

Algebra relațională

■ Operatori de bază:

- Proiectia (π) Elimină attributele nedorite ale unei relații
- Selectie (σ) Selectează o submulțime de tupluri ale unei relații.
- Prod cartezian (\times) Permite combinarea a două relații.
- Diferenta ($-$) Tuplurile ce aparțin unei relații dar nu aparțin celeilalte
- Reuniunea (\cup) Tuplurile aparținând ambelor relații

■ Operatori adiționali:

- Intersecția, join, câtul, redenumirea: nu sunt esențiale dar sunt foarte folosite.

■ Deoarece fiecare operator returnează o relație, **operatorii pot fi compuși** (algebra este “închisă”).

Extensii ale operatorilor algebrici relaționali

- Generalizarea proiecției
- Funcții de agregare
- Left/Right/Full Outer Join
- Modificarea bazei de date

Generalizarea proiecției

- Operatorul *proiecție* este extins prin permiterea utilizării funcțiilor aritmetice în lista de definire a proiecției.

$$\pi_{F_1, F_2, \dots, F_n}(R)$$

- R poate fi orice expresie din algebra relatională
- Fiecare dintre F_1, F_2, \dots, F_n sunt expresii aritmetice ce implică attribute din R și constante.

Funcții de agregare

- **Funcția de agregare** returnează ca rezultat o valoare pe baza unei colecții de valori primite ca input

avg: valoarea medie

min: valoarea minimă

max: valoarea maximă

sum: suma

count: numărul înregistrărilor

- **Operator de agregare în algebra relațională**

$$G_1, G_2, \dots, G_n \quad \mathcal{G}_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(R)$$

- R poate fi orice expresie din algebra relațională
 - G_1, G_2, \dots, G_n e o listă de attribute pe baza cărora se grupează datele (poate fi goală)
 - Fiecare F_i este o funcție de agregare
 - Fiecare A_i este un nume de atribut

Agregarea – Example

Relație R :

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

$\rho_{\text{sum}(C)}(R)$

sum(C)
27

- Rezultatul agregarii nu are un nume
 - se pot folosi operatorii de redenumire
 - se poate permite redenumirea ca parte a unui operator de agregare

Agregarea – Example

Relație R :

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

$\rho_{A, \text{sum}(C)}(R)$

A	sum(C)
α	14
β	13

- Rezultatul agregarii nu are un nume
 - se pot folosi operatorii de redenumire
 - se poate permite redenumirea ca parte a unui operator de agregare

Left/Right/Full Outer Join

■ Extensii ale operatorului join natural care împiedică pierderea informației:

- Left Outer Join
- Right Outer Join
- Full Outer Join



■ Realizează joncțiunea și apoi adaugă la rezultat tuplurile dintr-una din relatii (din stânga, dreapta sau ambele părți ale operatorului) care nu sunt conectate cu tupluri din celaltă relație.

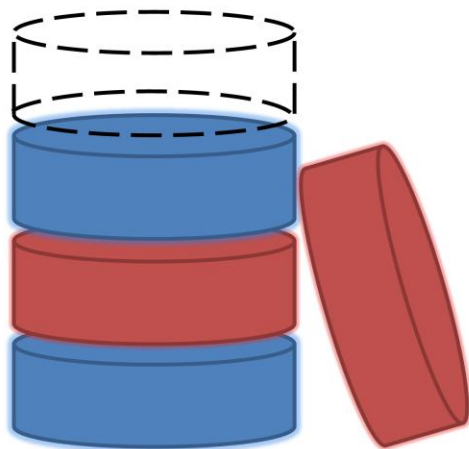
■ Utilizează valoarea *null*:

- *null* semnifică faptul că valoarea e necunoscută sau nu există
- Toate comparațiile ce implică *null* sunt (*simply spus*) **false** prin definiție.

Modificarea bazei de date

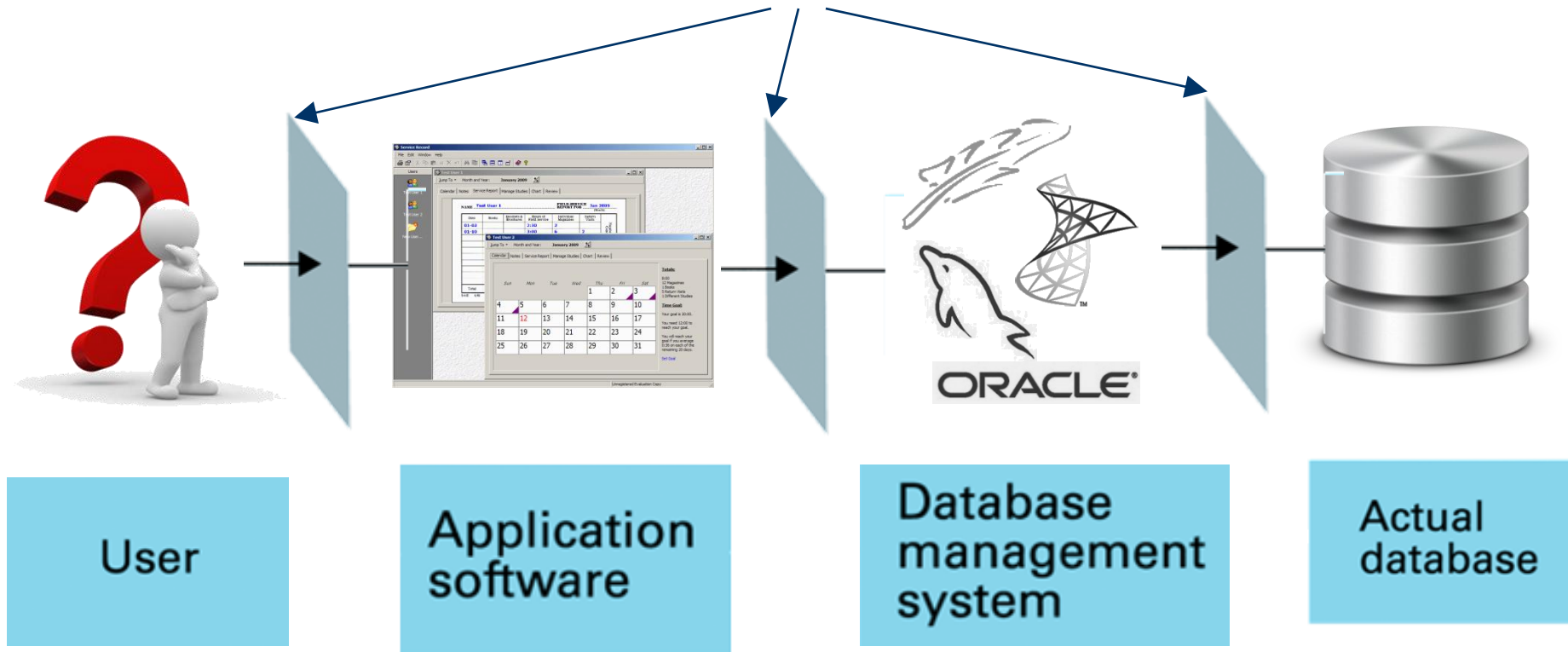
- Conținutul bazei de date poate fi modificat folosind următorii operatori:
 - Ștergere $R \leftarrow R - E$
 - Inserare $R \leftarrow R \cup E$
 - Modificare $R \leftarrow \pi_{F_1, F_2, \dots, F_n}(R)$
- Toți acești operatori sunt exprimați prin utilizarea operatorului de atribuire.

Structura fizică a bazelor de date



Nivelele de abstractizare

Nivele diferite
de abstractizare



STUDENT

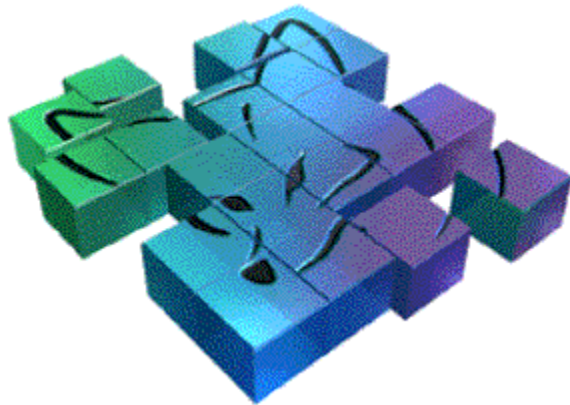
Name

Date of birth

Gender

CNP

Group



Structura fizică

Faculty.dbc

42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

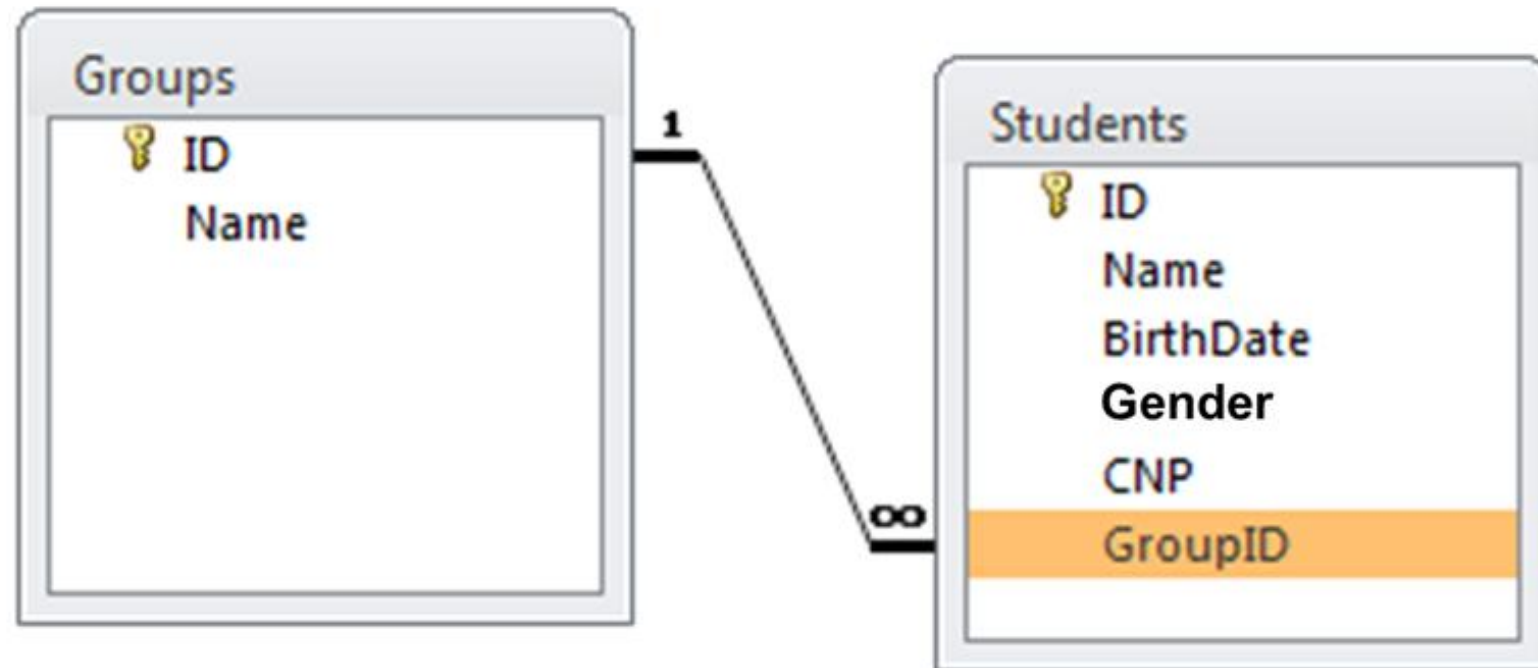
Students.dbf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

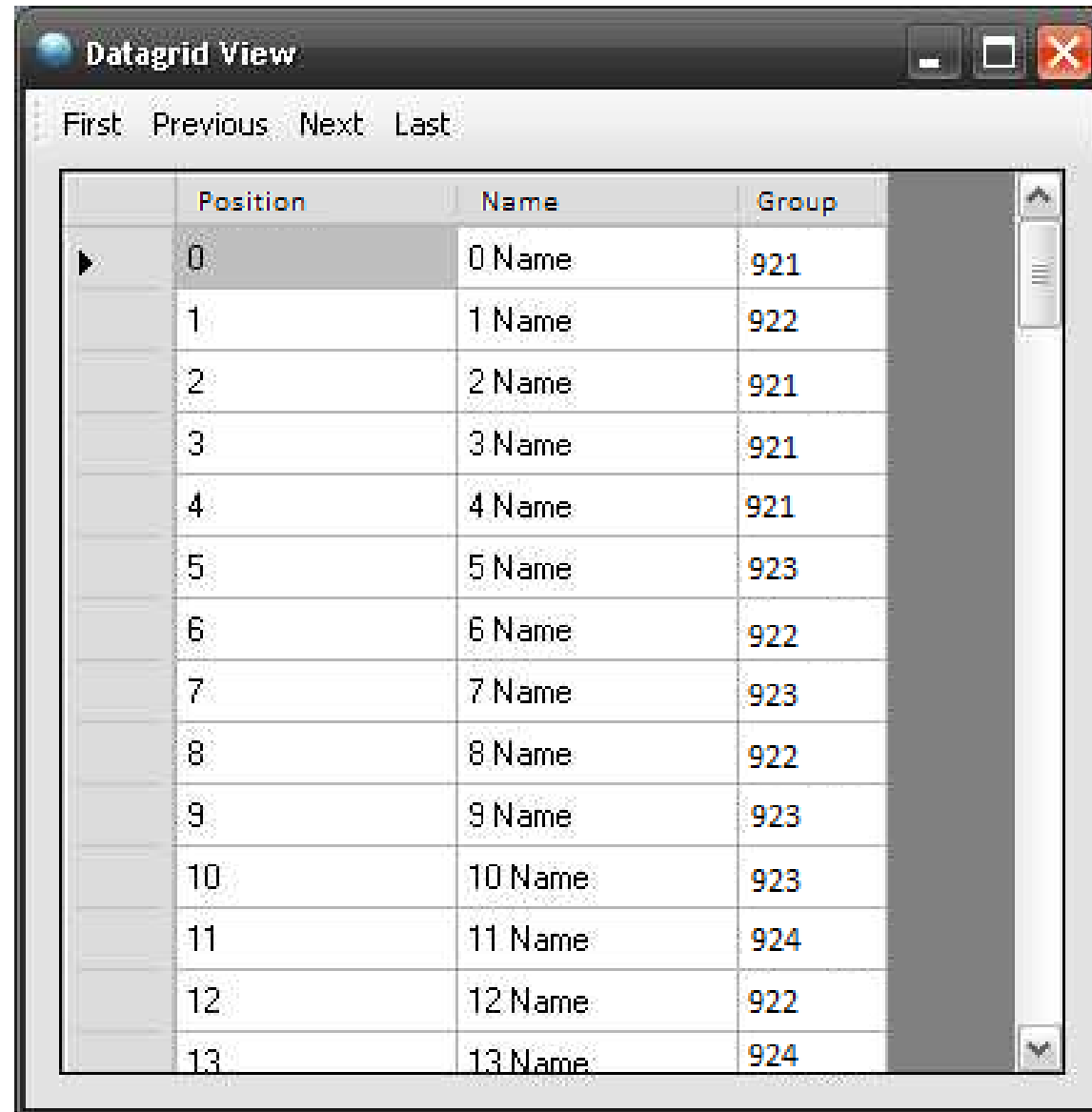
Groups.dbf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

Structura conceptuală



Vizualizare informații de către utilizator



Datagrid View

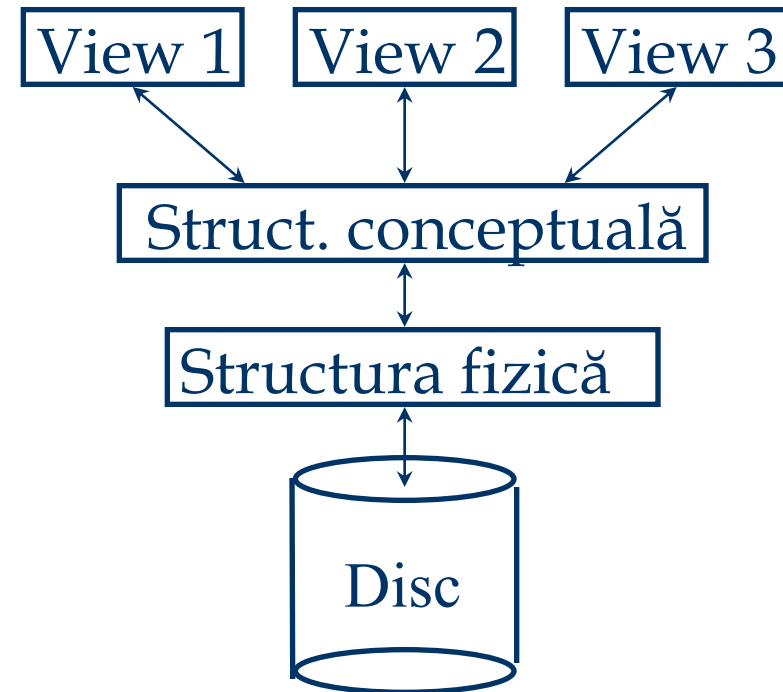
First Previous Next Last

	Position	Name	Group
▶	0	0 Name	921
	1	1 Name	922
	2	2 Name	921
	3	3 Name	921
	4	4 Name	921
	5	5 Name	923
	6	6 Name	922
	7	7 Name	923
	8	8 Name	922
	9	9 Name	923
	10	10 Name	923
	11	11 Name	924
	12	12 Name	922
	13	13 Name	924

Nivelele de abstractizare

■ Mai multe structuri externe (views), câte o singură structură conceptuală (logică) și o structură fizică (internă).

- *Views* – cum văd utilizatorii datele.
- *Conceptual* - modelul logic compus din relații, attribute, etc
- *Fizic* - fișierele de date și indecși

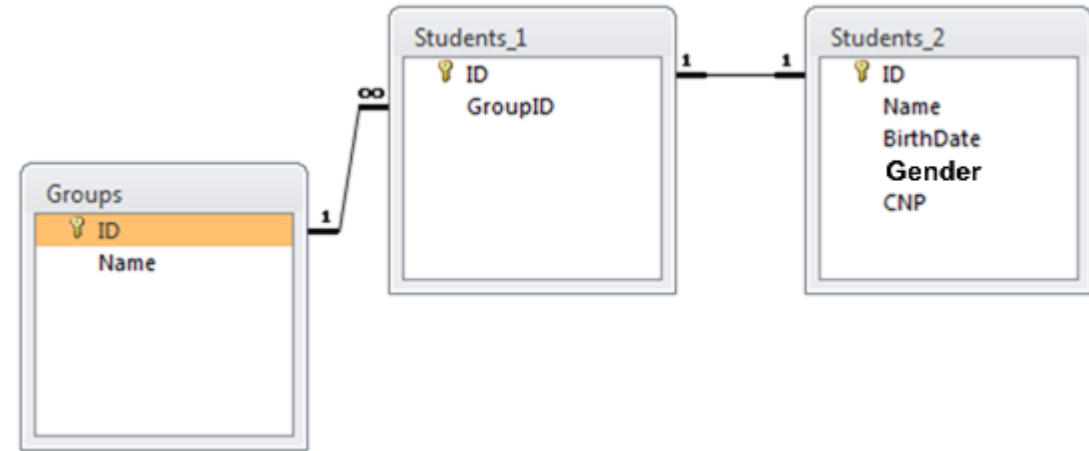
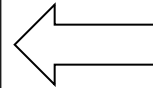


Independența logică a datelor

Datagrid View

First Previous Next Last

	Position	Name	Group
▶	0	0 Name	921
	1	1 Name	922
	2	2 Name	921
	3	3 Name	921
	4	4 Name	921
	5	5 Name	923
	6	6 Name	922
	7	7 Name	923
	8	8 Name	922
	9	9 Name	923
	10	10 Name	923
	11	11 Name	924
	12	12 Name	922
	13	13 Name	924

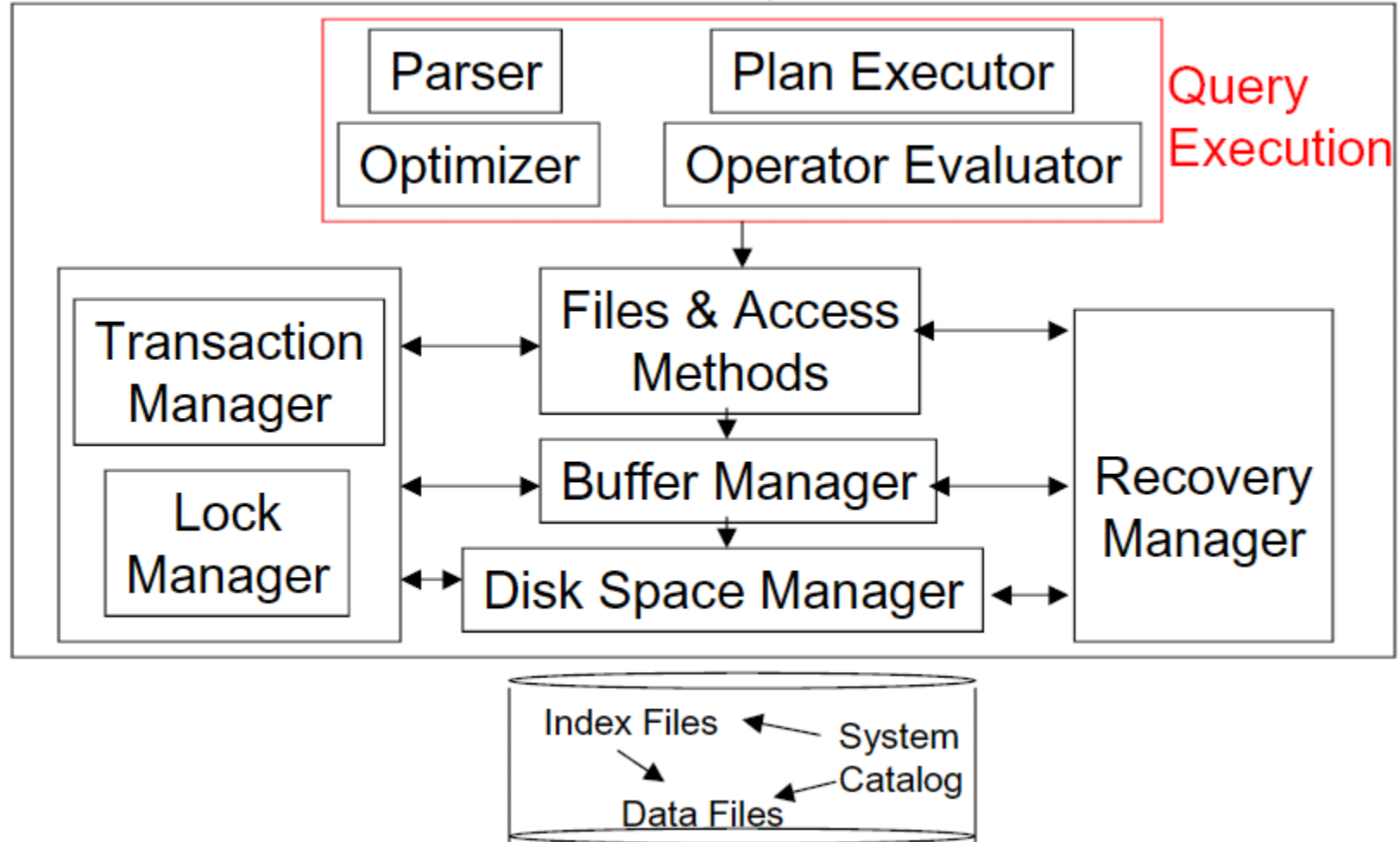


Faculty.mdf

Faculty.dbc

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

Structura unui SGBD



Structura fizică a fișierelor BD

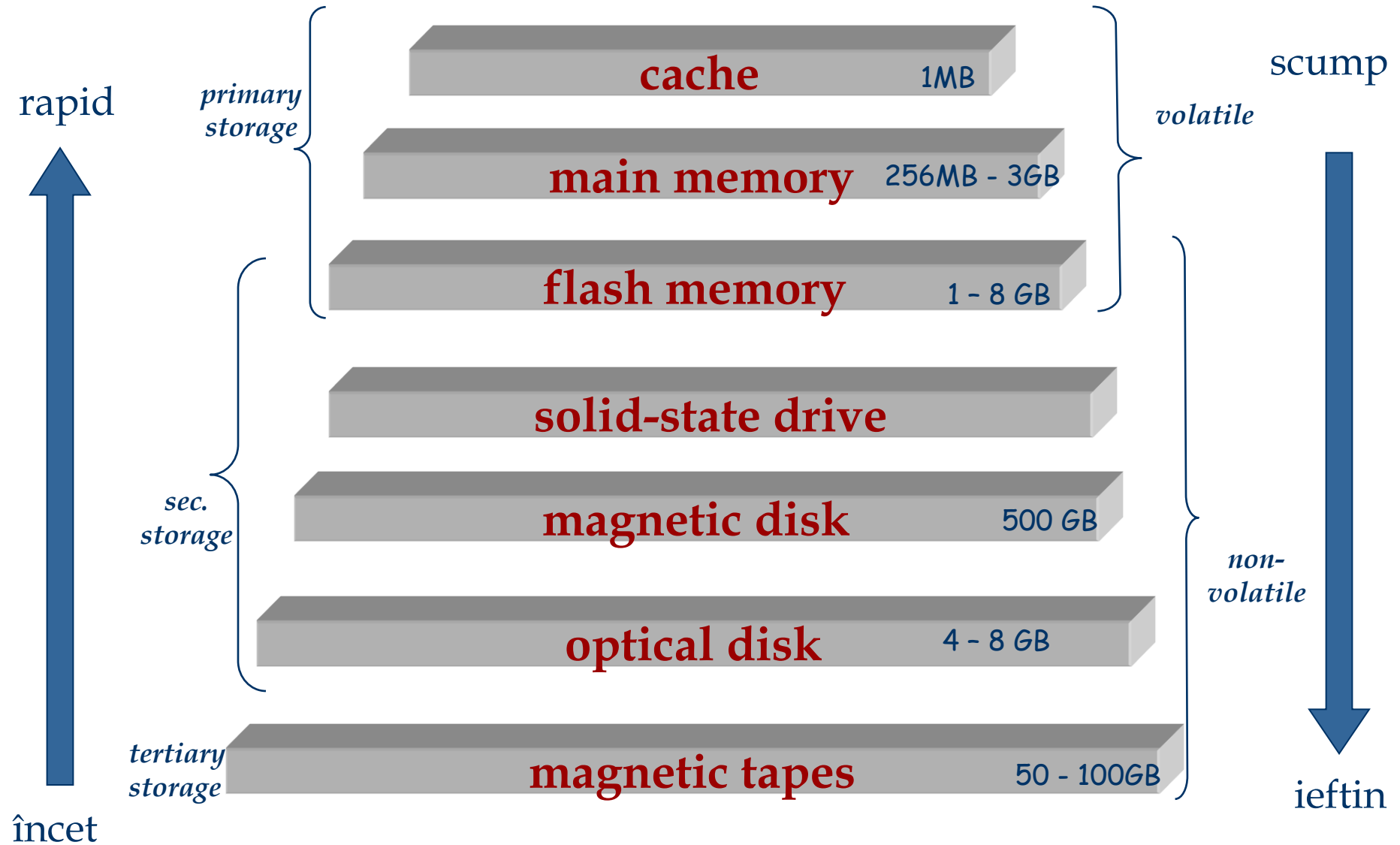
- SGBD-urile stochează informația pe disc magnetic
- Acest lucru are implicații majore în proiectarea unui SGBD!
 - **READ**: transfer date de pe disc în memoria internă
 - **WRITE**: transfer date din memoria internă pe disc

Ambele operații sunt costisitoare, comparativ cu operațiile *in-memory*, deci trebuie planificate corespunzător!

De ce nu stocăm totul în memoria internă?

- Răspuns (tipic):
 - Costă prea mult
 - Memoria internă este volatilă (datele trebuie să fie persistente)
- Procedură tipică(“ierarhie de stocare”)
 - RAM – pentru datele utiliz. curent (*primary storage*)
 - *Hard-disks* – pentru baza de date (*secondary storage*)
 - Bandă – pentru arhivarea versiunilor anterioare ale datelor (*tertiary storage*)

Ierarhia mediilor de stocare

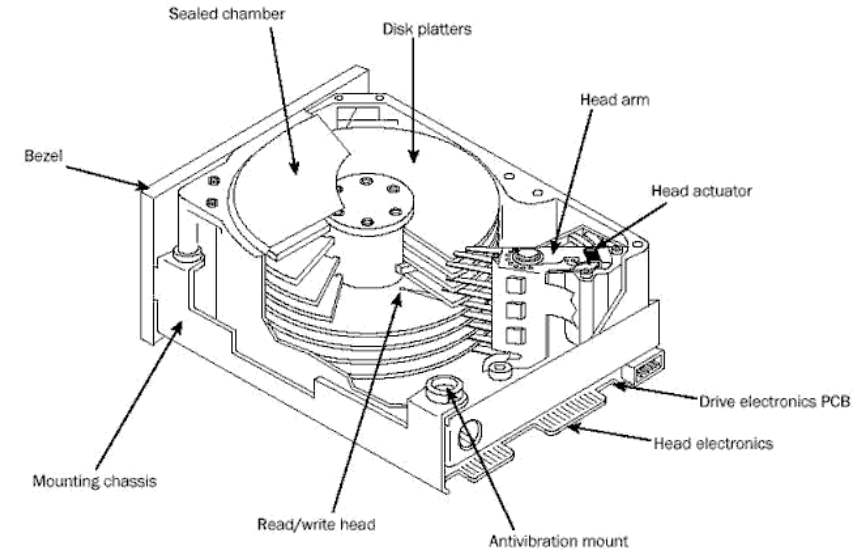


Legea lui Moore

- Gordon Moore: *“Integrated circuits are improving in many ways, following an exponential curve that doubles every 18 months”*
 - Viteza procesoarelor
 - Numărul de biți de pe un chip
 - Numărul de octeți (*bytes*) pe un hard disk
 - Parametrii ce NU urmează legea lui Moore:
 - Viteza de accesare a datelor în memoria internă
 - Viteza de rotație a discului
- ⇒ Latența devine progresiv mai mare
- Timpul de transfer între nivelele ierarhiei mediilor de stocare este tot mai mare în comparație cu timpul de calcul

Discuri magnetice

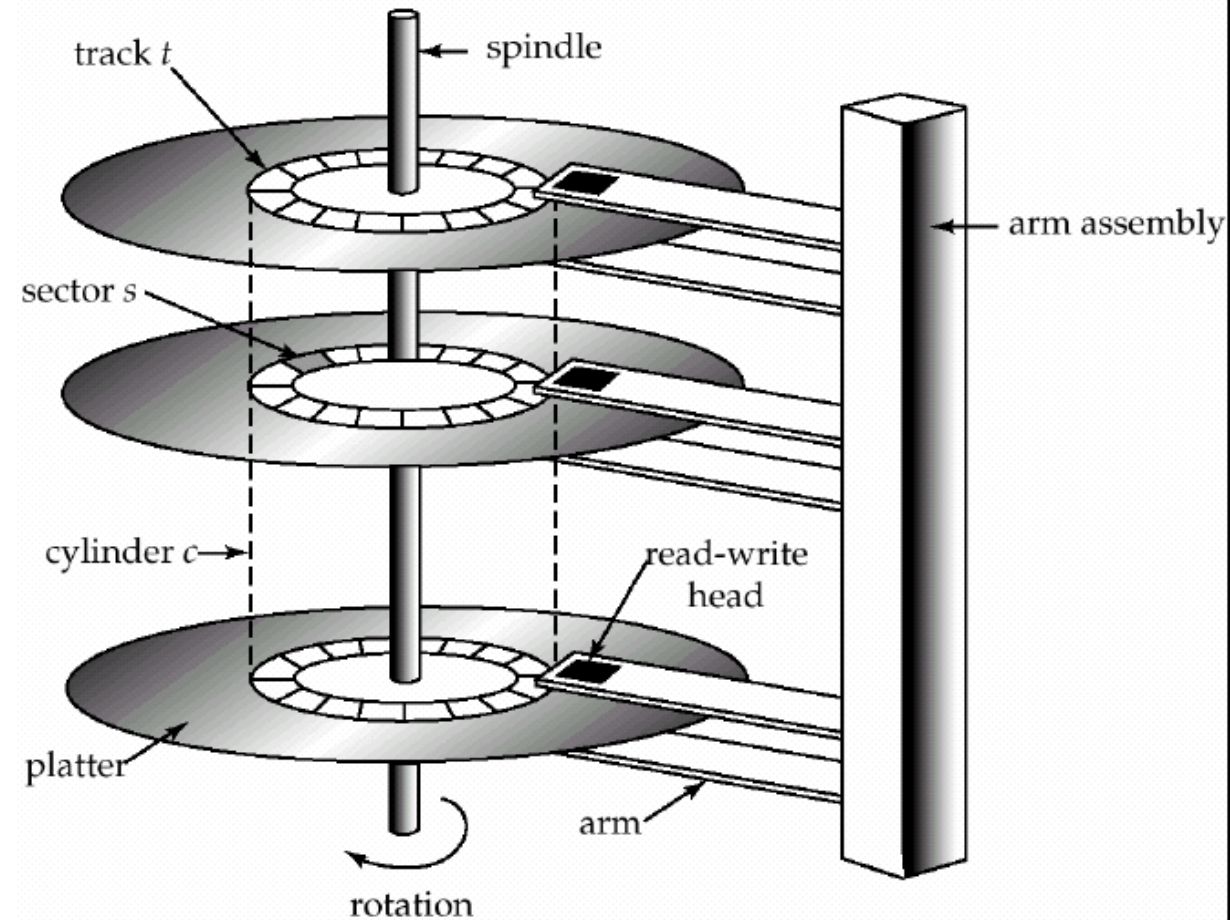
- Utilizat ca mediu de stocare secundar
- Avantaj major asupra benzilor: *acces direct*
- Datele sunt stocate și citite în unități numite **blocuri** sau **pagini**.
- Spre deosebire de memoria internă, timpul de transfer blocurilor/paginilor variază în funcție de poziția acestora pe disc.



! Poziția relativă a paginilor pe disc
are un impact major asupra performanței unui SGBD!

Componentele unui disc dur (*hard disk*)

- Rotația platanelor (90rps)
- Ansamblu de brațe ce se deplasează pentru poziționarea capului magnetic pe pista dorită. Pistele aflate la aceeași distanță de centrul platanelor formează un **cilindru** (imaginar!).
- Un singur cap citește/ scrie la un moment dat.
- Un **bloc** e un multiplu de **sectoare** (care e fix).



Accesarea unei pagini (bloc)

- Timp de acces (citire/scriere) a unui bloc:
 - seek time* (mutare braț pentru poziționarea capului de citire/scriere pe pistă)
 - rotational delay* (timp poziționare bloc sub cap)
 - transfer time* (transfer date de pe/pe disc)
- *Seek time* și *rotational delay* domină.
 - *Seek time* variază între 1 și 20msec
 - *Rotational delay* variază între 0 și 10msec
 - *Transfer rate* e de aproximativ 1msec pe 4KB (pagină)
- Reducerea costului I/O: *reducere seek/rotational delays!*
- Soluții *hardware* sau *software*?

Aranjarea paginilor/blocurilor pe disc

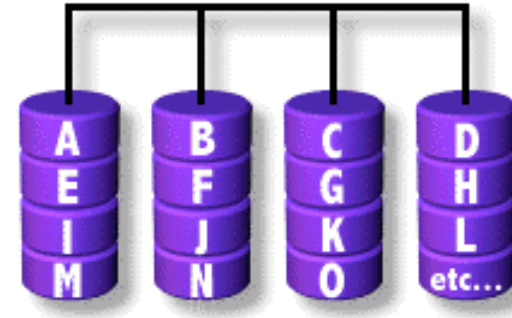
- Conceptul de *next block*:
 - blocuri pe aceeași pistă, urmate de
 - blocuri pe același cilindru, urmate de
 - blocuri pe cilindri adiacenți
- Blocurile dintr-un fișier trebuie dispuse secvențial pe disc (*'next'*), pentru a minimiza *seek delay* și *rotational delay*.
- În cazul unei *scanări secvențiale*, citirea de pagini în avans (*pre-fetching*) este esențială!

RAID

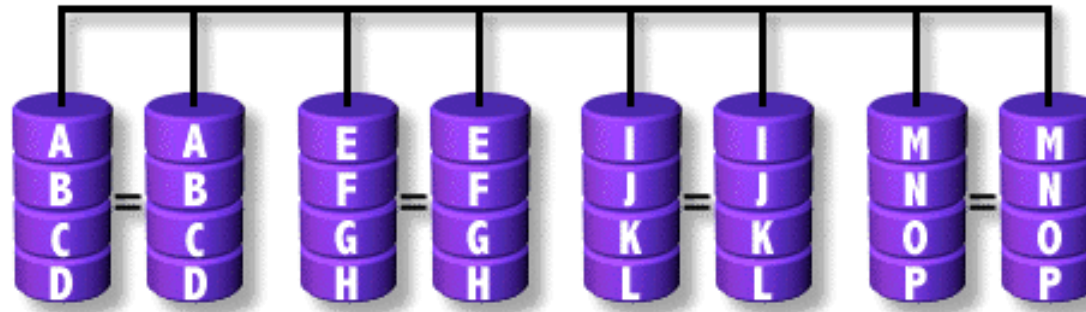
- *Disk Array*: configurație de discuri magnetice ce abstractizează un singur disc.
 - mult mai puțin costisitor; se utilizează mai multe discuri de capacitate mică și ieftine în locul unui disc de capacitate ridicată
- Scop: Creșterea performanței și fiabilității.
- Tehnici:
 - Data striping*: distribuirea datelor pe mai multe discuri (în partiții prestabilite - *striping unit*)
 - Mirroring*: stocarea automată a unei copii a datelor pe alte discuri → redundanță. Permite reconstruirea datelor în cazul unor defecte ale discurilor.

Nivele RAID

- Nivel 0: Fără redundanță

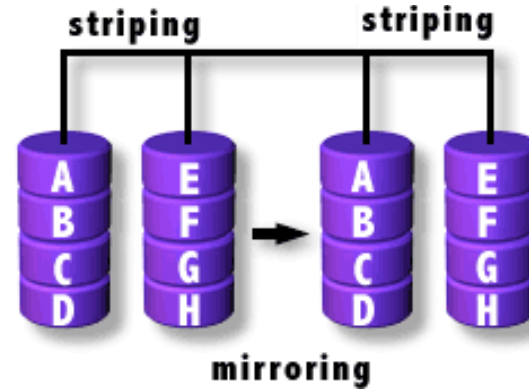


- Nivel 1: Discuri oglindite (*mirrored*)
 - Fiecare disc are o “oglină” (*check disk*)
 - Citiri paralele, o scriere implică două discuri.
 - Rata maximă de transfer = rata de transfer a unui disc

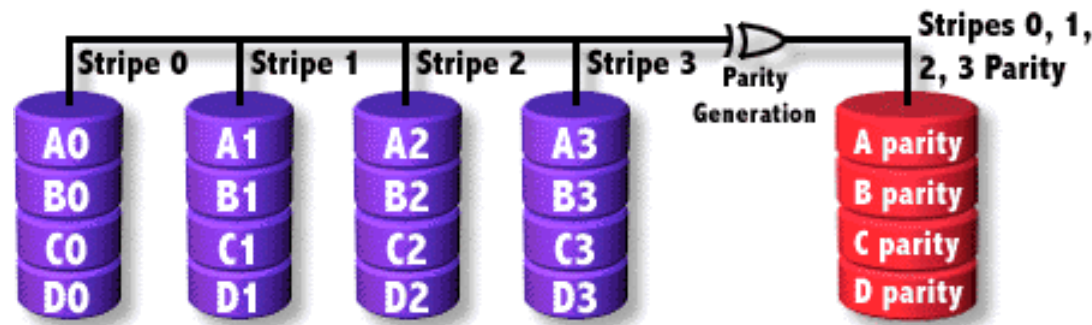


Nivele RAID

- Nivel 0+1:
Întrețesut și oglindit

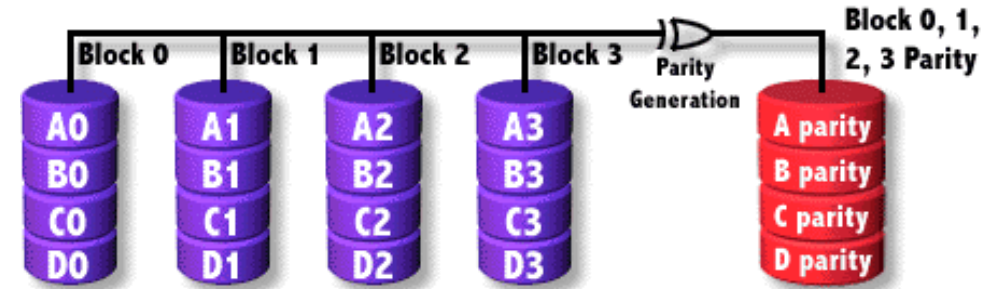


- Nivel 3: Bit de paritate intercalat
 - *Striping Unit*: un bit. un singur disc de verificare
 - Fiecare citire și scriere implică toate discurile

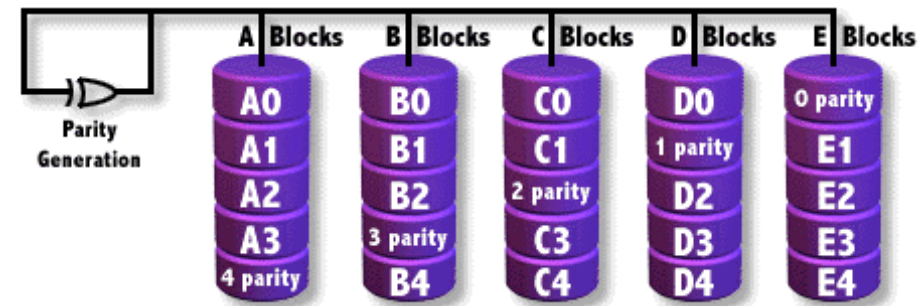


Nivele RAID

- Nivel 4: Block de paritate
 - *Striping unit*: un bloc. un singur disc de verificare.
 - Citiri în paralel pentru cereri de dimensiune mică
 - Scrierile implică blocul modificat și discul de verificare

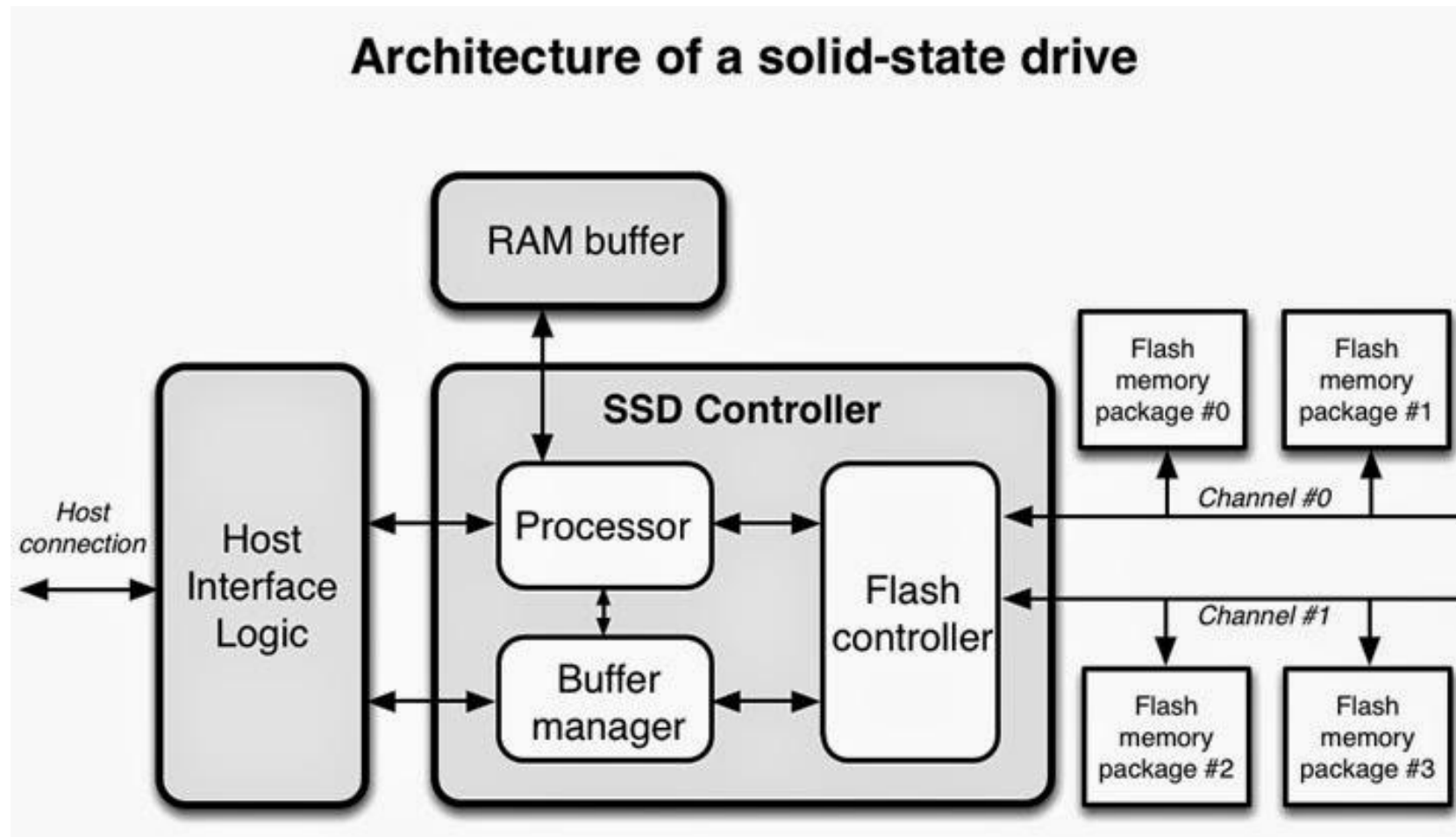


- Nivel 5: Bloc de paritate distribuit
 - Similar cu RAID 4, dar blocurile de paritate sunt distribuite pe toate discurile



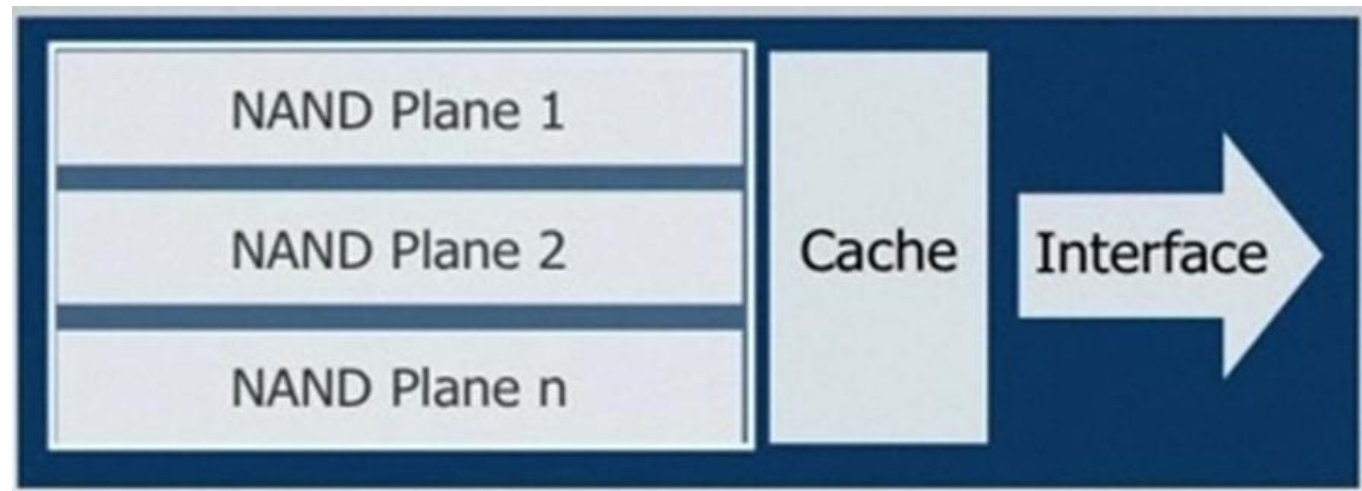
Solid State Drive

- SSD-urile conțin mai multe componente NAND flash (16/32 GB)



Solid State Drive

- NAND Flash Media e format din mai multe celule NAND aranjate pe planuri multiple:
 - aceste planuri permit accesul paralel la NAND
 - permit, de asemenea, întrețeserea datelor
- Datele sunt transferate printr-un *cache*



Solid State Drive - Avantaje

- Latență foarte mică
 - *seek time* este zero
- Viteze mari de citire și scriere
- Mai robuste fizic
 - Rezistente la șocuri
 - Zero părți mobile
 - Silențioase
 - Consumă puțin
- Excelează la citiri/scrieri de dimensiuni reduse
- “Imun” la fragmentarea datelor

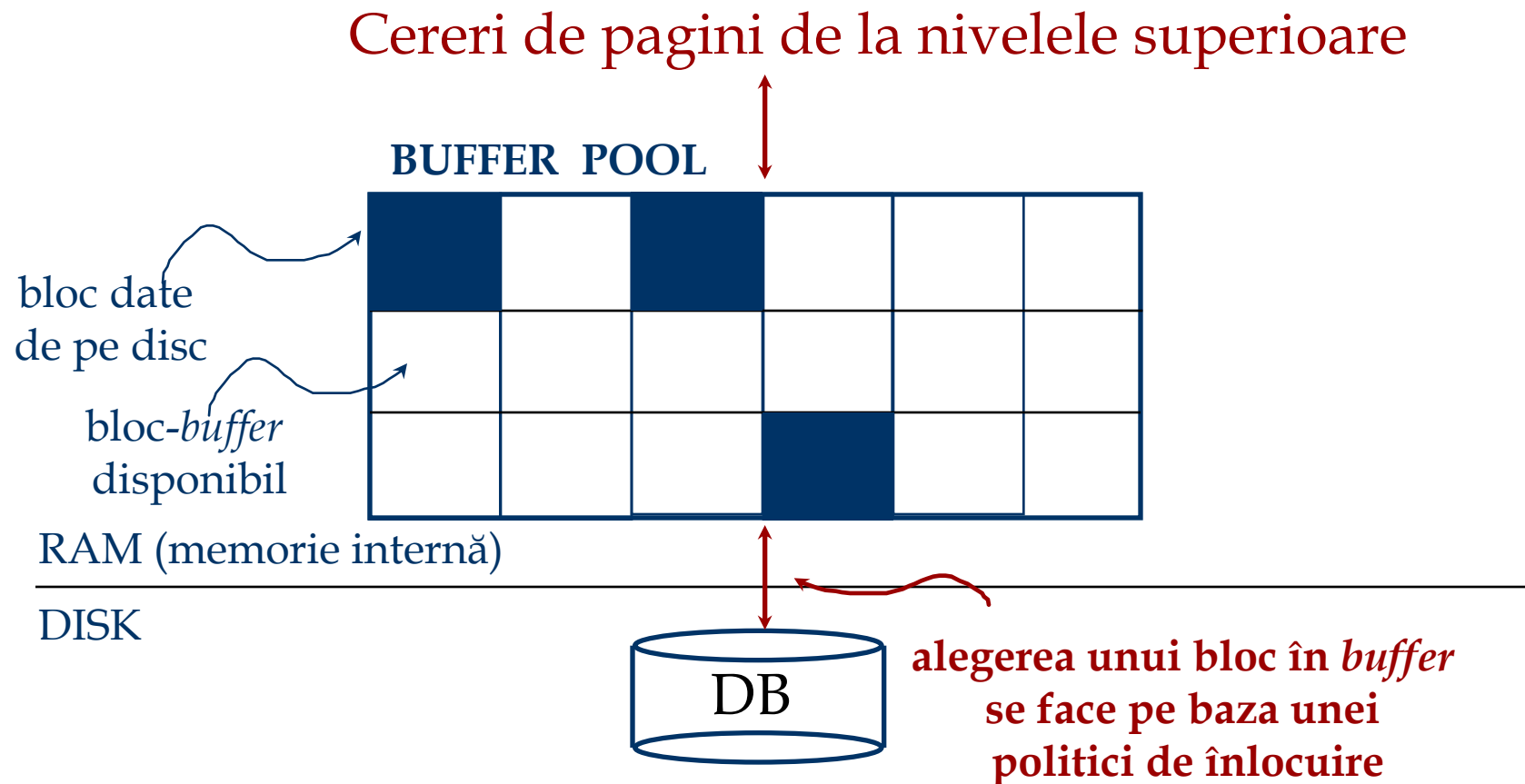
Solid State Drive – Dezavantaje

- Cost per GB mult mai mare comparativ cu discurile magnetice
- Dimensiuni
 - HDD 3.5'' cu 4TB sunt relativ comune
 - SSD 3.5'' cu 2TB sunt disponibile (rare și scumpe)
- Cicluri de citire/scriere limitate
 - 1 - 2 milioane cicluri de scriere \Rightarrow uzura MLC (multi-level cell)
 - Sub 5 milioane cicluri de scriere \Rightarrow uzura SLC

Gestionare *buffer* (zona de lucru) de către SGBD

- *Buffer* – partiție a memoriei interne utilizată pentru stocarea de copii ale blocurilor de date.
- *Buffer manager* – modul SGBD responsabil cu alocarea spațiului de *buffer* în memoria internă.
- *Buffer manager*-ul este apelat când este necesară accesarea unui bloc de pe disc
 - SGBD-ul operează asupra datelor din memoria internă

Gestionare *buffer* de către SGBD



- Se actualizează o tabelă cu perechi $\langle \text{nr_bloc_buffer}, \text{id_bloc_date} \rangle$

La cererea unui bloc de date...

- Dacă blocul nu se regăsește în *buffer*:
 - Se alege un bloc disponibil pt. **înlocuire**
 - Dacă blocul conține modificări acesta este transferat pe disc
 - Se citește blocul dorit în locul vechiului bloc
- Blocul e *fixat* și se returnează adresa sa

*! Dacă cererile sunt predictibile (ex. scanări secvențiale)
pot fi citite în avans mai multe blocuri la un moment dat*

Gestionare *buffer* de către SGBD

- Programul care a cerut blocul de date trebuie să îl elibereze și să indice dacă blocul a fost modificat:
 - folosește un **dirty bit**.
- Același bloc de date din *buffer* poate fi folosit de mai multe programe:
 - folosește un **pin count**. Un bloc-*buffer* e un candidat pentru a fi înlocuit ddacă **pin count** = 0.
- Modulele *Concurrency Control & Crash Recovery* pot implica acțiuni I/O adiționale la înlocuirea unui bloc-*buffer*.

Politici de înlocuire a blocurilor în *buffer*

- *Least Recently Used* (LRU): utilizează șablonul de utilizare a blocurilor ca predictor al utilizării viitoare. Interogările au șabloane de acces bine definite (ex, scanările secvențiale), iar un SGBD poate utiliza informațiile din interogare pentru a prezice accesările ulterioare ale blocurilor.
- *Toss-immediate*: eliberează spațiul ocupat de un bloc atunci când a fost procesat ultima înregistrare stocată în blocul respectiv
- *Most recently used* (MRU): după procesarea ultimei înregistrări dintr-un bloc, blocul este eliberat (*pin count* e decrementat) și devine blocul utilizat cel mai recent.

Politici de înlocuire a blocurilor în *buffer*

- *Buffer Manager* poate utiliza **informații statistice** cu privire la probabilitatea ca o anumită cerere să refere un anumit bloc sau chiar o anumită relație
- Politicile de înlocuite pot avea un impact determinant în ceea ce privește numărul de I/Os – dependent de **șablonul de acces**.
- *Sequential flooding*: problemă generată de LRU + scanări secvențiale repetate.
 - **Nr blocuri-buffer < Nr blocuri în tabelă** → fiecare cerere de pagină determină un I/O. MRU e preferabil într-o astfel de situație.

SGBD vs. Sistemul de fișiere al SO

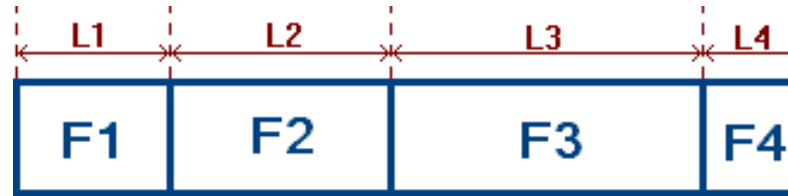
- SO gestionează spațiul pe disc și *buffer*-ul.
De ce o face și un SGBD?
- Diferențe de suport oferit de SO: probleme de portabilitate
- Existența unor limitări (ex. fișierele nu pot fi salvate pe mai multe discuri)
- Gestionarea *buffer*-ului de SGBD presupune abilitatea de a:
 - fixa/*elibera* blocuri, *forța salvarea unui bloc* pe disc (important pentru *concurrency control & crash recovery*),
 - ajustarea *politicii de înlocuire* și citirea de blocuri în avans pe baza șablonului de acces ale operațiilor tipice BD.

Fișiere de înregistrări

- Modulele de nivel superior ale unui SGBD operează cu **înregistrări** și **fișiere de înregistrări**, nu cu pagini sau blocuri
- **Fișier** = colecție de pagini; fiecare pagină conține o colecție de înregistrări; trebuie să permită:
 - inserarea/ștergerea/modificare înregistrărilor
 - citirea unei înregistrări particulare (folosind un **record id**)
 - scanarea tuturor înregistrărilor (eventual filtrate)
- O pagină ce conține o înregistrare poate fi identificată prin intermediul referinței acestuia (**rid**)

Formatarea înregistrărilor

- Lungime fixă

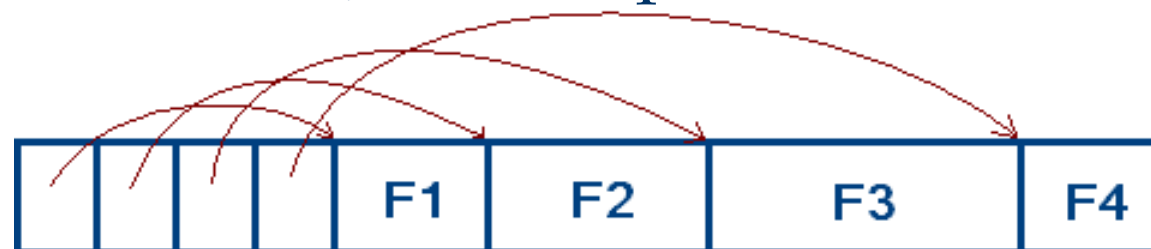


- Lungime variabilă

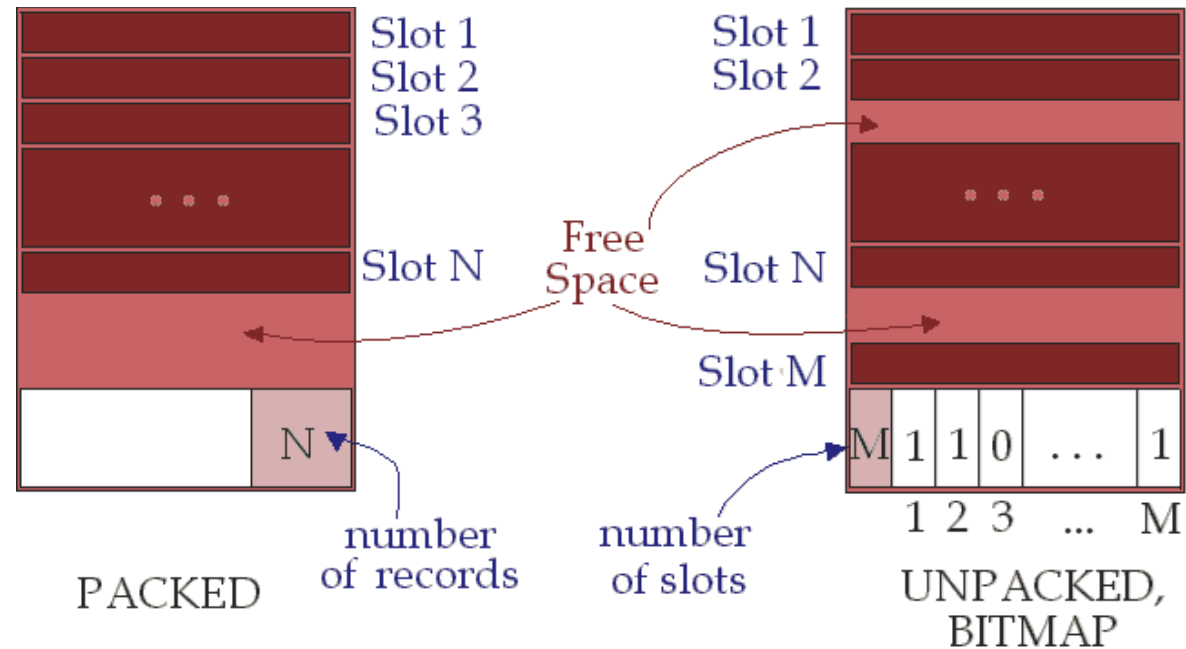
- Câmpuri delimitate prin simboluri speciale



- Șir de referințe la câmpuri

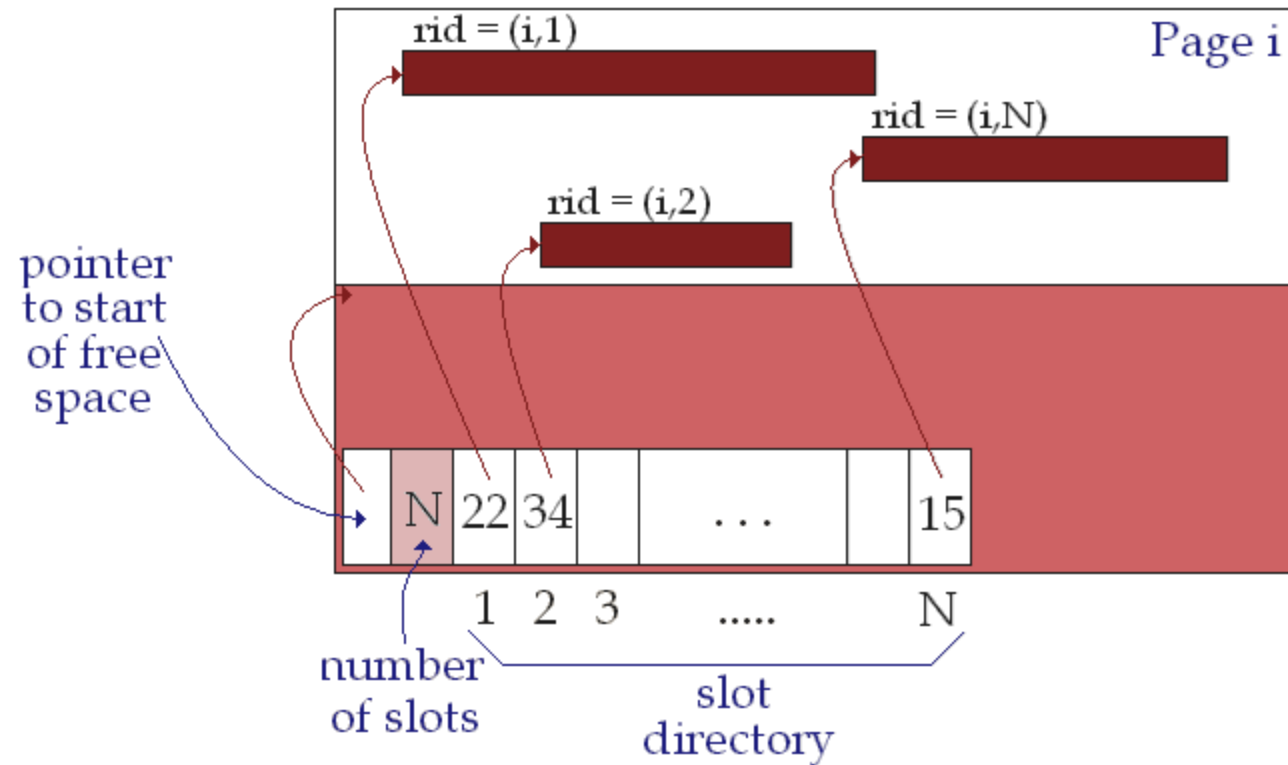


Formatarea paginilor: înregistrări cu lungime fixă



- Record id = <page id, slot no>. În varianta împachetată, mutarea înregistrărilor pentru gestionarea spațiului liber implică modificarea ridului, ceea ce nu este acceptabil în anumite situații.

Formatarea paginilor: înregistrări cu lungime variabilă



- Pot fi mutate înregistrări în pagină fără a modifica *rid-ul*; utilizabil și în cazul înregistrărilor cu lungime fixă