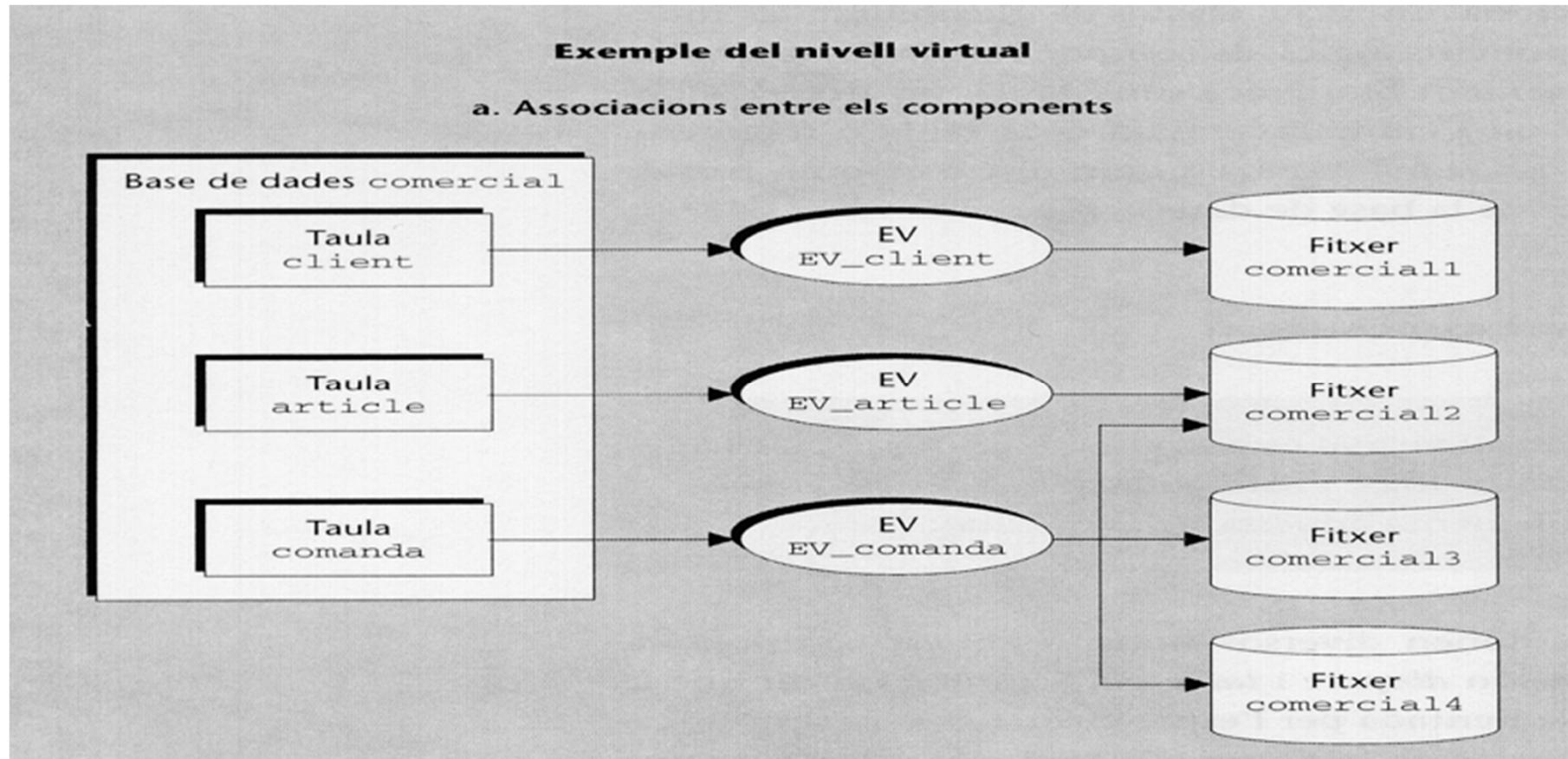
Nivell intermedi que permet relacionar components lògics amb físics; proporciona independència entre nivells. Flexibilitat

S'anomena Espai Virtual (EV) el component que implementa aquesta funcionalitat

- S'anomenen com Tablespace en la majoria de sistemes comercials, encara que hi pot haver sistemes on se'ls anomeni diferent.
- Normalment,



El nivell virtual: Exemple



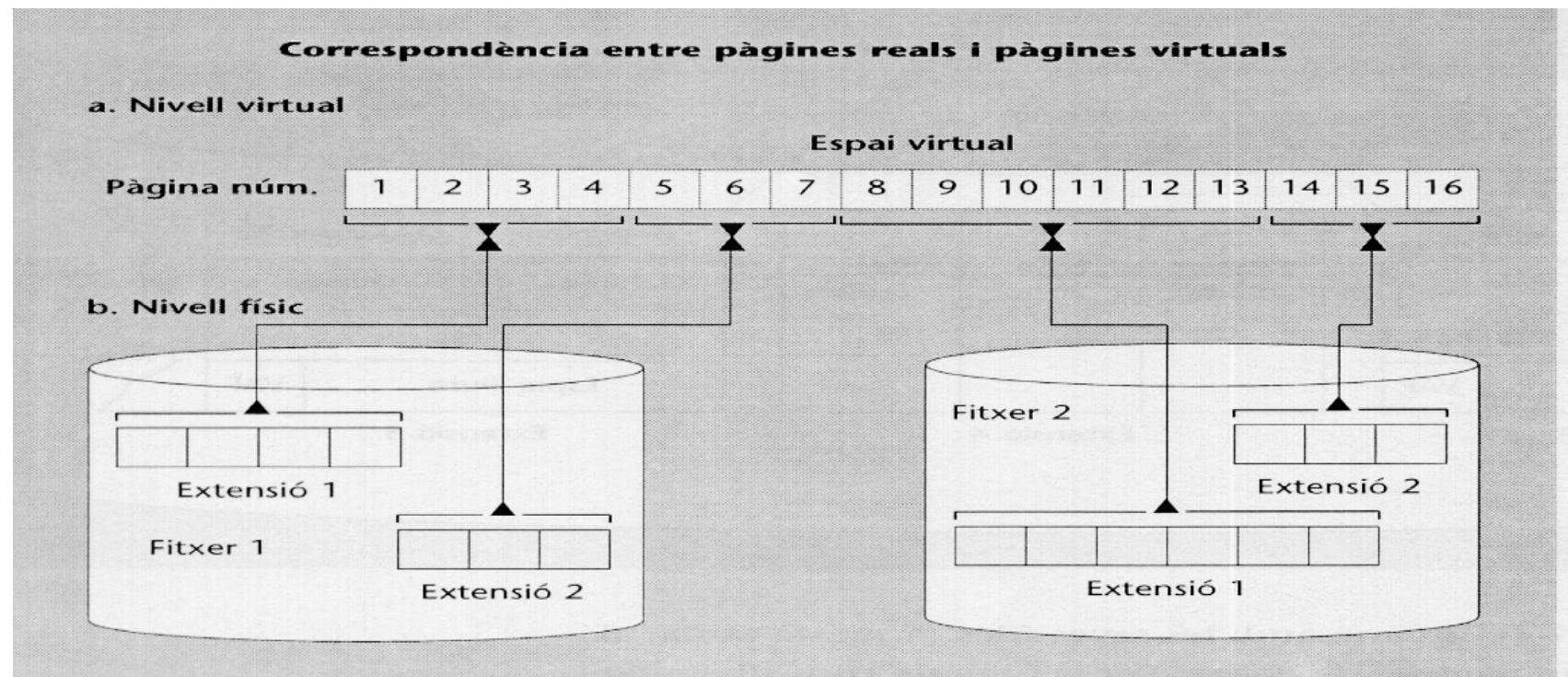
Exemple de sentències per crear l'espai virtual EV_client en PostgreSQL

```
CREATE TABLESPACE EV_client LOCATION 'c:/comercial';
```

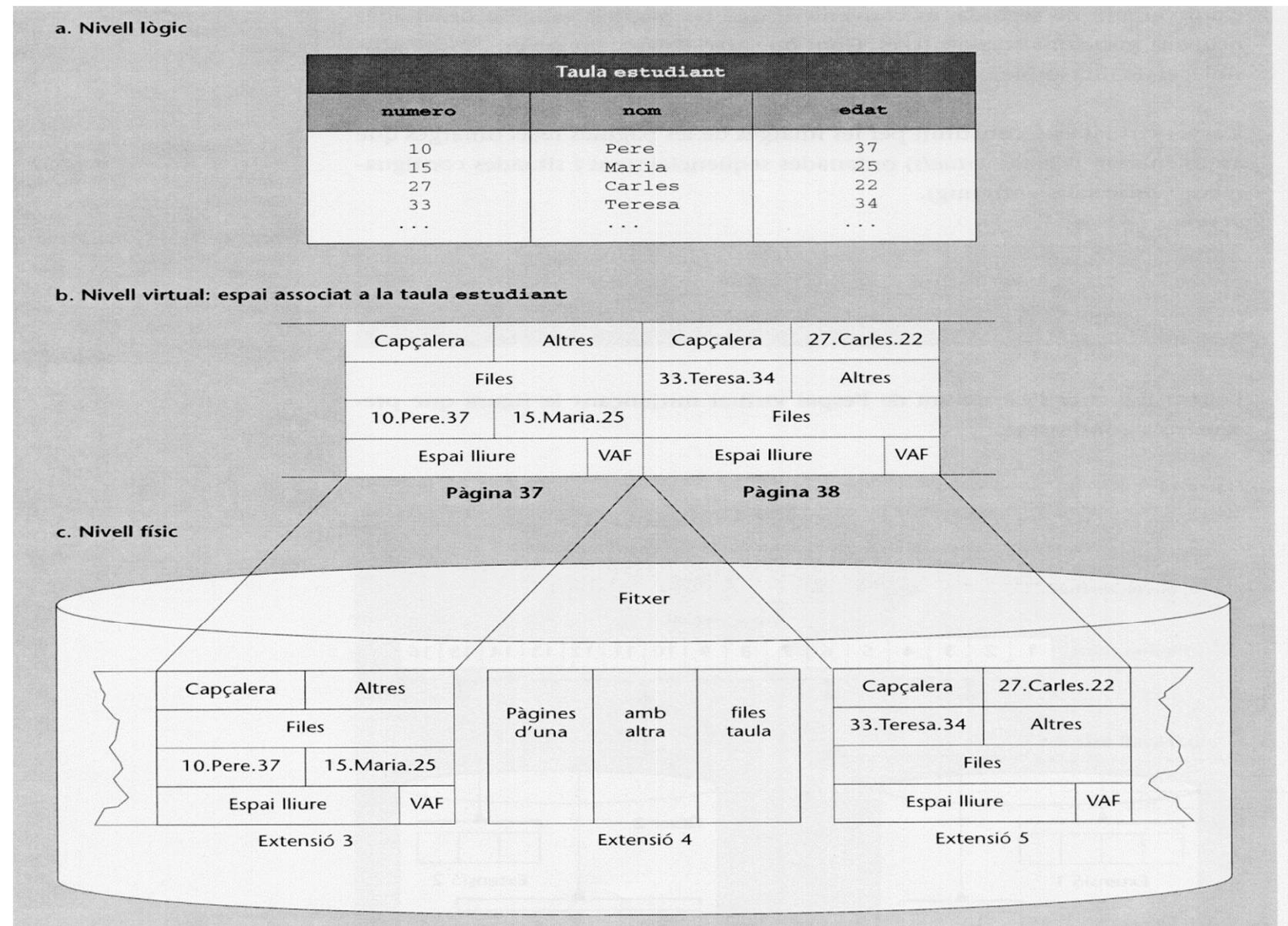
```
CREATE TABLE client (codiClient integer, ...) IN EV_client;
```

Estructura de l'espai virtual

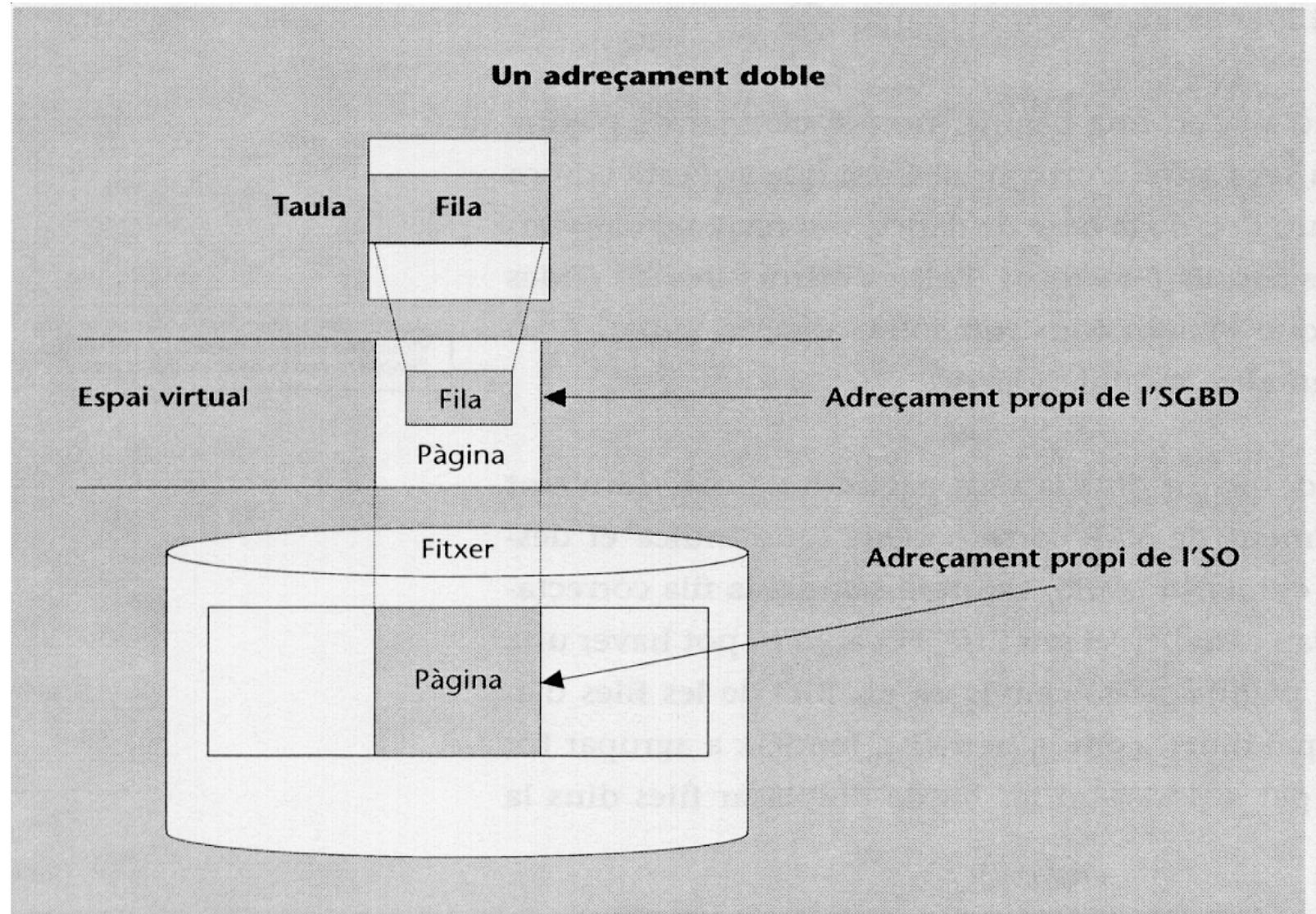
- Espai virtual: visió diferent de les pàgines físiques, sense duplicar
 - Paral·lelisme: Vistes(taules) Memòria virtual(real)
- Espai virtual: Seqüència de pàgines virtuals que es relacionen una a una amb les reals del nivell físic



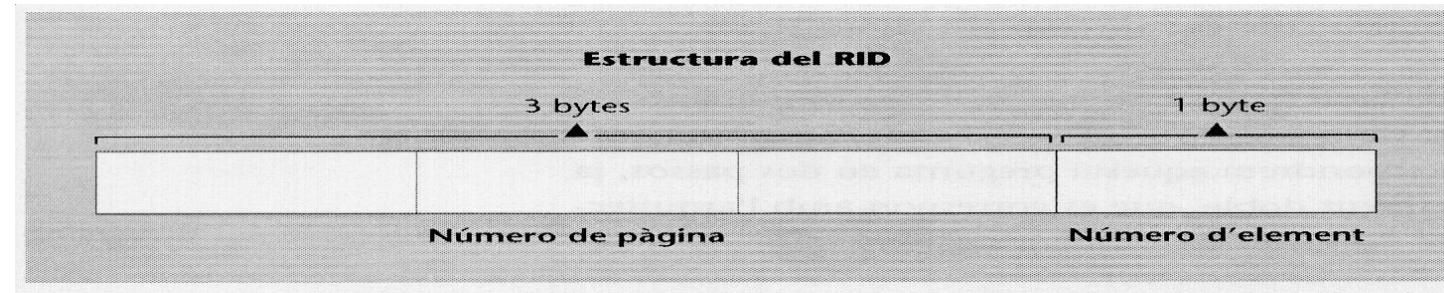
Estructura de l'espai virtual: Exemple dels tres nivells



Adreçament en un SGBD



Record IDentifier (RID)



- Una fila està clavada (pinned) a les pàgines, fins i tot, si creix i no hi cap !!
- Una fila es pot moure dintre la pàgina
- Això ens dóna independència del dispositiu concret

Tipus d'espais virtuals

■ Espai de taules

- Per a taules no molt grans, ni lligades a altres taules
- Seleccions eficients, però joins no

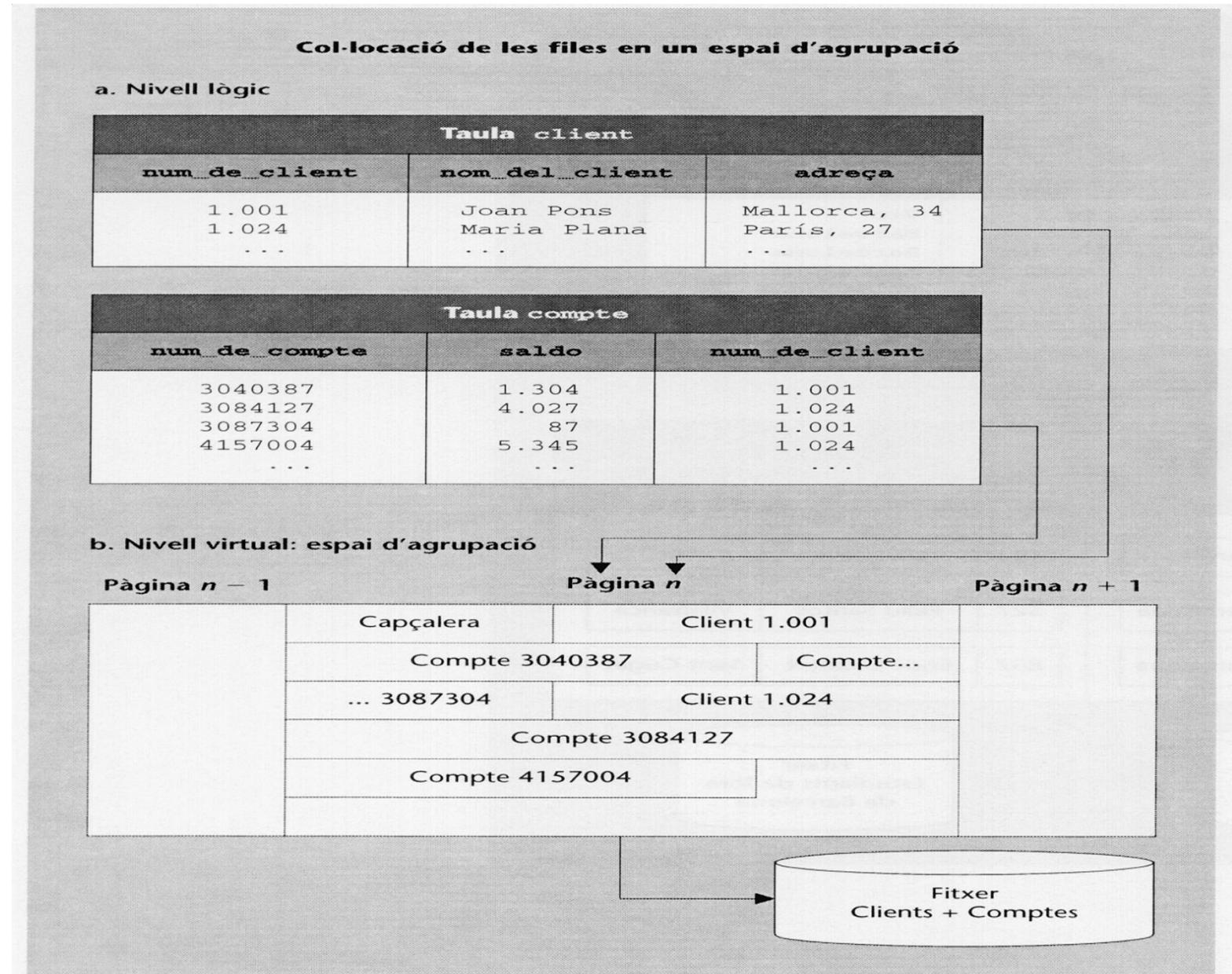


■ Espai d'agrupació

- Per a taules molt relacionades on l'accés quasi sempre és conjunt
- Join eficient, però la selecció parcial no



Tipus d'espais virtuals: Exemple d'espai virtual d'agrupació



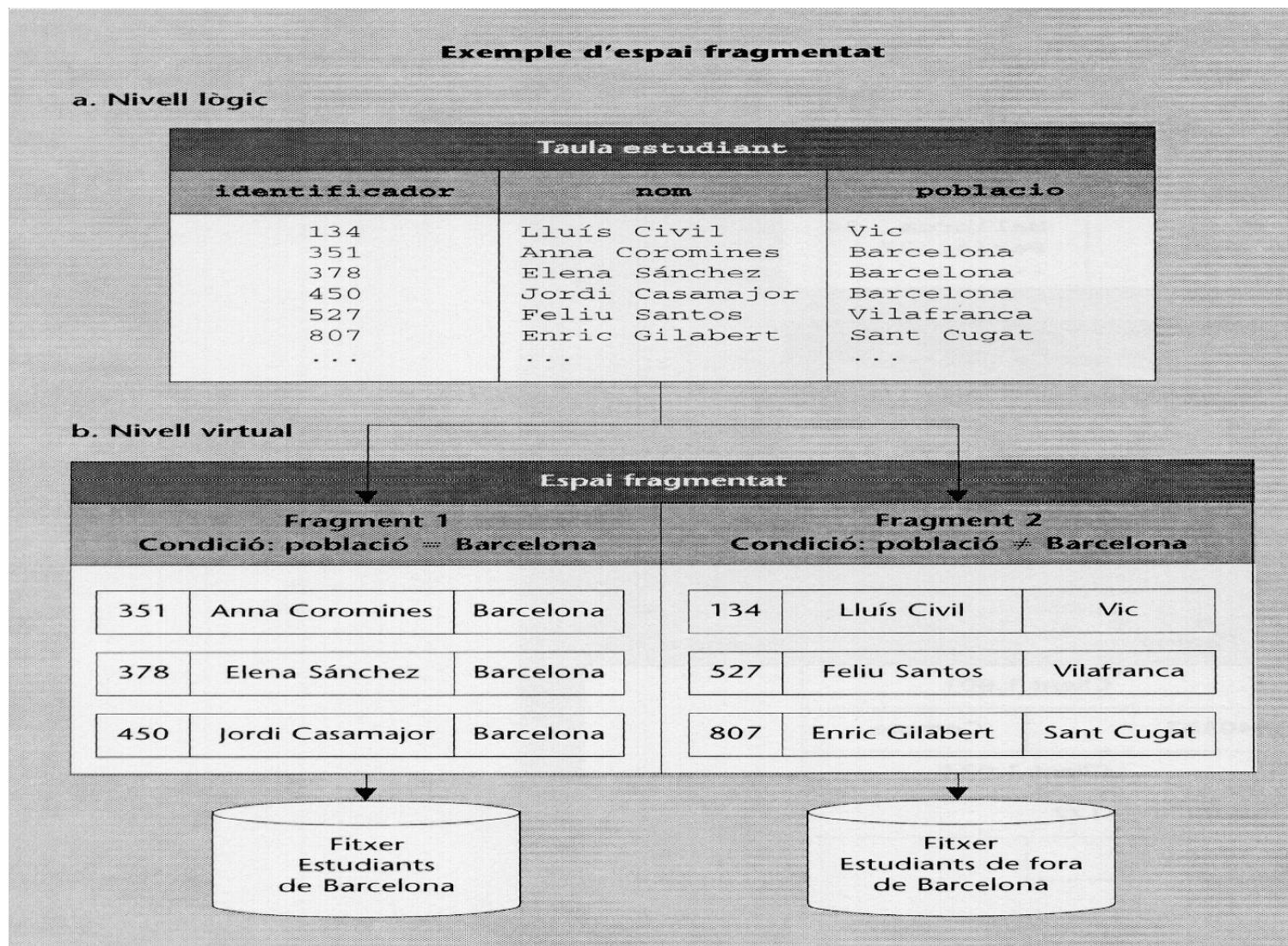
Tipus d'espais virtuals

■ Espai fragmentat

- Per a taules molt grans
 - » Ex: Info. Trucades telefòniques 2 mesos
 - » 10 milion abonats 100 trucades 2 mesos = 1000 millions files
 - » 1000 millions 50 bytes trucada = 50 GB
- La taula s'associa a un espai virtual d'aquest tipus i cal establir el criteri de fragmentació. Exemples de possibles criteris:
 - Segons el valor d'un camp. Ex: `id_client < 20.000`
 - Aleatoriament
- Es pot associar cada fragment a un fitxer diferent per optimitzar l'accés:



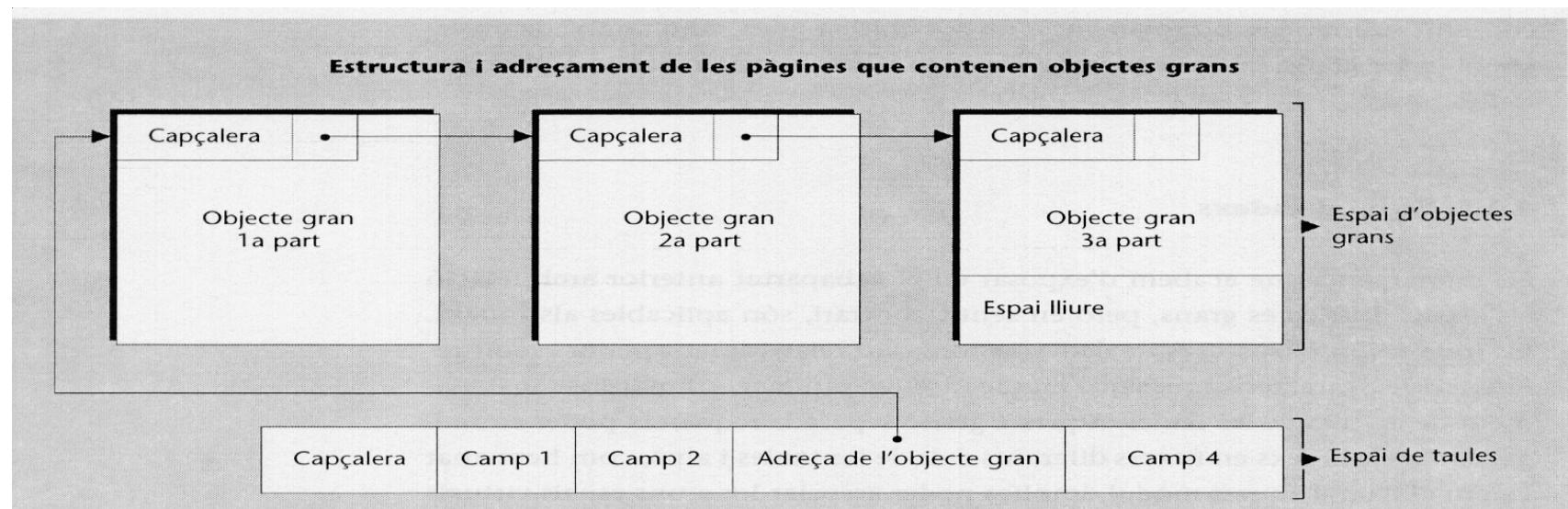
Tipus d'espais virtuals: Exemple espai virtual fragmentat



Tipus d'espais virtuals

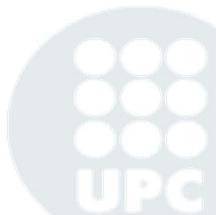
■ Espai d'objectes grans

- Els objectes grans s'emmagatzemen separadament
 - L'accés als atributs amb tipus tradicionals és més freqüent
 - Perjudicaria que estiguessin guardats junts
- La mida de les pàgines que emmagatzemen objectes grans no ha de ser la mateixa que la de les pàgines que emmagatzemen les files de la taula.
- En una pàgina, no hi ha dos objectes grans i no hi ha vaf



Altres aspectes

- Catàleg
 - Normalment, és un EV de taules
- Taules temporals
 - Select into, group by, funcions d'agregació i unique
 - Normalment, és un EV de taules
- Dietari (log)
 - Fitxer(s) que guarden les còpies dels canvis efectuats a la BD a efectes de restauració
 - Normalment, és un EV de taules
- L'ABD ha de calcular l'espai necessari per emmagatzemar la BD
 - Evitar un càlcul erroni de files*bytes que ocupen
 - A més, cal tenir en compte que també fan falta aquests components:
 - Índexos, Catàleg, Taules temporals, Dietari,...



Implementació de mètodes d'accés

- Introducció
- Objectius
- Mètodes d'accés: per posició
- Implementació dels accessos per posició
- Mètodes d'accés: per valor
- Implementació dels accessos per valor
 - Característiques generals dels índexs
 - Arbre B+
 - Índexs agrupats
- Implementació dels accessos per diversos valors
- Annex: Càlcul de costos pels diferents mètodes d'accés

Implementació de mètodes d'accés: Introducció

- Sempre que es llegeix o s'actualitza alguna dada d'una base de dades es fa mitjançant algun dels mètodes d'accés disponibles en un SGBD.
- Els SGBD estructuren les seves dades en pàgines dels espais virtuals i aquests físicament s'emmagatzemen en pàgines físiques dels discos magnètics. De manera que una lectura o actualització a nivell lògic implica:
 - un conjunt de lectures o actualitzacions de pàgines del nivell virtual.
 - un conjunt de lectures o actualitzacions de pàgines del nivell físic.
- **Estudiarem únicament el cas d'accisos a dades emmagatzemades en una única taula situada en un únic espai virtual**, encara que el que s'estudia és directament aplicable a accessos a diverses taules si les tenim en un únic espai d'agrupació.
- Implementar un mètode d'accés d'una o altra manera pot comportar un cost més alt o més baix. **Establirem com a convenció per a l'estimació del costos:**
 - Considerem només el cost d'entrades/sortides, no de CPU
 - Simplificació: **Compten només el nombre de pàgines que es llegeixen o escriuen**



Implementació als mètodes d'accés: Objectius

■ Objectius Generals

- Conèixer els diferents mètodes d'accés que són necessaris per poder fer consultes i actualitzacions a les dades emmagatzemades a les BD.

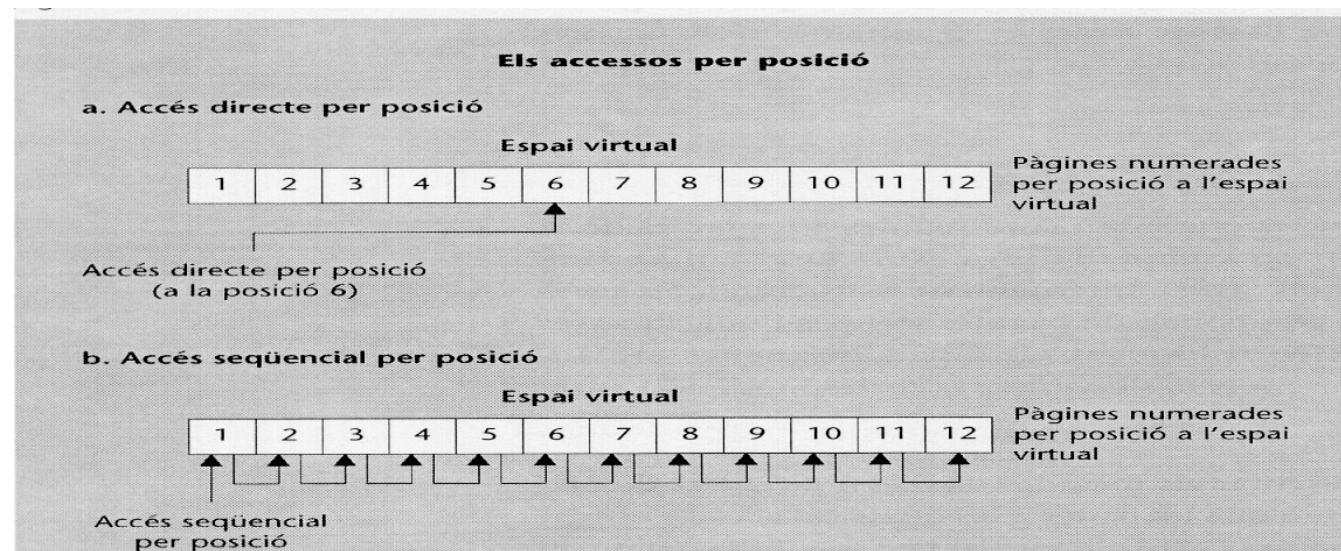
■ Objectius Concrets

- Comprendre la utilitat dels índexs per a la implementació dels accessos per valor.
- Entendre la importància de la reducció del nombre d'E/S (accessos a disc) en les implementacions.
- Entendre els avantatges i inconvenients dels índex arbres B⁺ i dels agrupats per fer accessos per valor.



Mètodes d'accés: Accés per posició

- **Directe:** Obtenir una pàgina que té un número de pàgina determinat.
- **Seqüencial:** Obtenir les pàgines d'un espai seguint l'ordre definit pels seus números de pàgina.



■ Suficient per casos simples:

Empleats(nemp, nom, despatx, sou)

Directe: Insert into Empleats values (25, 'Joan', 150, 2000)

Seqüencial: Select * from Empleats

Implementació dels accessos per posició

- Els SGBD es basen en el sistema operatiu. Les rutines d'E/S del sistema operatiu permeten obtenir una pàgina física donada la seva adreça, i també permeten enregistrar una pàgina física concreta.
- Accés directe per posició
 - L'SGBD transforma els números de posició de les pàgines virtuals en adreces físiques
 - » Cost : 1 pàgina
- Accés seqüencial per posició
 - Similar
 - » Cost : N pàgines



Mètodes d'accés: Accés per valor

- **Directe:** Obtenir totes les files que contenen un valor determinat d'un atribut
- **Seqüencial:** Obtenir totes les files per l'ordre dels valors d'un atribut

■ Exemples

- Directe:
 - Select * from Empleats where despatx=150
 - Update Empleats set sou=2500 where despatx=200
 - Delete from Empleats where despatx=150

- Seqüencial
 - Select * from Empleats order by despatx
 - Select * from Empleats where despatx>100 and despatx<300
 - Update Empleats set sou=2500
where despatx >100 and despatx<300
 - Delete from Empleats where despatx>100 and despatx<300

Una manera efectiva
de resoldre les
consultes és obtenir
els valors per ordre;
no és la única



Mètodes d'accés: Accés per diversos valors

- Accedir a les files per valors de diversos atributs
- Es poden fer accessos directes o seqüencials

- Exemples
 - Select * from Empleats order by despatx, sou
Directe | seqüencial
 - Select * from Empleats where despatx=150 and sou=2000
Directe | seqüencial
 - Delete from Empleats where despatx>100 and sou>1800
Directe | seqüencial
 - Select * from Empleats where despatx=150 and sou>2000
Directe | seqüencial
- Exemples d'accessos mixtos (seqüencial i directe) per diversos valors
 - Select * from Empleats where sou=2000 order by despatx
 - Delete * from Empleats where despatx>100 and despatx<200 and sou=2000



Implementació dels accessos per valor

- Per implementar de manera eficient l'accés per valor s'usa unes estructures de dades auxiliars anomenades ÍNDEXS
- Per què són necessaris? Cost de les solucions alternatives a l'ús d'índexs:

Select * from Empleats where despatx=150

S e q ü e n c i a	a c c é s s e q ü e n c i a l p e r p o s i c i ó	MIG	MAX
T a u l a	I d e m	N / 2	N
T a u l a o r d . físicament	C e r c a d i c t ó m i c a A c c é s directe p o s i c i ó	$\log_2 N$	$\log_2 N + 1$
	I n t e r p o l a c i ó	$\log_2 \log_2 N$	$\rightarrow N$
T a u l a o r d . lò g i c a m e n t	C a d e n a c u r t a i ll a r g a	\sqrt{N}	$2\sqrt{N}$

Els costos d'ambdues consultes pel cas de 300000, i 23M d'empleats seria:

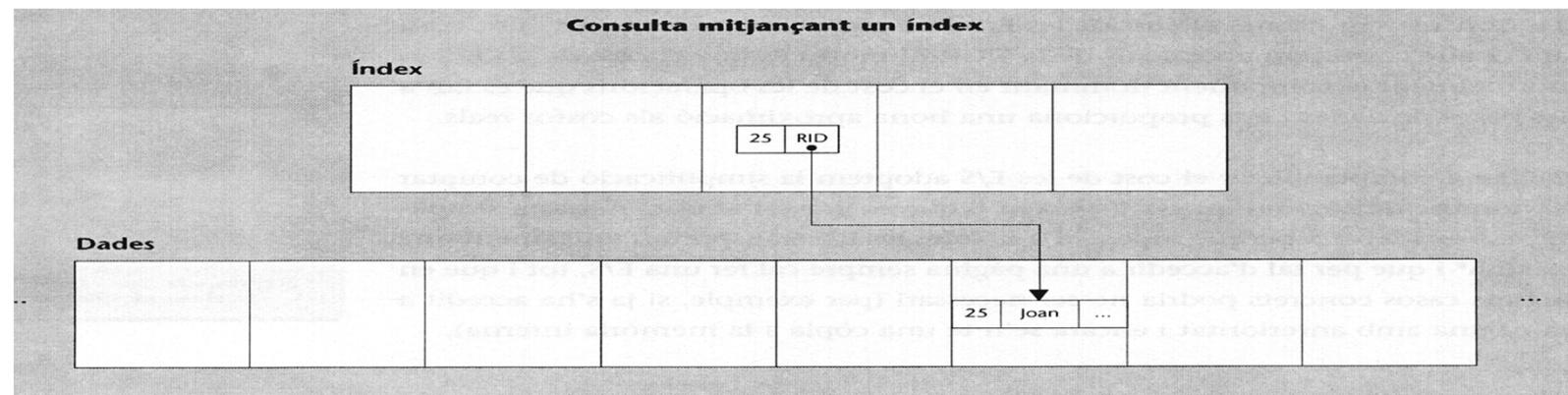
- 300.000 empleats (registres), 10 registres/pàgina, N=30.000 pàgines
Costos: 15.000; 15.000; **15; 4; 174** ordre físic !!
- 23.000.000 empleats (registres), 10 registres/pàgina, N=2.300.000 pàgines
Costos: 1.150.000; 1.150.000; **22; 5; 1516** ordre físic !!

Select * from Empleats where sou < 200000



Característiques generals dels índexs

- Utilitat semblant als índexs de llibres
- Les cerques poden ser més ràpides que si es fan sobre les dades ja que els índexs ocupen molt menys espai que les dades
 - Els índexs contenen parelles (valor, RID) → **entrades**

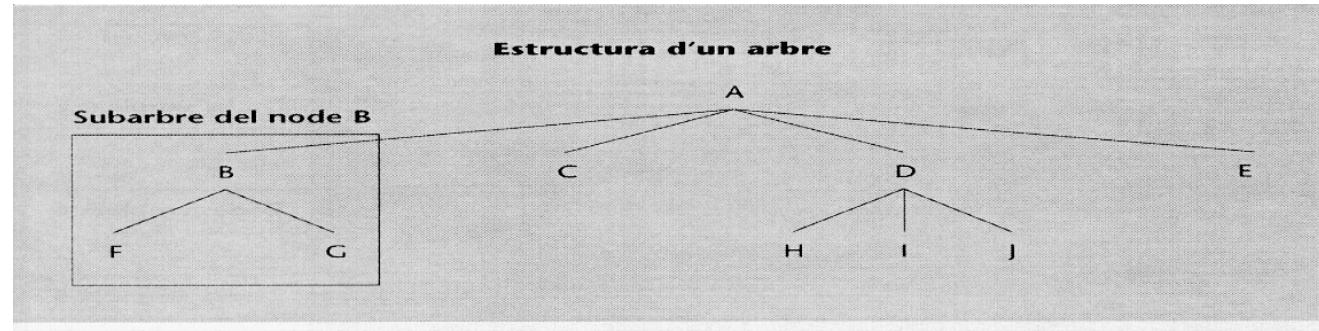


- Fan accés per valor via accés per posició!!!
- Diverses maneres d'estructurar els índexs:
 - Arbre B⁺ (accés per valor i seqüencial per valor)
 - Altre tipus que no veurem (Funcions de dispersió, Bounded disorder files, R-tree,...)
- Una característica molt útil és usar múltiples índexs sobre una relació

Arbres B+ Introducció

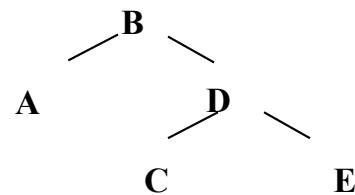
- Arbres:

- Nodes. Arrel. Fulles. Nodes interns. Subarbre.
- Nivell de l'arrel = 1; Nivell d'un node = nivell del pare + 1



- Un arbre B+: És un tipus particular d'arbre de cerca

- Objectiu d'aquest tipus d'arbre: Minimitzar les E/S
- Cost arbre B+:
 - Accedir a 1 registre d'entre 300.000 amb 3 accessos a pàgines
i a 1 registre d'entre 23.000.000 amb 4 accessos No ordre físic!
 - Si es compara amb el cost en el cas d'un arbre binari (arbre orientat normalment a fer cerques a memòria interna):



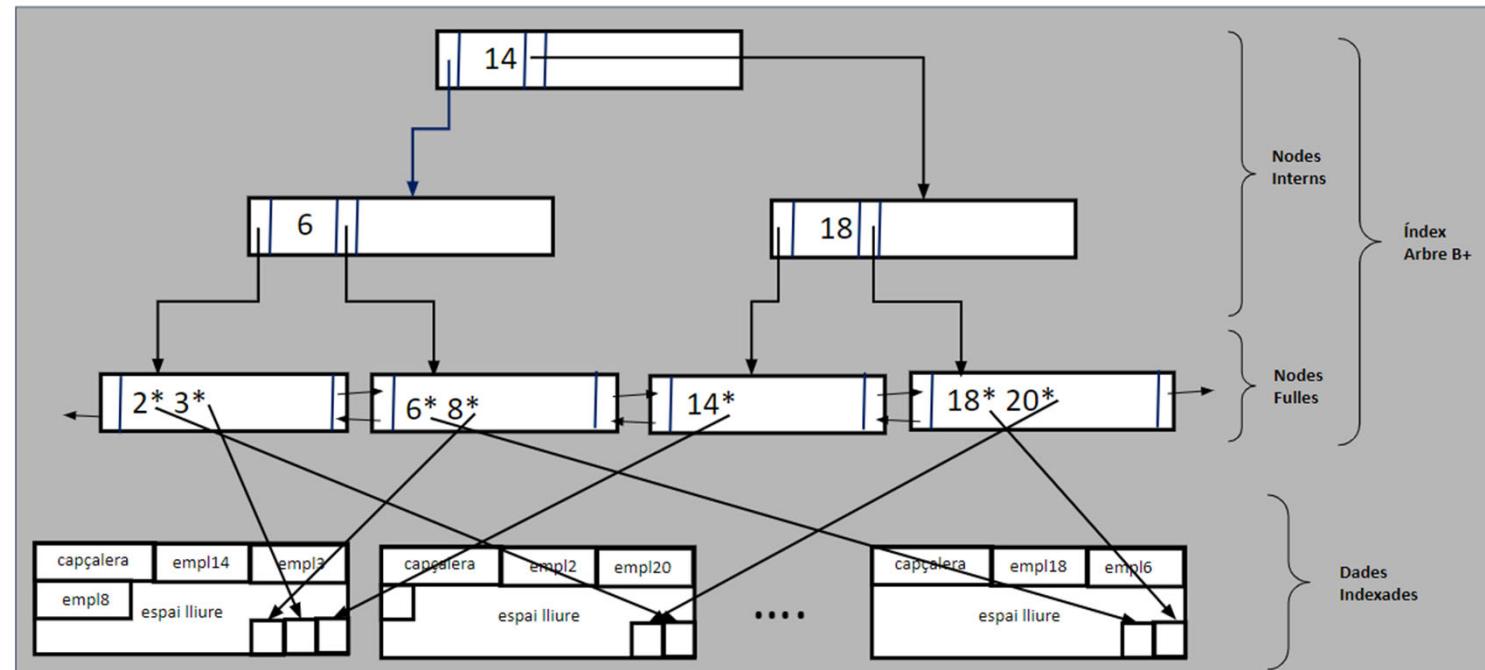
$$\text{Cost} = (\log_2 \text{número de registres})$$

300.000	registres	18 accessos
23 Milions	registres	24 accessos

No ordre físic!

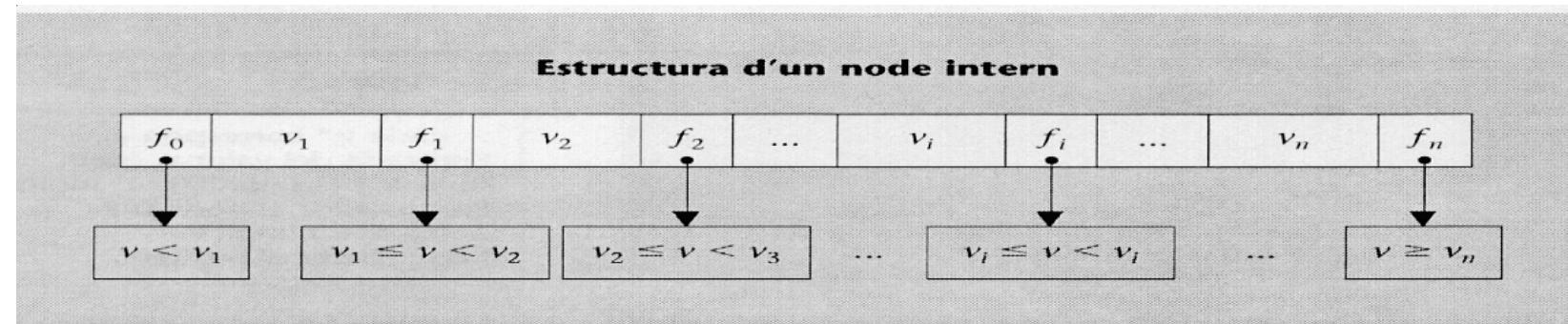
Arbres B⁺ Estructura

- Tot arbre té un número d'ordre d que indica la capacitat dels nodes.
 - Arbre B⁺ d'ordre d , els nodes contenen com a **màxim $2d$** valors
- Els nodes interns i els nodes fulla tenen estructures diferents:
 - Els nodes fulla són els que contenen totes les entrades del índexs (valor i RID)
 - Els nodes interns tenen com a objectiu dirigir la cerca !
 - No tenen entrades, només valor i apuntadors a altres nodes
 - Els nodes fulla estan connectats per apuntadors dobles



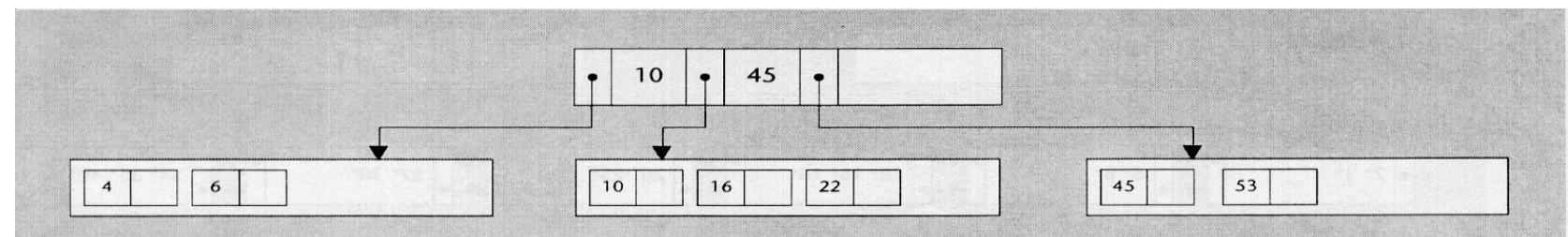
Arbres B⁺ Estructura node intern

- Un node intern conté valors i apuntadors cap als seus nodes fills
- L'estructura d'un node intern que conté n valors, amb $n \leq 2d$:



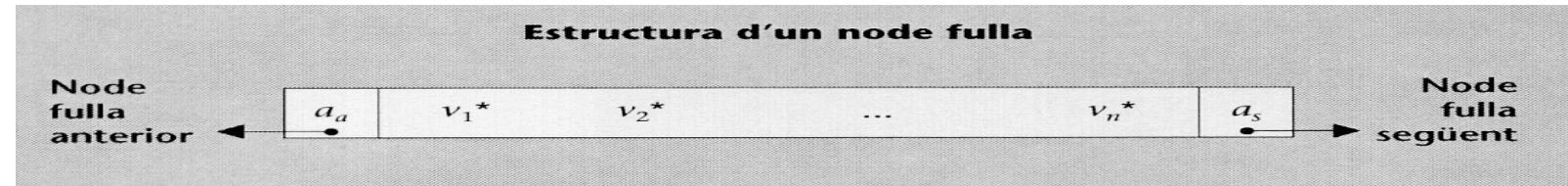
En el cas màxim: $2d$ valors i $2d+1$ apuntadors

- Exemple d'ordre 2:



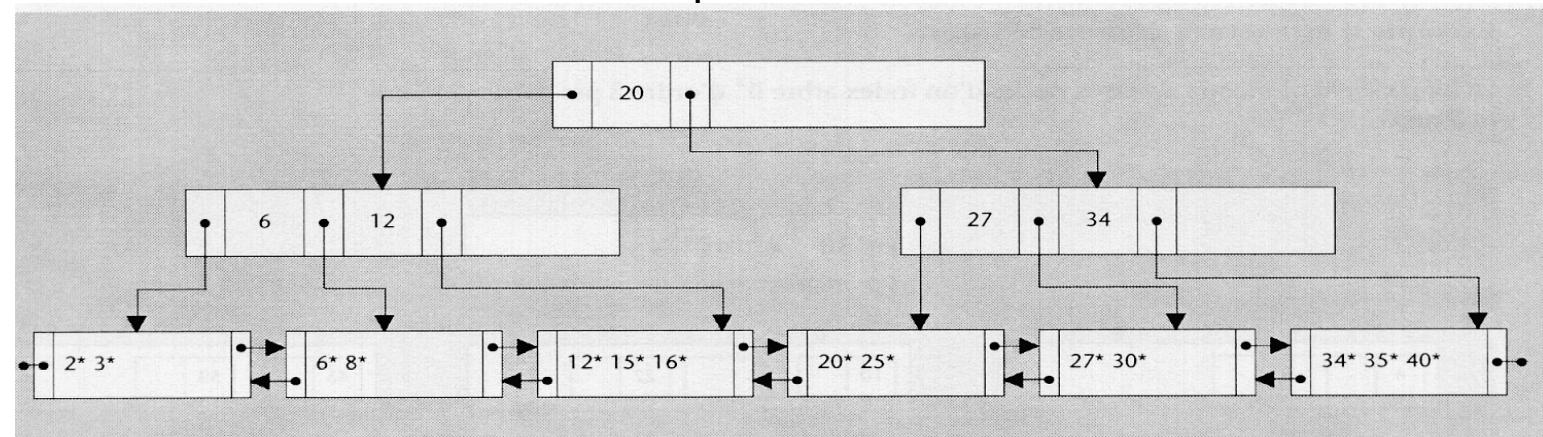
Arbres B+ Estructura node fulla

- Els nodes fulla contenen:
 - Entrades formades per [valor, RID] que escriurem v^*
 - Apuntadors a la fulla anterior i següent



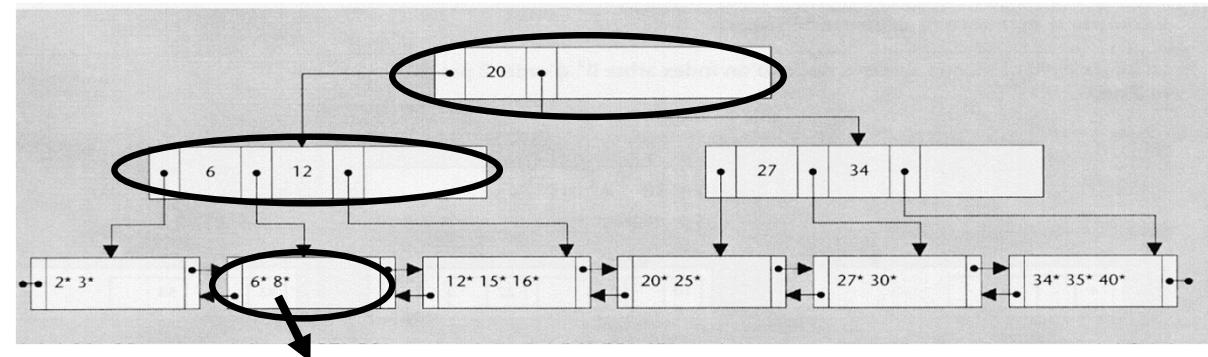
- Cas màxim: $2d$ (valors+rid) + 2 apuntadors a fulla

- Condicions addicionals:
 - Tots els valors d'un node fulla són més petits que els valors de les fulles següents
 - Tots els valors d'un node intern estan repetits a les fulles. Les fulles tenen tots els valors.
 - Els nodes interns no tenen valors repetits.

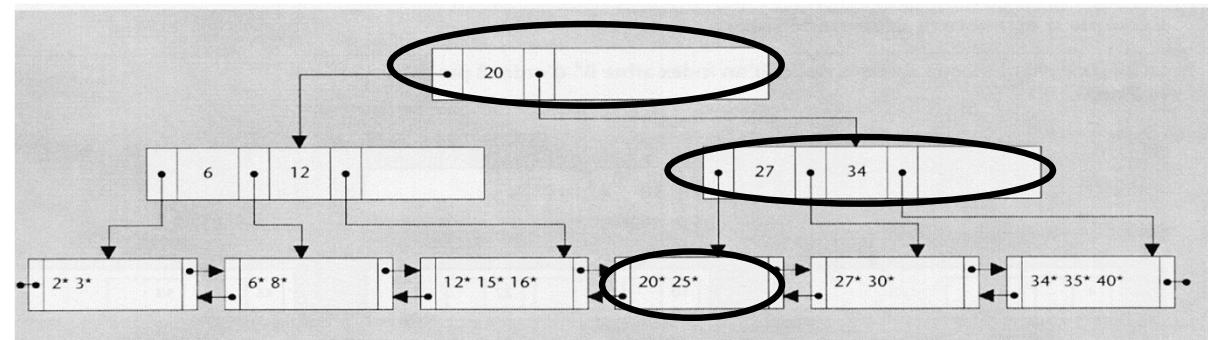


Arbres B⁺ Accés directe per valor

- Per accedir al registre amb valor v cal
 - Localitzar la fulla que té l'entrada v
 - Usar el RID de l'entrada per trobar la fila cercada
- Exemples: Localitzar el registre amb el valor 8



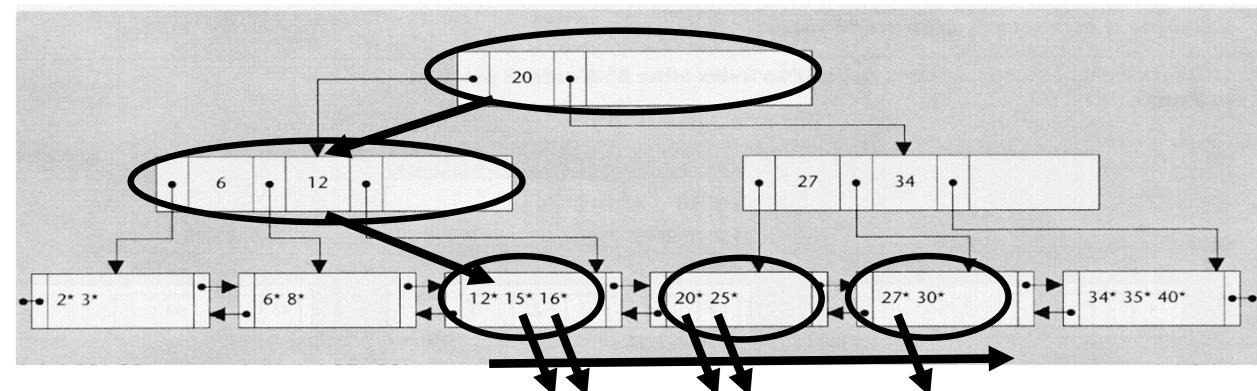
Localitzar el registre amb el valor 26



- Lectures: El nombre de nodes als que cal accedir és igual al nivell de la fulla on hauria d'estar el valor buscat. A part, també cal accedir a la pàgina de dades que conté la fila buscada, si és que s'ha trobat el valor a l'índex.

Arbres B⁺ Accés seqüencial per valor

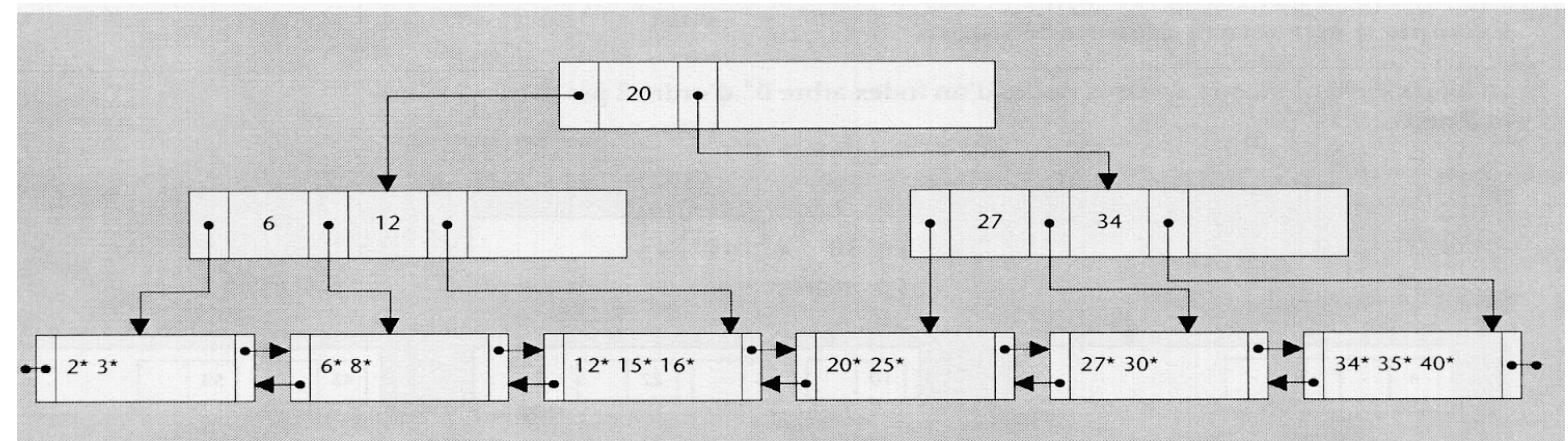
- Per fer un accés seqüencial per valor total cal:
 - Localitzar la fulla de més a l'esquerra
 - Recuperar totes les seves entrades
 - Per cada entrada, usar el RID per trobar la fila cercada
 - Localitzar la fulla següent
- Per tal de fer cerques parcials: per exemple, accedir als registres amb valor més gran o igual **v1** (de manera similar en cerques tal que: $v_1 < \text{valor} < v_2$) cal:
 - Localitzar la fulla que té (hauria de tenir) l'entrada v
 - Recuperar totes les seves entrades que compleixin la condició
 - Per cada entrada, usar el RID per trobar la fila cercada
 - Localitzar la fulla següent
- Exemple: Localitzar els registres amb els valors entre el 15 i el 28



- Lectures: El nombre de nodes als que cal accedir és igual al nivell de la fulla on hauria d'estar el valor inicial buscat, més els nodes fulla consecutius on hi ha valors dins del rang buscat. A part, també cal accedir a les pàgines de dades que contenen les files buscades.

Arbres B⁺ Millors de rendiment

- Per localitzar una entrada en una fulla d'un arbre, el nombre de nodes a recórrer és igual al nivell de la fulla!!!
 - Si reduïm el nivell de les fulles ...
- **Propietat 1:** Tots els nodes de l'arbre, excepte l'arrel (per què?), han d'estar plens com a mínim al 50%.
 - En un arbre d'ordre $d \rightarrow$ almenys d valors per node
 - El arbre següent d'ordre 2 compleix aquesta propietat



- Que passaria en l'extrem si l'arbre fos binari?
- **Propietat 2: Equilibrats.** Totes les fulles al mateix nivell.
 - El nombre de nodes a recórrer és sempre el mateix

Arbres B⁺ Emmagatzematge

- Espai d'índexs
- La mida dels nodes depèn de l'ordre i de la mida dels valors i apuntadors
 - Nodes molts grans → un arbre tingui pocs nivells
 - Nodes molts grans → potser més d'una E/S

La mida habitual d'un node coincideix amb la mida de les pàgines.
Cada node s'emmagatzema en una pàgina.
Per tant, en un arbre de **3 nivells, calen 3 E/S** per localitzar la fulla
- En cas que l'arrel estigui a memòria → 1 accés menys
- Càcul de l'ordre **d** òptim
 - Suposant pàgines de 4Kb (4096b) → ordres **d** entre 50 i 100
 - Nodes interns: $2d * (\text{mida valor}) + (2d + 1) * \text{mida apuntador fulla} \leq 4096b$
» $2d * (20) + (2d + 1) * 3 \leq 4096 ; d \leq 88$
 - Nodes fulla: $2d * (\text{mida valor} + \text{mida RID}) + 2 * \text{mida apuntador fulla} \leq 4096b$
» $2d * (20 + 4) + 6 \leq 4096 ; d \leq 85$
 - Ordre òptim. Simplificació: Escollim $d = 85$

Arbres B⁺ Exemple

- Càlcul del nombre **màxim** de files que es pot indexar:
 - Suposem que $d = 50$ i que cada node té una ocupació 69%
 - Nivell 1: 1 node amb 69 valors i 70 apuntadors
(Atenció: s'està suposant el node arrel ple al 69% també)
 - Nivell 2: 70 nodes amb 69 valors i 70 apuntadors cadascun
 - Nivell 3: 4900 nodes amb 69 entrades

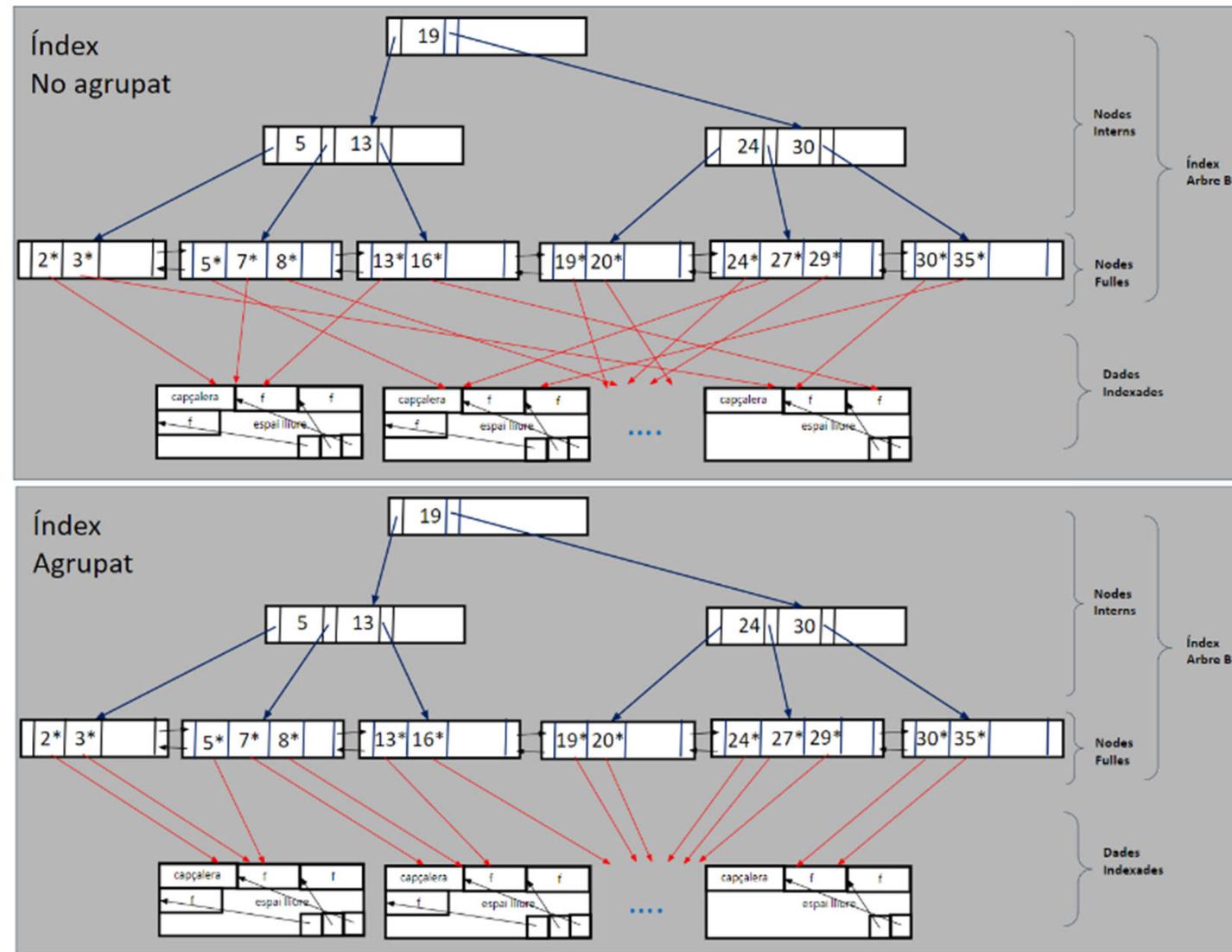
$$4900 * 69 = 338.100 \text{ entrades}$$

Localitzar 1 registre entre 338100 es pot fer amb 3 accessos a pàgines de l'índex
(+1 per accedir al registre –pàgina de dades)
(-1 en cas que l'arrel estigui a memòria)

- I amb 4 nivells i 69% ocupació quantes entrades?
Nivell 4: 343.000 nodes amb 69 entrades = 23667000
- Càlcul de l'alçada aproximada d'un Arbre B⁺
 $\approx \log_{70} 23667000 = \ln 23667000 / \ln 70 = 4;$
 $\approx \log_{70} 338100 = 3$

Índexs agrupats

- Un índex agrupat (*cluster*) és aquell en què les dades que indexa estan ordenades físicament segons l'accés seqüencial per valor que proporciona



Índexs agrupats

- Quants índex agrupats podem tenir sobre una taula?
- Problema: Mantenir l'ordre físic?
 - Cost prohibitiu!
 - Solució pràctica:
 - Deixar % espai lliure a pàgines
 - Si s'omple, a pàgines d'excedents encadenades
 - Si % pàgines excedents elevat, regeneració

Implementació dels accessos per diversos valors: Accessos directes, Estratègia intersecció de RIDs

Empleats(nemp, nom, despatx, sou)

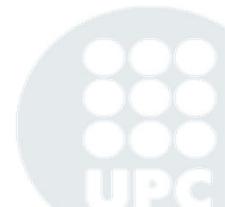
Arbre B⁺ sobre despatx

Arbre B⁺ sobre sou

```
SELECT *
FROM Empleats
WHERE despatx = 150
AND sou = 200000
```

- **Estratègia Intersecció de RIDs:** Us de dos índexs arbres B+
 - Obtenir el conjunt dels RIDs tals que despatx = 150
 - Obtenir el conjunt dels RIDs tals que sou = 200000
 - Intersecció entre els dos conjunts de RIDs, obtenint els RIDs dels empleats que compleixen les dues condicions.

En general aquesta estratègia dóna un bon rendiment, però si hi ha molts RIDs que compleixen una condició (200000) i pocs les dues (150), és ineficient



Implementació dels accessos per diversos valors: Accessos directes, Estratègia índex multi-atribut

Empleats(nemp, nom, despatx, sou)

Índex amb valors compostos
pels atributs [num_despatx, sou].
Els valors de l'índex seran valors
com ara [150, 200000], [150, 300000],
[200, 100000], [200, 150000] ...

```
SELECT *  
FROM Empleats  
WHERE despatx = 150  
AND sou = 200000
```

■ Estratègia Índex (multiatribut)

Aquests índexs tenen la mateixa estructura **que els altres**, però v seran
llistes de elements $[v_1, v_2, \dots]$

La gestió de l'índex fa necessari establir una relació d'ordre lineal entre les
llistes: s'ordenen primer segons el primer element, entre els que tenen
el mateix valor pel primer, s'ordenen segons el segon element, ...



Implementació dels accessos per diversos valors: Accessos seqüencials i mixtos

■ Estratègia Índex multi-atribut

Empleats(nemp, nom, despatx, sou)
Índex multiatribut sobre [despatx, sou]

Els valors de l'índex seran valors
com ara [150, 200000], [150, 300000],
[200, 100000], [200, 150000] ...

```
SELECT *
FROM Empleats
ORDER BY despatx, sou
```

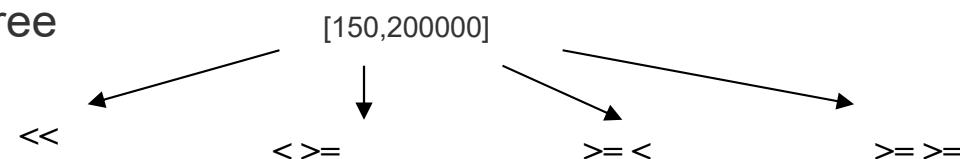
```
SELECT *
FROM Empleats
WHERE despatx = 150
AND sou > 200000
```

```
SELECT *
FROM Empleats
ORDER BY sou, despatx
```

```
SELECT *
FROM Empleats
WHERE despatx > 150
AND sou = 200000
```

■ Estratègia índex multi-atribut multi-dimesional: no ordre lineal!!

Ex: Quad-tree



Els índexs en un SGBD concret

- Per implementar accessos per valor, directes o seqüencials, la majoria de SGBDs proporcionen índexs arbre B+ per un atribut que poden ser agrupats o no
 - Creació d'un índex:
Empleats(nemp, nom, despatx, sou)
CREATE INDEX index_despatx ON Empleats(despatx)
 - Per a que l'índex sigui descendant:
CREATE INDEX index_despatx ON Empleats(despatx DESC)
 - Creació d'un índex agrupat:
CREATE INDEX CLUSTER index_despatx ON Empleats(despatx)
 - Creació d'un índex que no admeti valors repetits:
CREATE UNIQUE INDEX index_despatx ON Empleats(despatx)
 - Creació d'un índex multiatribut:
CREATE INDEX index_despatx_sou ON Empleats(despatx,sou)

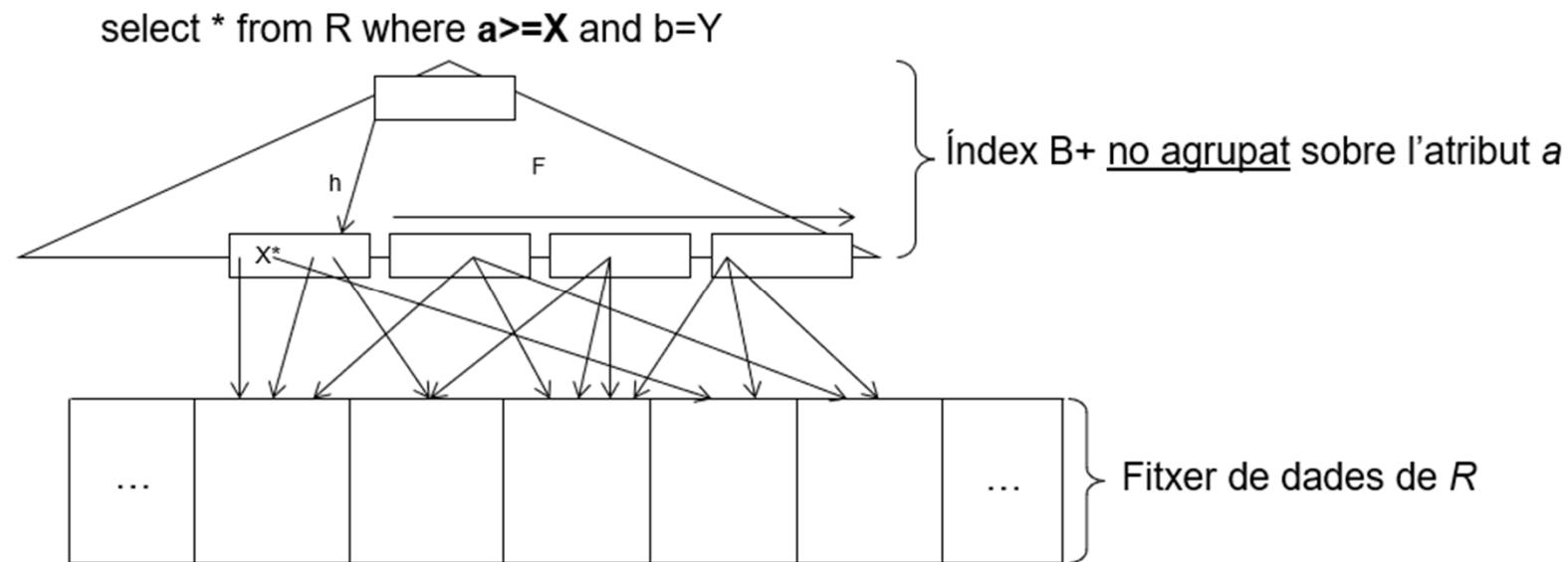


Annex

- Cost accés seqüencial per valor:
 - Índex arbre B+, no agrupat
 - Índex arbre B+, agrupat
- Cost accés directe per valor:
 - Índex arbre B+, agrupat/no agrupat, no valors repetits
 - Índex arbre B+, no agrupat, pot prendre valors repetits
 - Índex arbre B+, agrupat, pot prendre valors repetits
- Cost accés per diversos valors:
 - Alternatives possibles
 - Estratègia intersecció de RIDs

Cost accés seqüencial per valor: Índex arbre B+, no agrupat

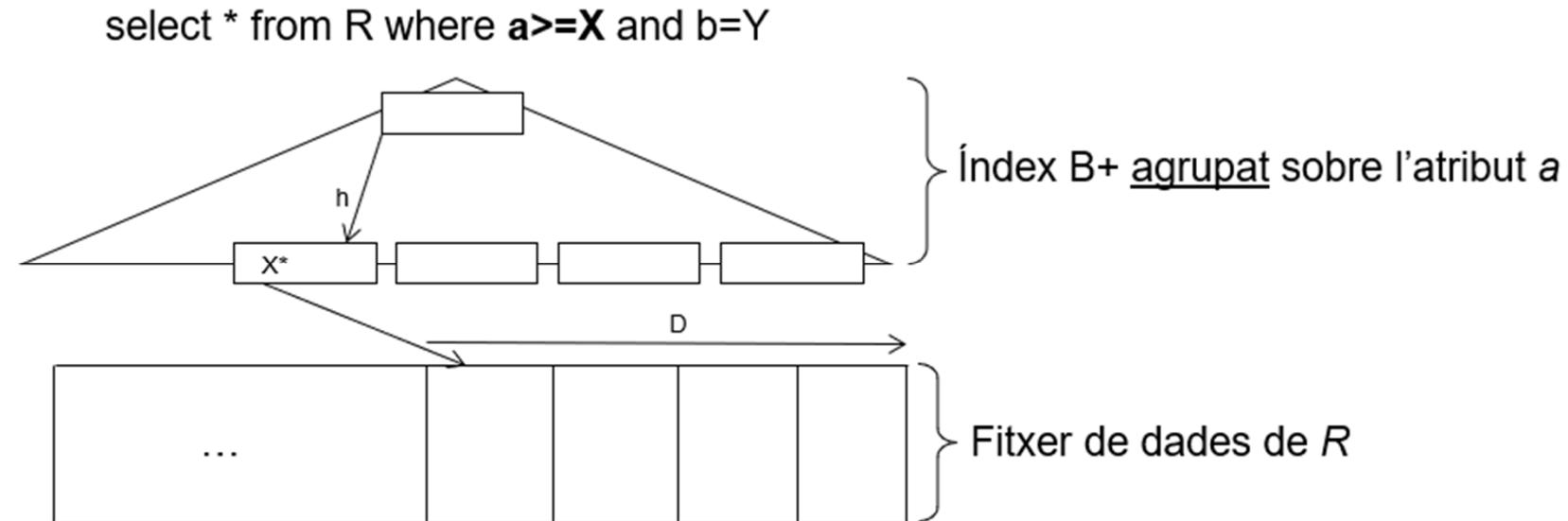
- Índex B+ no agrupat sobre un atribut a d'una taula R , l'atribut a pot prendre (o no) valors repetits



- Cost = Cost accés índex + Cost accés fitxer de dades = $(h + F) + |R(a \geq X)|$
- h alçada índex B+, F nombre de nodes fulla addicionals (a més del primer node fulla) de l'índex B+ que cal recórrer
- $|R(a \geq X)|$ nombre de files de la taula R que verifiquen la condició $a \geq X$ expressada a la consulta
- El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

Cost accés seqüencial per valor: Índex arbre B+, agrupat

- Índex B+ agrupat sobre un atribut a d'una taula R , l'atribut a pot prendre (o no) valors repetits

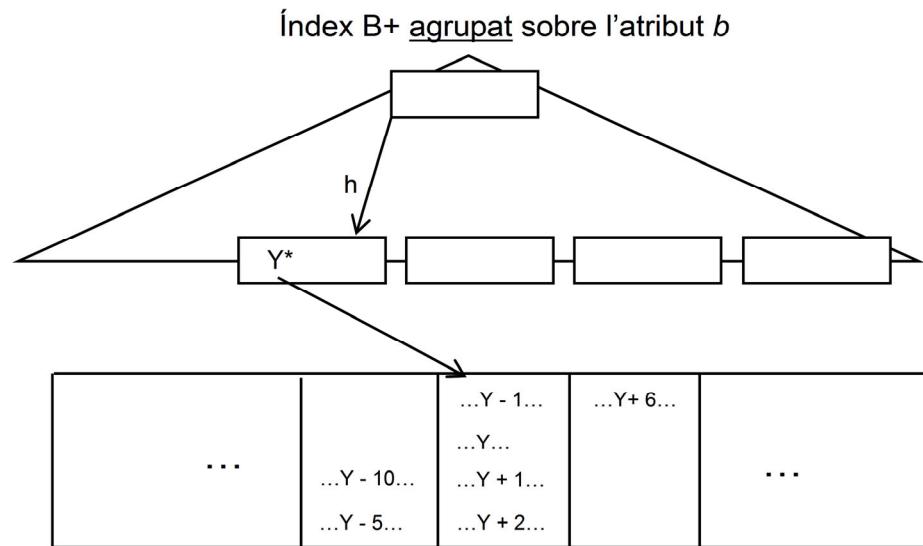


- Cost = Cost accés índex + Cost accés fitxer de dades = $h + D$
- h alçada índex B+, D nombre de pàgines del fitxer de dades que cal recórrer
- $D = \lceil |R(a \geq X)|/f \rceil$
- $|R(a \geq X)|$ nombre de files de la taula R que verifiquen la condició $a \geq X$ expressada a la consulta
- f factor de bloqueig del fitxer de dades (nombre de registres per pàgina)
- El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

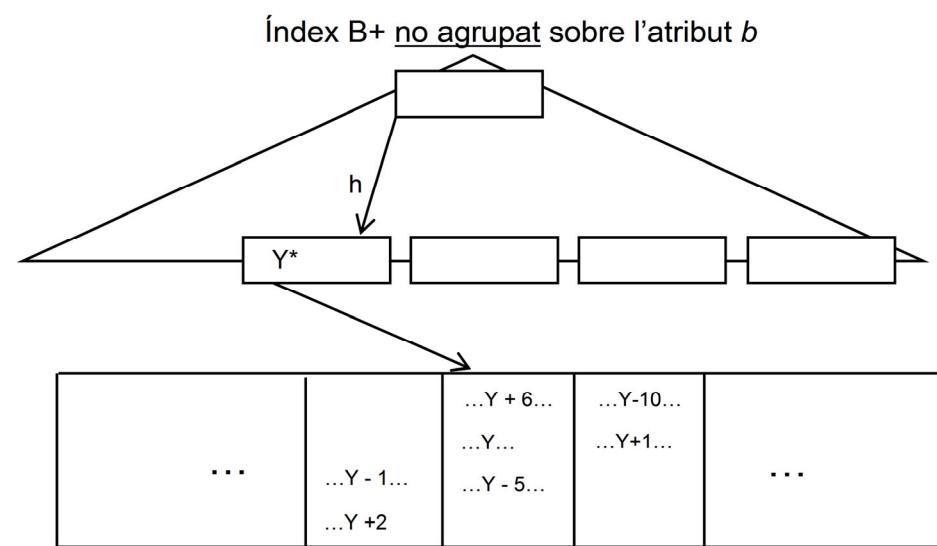
Cost accés directe per valor: Índex arbre B+, no valors repetits

- Índex B+ agrupat/no agrupat sobre un atribut b d'una taula R , l'atribut b no pren valors repetits
- Cost = Cost accés índex + Cost accés fitxer de dades = $h + 1$
- h alçada de l'arbre B+
- El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

select * from R where a>=X and **b=Y**



Fitxer de dades de R
(les dades estan ordenades segons el valor de l'atribut b)

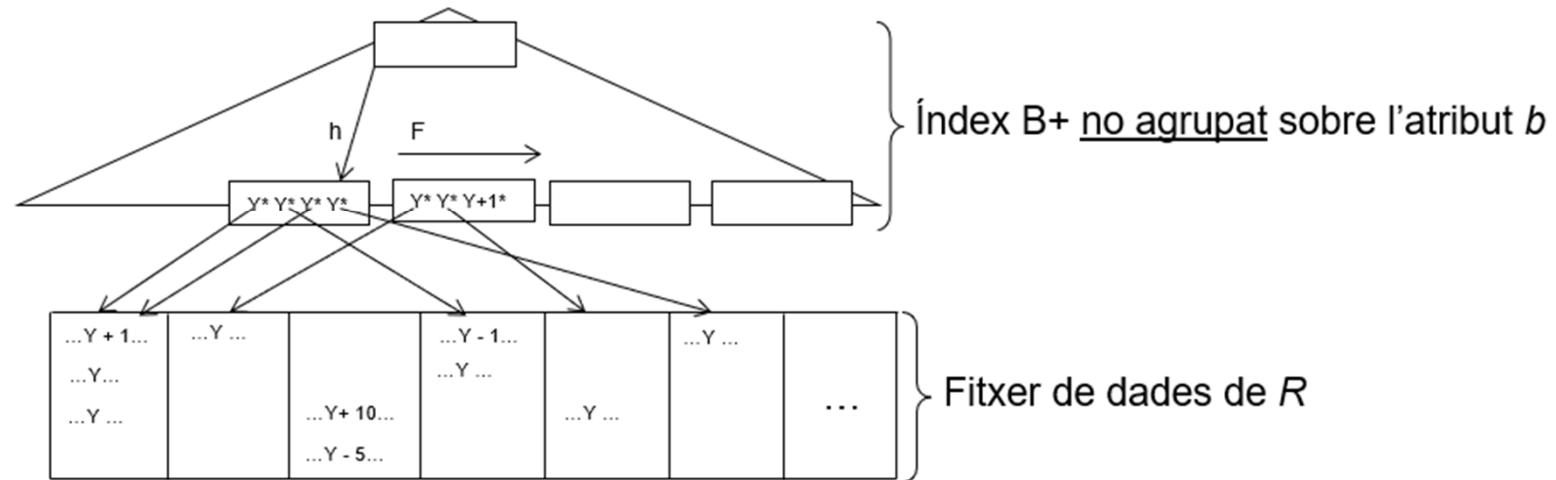


Fitxer de dades de R
(les dades no estan ordenades segons el valor de l'atribut b)

Cost accés directe per valor: Índex arbre B+, no agrupat, pot prendre valors repetits

- Índex B+ no agrupat sobre un atribut b d'una taula R , l'atribut b pot prendre valors repetits

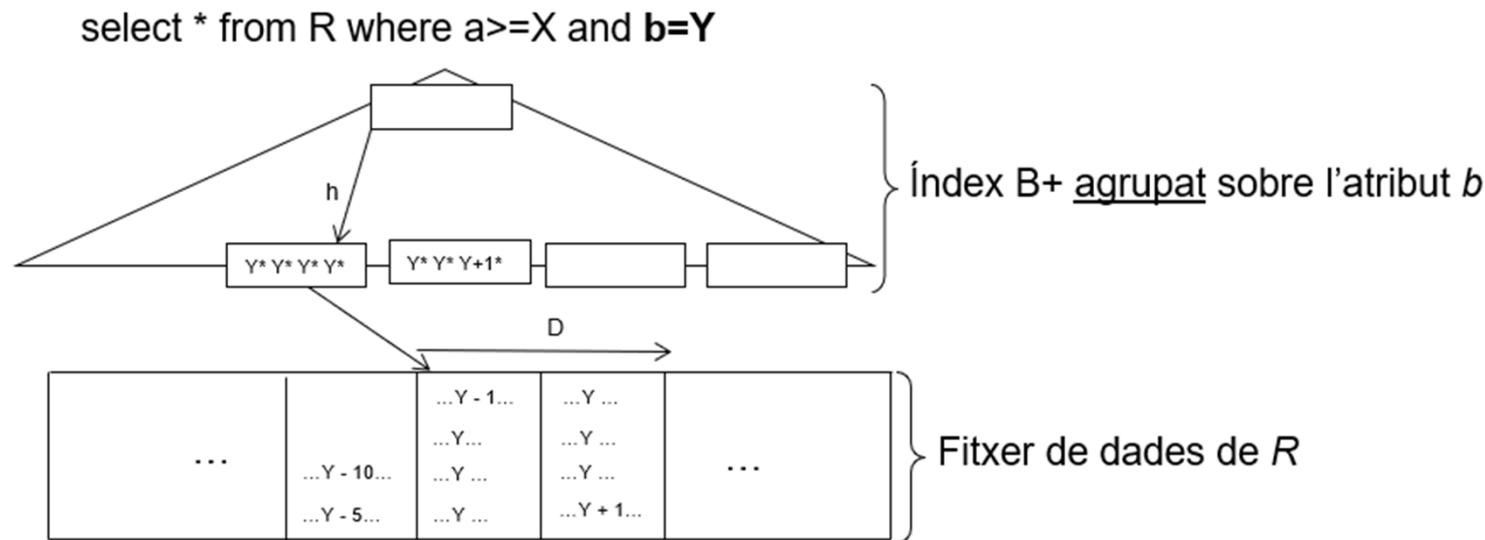
select * from R where a>=X and b=Y



- Cost = Cost accés índex + Cost accés fitxer de dades = $(h + F) + |R(b=Y)|$
- h alçada índex B+, F nombre de nodes fulla addicionals (a més del primer node fulla) de l'índex B+ que cal recórrer
- $|R(b=Y)|$ nombre de files de la taula R que verifiquen la condició $b=Y$ expressada a la consulta
- El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

Cost accés directe per valor: Índex arbre B+, agrupat, pot prendre valors repetits

- Índex B+ agrupat sobre un atribut b d'una taula R , l'atribut b pot prendre valors repetits



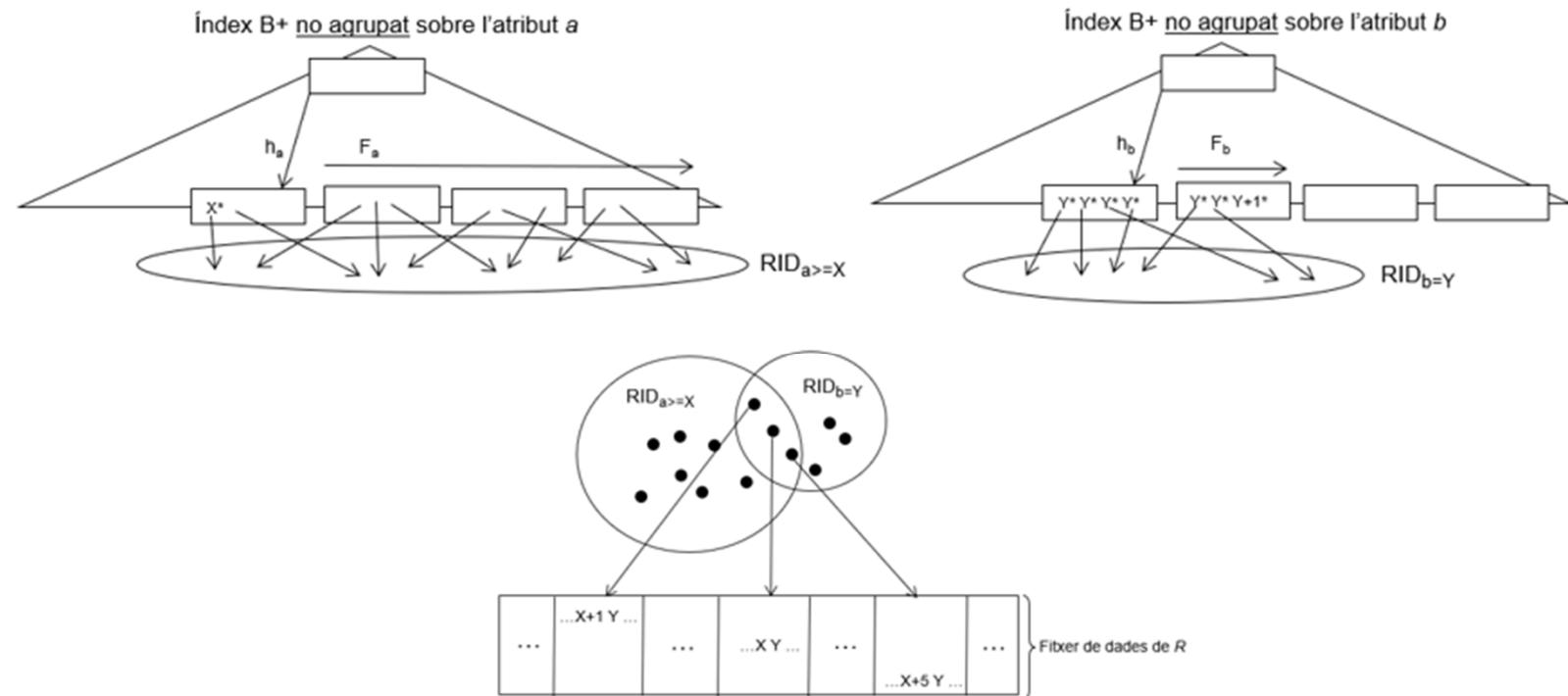
- Cost = Cost accés índex + Cost accés fitxer de dades = $h + D$
- h alçada índex B+, D nombre de pàgines del fitxer de dades que cal recórrer
- $D = \lceil |R(b=Y)|/f \rceil$
- $|R(b=Y)|$ nombre de files de la taula R que verifiquen la condició $b=Y$ expressada a la consulta
- f factor de bloqueig del fitxer de dades (nombre de registres per pàgina)
- El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

Cost accés per diversos valors: Alternatives possibles

- Imaginem que tenim la consulta següent:
`select * from R where a>=X and b=Y`
- Disposem de dos índexs B+ no agrupats, un sobre a i un altre sobre b , b pot prendre valors repetits
- Per resoldre la consulta, usant els índexs, tenim diverses alternatives:
 - Utilitzar només l'índex B+ definit sobre a
 - Utilitzar només l'índex B+ definit sobre b
 - Utilitzar simultàniament tots dos índexs B+:
 - Accés índex B+ sobre a : recuperem els RID de les files del fitxer de dades que verifiquen la condició $a>=X$. Anomenem $RID_{a>=X}$ el conjunt d'RID trobats
 - Accés índex B+ sobre b : recuperem els RID de les files del fitxer de dades que verifiquen la condició $b=Y$. Anomenem $RID_{b=Y}$ el conjunt d'RID trobats
 - Intersecció dels dos conjunts de RID: $RID_{a>=X} \cap RID_{b=Y}$, obtenim els RID de les files del fitxer de dades que simultàniament verifiquen les dues condicions ($a>=X$ and $b=Y$)
 - Es desencadenen els accessos corresponents ($RID_{a>=X} \cap RID_{b=Y}$) al fitxer de dades de R
- No es pot saber *a priori* quina és la millor alternativa, pot variar en funció de la consulta concreta a resoldre

Cost accés per diversos valors: Estratègia intersecció de RIDs

- Cost = Cost accés índex₁ + ... + Cost accés índex_n + Cost accés fitxer de dades
- Sobre el nostre exemple: select * from R where **a>=X** and **b=Y**



- Cost = $(h_a+F_a) + (h_b+F_b) + |RID_{a>=X} \cap RID_{b=Y}|$
- $|RID_{a>=X} \cap RID_{b=Y}|$ és equivalent a $|R(a>=X \wedge b=Y)|$ que és el nombre de files de la taula R que verifiquen la condició expressada a la consulta ($a>=X$ and $b=Y$)
- El cost d'accés a cada índex es pot disminuir en 1 unitat si les arrels estan a memòria