

Emmagatzematge de les BD i implementació dels accessos a les BD

- Memòria externa
- L'arquitectura dels components d'emmagatzematge
- Implementació de mètodes d'accés

Memòria externa

- Necessitat de la memòria externa
 - Fitxers
 - Esquema bàsic de l'E/S per fitxers
 - Suports
 - Discos magnètics
-

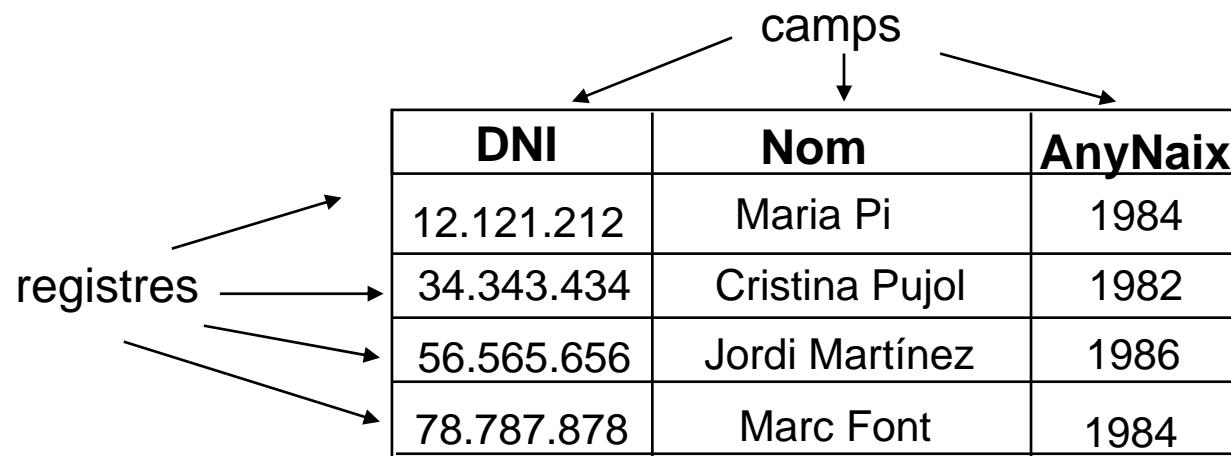
Necessitat de la memòria externa

	mem. interna	mem. externa
preu	alt	baix
temps 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.

Fitxers (1)

- Les dades emmagatzemades a memòria externa, s'estructuren en **fitxers**
- **Exemple:** fitxer d'alumnes



DNI	Nom	AnyNaix
12.121.212	Maria Pi	1984
34.343.434	Cristina Pujol	1982
56.565.656	Jordi Martínez	1986
78.787.878	Marc Font	1984

Fitxers (2)

- El SO gestiona els fitxers mitjançant el **sistema de fitxers**
- Normalment s'ofereixen com a mínim **dos tipus d'accés** als fitxers:

	codi assig	crèdits
R1	DABD	7,5
R2	GSI	6
R3	BD	9
R4	ADA	7,5

- **Accés seqüencial per posició**
 - DABD 7,5 GSI 6 BD 9 ADA 7,5
- **Accés directe per posició**
 - posició 3: BD 9
 - posició 2: GSI 6

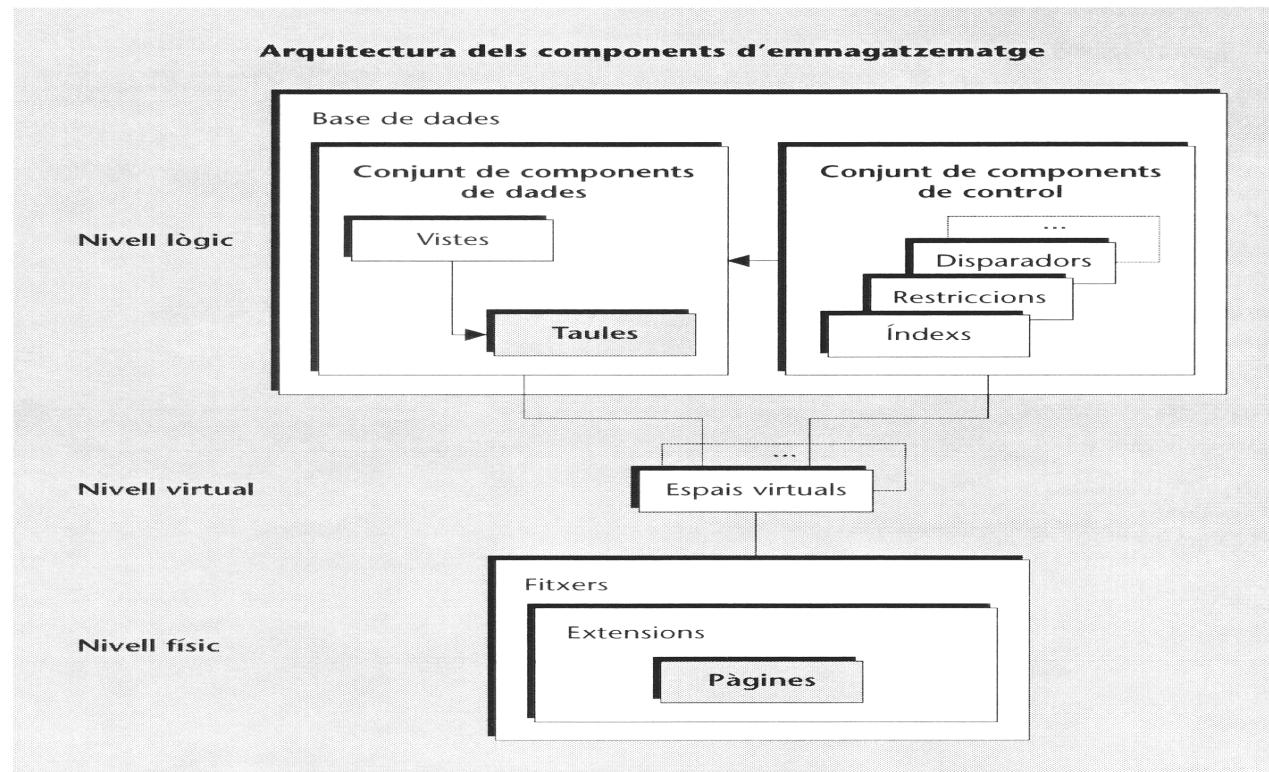
Emmagatzematge d'una base de dades i Mètodes d'accés

- 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 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.
- **Objectius Concrets**
 - Distingir els components i poder situar-los en una arquitectura funcional
 - Entendre el concepte de pàgina, la seva estructura, agrupacions
 - Entendre la funcionalitat, l'estructura i els tipus d'espai virtual

L'arquitectura dels components

- La manera més senzilla d'explicar que és una BD: un conjunt de dades persistents ! -> no han de desaparèixer entre execucions -> medi no volàtil (disc magnètic)
- Una arquitectura és una manera d'esquematitzar la realitat, és una eina ideada per abstraure i entendre els trets fonamentals dels sistemes més complexos (ex: l'arquitectura ANSI/SPARC)
- Per a comprendre la relació entre les dades tal i com les veu el programador o usuari final i tal com estan emmagatzemades:

Figura 1

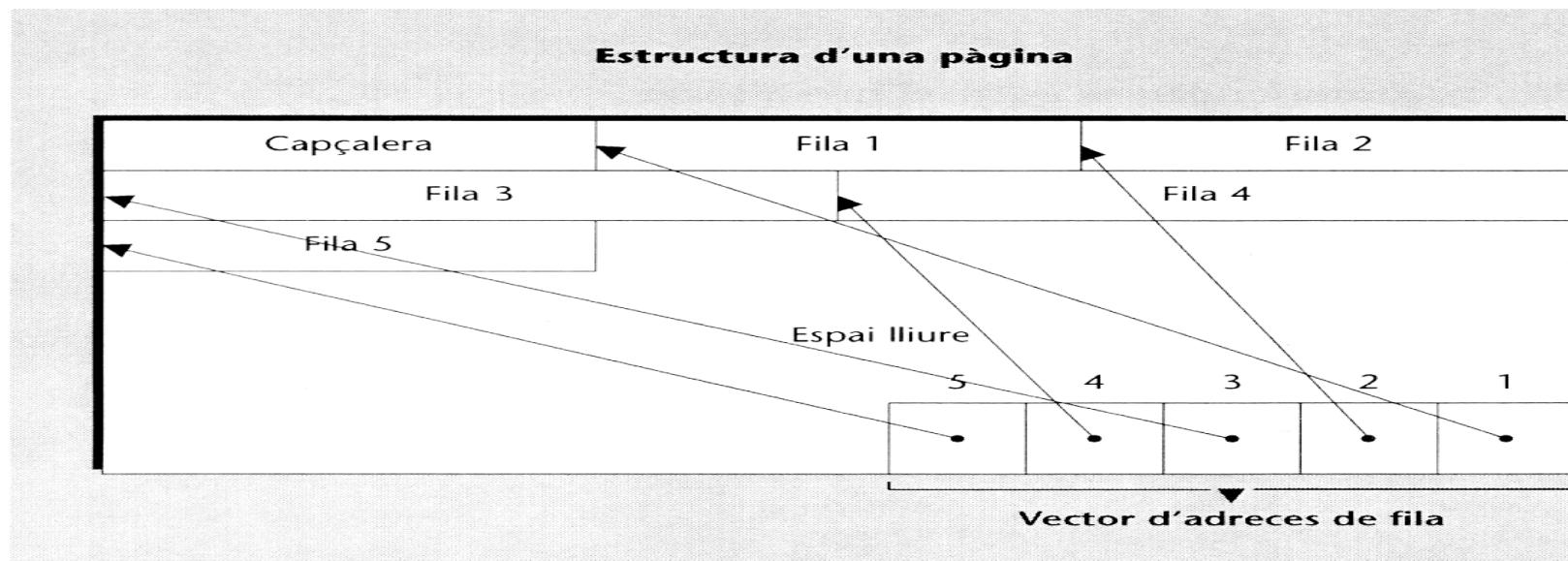


El nivell físic

- Les dades s'emmagatzemen a discs 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 discs 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.
- En canvi, en els sistemes clàssics de fitxers, aquestes dues unitats no tenen perquè coincidir.
- És pot accedir a menys d'una pàgina? I a més?
- **Estructura física :** Longitud fixa (2K,4K, 8K).



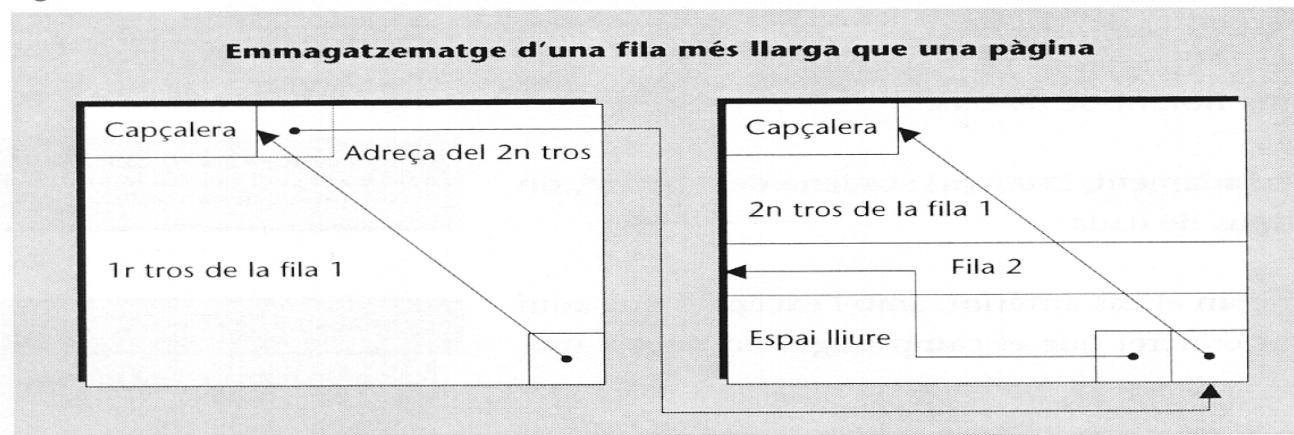
La fila

- Estructura fila:

Capçalera	Camp1	Camp2	Camp3	...
-----------	-------	-------	-------	-----

- 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:

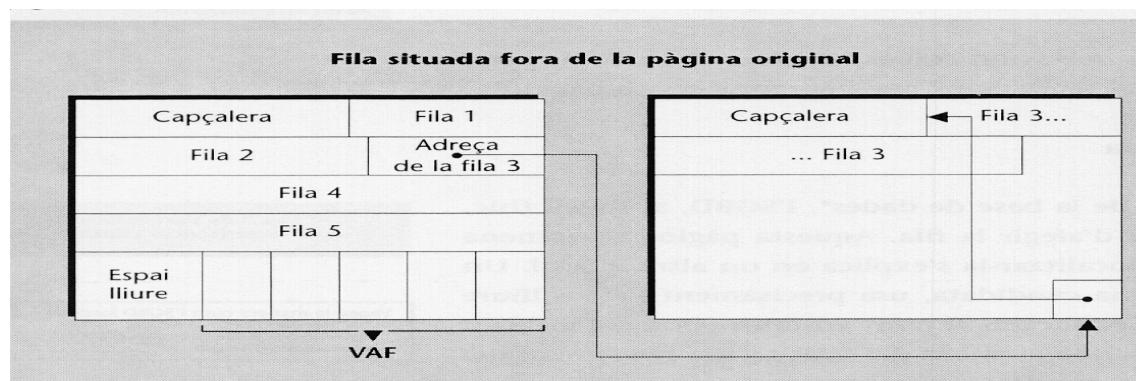
Capçalera	Contingut
-----------	-----------
- A la capçalera del camp:
 - Ens diu si el camp és Null / not null (si el camp admet valors nuls)
 - Longitud del camp

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: aaaa | 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
0		123

Gestió de la pàgina

- **Formatació**
- **Càrrega inicial de files**
 - La primera darrere la capçalera i el vaf que l'apunta al final
 - La segona darrere la primera i el vaf just abans del darrer
 - Hem de deixar un % d'espai lliure
- **Alta posterior d'una nova fila**
 - Localitzar la pàgina candidata
 - Usar l'espai lliure; si ple, hi ha diverses opcions en funció del tipus d'espai en el que s'emmagatzemi (pàgina següent, punters, nova candidata)
- **Baixa d'una fila**
 - Alliberar l'espai ocupat
 - Reorganització interna de la pàgina
- **Canvi de longitud**
 - Si disminució aleshores procés similar a la baixa
 - Si augment i espai suficient aleshores desplaçament de les files que segueixen
 - Si augment i no espai



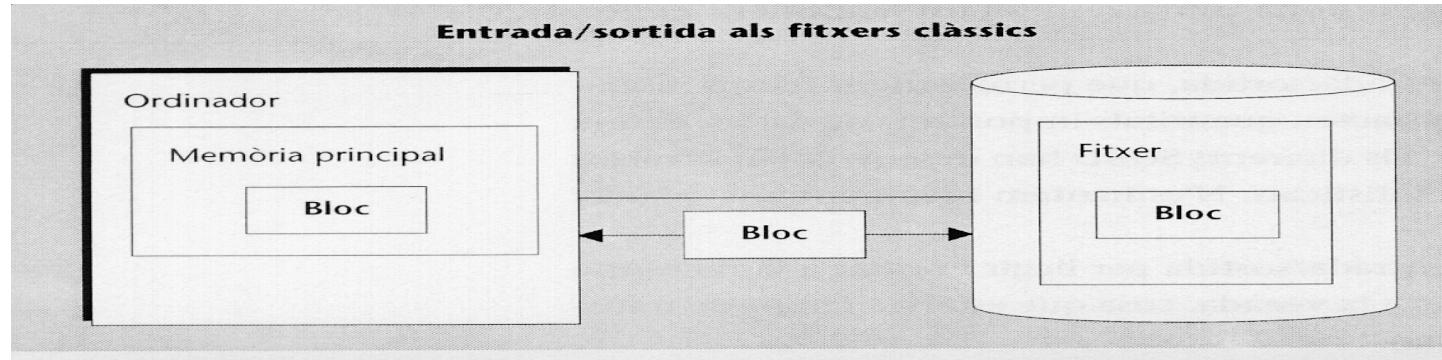
Altres tipus de pàgines

- **Pàgines d'altres components:** vistes, disparadors, procediments
 - Totes les definicions al catàleg → com les taules
- **Pàgines d'índexs**
 - Un índex és un conjunt d'enregistraments, *entrades*, que pot ser molt voluminos → com les taules
- **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 → més endavant el veurem

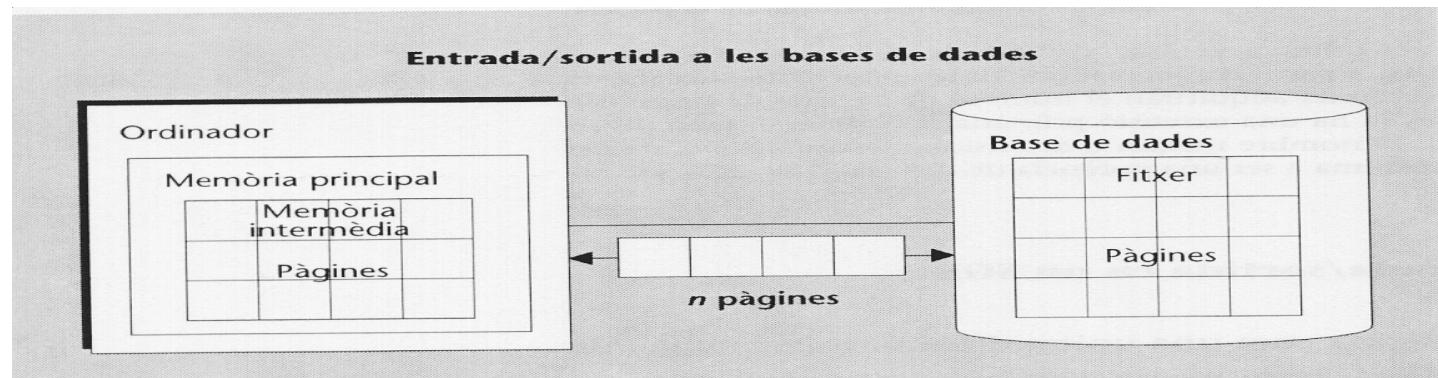
Extensió i Fitxer

- 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
- Extensió Primària i Secundària.
 - Normalment, l'ABD pot definir el tamany de l'extensió primària del fitxer associat a una taula i el de les secundàries, que sol ser per defecte 32k.

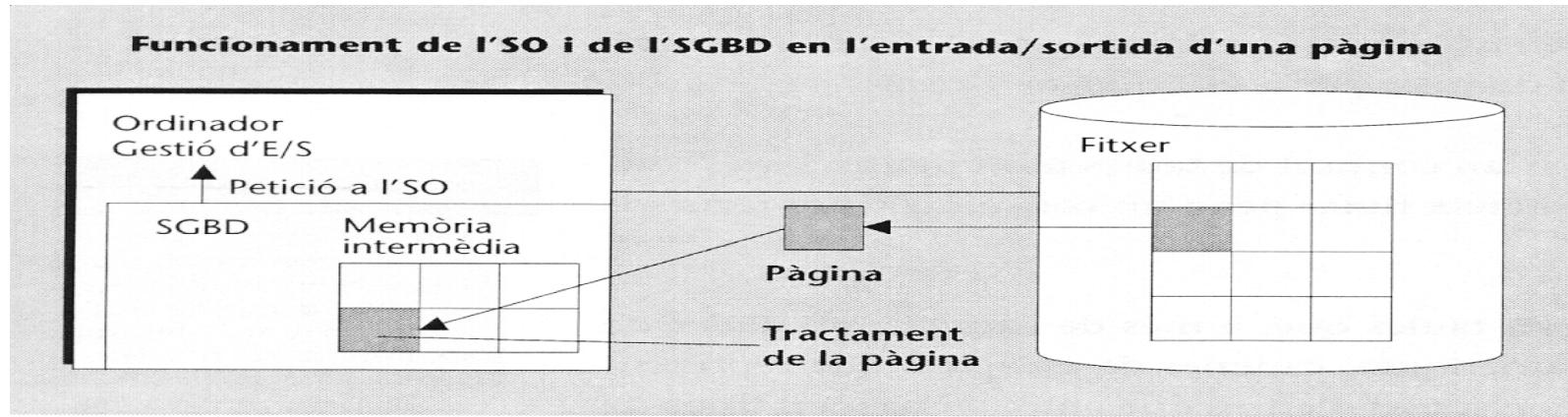
E/S fitxers tradicionals vs E/S SGBD



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



SGBD - SO



- 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:

Si Taula \Leftrightarrow Fixter, no cal un nivell intermedi que faci correspondre 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:

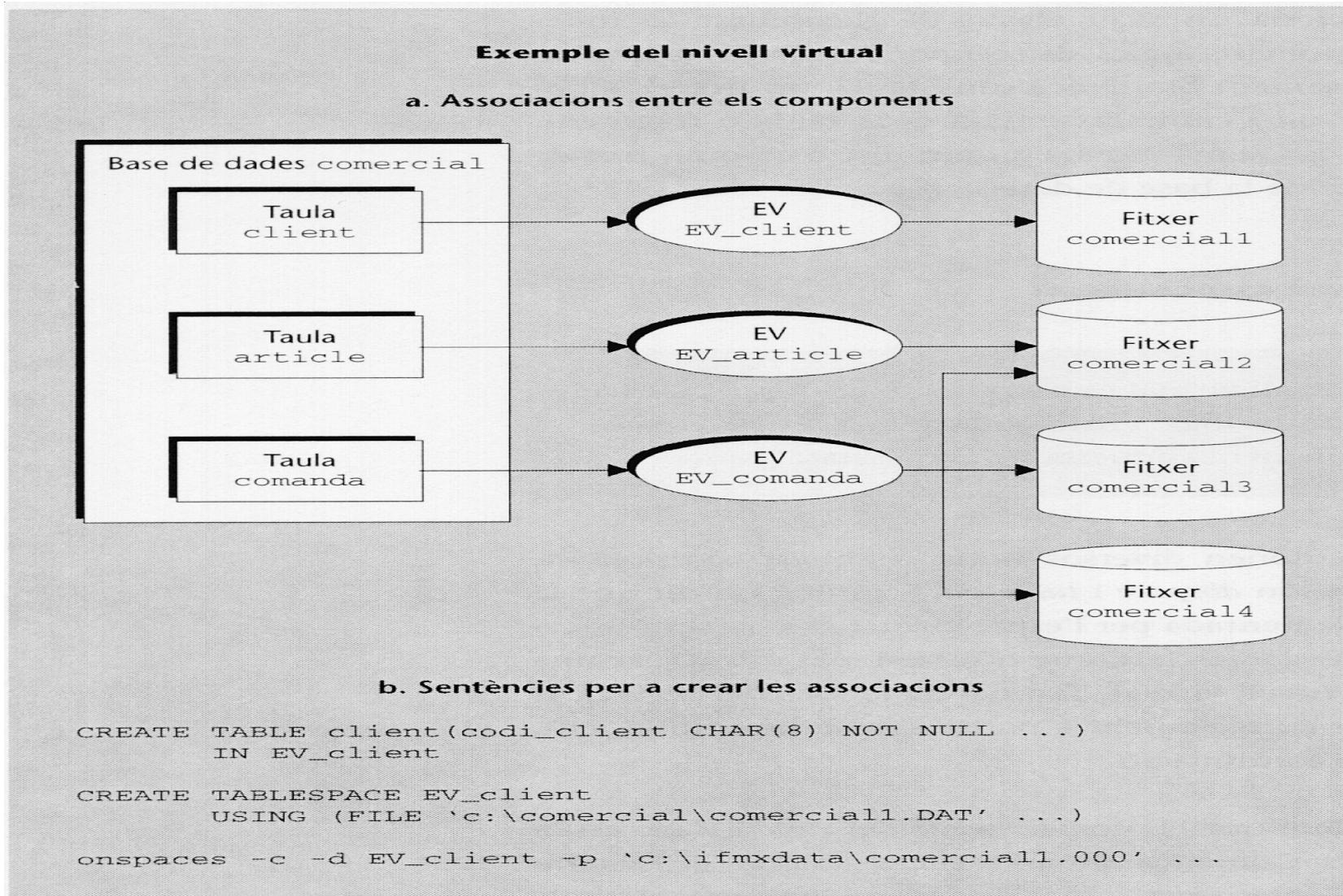
Nivell intermedi que permet relacionar components lògics amb físics; proporciona independència entre nivells. Flexibilitat

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

- Noms comercials: dbspace (informix); tablespace (db2) ...
- Normalment,

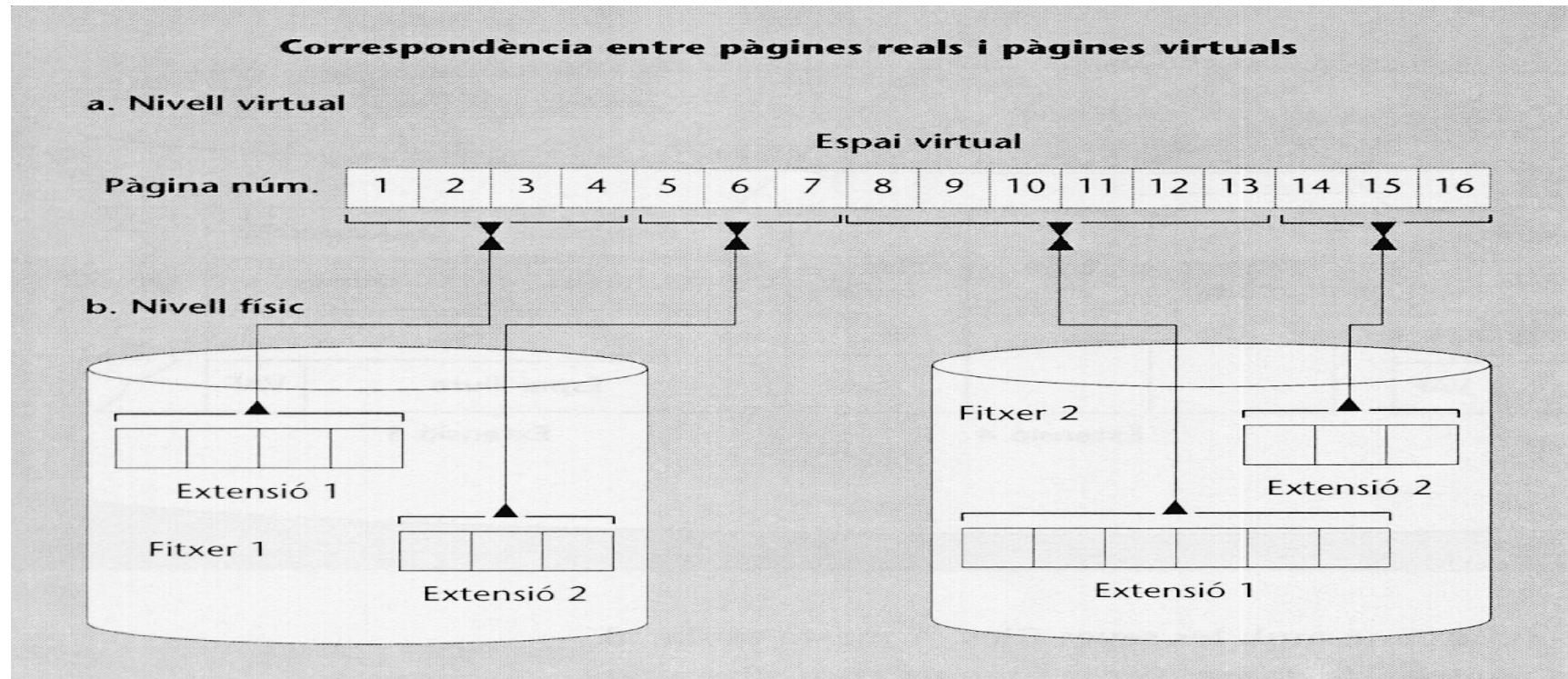


Exemple nivell virtual

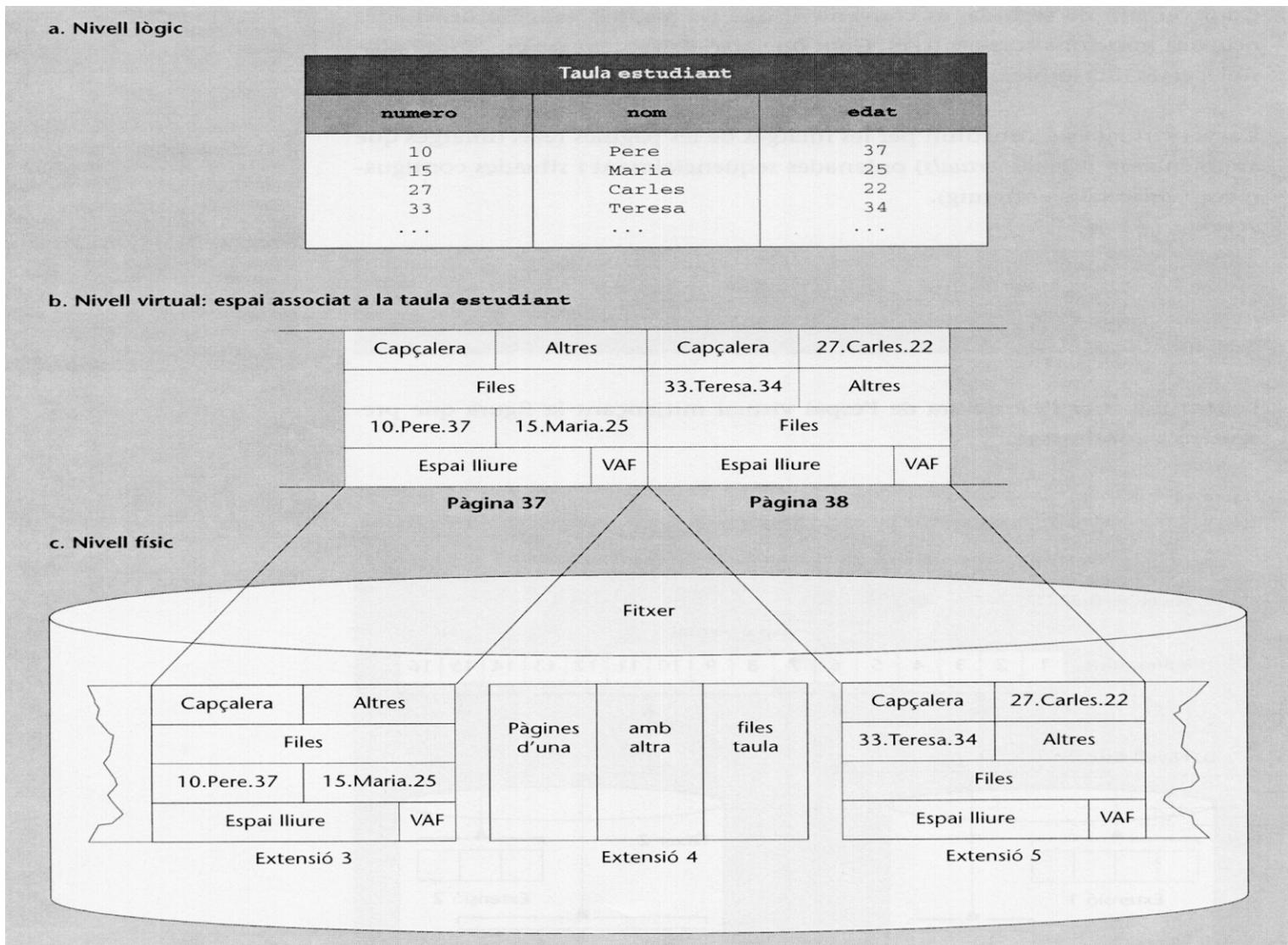


Estructura de l'espai virtual

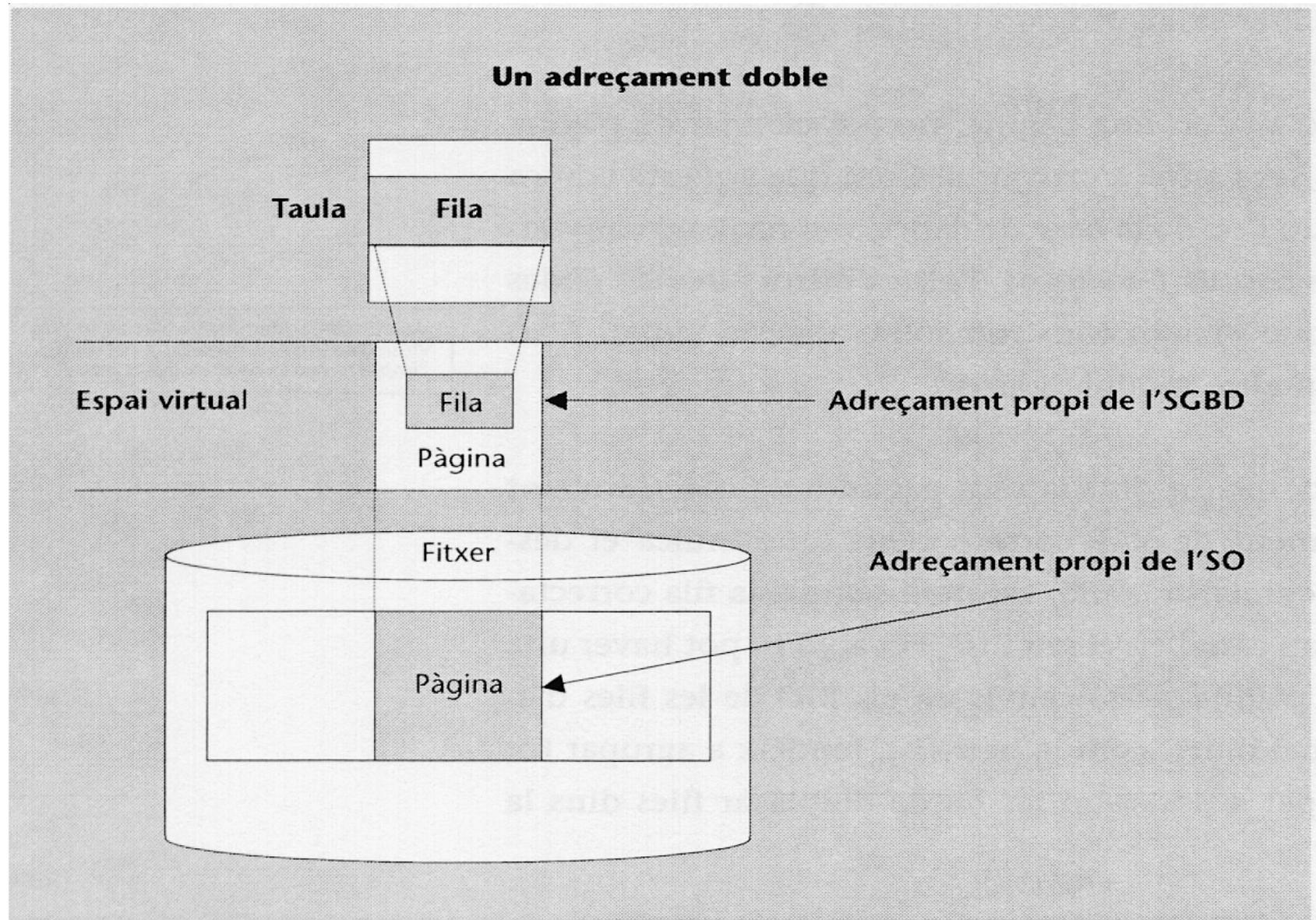
- Espai virtual: visió diferent de les pàgines físiques, sense duplicar
 - Paral·lelisme: Vistes(taules) Memòria virtual(real)
- Espai virtual: Seqüència de pàgines virtuals que es relacionen una a una amb les reals del nivell físic



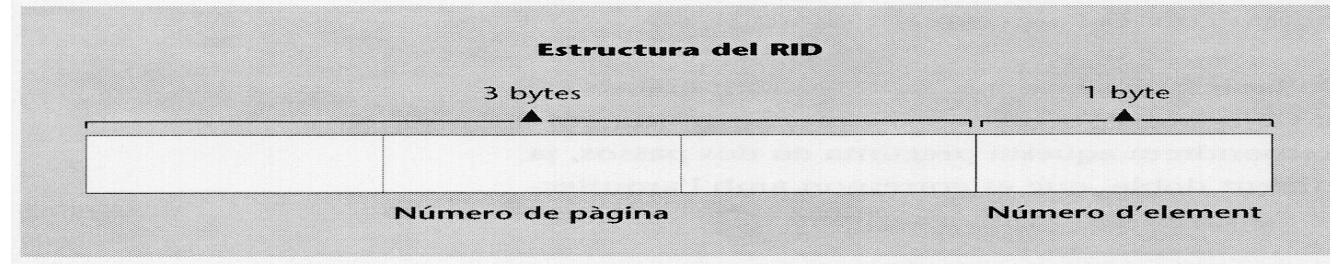
Exemple dels tres nivells



Adreçament en un SGBD



Record Identifier



- 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

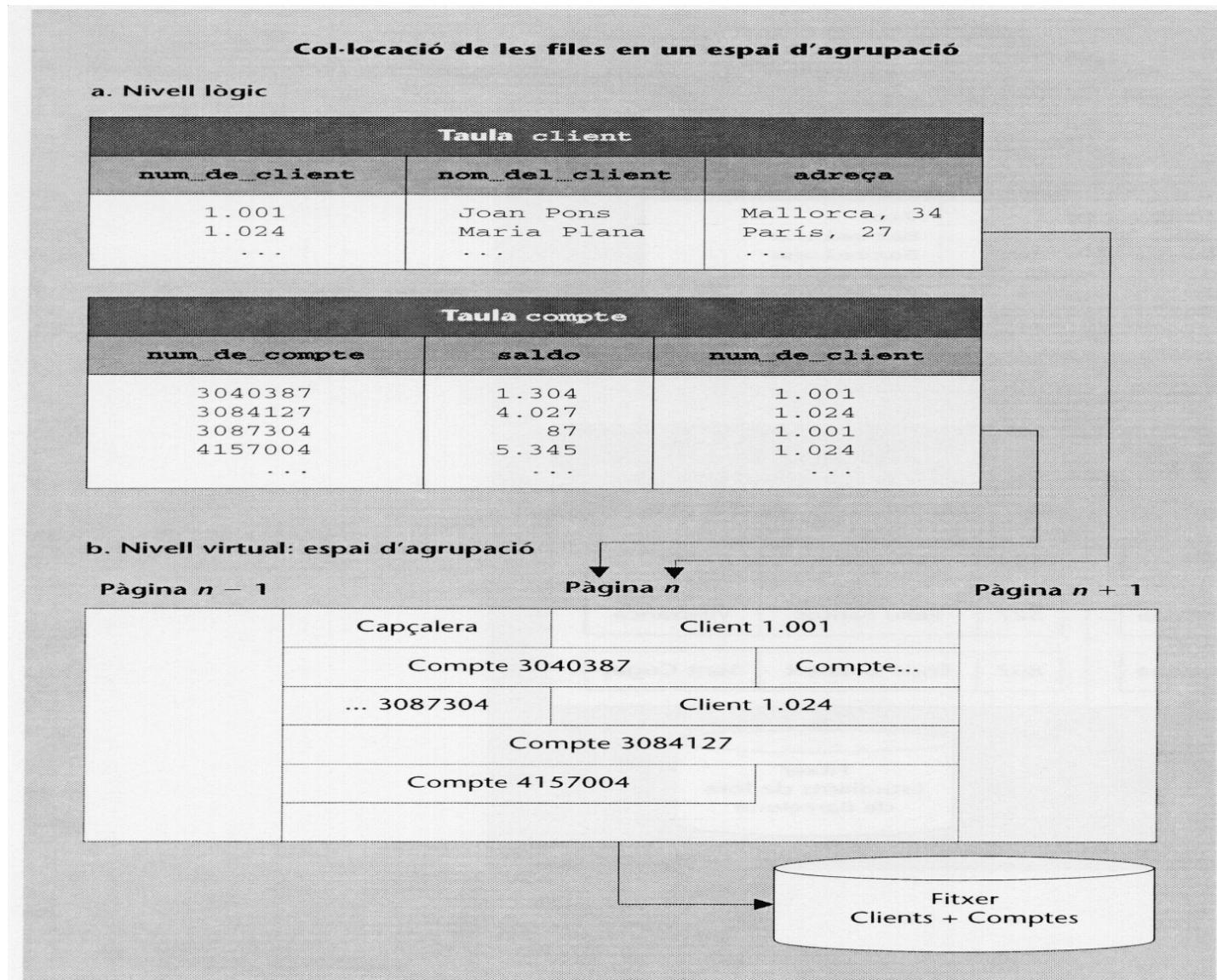


- **Espai d'agrupació**

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



Exemple d'espai virtual d'agrupació

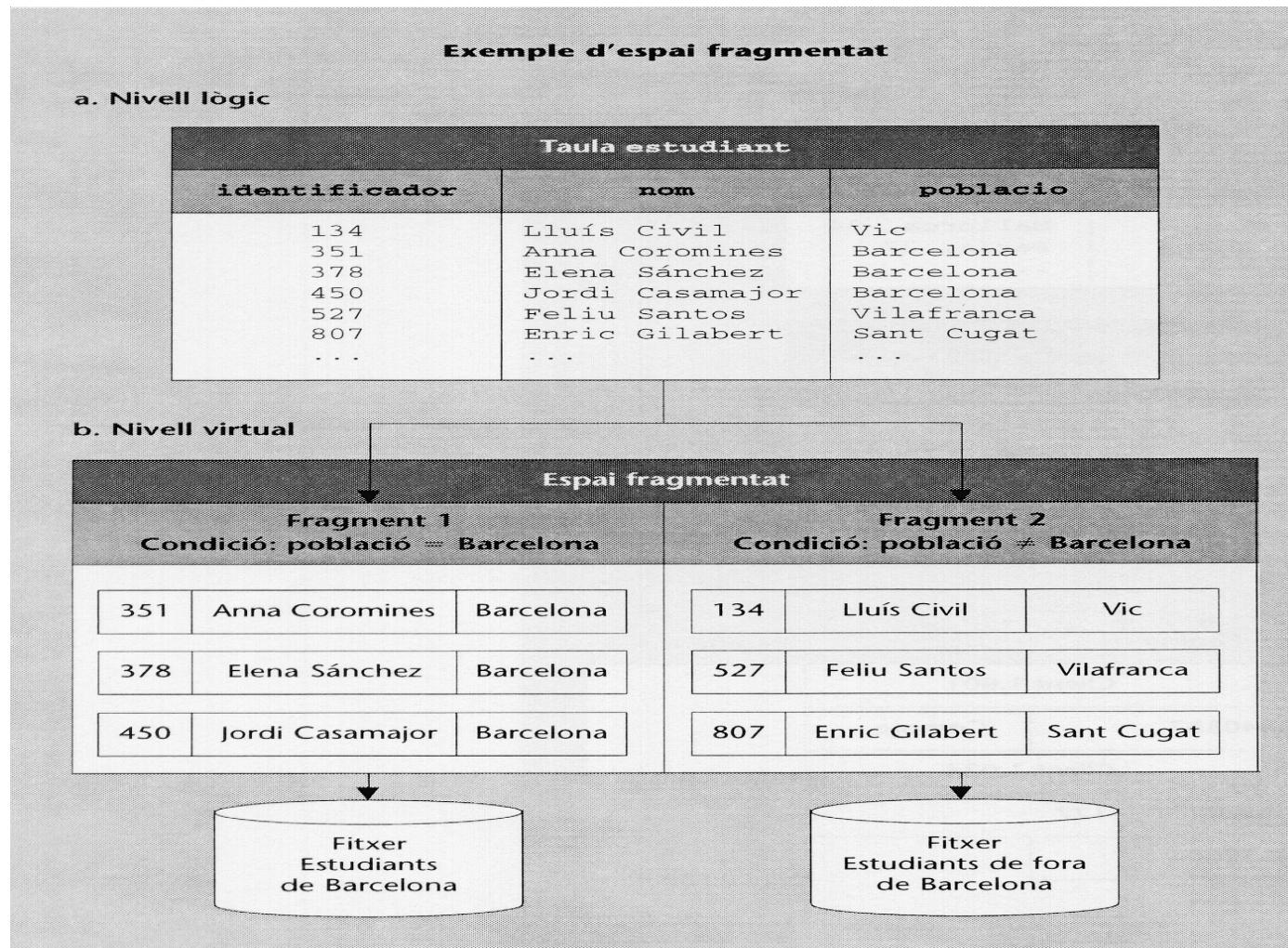


Tipus d'espais virtuals

- **Espai fragmentat**
 - Per a taules molt grans
 - » Ex: Info. Trucades telefòniques 2 mesos
 - » 10 milion abonats 100 trucades 2 mesos = 1000 milions files
 - » 1000 milions 50 bytes trucada = 50 GB
 - La taula s'associa a l'espai i cal establir el criteri de fragmentació
 - Ex: valor d'un camp clients < 20.000 ...
 - Aleatoriament i uniforme (Round robin)
 - Es pot associar cada fragment a un fixter diferent
 - Optimització



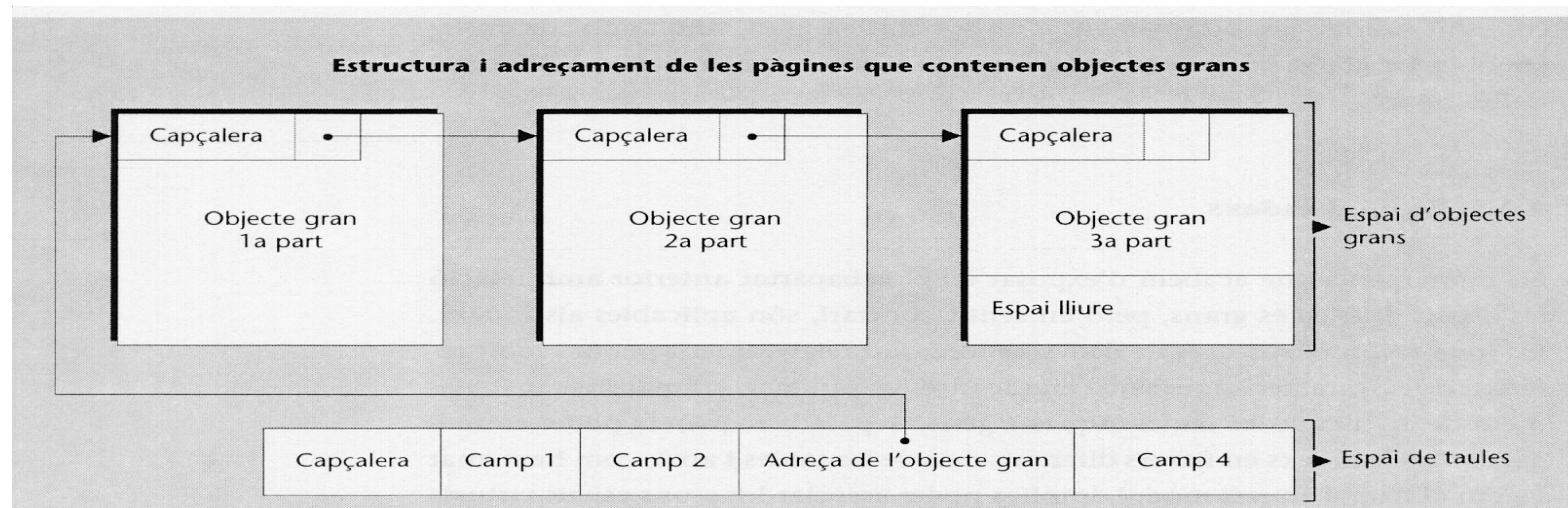
Exemple espai virtual fragmentat



Tipus d'espais virtuals

- **Espai d'objectes grans**

- Els objectes grans s'emmagatzemen separadament
 - L'accés als tipus tradicionals és més freqüent
 - E/S diferent
- La mida de les pàgines més grans és de 32k
- 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
 - 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:
 - Catàleg
 - Taules temporals
 - Dietari
 - Miralls
 - Mirar manuals!

Implementació de mètodes d'accés

- Els SGBD estructuren les seves dades en pàgines dels espais virtuals i aquests físicament s'emmagatzemen en pàgines físiques dels discos magnètics.
- 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.
- **Objectius Generals**
 - Coneixer 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 amb vista als accessos per valor.

Implementació de mètodes d'accés

Els mètodes d'accés a una base de dades

Implementació dels accessos per posició

Implementació dels accessos per valor

Arbre B⁺

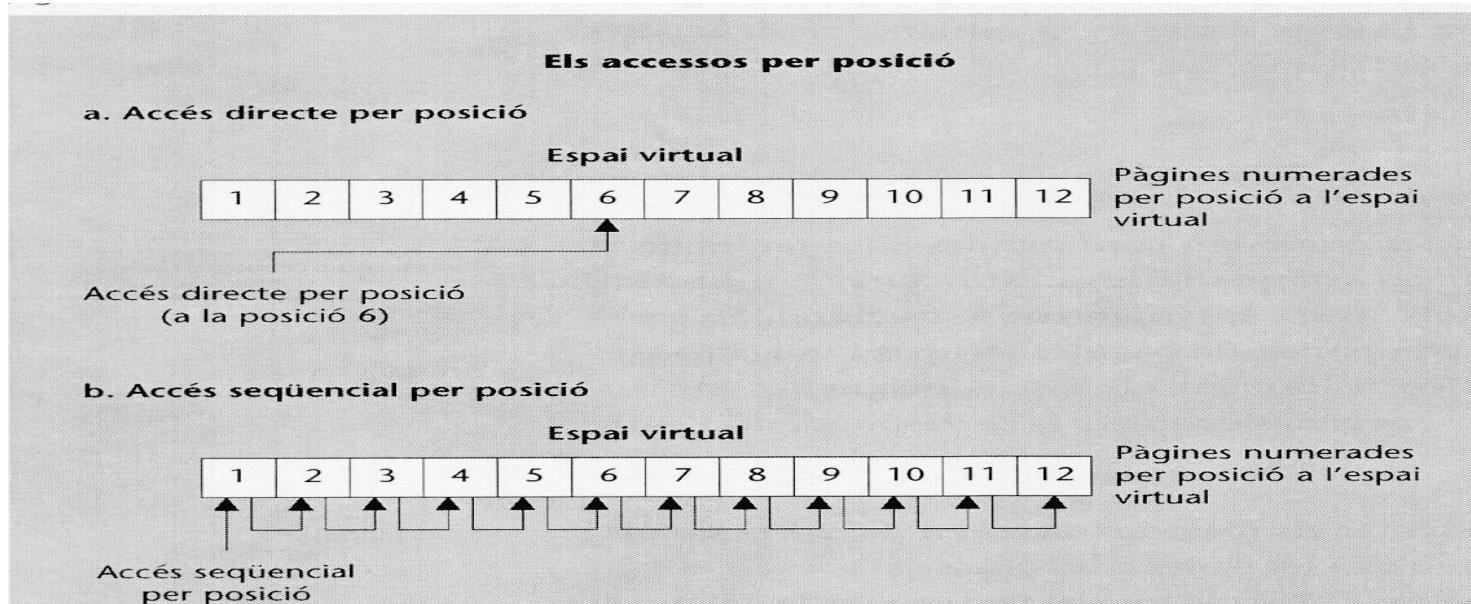
Índexs agrupats

Implementació dels accessos per diversos valors

Introducció a l'optimització

Els mètodes d'accés a una base de dades

- **Els accessos per posició**



- Tecnologia de fitxers: fitxers seqüencials i fitxers relatius !!

- Suficients per casos simples:

Empleats(nemp, nom, despatx, sou)

Directe: Insert into Empleats values (25, 'Joan', 150, 200k)

Seqüencial: Select * from Empleats

Els mètodes d'accés a una base de dades

- **Els accessos 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
- Tecnologia de fitxers: fitxers per valor i fitxers seqüencials per valor !!!!
- Exemples
 - Directe:
 - Select * from Empleats where despatx=150
 - Update Empleats set sou=250k 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=250k
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

Els mètodes d'accés a una base de dades

- **Els accessos per diversos per valors**
 - Accedir a les files per valors de diversos atributs
 - Accessos directes o seqüencials
 - No tenen corresponent a la tecnologia de fitxers
- Exemples
 - Select * from Empleats order by despatx, sou
Directe | seqüencial
 - Select * from Empleats where despatx=150 and sou=200k
Directe | seqüencial
 - Delete from Empleats where despatx>100 and sou>180k
Directe | seqüencial
 - Select * from Empleats where despatx=150 and sou>200k
Directe | seqüencial
- **Els accessos mixtos per diversos per valors**
 - Select * from Empleats where sou=200 order by despatx
 - Delete * from Empleats where despatx>100 and despatx<200 and sou=200k

Implementació dels accessos per posició

- Els SGBD es basen en el sistema operatiu !
- Rutines d'entrada sortida permeten obtenir una pàgina física
- Convencions per a l'estimació del cost
 - Considerem només el cost d'E/S, no CPU
 - Simplificació: Compten només el nombre de pàgines que es llegeixen o escriuen
 - NO: la pàgina està a memòria; no seek no latency
- Accés directe per posició
 - L'SGBD transforma els números de posició de les pàgines virtuals en adreces físiques
 - » Cost : 1
- Accés seqüencial per posició
 - Similar
 - » Cost : N

Implementació dels accessos per valor

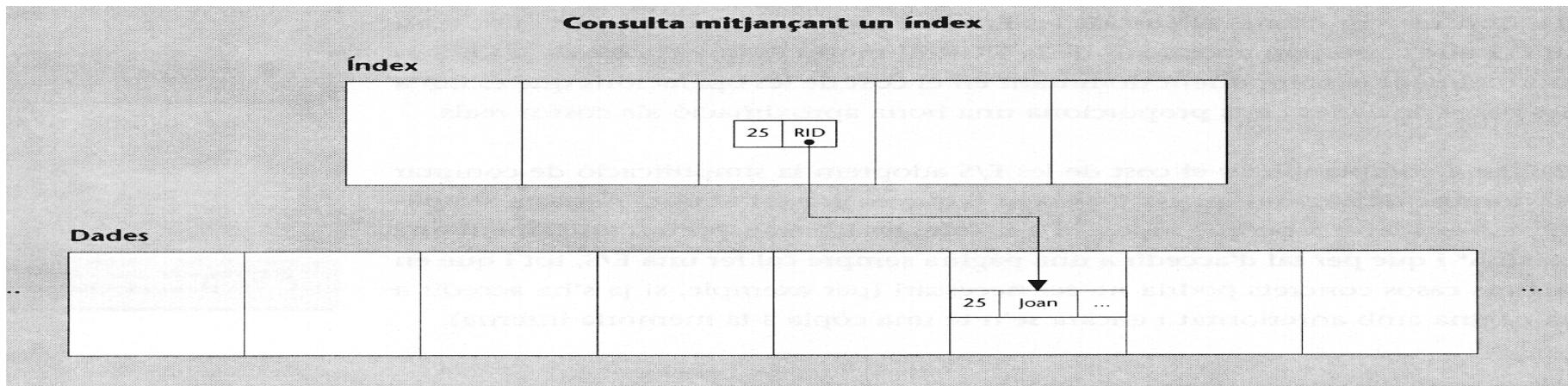
- Per implementar de manera eficient l'accés per valor s'usa unes estructures de dades auxiliars anomenades índexs
- Necessitat:
 - **Select * from Empleats where despatx=150**

Solucions:		MIG	MAX
Seqüència	accés seqüencial per posició	$N / 2$	N
Taula	Idem	$N / 2$	N
Taula ord. físicament	Cerca dicotòmica Accés directe posició	$\log_2 N$	$\log_2 N + 1$
	Interpolació	$\log_2 \log_2 N$	$\rightarrow N$
Taula ord. lògicament	Cadena curta i llarga	\sqrt{N}	$\sqrt[2]{N}$
$N=30.000$	300.000 reg 10 reg/pàgina 15.000; 15.000; 15; 4; 174 ordre físic !!		
$N=2.300.000$	23.000.000 reg 1.150.000; 1.150.000; 22; 5; 1516		

- **Select * from Empleats
where despatx>150 and despatx<200**

Característiques generals dels índexs

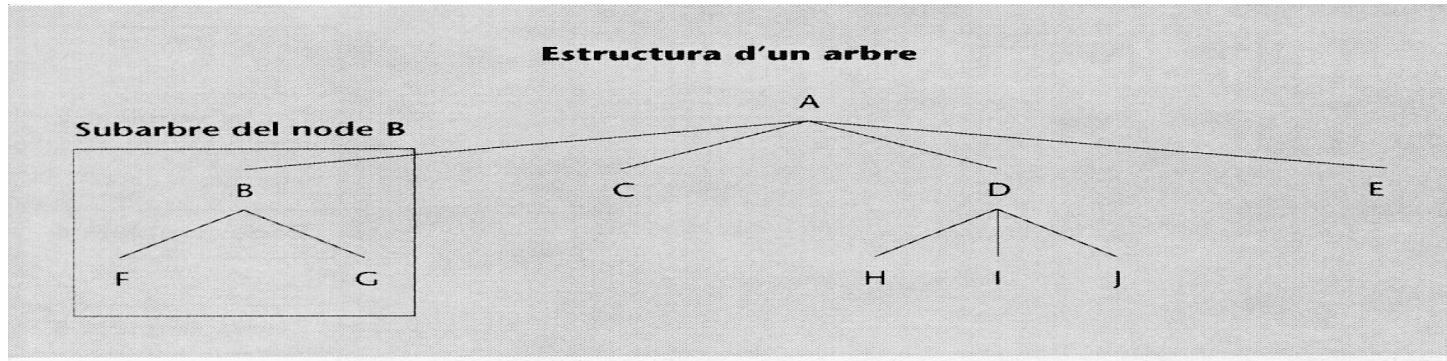
- Utilitat semblant als índexs de llibres
- Les cerques poden ser més ràpides que si es fan sobre les dades ja que els índexs ocupen molt menys espai que les dades
 - Els índexs contenen parelles (valor, RID) → *entrades*



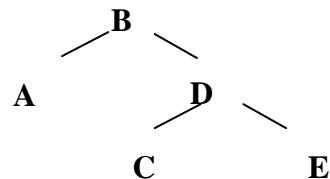
- Fan accés per valor via accés per posició!!!
- Diverses maneres d'estructurar els índexs:
 - Arbre B⁺ (accés per valor i seqüencial per valor)
 - Funcions de dispersió (accés per valor)
 - Altre tipus que no veurem (Bounded disorder files, R-tree,...)
- Una característica molt útil és usar múltiples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

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⁺: Un tipus particular d'arbre de cerca
Recordeu el concepte d'arbre binari, arbre **orientat a fer cerques a memòria interna**



Cost = $(\log_2 \text{número de registres})$ 300.000 reg 18

23 Milions reg 24 No ordre físic!

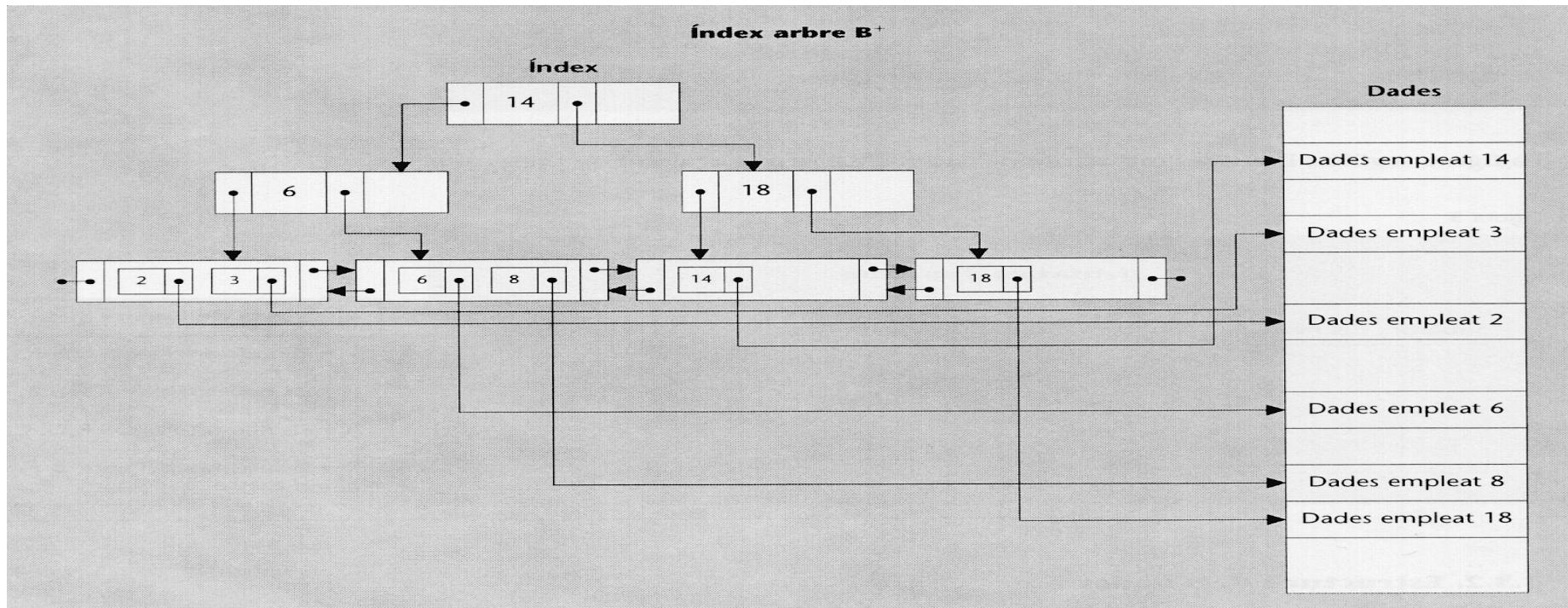
- Un arbre B⁺: **Orientat a Disc; l'objectiu és minimitzar les E/S**

300.000 reg 3 Ja ho veurem!

23 Milions reg 4 No ordre físic!

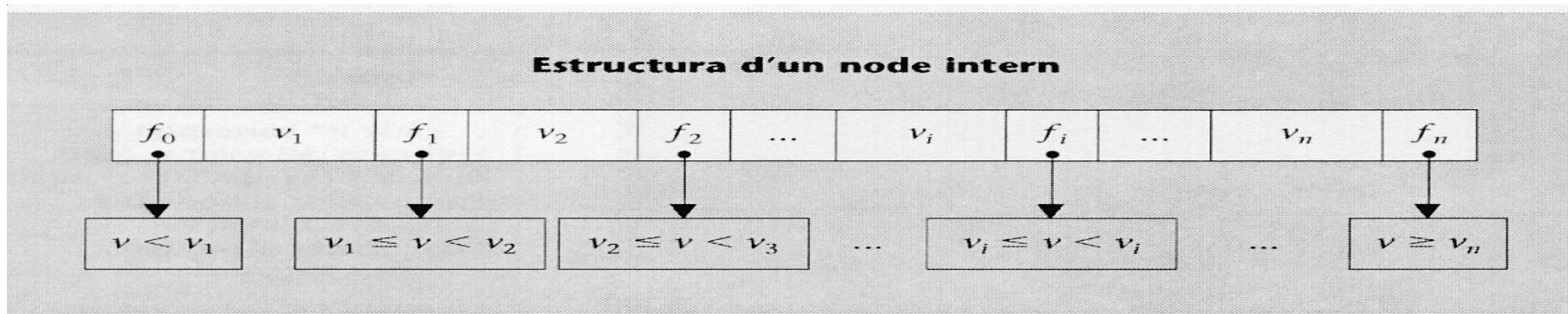
Arbres B+ Estructura

- 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



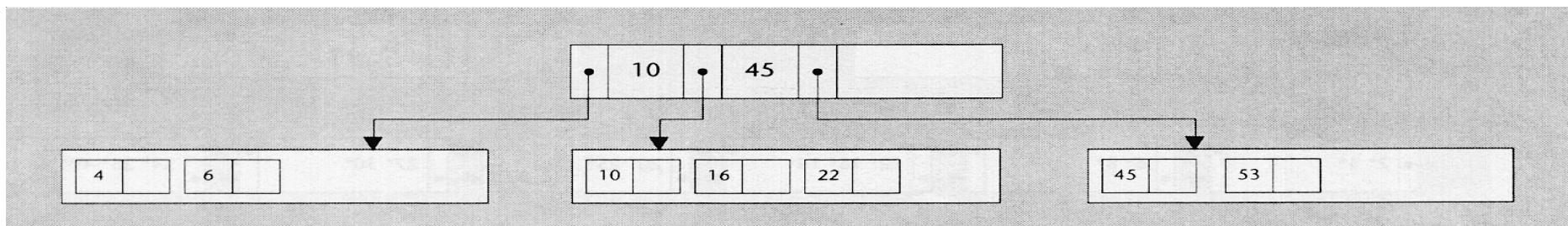
Arbres B+ Estructura node intern

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



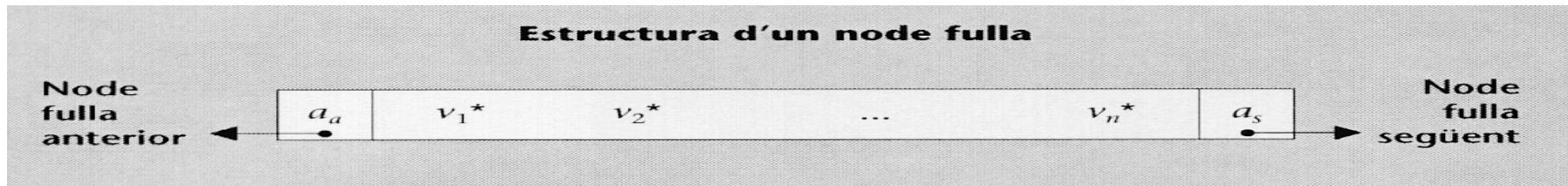
En el cas màxim: $2d$ valors i $2d+1$ apuntadors

- Exemple d'ordre 2:

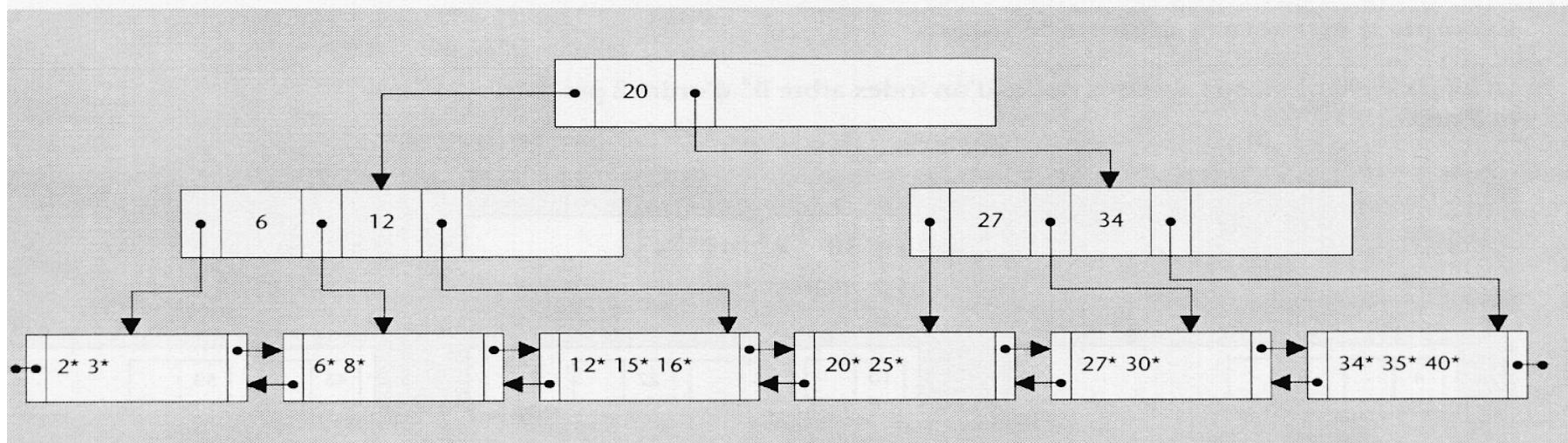


Arbres B+ Estructura node fulla

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

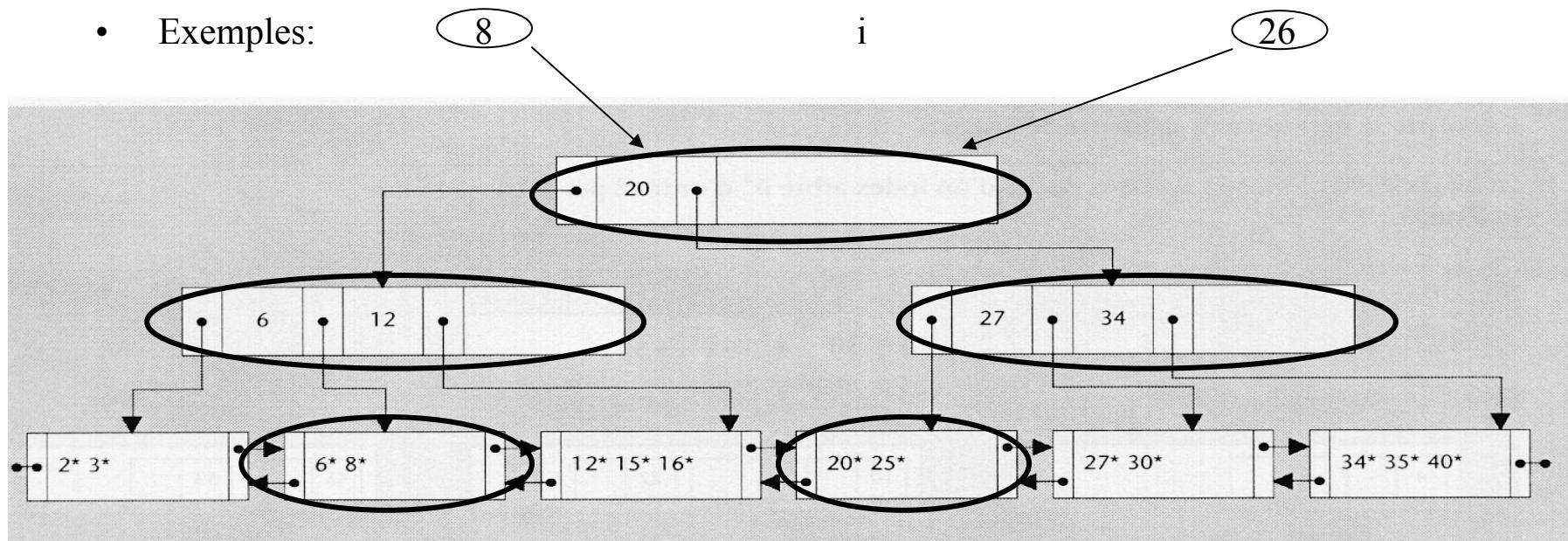


- Cas màxim: $2d$ (valors+rid) + 2 apuntadors a fulla
- Condicions addicionals:
 - Tots els valors d'un node fulla són més petits que els valors de les fulles següents
 - Tots els valors d'un node intern estan repetits a les fulles. Les fulles tenen tots els valors



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 a trobar la fila cercada
- Exemples:



- Per lectures cal: el nombre de nodes que cal accedir és igual al nivell de la fulla on hauria d'estar + RID = 3 + 1
Per actualitzacions?

Arbres B+ Accés seqüencial per valor

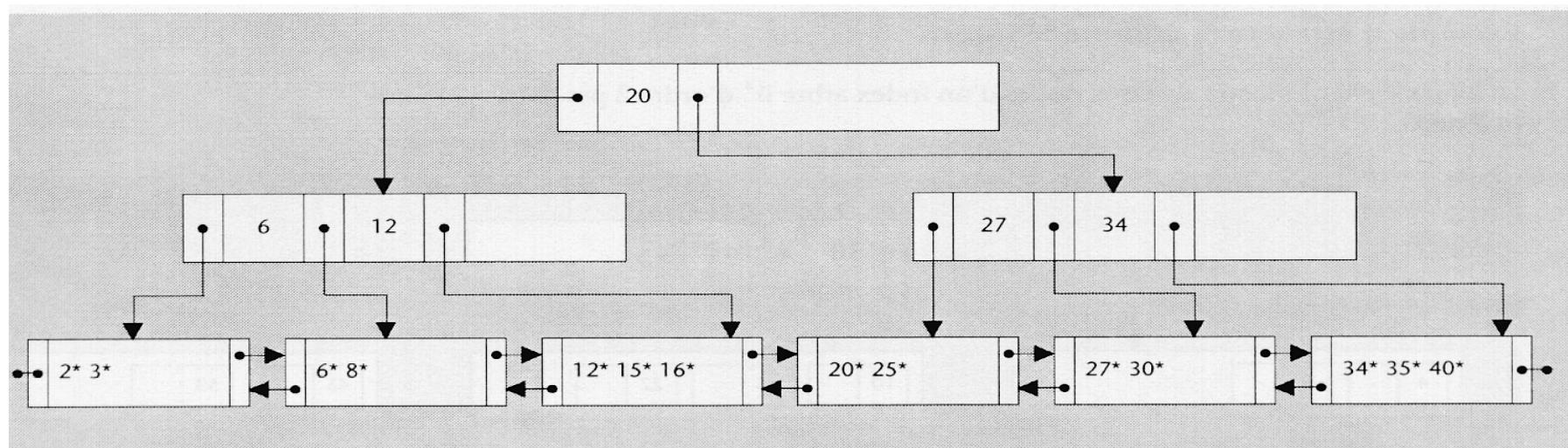
- Per fer un accés seqüencial per valor total cal:
 - Localitzar la fulla de més a l'esquerra
 - Recuperar totes les seves entrades
 - Usar el RID de l'entrada per a trobar la fila cercada
 - Localitzar la fulla següent

Si ho apliquem a l'arbre de la figura anterior:

- 2, 3, 6, 8,
- Per tal de fer cerques parcials: per exemple, accedir als registres amb valor més gran o igual v cal:
 - Localitzar la fulla que té (hauria de tenir) l'entrada v
 - Recuperar totes les seves entrades que compleixin la condició
 - Usar el RID de l'entrada per a trobar la fila cercada
 - Localitzar la següent fulla
- De manera similar, cerques tal que: $v_1 < \text{valor} < v_2$

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 següent arbre d'ordre 2 compleix aquesta propietat



- Que passaria en l'extrem si l'arbre fos binari?
- **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, 3 E/S** per a localitzar la fulla
- L'arrel pot estar a memòria → 1 accés menys
 - Pàgines de 4Kb → ordres entre 50 i 100
 - Nodes interns: $2d * (\text{mida valor}) + (2d+1) \text{ mida apuntador fulla} = 4k$
» $2d * (20) + (2d+1) * 3 = 4096 ; \quad d = 88$
 - Nodes fulla: $2d * (\text{mida valor} + \text{mida rid}) + 2 \text{ mida apuntador fulla} = 4k$
» $2d * (20 + 4) + 6 = 4096 ; \quad d = 85$
 - Simplificació: escollim mida igual $d = 85$
 - Quants registres podem indexar i quin cost té localitzar-los???

Arbres B+ Volum i Cost

- Suposem que $d = 50$ i que cada node té una ocupació 69%
 - Nivell 1: 1 node amb 69 valors i 70 apuntadors
 - Nivell 2: 70 nodes amb 69 valors i 70 apuntadors cadascun
 - Nivell 3: 4900 nodes amb 69 entrades

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

**Localitzem 1 registre de 338100 amb 3 accessos
(+1 per accedir al registre)
(-1 arrel a memòria)**

- I amb 4 nivells? Nivell 4: 343.000 nodes amb 69 entrades = 23667000
- Arbre B+ $\approx \log_{70} 23667000 = \ln 23667000 / \ln 70 = 4;$
 $\approx \log_{70} 338100 = 3$

Dicotomies $N=30.000 \quad 300.000 \text{ reg} \quad 10 \text{ reg/pàgina}$

$\log_2 N = 15 \quad \text{ordre físic}$

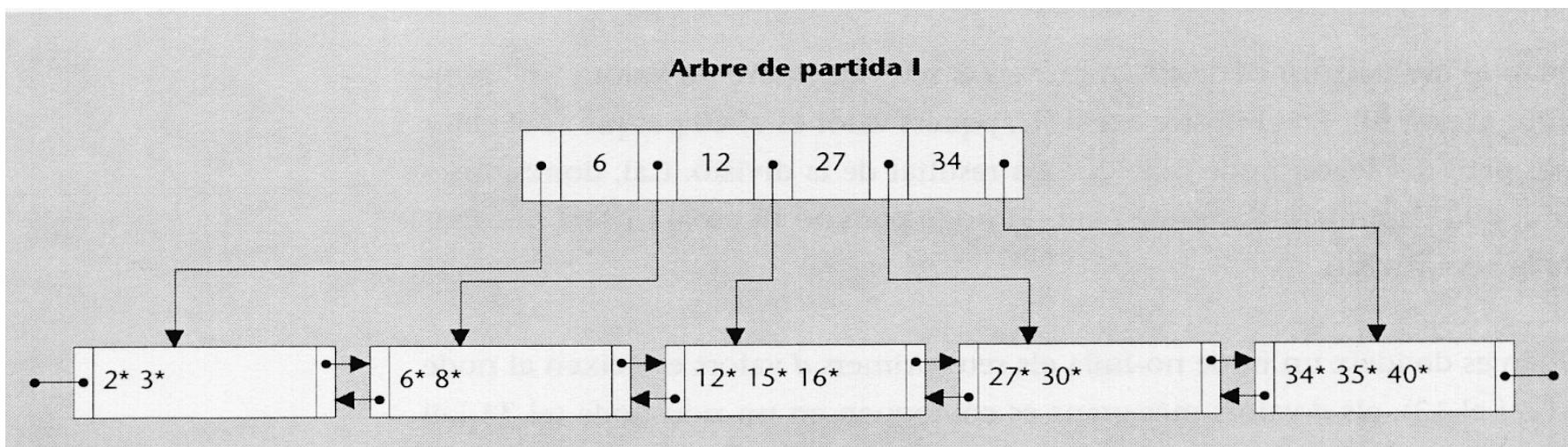
$N=2.300.000 \quad 23.000.000 \text{ reg}$

$\log_2 N = 22 \quad \text{ordre físic}$

Arbre Binari $\log_2 338100 = 19; \log_2 232667000 = 26$

Arbres B+ Insercions

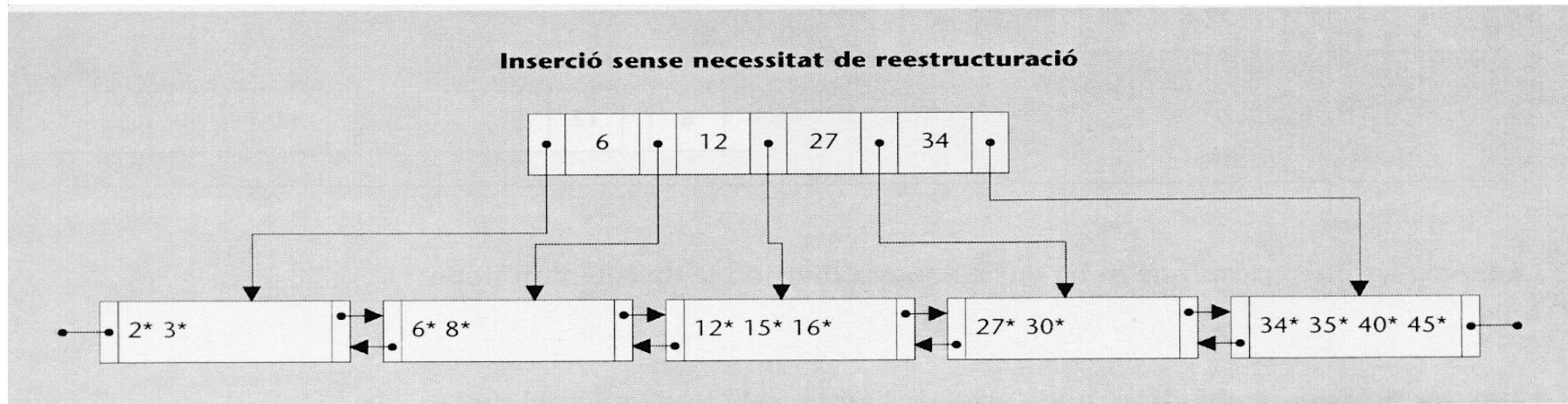
- Algoritme d'inserció: Cal que compleixi la definició d'arbre B⁺
 - Localitzar la fulla a on hauria d'anar
 - **Si** no està plena, **llavors** insertar ordenadament
 - si no** SPLIT cap al pare
 - (si pare ple, split)



- Cas trivial: Inserció de l'entrada 45*

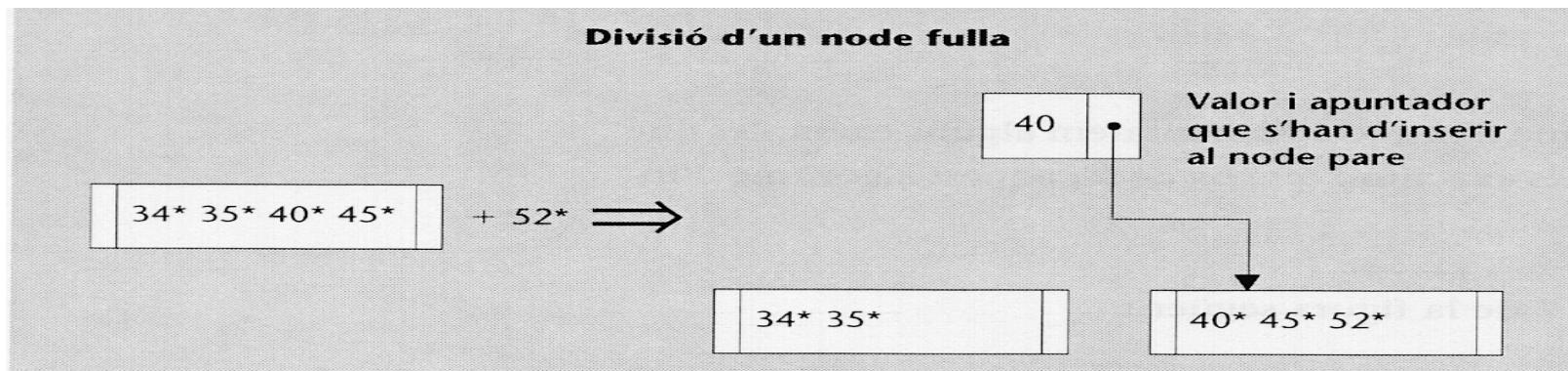
Arbres B+ Insercions

- Resultat



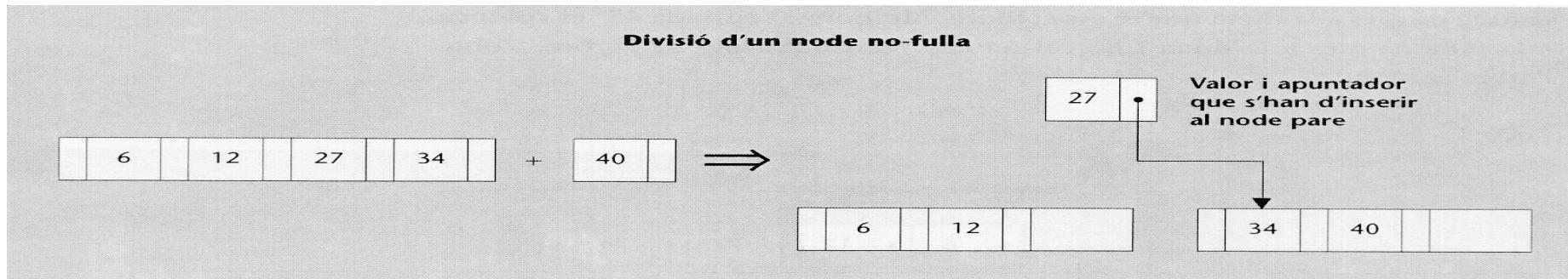
- Doble split: inserció de 52*

- Primer split:

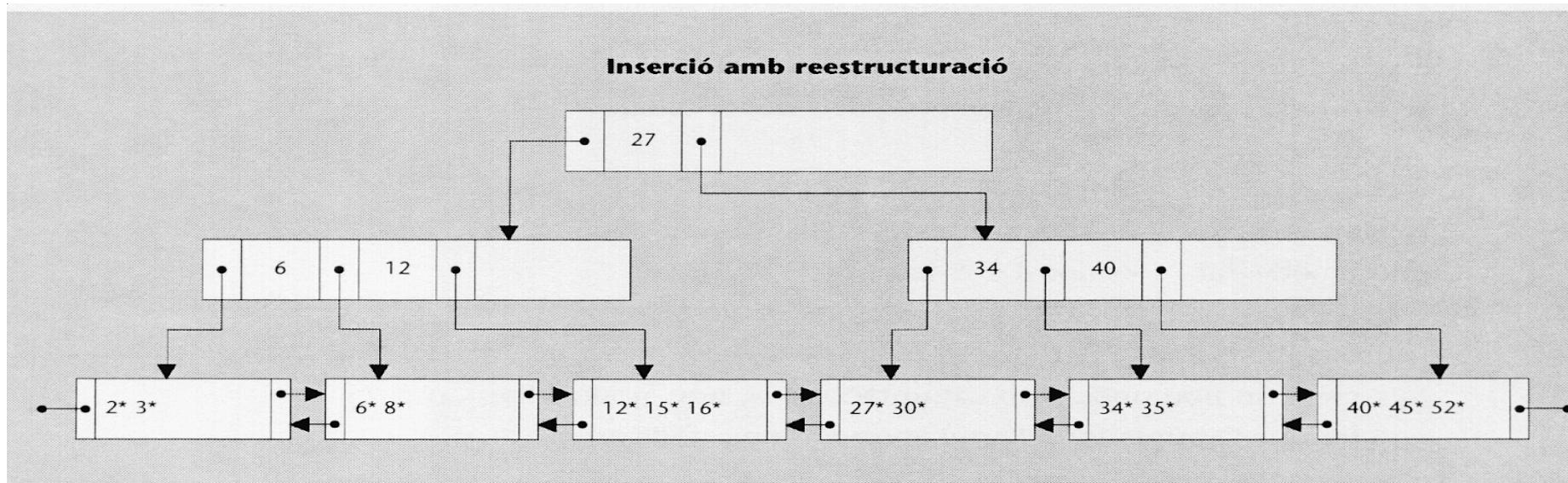


Arbres B+ Insercions

- Segon split



- Resultat



Arbres B+ Supressions

- Algoritme: Cal que preservi l'estructura d'arbre B⁺

- Pas 1: **Supressió**

- Si** el valor es troba a un node no fulla

- llavors** substituir-lo pel següent lexicogràficament

- (sempre està en una fulla);

- esborrar la còpia (entrada) de la fulla

- Si no** esborrar-lo

- Pas 2: **Reestructuració**

- Si** la fulla modificada sota mínims (menys de d entrades)

- llavors**

- Si** hi ha fulla veïna (mateix pare) dreta

- llavors Si** més de d entrades **llavors REDISTRIBUIR**

- si no** FUSIONAR

- (si pare sota mínims, reestructuració)

- si no** Usar fulla veïna esquerra

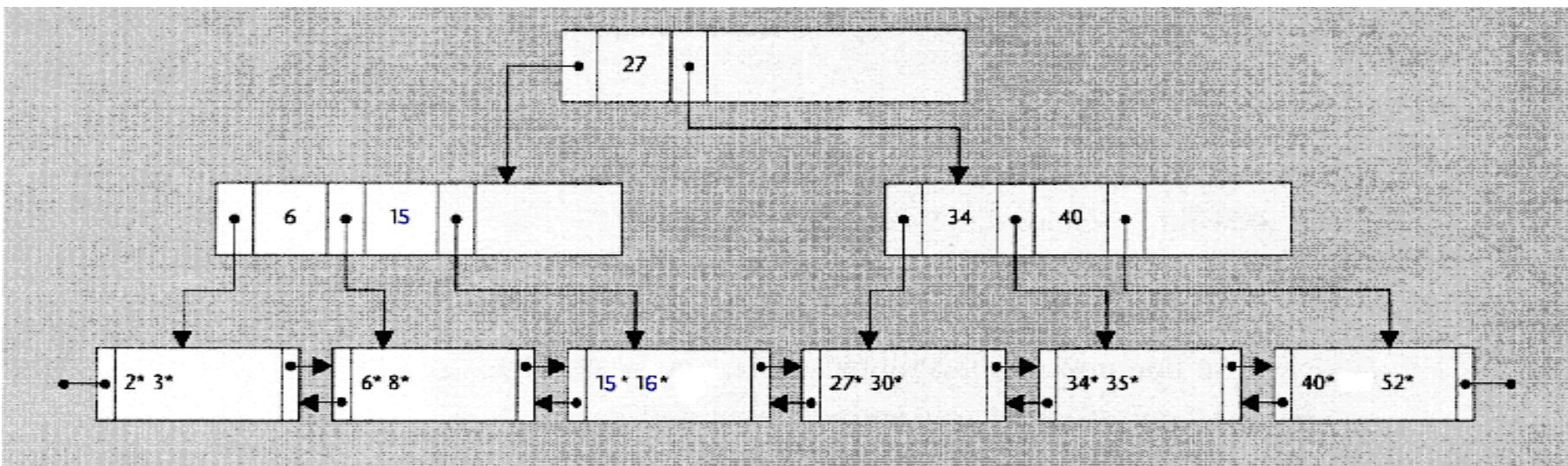
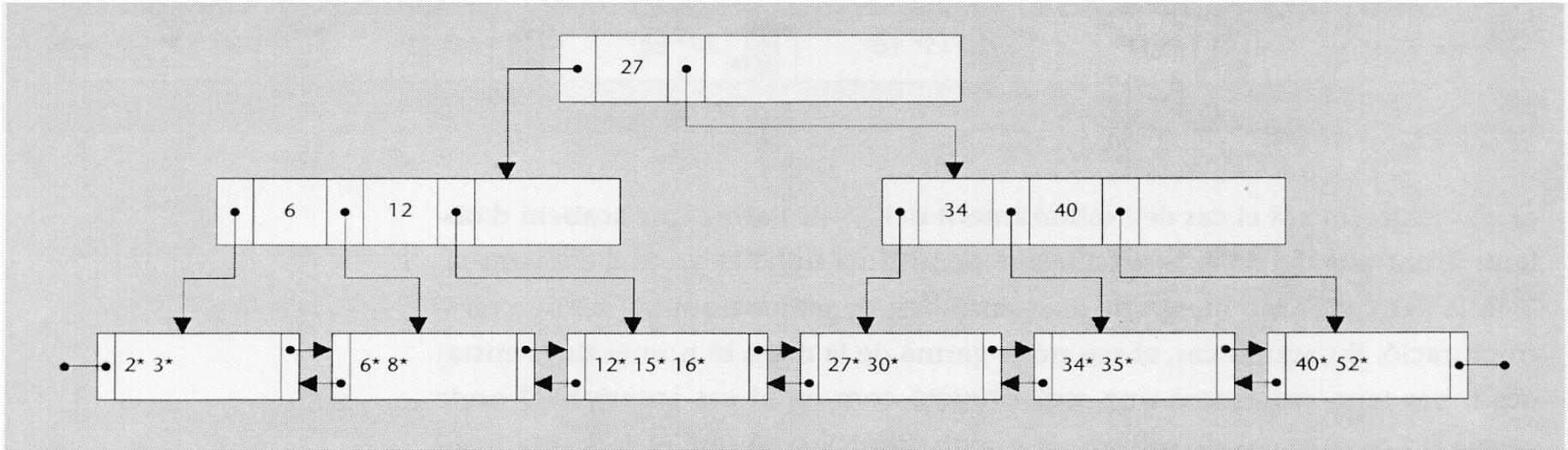
- Si** més de d entrades **llavors REDISTRIBUIR**

- si no** FUSIONAR

- (si pare sota mínims, reestructuració)

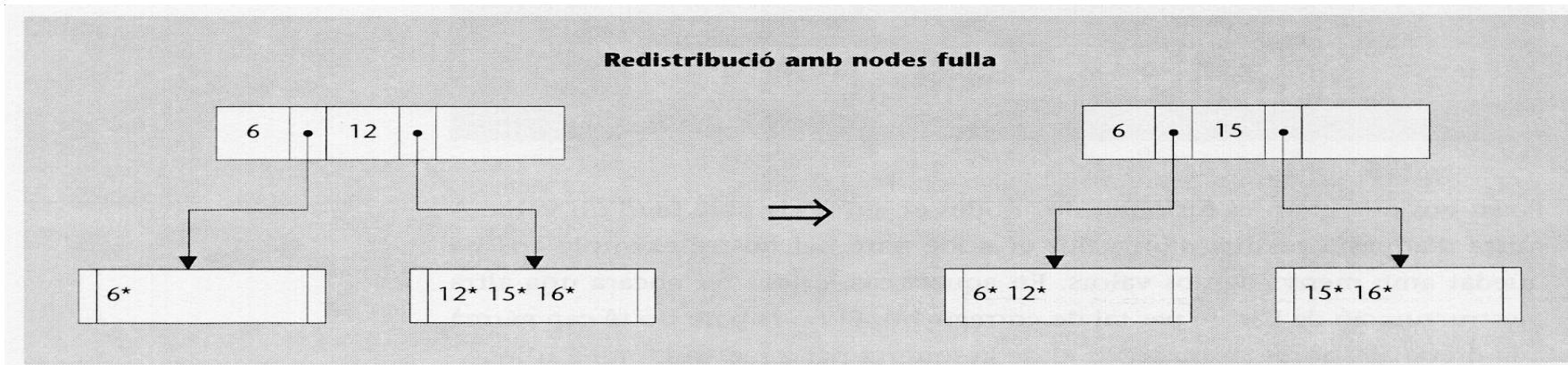
Arbres B+ Supressions

Donat el següent arbre suprimir el 12

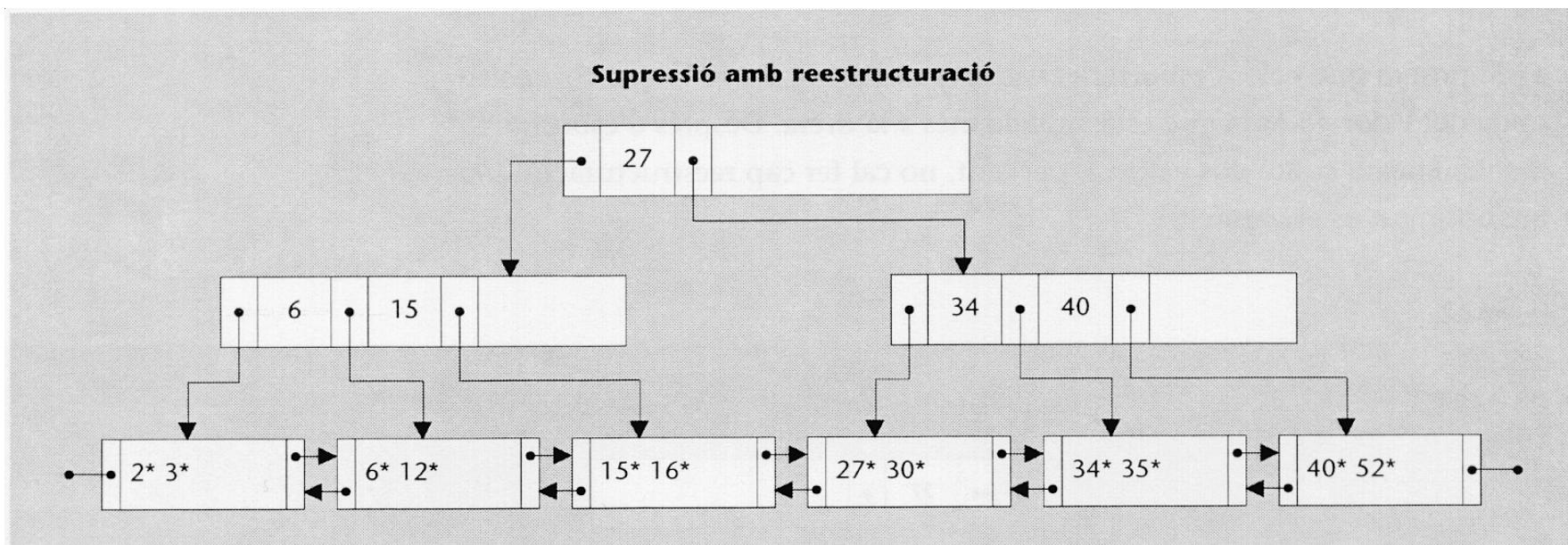


Arbres B+ Supressions

- **Redistribució al nivell fulla:** Supressió del 8 del darrer arbre



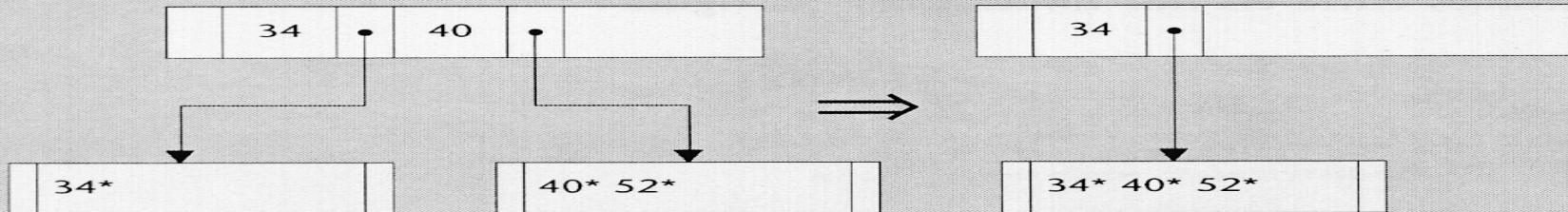
- Arbre resultat



Arbres B+ Supressions

- **Doble fusió:** Supressió del 35

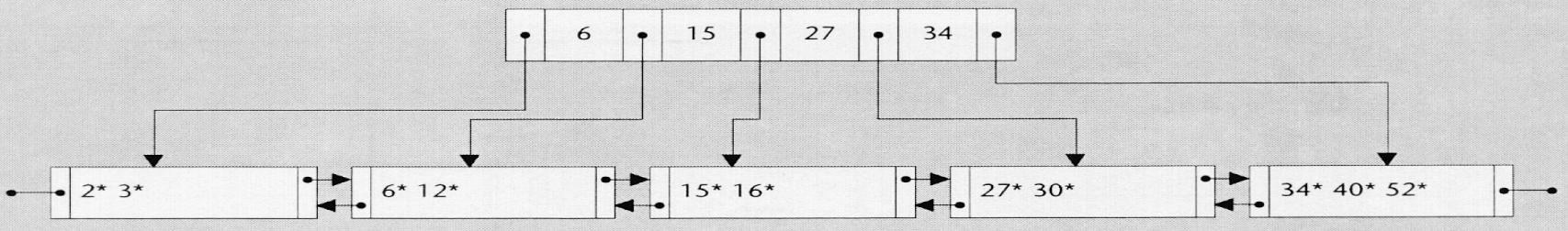
Fusió de nodes fulla



Fusió de nodes no-fulla

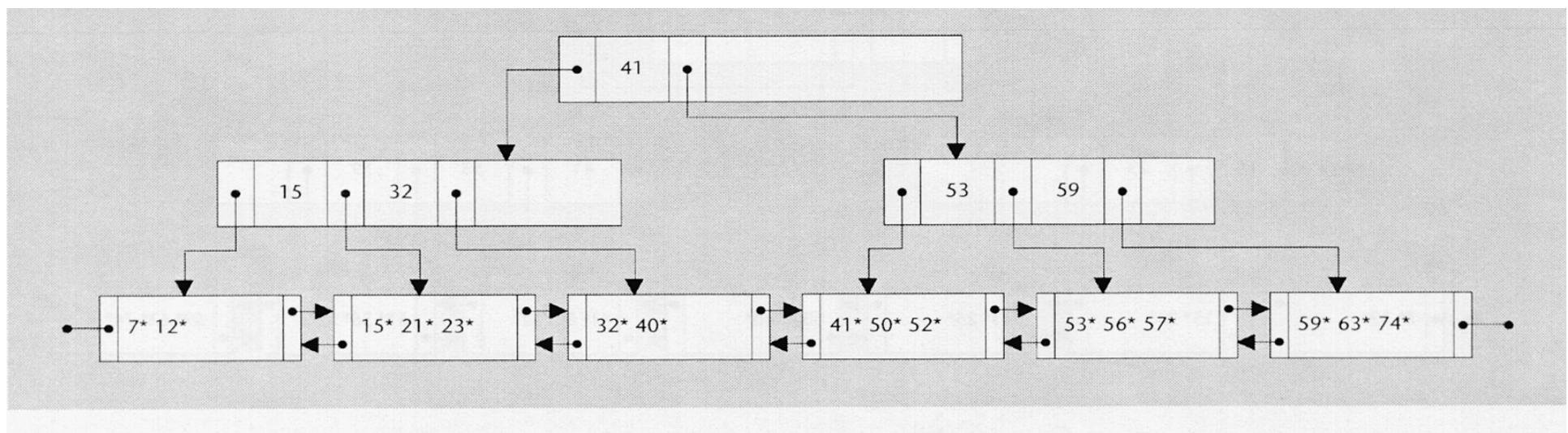
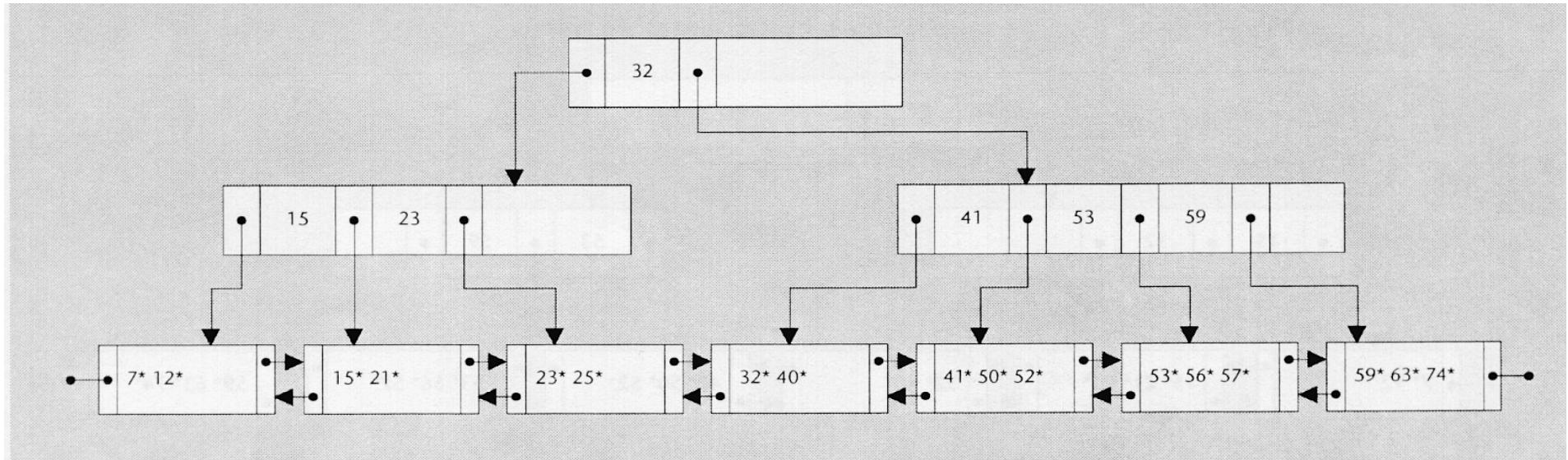


Supressió amb reestructuració i pèrdua d'un nivell



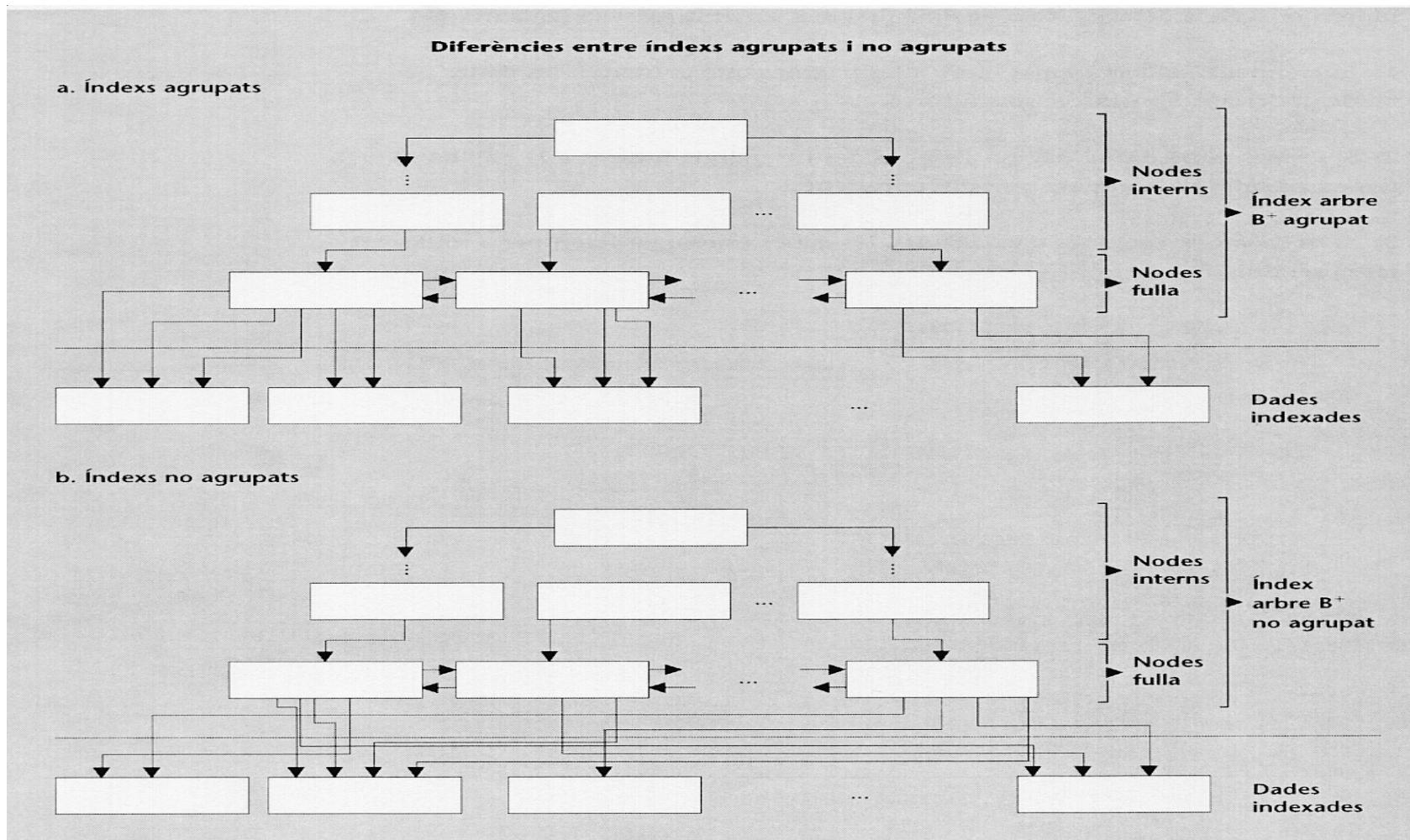
Arbres B+ Supressions

- Redistribució de nodes no fulla i fusió: Supressió a l'arbre següent del 25



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

- Cost:
 - Accés directe per valor:
 - si no repetits :
 - Cost de l'accés per valor en un índex no agrupat
 - Si repetits
 - Cost de l'accés per valor en un índex no agrupat / f_b
 - Accés seqüencial per valor
 - Cost de l'accés per valor en un índex no agrupat / f_b
- 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

- **Implementació dels accessos directes**

Empleats(nemp, nom, despatx, sou)

Arbre B⁺ sobre despatx

Arbre B⁺ sobre sou

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

Intersecció de RIDs

- Obtenir tots els RIDs tals que despatx = 150
- Obtenir tots els RIDs tals que sou = 200000
- Intersecció entre els dos conjunts

En general bon rendiment, però si molts 150, 200000 i pocs les dues condicions, inefficient

Índex (multiatribut)

Mateixa estructura però v seran llistes de elements [v₁, v₂, ...]

És defineix una relació d'ordre lineal entre les llistes:

s'ordenen primer segons el primer element
segon segons el segon element ...

[150, 200000], [150, 300000], [200, 100000], [200, 150000] ...

Implementació dels accessos per diversos valors

- **Implementació dels accessos seqüencials i mixtos**

- **Índex multiatribut**

Empleats(nemp, nom, despatx, sou)

Índex multiatribut sobre [despatx, sou]

[150, 200000], [150, 300000], [200, 100000], [200, 200000] ...

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

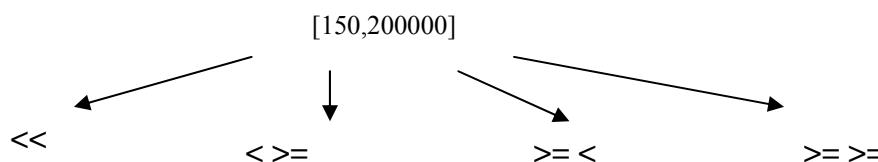
```
SELECT *  
FROM Empleats  
WHERE despatx = 150  
AND sou > 200000
```

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

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

- **Índex multiatribut multidimesional: no ordre lineal!!**

Ex: Quad-tree

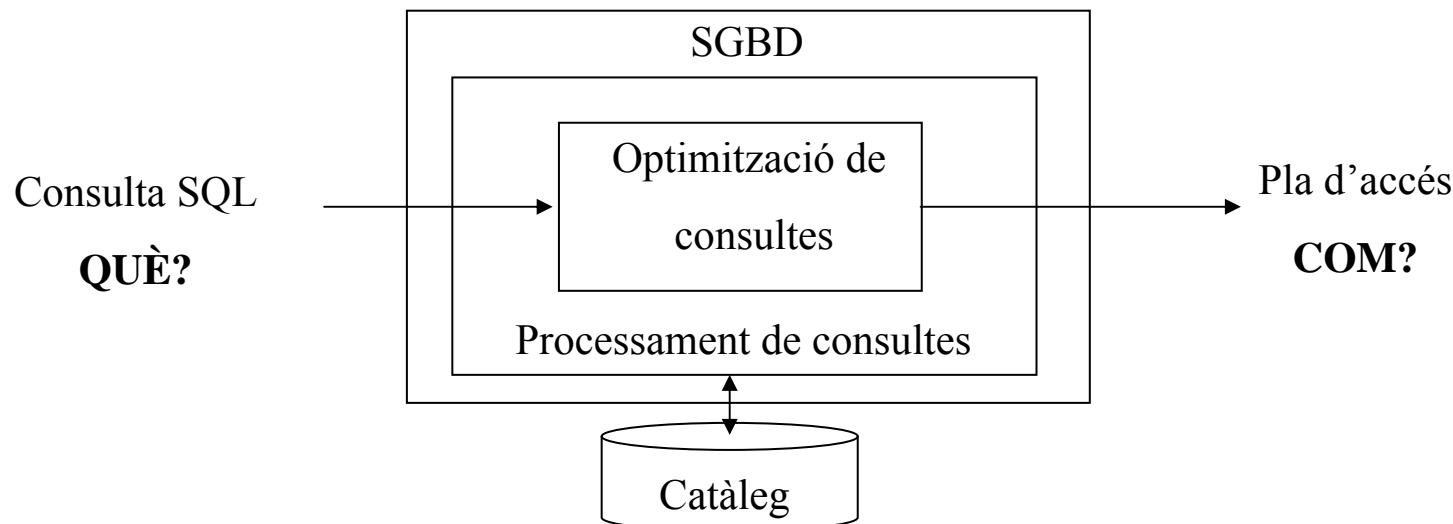


Els índexs en un SGBD concret

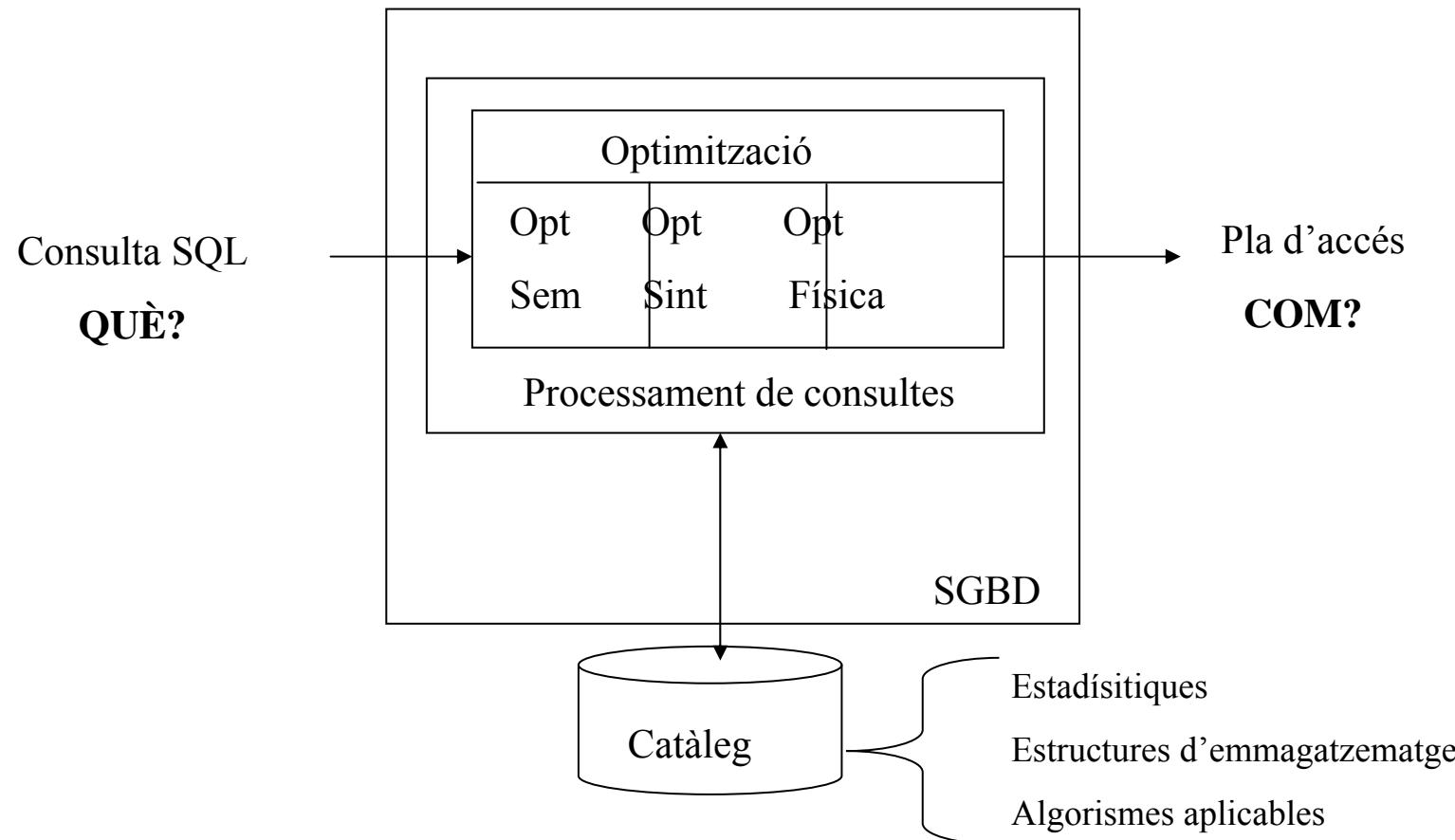
- Per implementar accessos per valor, directes o seqüencials, la majoria d'SGBDs proporcionen índexs arbre B+ per un atribut que poden ser agrupats o no
- Creació d'un índex:
Empleats(nemp, nom, despatx, sou)
CREATE INDEX index_despatx ON Empleats(despatx)
- Per a que l'índex sigui descendant:
CREATE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

Introducció a l'optimització

- El procés d'optimització de consultes està inclòs dins d'un procés més general que es coneix com **processament de consultes**
- El processament de consultes consisteix en la transformació d'una consulta (expressada en SQL) en un **conjunt d'instruccions de baix nivell**
- El conjunt d'instruccions de baix nivell constitueix una **estratègia** per accedir a les dades **amb un consum de recursos mínim**
- L'estratègia que minimitza el consum de recursos rep el nom de **pla d'accés de la consulta**



El procés d'optimització de consultes



El procés d'optimització de consultes

- Hem estudiat els dos tipus d'optimització que efectuen els SGBD relacionals del mercat. Existeix, però, un tercer tipus d'optimització possible conegut amb el nom **d'optimització semàntica**
- L'objectiu de l'optimització semàntica d'una consulta és simplificar les consultes formulades pels usuaris mitjançant l'ús de la informació que proporcionen les regles d'integritat que s'hagin definit sobre la BD i les lleis de la lògica
- Exemple:

```
SELECT e2.nom_empl  
FROM empleats e1, empleats e2  
WHERE e1.num_empl= e2.cap AND e2.salari>e1.salari;
```

RI: Cap empleat pot guanyar més que el seu superior

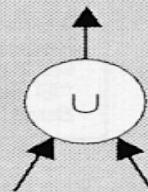
L'àlgebra relacional en el processament de consultes

- Donada una consulta formulada en llenguatge SQL, és possible trobar diverses consultes equivalents expressades en àlgebra relacional
- Cada consulta en àlgebra relacional es pot representar gràficament mitjançant una estructura en forma d'arbre que es coneix amb el nom **d'arbre sintàctic de la consulta**
- L'arbre sintàctic d'una consulta és una estructura en forma d'arbre que correspon a una expressió en àlgebra relacional en la qual les diferents parts de l'arbre representen els elements següents:
 - Les fulles de l'arbre representen les relacions de base (taules SQL) que intervenen a la consulta
 - Els nodes intermedis són les operacions d'àlgebra relacional que intervenen a la consulta original. L'aplicació de cadascuna d'aquestes operacions dóna lloc a una relació intermèdia
 - L'arrel de l'arbre constitueix la resposta a la consulta que ha estat formulada

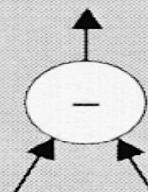
L'àlgebra relacional en el processament de consultes

Representació gràfica de les operacions de l'àlgebra relacional

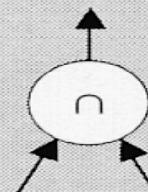
Unió



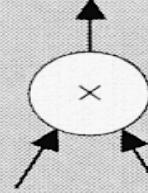
Diferència



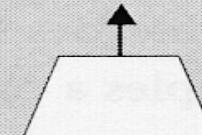
Intersecció



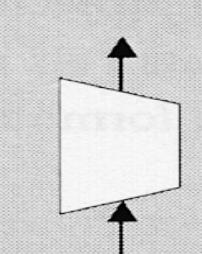
Producte cartesià



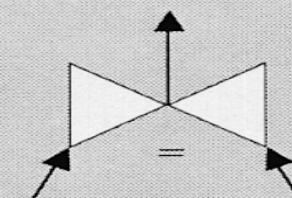
Projecció



Selecció



Combinació natural

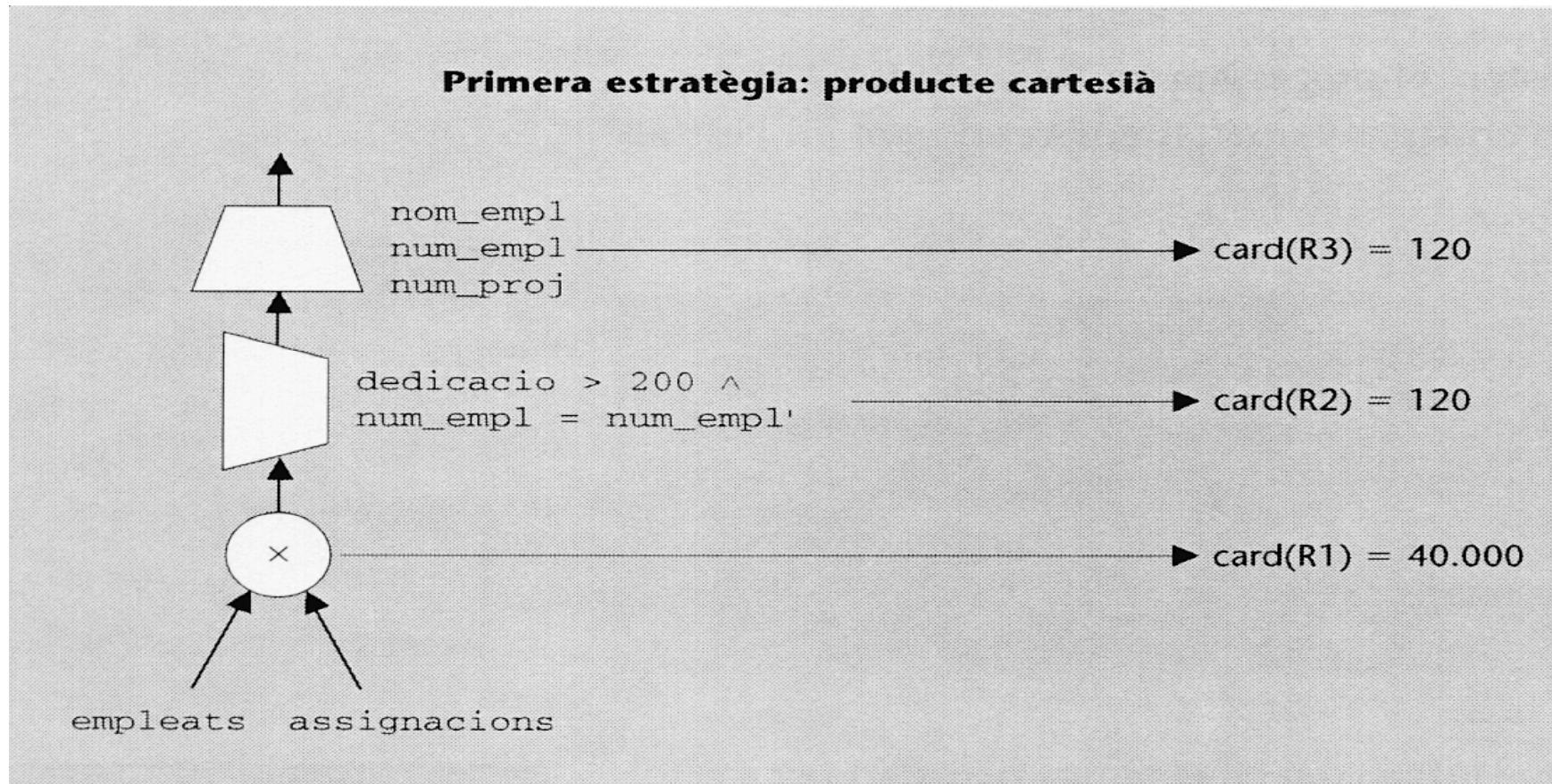


L'àlgebra relacional en el processament de consultes

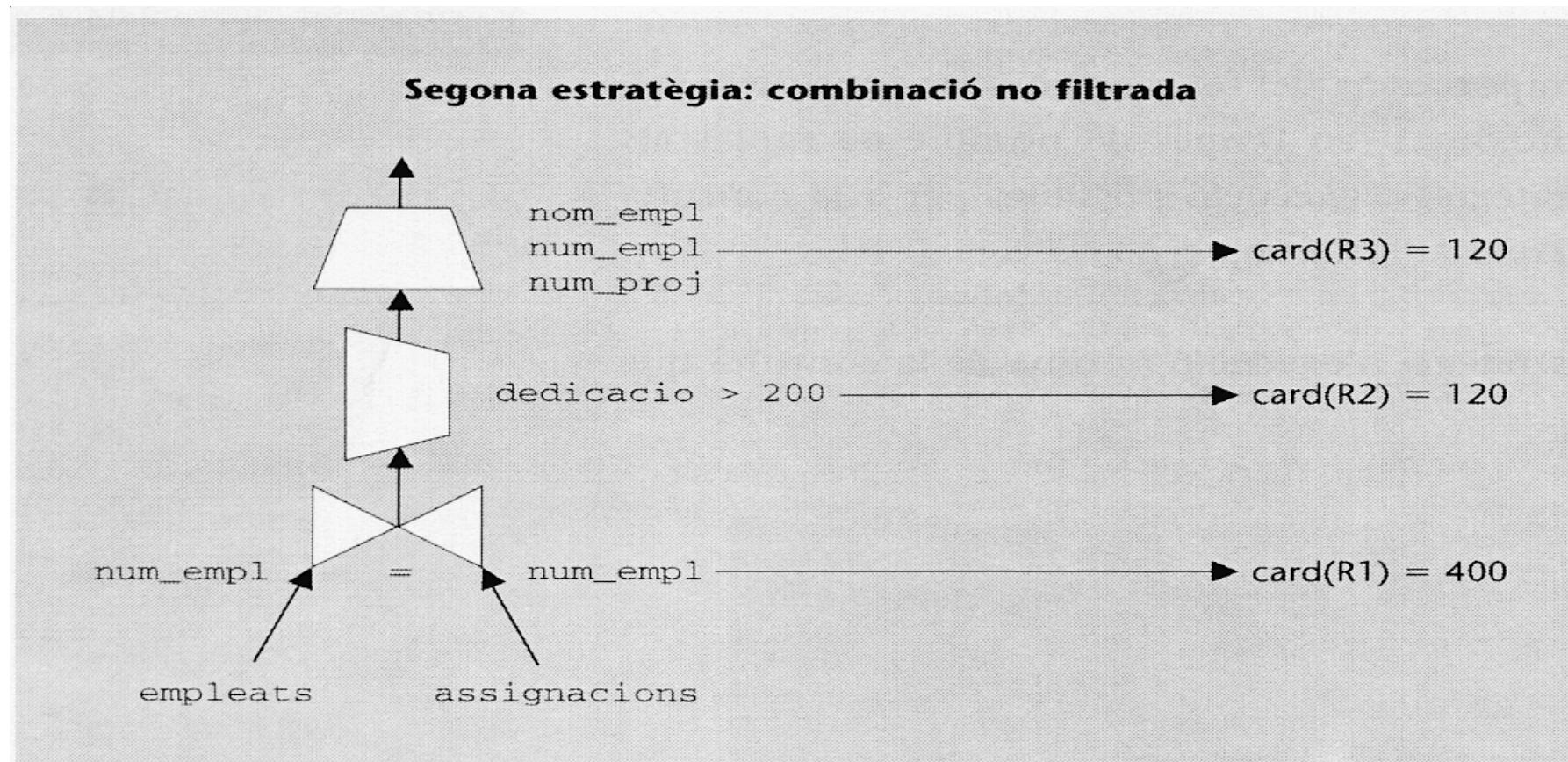
- Imaginem que tenim una BD amb les taules següents:
 - empleats(num_empl, nom_empl, cat_laboral, divisió, sou, cap)
{num_empl} és la clau primària
{cap} és clau forana que referencia a empleats
 - projectes(num_proj, nom_proj, producte, duració)
{num_proj} és la clau primària
 - assignacions(num_empl, num_proj, dedicació)
{num_empl, num_proj} és la clau primària
{num_empl} és clau forana que referencia a empleats
{num_proj} és clau forana que referencia a projectes
- Suposem que un usuari vol saber les dades dels empleats que dediquen més de 200 hores a un projecte. En concret, l'usuari vol trobar el nom i el número d'aquests empleats i els números dels projectes als quals estan assignats:

```
SELECT e.nom_empl, e.num_empl, a.num_proj
FROM empleats e, assignacions a
WHERE e.num_empl=a.num_empl AND a.dedicacio>200;
```

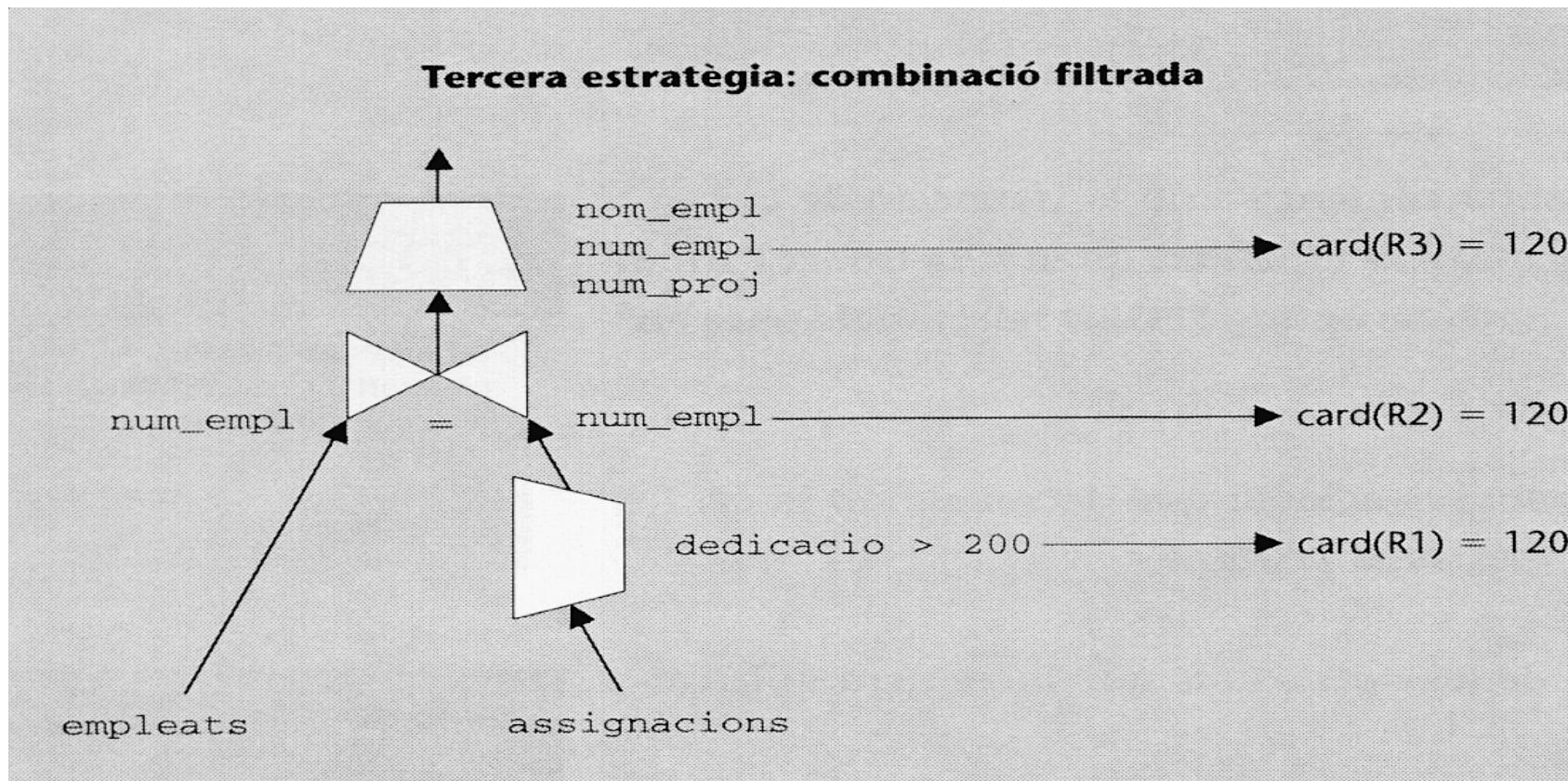
El procés d'optimització de consultes



El procés d'optimització de consultes

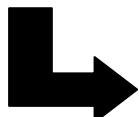


El procés d'optimització de consultes



El procés d'optimització de consultes

- Acabem de veure que l'ordre d'execució de les operacions d'una consulta repercutex directament en el cost d'execució de la consulta
- És important trobar el millor ordre d'execució possible per a les operacions de cada consulta:
 - **Problema:** trobar el millor ordre d'execució de les operacions requereix molt de temps si les consultes no són simples
 - **Solució:** els SGBD relacionals apliquen **mètodes heurístics** per trobar una estratègia d'execució raonablement òptima per resoldre una consulta determinada
- L'**optimització sintàctica** d'una consulta és el procés que determina un ordre d'execució raonablement òptim de les operacions que inclou una consulta
- L'optimització sintàctica de la consulta comença amb la traducció de la consulta original a un arbre sintàctic equivalent i finalitza quan s'ha trobat l'arbre sintàctic òptim
- Per trobar l'arbre sintàctic òptim l'SGBD aplica el mètode heurístics següents:
 - Executar les operacions de l'àlgebra relacional que disminueixen la cardinalitat dels resultats intermedis el més aviat possible



Endarrerir al màxim l'execució de les operacions
que incrementen la cardinalitat dels resultats intermedis

El procés d'optimització de consultes

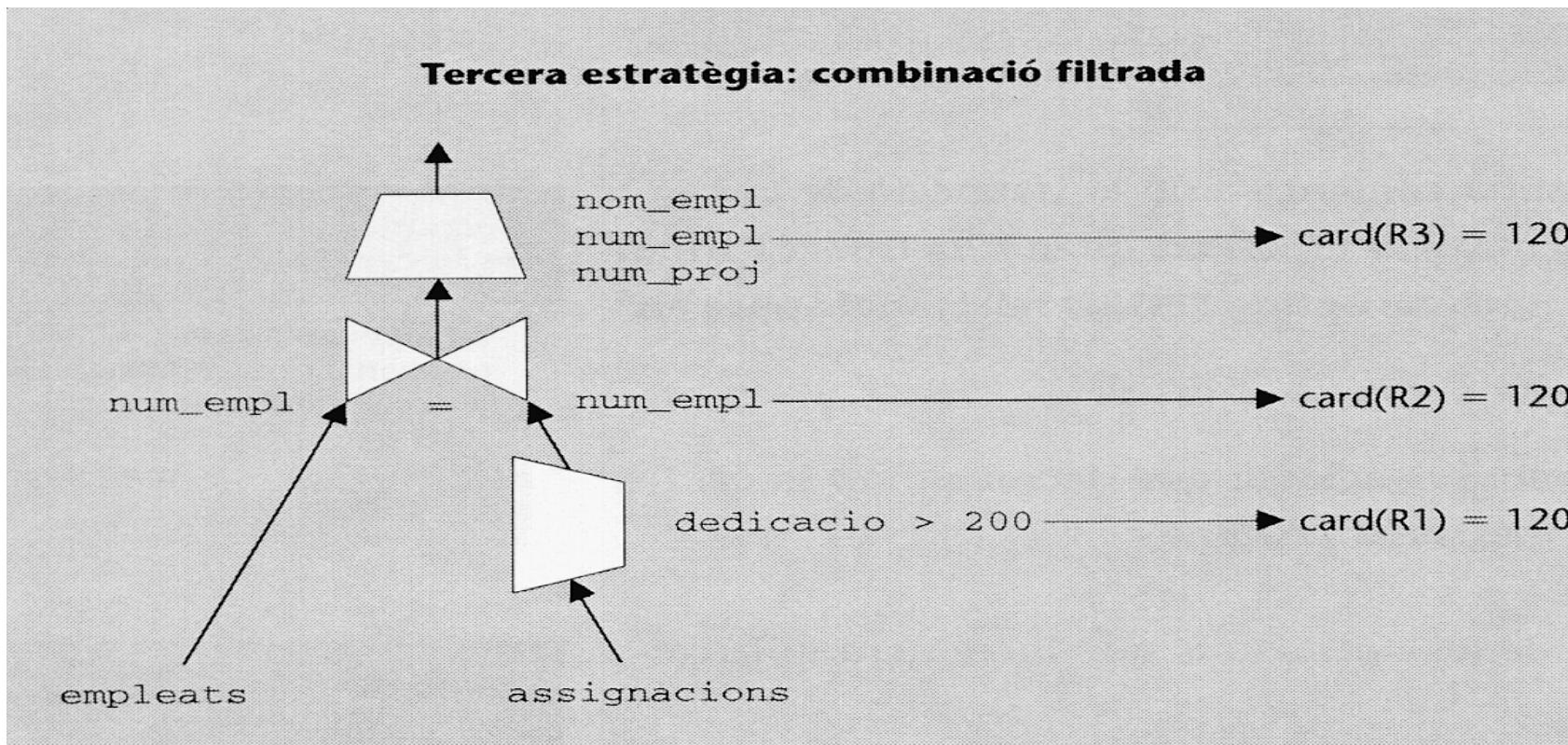
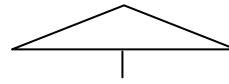
- L'optimització sintàctica simplement reordena les operacions de l'àlgebra relacional que figuren a la consulta; l'avaluació del cost de l'arbre sintàctic òptim obtingut com a resultat de l'optimització sintàctica es fa posteriorment, en l'etapa d'**optimització física** de la consulta
- L'objectiu de l'optimització física d'una consulta és avaluar el cost total de l'execució de l'arbre sintàctic òptim associat a una consulta d'un usuari. Aquest cost és la suma dels costos de totes les operacions que figuren a l'arbre
- El cost de cada operació inclou tant el cost d'execució de l'operació en si com el d'escriure el resultat de l'operació en una relació intermèdia
- Per aproximar el cost de les diferents operacions, l'SGBD necessita conèixer les dades següents:
 - Les estadístiques de la BD, que serveixen per a estimar la longitud dels resultats intermedis i la longitud de les relacions implicades en l'execució de les diferents operacions. Les estadístiques s'emmagatzemem al catàleg de la BD i és missió de l'ABD mantenir-les actualitzades:
 - Exemples de dades estadístiques: $\text{card}(R)$, longitud dels tuples d'una relació, nombre de valors diferents que prenen els atributs d'una relació

El procés d'optimització de consultes

- Les estructures d'emmagatzematge definides a l'esquema intern. D'aquestes estructures, és important tenir dades sobre els aspectes següents:
 - A quins fitxers s'han emmagatzemat les relacions de base i a on es troben aquests fitxers
 - Si s'han definit estructures d'agrupació per als tuples d'una relació segons el valor d'un o més atributs
 - Si hi ha altres camins d'accés a les dades com, per exemple, índexs secundaris sobre atributs que no són part dels criteris d'agrupació
- Els algorismes d'implementació de les operacions de l'àlgebra relacional disponibles
 - Aquestes dades permeten que l'SGBD pugui decidir quins algorismes són aplicables i quins no
 - Per cada algorisme que sigui aplicable, l'SGBD haurà de avaluar-ne el seu cost i triar l'algorisme que tingui el cost més baix

El procés d'optimització de consultes

- Exemple:
assignacions(num_empl, num_proj, dedicació)



- 1) algorisme seqüencial
- 2) algorisme que fes ús de l'índex