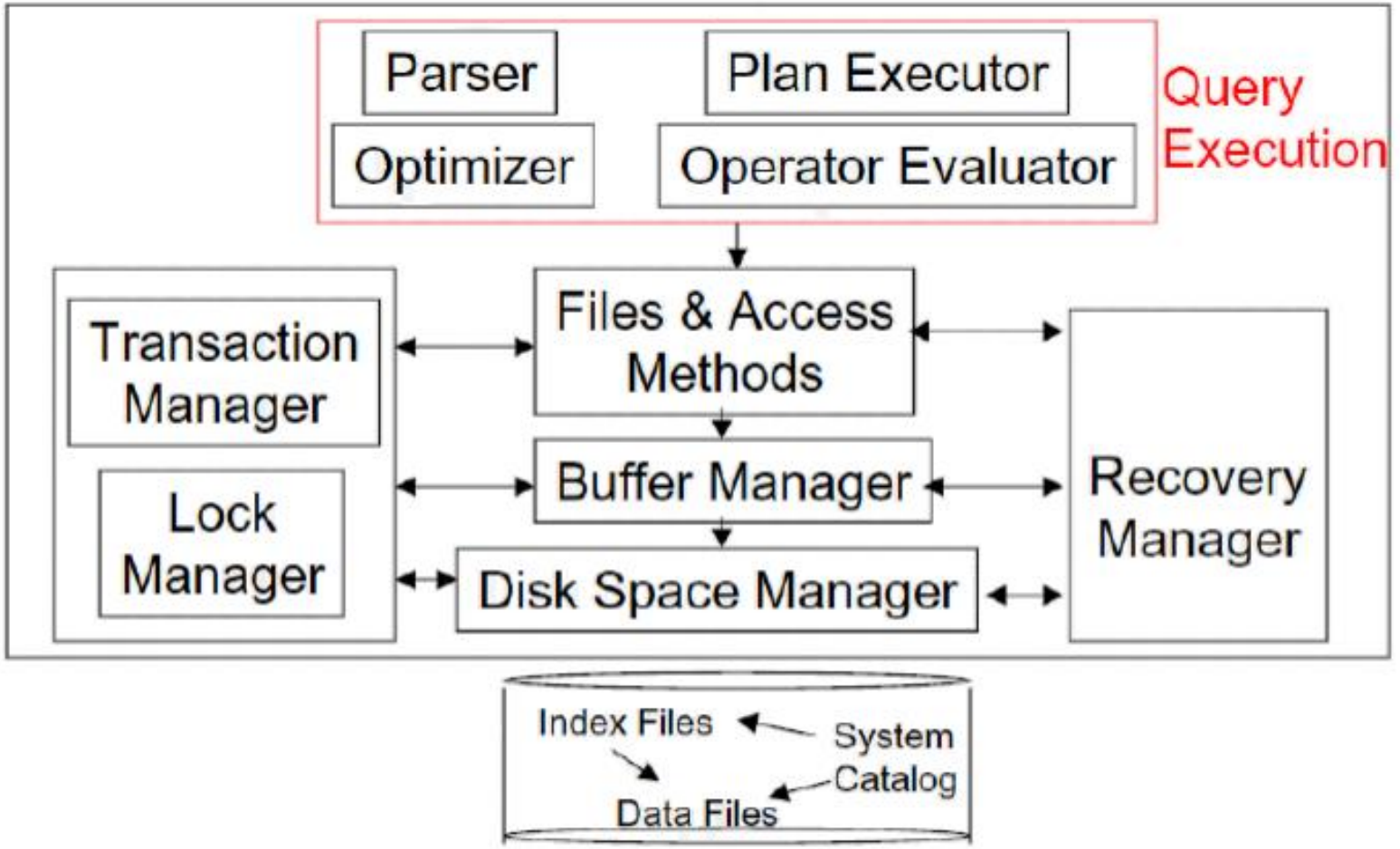
Structura fizică a bazelor de date.

Curs 4

Structura unui SGBD



Structura fizică a fişierelor BD

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

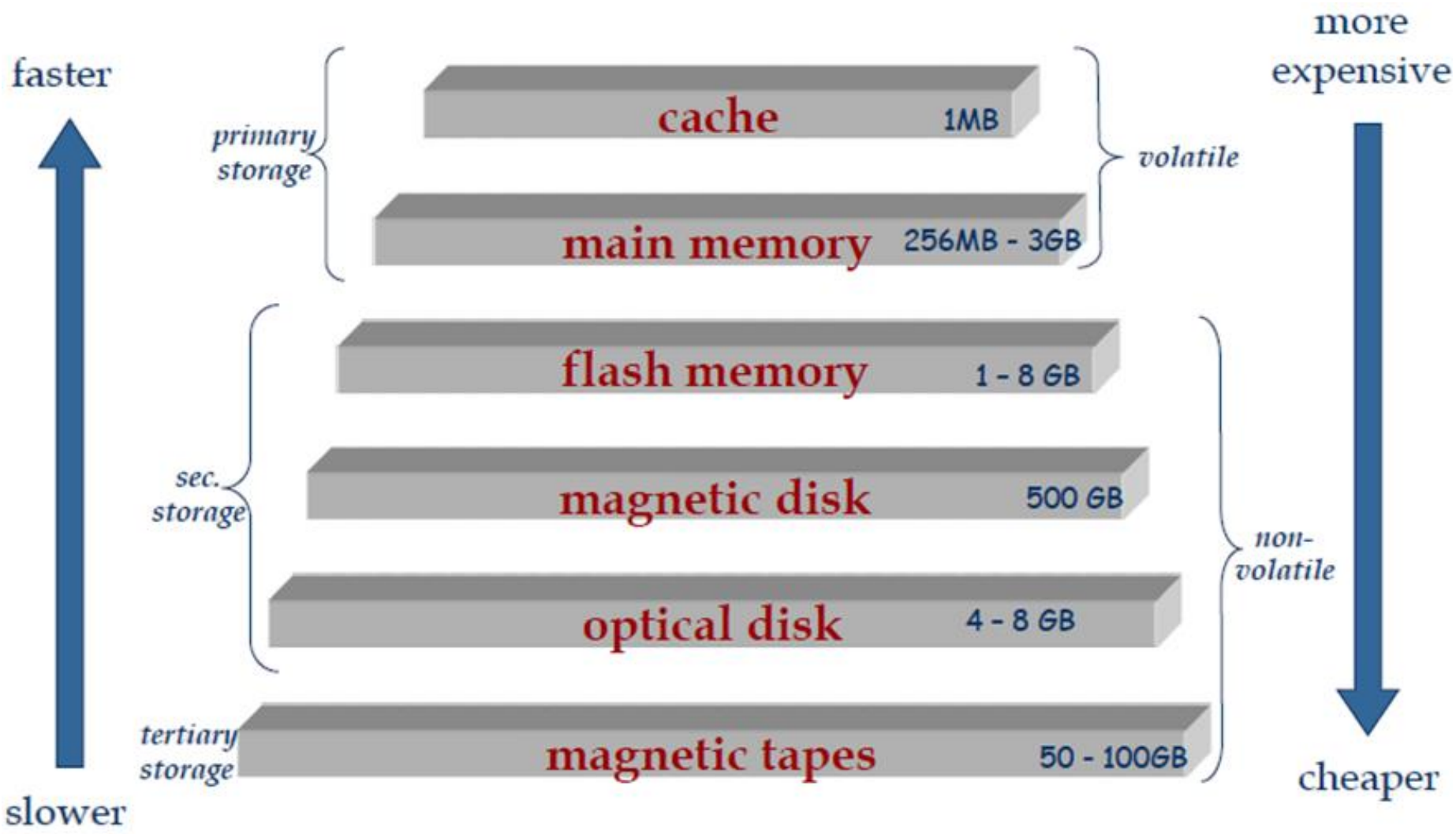
- Ambele operaţii sunt costisitoare, comparativ cu operaţiile *in-memory*, deci trebuie planificate corespunzător!

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 utilizate curent (*primary storage*)
  - *Hard-disks* – pentru baza de date (*secondary storage*)
  - Bandă – pentru arhivarea versiunilor anterioare ale datelor (*tertiary storage*)

Ierarhia mediilor de stocare



Ierarhia de memorii

- Stocare primară (internă)
  - cache și memorie principală
  - acces foarte rapid la date
  - nepersistentă
- Stocare secundară (externă)
  - dispozitive mai lente, e.g., discurile magnetice
  - persistentă
  - discuri – acces secvențial și direct
- Stocare terțiară (externă)
  - cele mai lente dispozitive de stocare, e.g., discuri optice și benzi magnetice
  - persistentă

Stocare terțiară

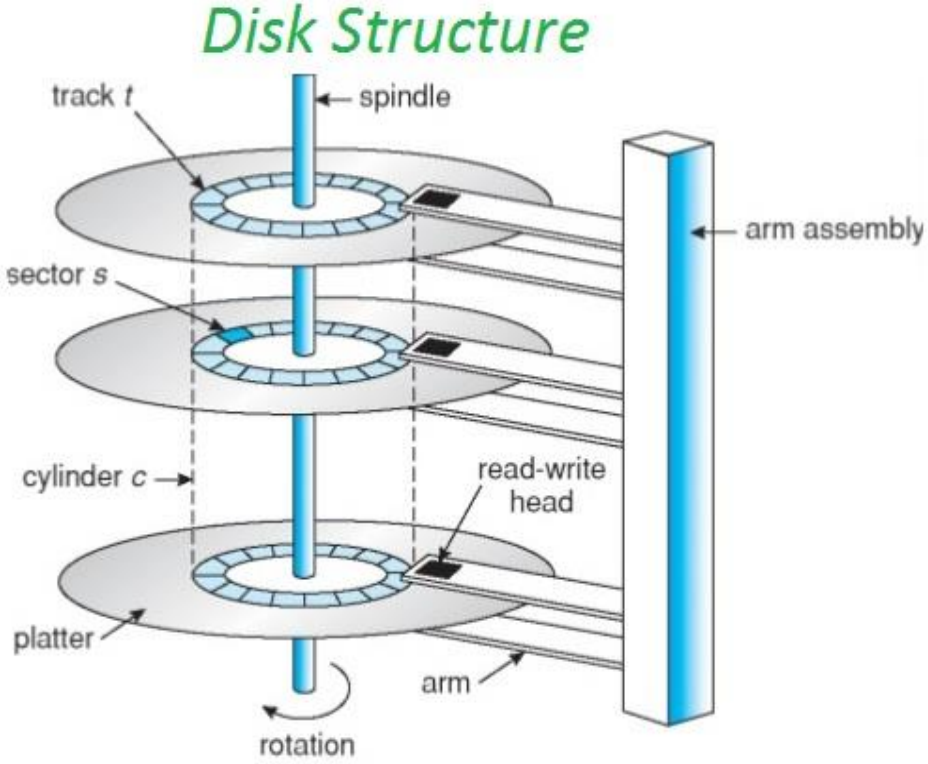
- Benzi
  - numai acces secvențial – trebuie să trecem prin toate datele în ordine, nu se poate accesa direct o locație dată de pe bandă
  - o alegere bună pentru stocarea arhivelor de date, a backup-urilor periodice
  - nepotrivite pentru stocarea datelor care sunt accesate / modificate frecvent
- Nevoia de SGBD care aduc date din memoria externă în memoria principală pentru procesare
  - costuri
  - persistarea datelor între execuții – nonvolatilitate

Stocare secundară – discurile magnetice

- avantaj major asupra benzilor: *acces direct*
- oferă acces direct la locația dorită
- Datele sunt stocate şi citite în unităţi numite blocuri sau pagini (secvențe de octeți învecinați)
- dimensiunea unui bloc poate fi stabilită când discul este inițializat ca fiind un multiplu al dimensiunii unui sector
- Spre deosebire de memoria internă, timpul de transfer blocurilor/paginilor variază în funcţie de poziţia acestora pe disc.

Componentele unui *hard disk*

- Blocurile sunt aranjate în cercuri concentrice numite *tracks* (piste) pe mai multe ***platters*** (discuri circulare) ale discului
- Fiecare ***track*** este împărțit în unul sau mai multe sectoare
- 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)

- pentru ca SGBD să poată opera asupra lor, datele trebuie să fie în memorie
- întrucât blocul e unitatea de transfer între disc și memoria principală, dacă trebuie citit doar un item de pe bloc, întregul bloc este transferat
- scrierea / citirea unui bloc: o operație I/O (input/output)
- capete de citire/scriere ale discului – pentru a scrie /citiun bloc, un cap de citire/scriere trebuie să fie poziționat deasupra blocului

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* (timpul de așteptare pentru ca blocul dorit să se rotească, ajungând sub capul de citire/scriere)
  - *transfer time* (transfer date de pe/pe disc)
- *Seek time* și *rotational delay* domină.
  - *Seek time* variază între 1 și 20 msec
  - *Rotational delay* variază între 0 și10 msec
  - *Transfer rate* e de aproximativ 1 msec pe 4KB (pagină)
- Reducerea costului I/O: reducere *seek*/*rotational delays*!
- **Poziţia relativă a paginilor pe disc are un impact major asupra perfomanţei unui SGBD!**

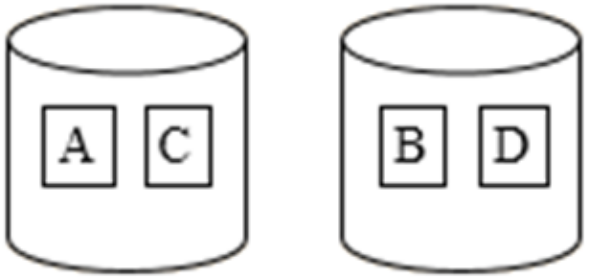
Aranjareapaginilor/blocurilorpe disc

- 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!

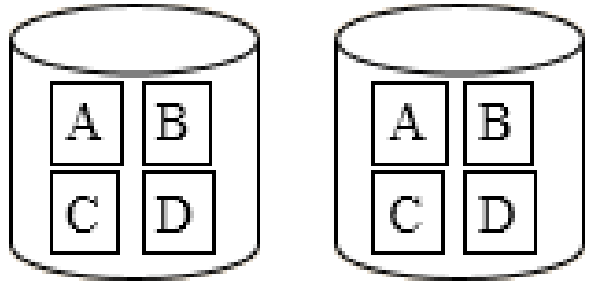
RAID (Redundant Array of Independent Disks)

- *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

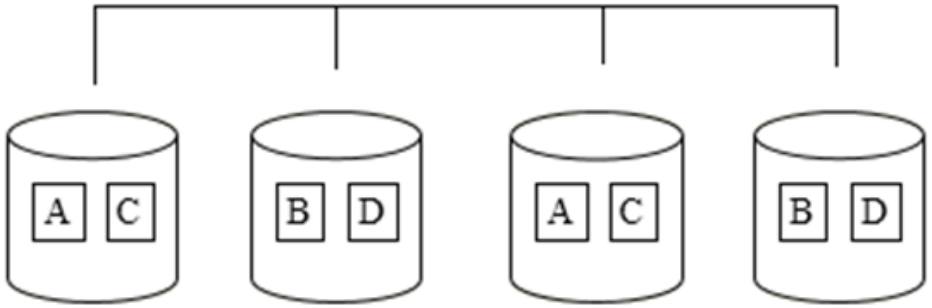


- Nivel 0: Fără redundanță
  - *Striping unit*: un bloc
  - Cea mai bună performanță la scriere



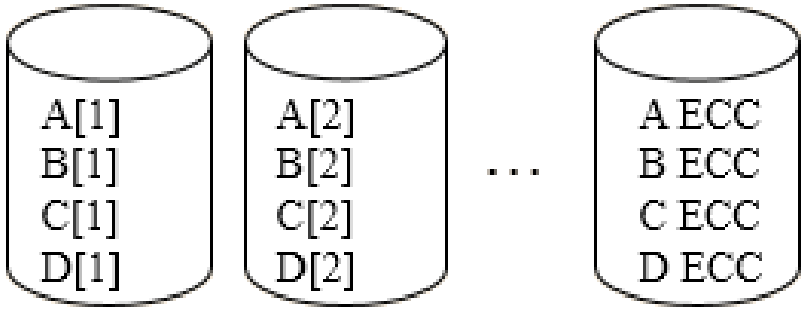
- Nivel 1: Discuri oglindite (*mirrored*)
  - Fiecare disc are o “oglindă” (*check disk*)
  - Citiri paralele, o scriere implică două discuri.
  - Rata maximă de transfer = rata de transfer a unui disc
  - Folosim 50% din spațiul de stocare

Nivele RAID



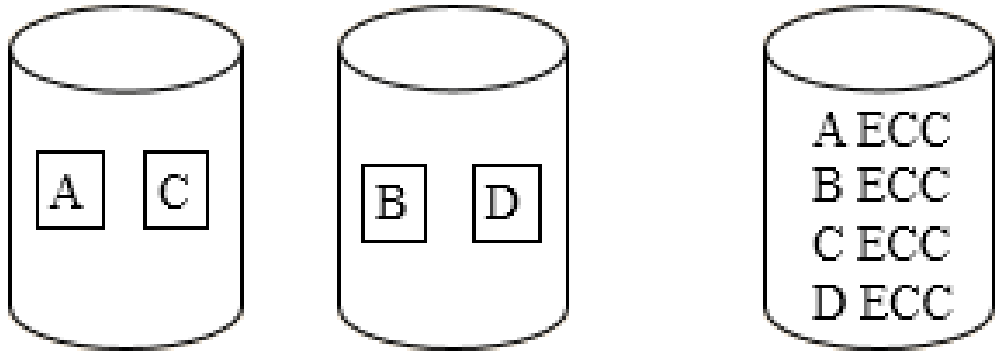
- Nivel 0+1: Întrețesut și oglindit
  - Putem citi in paralel de pe mai multe discuri

- Nivel 3: Bit de paritate intercalat
  - *Striping Unit*: un bit.
  - Un singur disc de verificare (ECC – error correction code)
  - Fiecare citire și scriere implică toate discurile



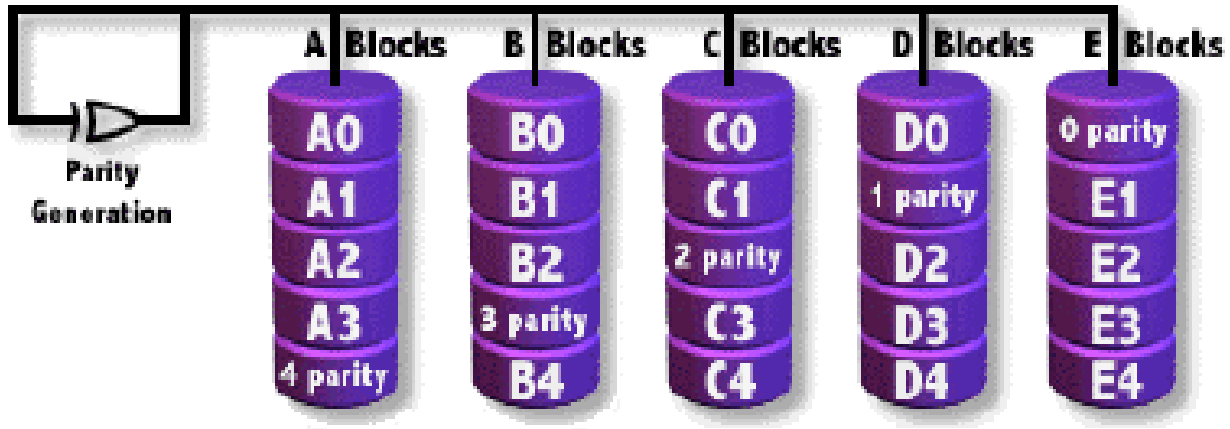
Nivele RAID

- 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



Nivele RAID

- Nivel 5: Bloc de paritate distribuit
  - Similar cu RAID 4, dar blocurile de paritate sunt distribuite pe toate discurile → nu mai avem bottleneck la scrieri



Solid State Drive (SSD)

- Avantaje:
  - Latență foarte mică
    - *seek time* este zero
  - Viteze mari de citire și scriere
  - Mai robuste fizic (rezistente la șocuri)
  - Silențioase
  - Consumă puțin
  - Excelează la citiri/scrieri de dimensiuni reduse

Solid State Drive (SSD)

- 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 millioane cicluri de scriere
    - Sub 5 milioane cicluri de scriere

    ⇒ uzura

Structura unui SGBD

1. disk space manager
   - monitorizare și gestiune spațiu disponibil pe disc
2. buffer manager
   - aducerea paginilor de pe disc într-o regiune a memoriei principale denumită *buffer pool* (zonă tampon)
3. file manager
   - fișier de înregistrări – abstractizare pentru nivelurile superioare din SGBD

Monitorizarea și gestiunea spațiului pe disc

- realizată de disk space manager
- la nivel abstract, disk space manager (DSM) utilizează conceptul de **pagină ca unitate de date** și oferă comenzi pentru alocarea / dealocarea unei pagini, comenzi pentru scrierea / citirea unei pagini
- dimensiunea unei pagini se alege să fie dimensiunea unui bloc
- paginile sunt stocate ca blocuri pe disc
- scrierea / citirea unei pagini poate fi realizată într-o operație I/O
- DSM permite nivelurilor superioare din SGBD să opereze cu datele sub forma unor colecții de pagini

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

- ***Buffer*** – partiție a memoriei interne utilizată pentru stocarea de copii ale blocurilor de date.

- ***Buffer manager*** – responsabil cu alocarea spațiului de *buffer* în memoria internă.

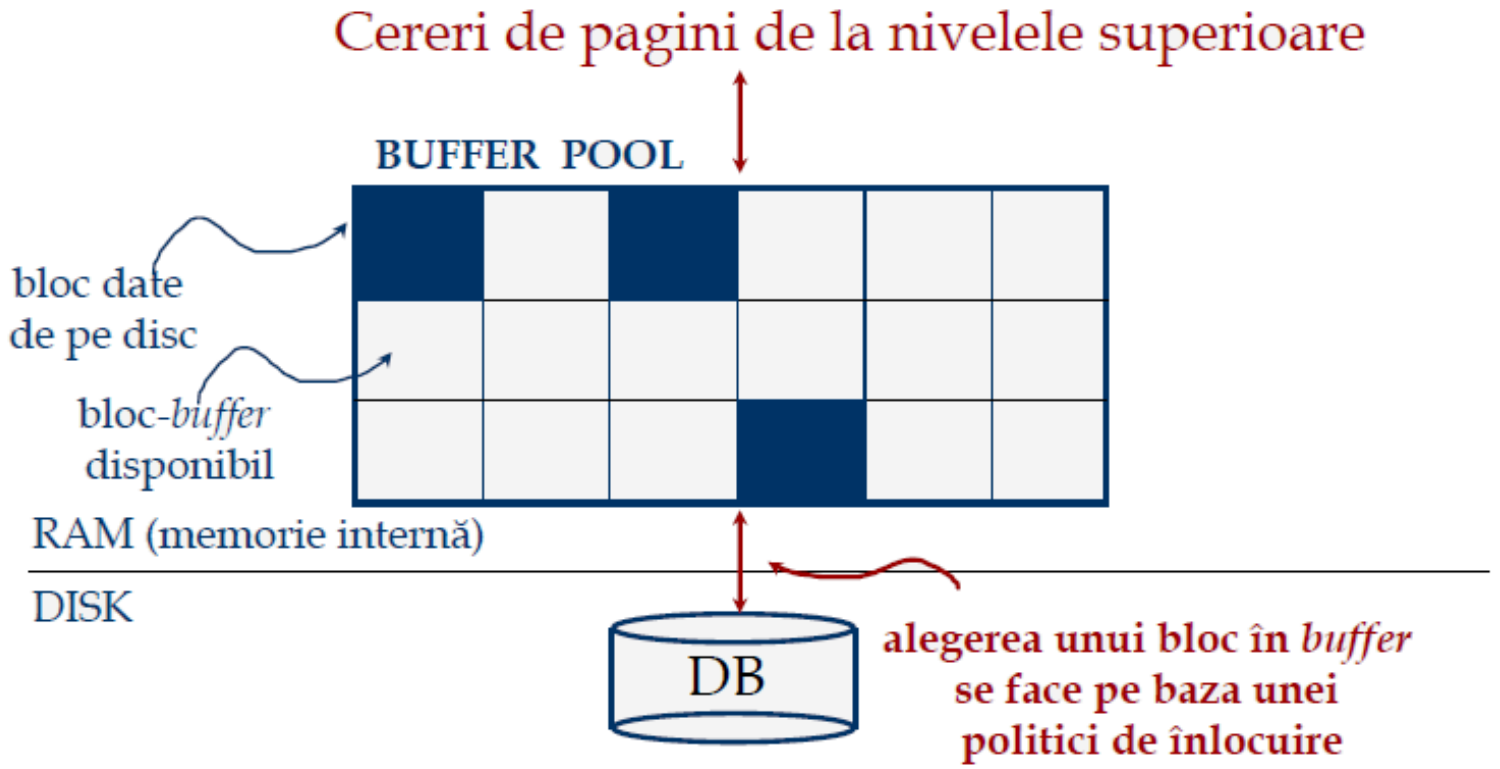
- ***Buffer manager*** gestionează memoria principală disponibilă partiționând-o într-o colecție de pagini: buffer pool(BP)

- ***Buffer manager*** este apelat când este necesară accesarea unui bloc de pe disc
  - SGBD-ul operează asupra datelor din memoria internă

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

- Paginile din buffer pool se numesc *frames* (cadre), i.e., sloturi care pot ține o pagină
- Din punctul de vedere al nivelurilor superioare din SGBD, nu contează dacă datele sunt sau nu în memoria principală
- Acestea solicită Buffer Manager-ului o pagină, dacă aceasta nu se află în Buffer Pool, este adusă într-un frame în BP
- Când pagina nu mai este utilizată, cel care a solicitat-o o eliberează, i.e., BM este informat și frame-ul care o conține poate fi reutilizat
- Dacă pagina a fost modificată, BM este informat de către codul care a solicitat pagina și propagă schimbările din BP în copia paginii de pe disc

Gestionare *buffer* de către SGBD



- Se actualizează o tabelă cu perechi <nr_bloc_buffer, id_bloc_date>

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 doritîn 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 dacă **pin count= 0.**

- Ce se întâmplă dacă o pagină este solicitată de mai multe tranzacții, e.g., dacă apar modificări conflictuale?
- 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 ulteriore 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 înlocuire 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)

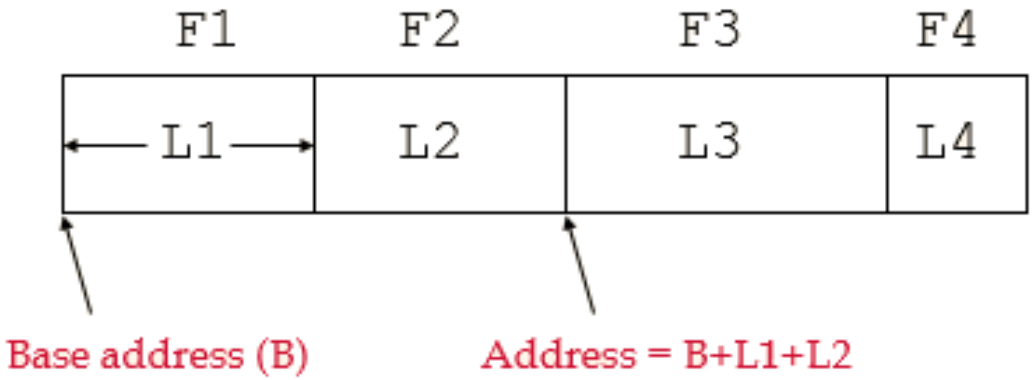
- Gestionarea *buffer*-ului de SGBD presupune abilitatea de a:
  - fixa/elibera blocuri, forța salvarea unui bloc pe disc (important pentru *concurrency control* & *crash recovery*),
  - Ajustarea *politicii de înlocuire* și citirea de blocuri în avanspe baza șablonului de accesale operațiilor tipice 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 tututror înregistrărilor(eventual filtrate)
- O pagină ce conține o înregistrare poate fi identificată prin intermediul referinței acestuia (rid)

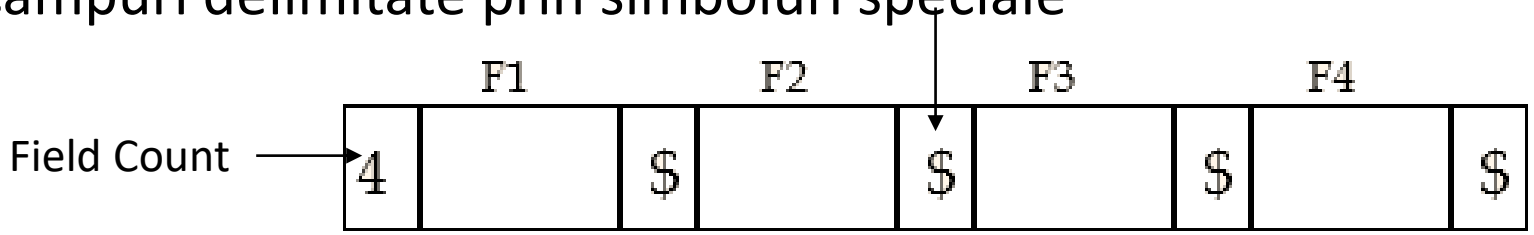
Formatarea înregistrărilor

- Lungime fixă

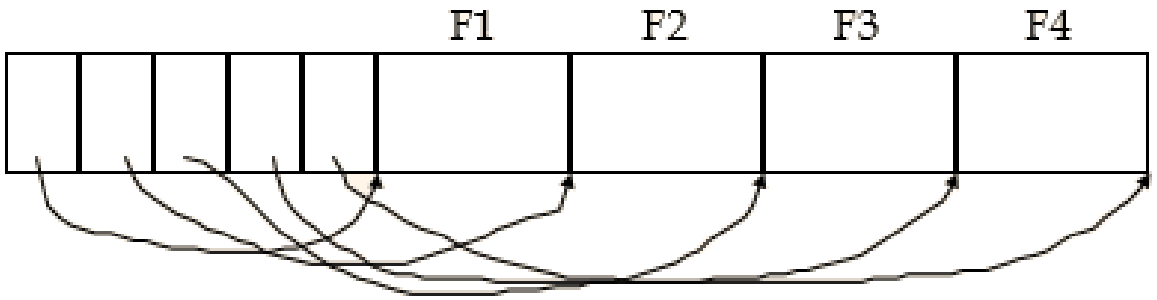


Formatarea înregistrărilor

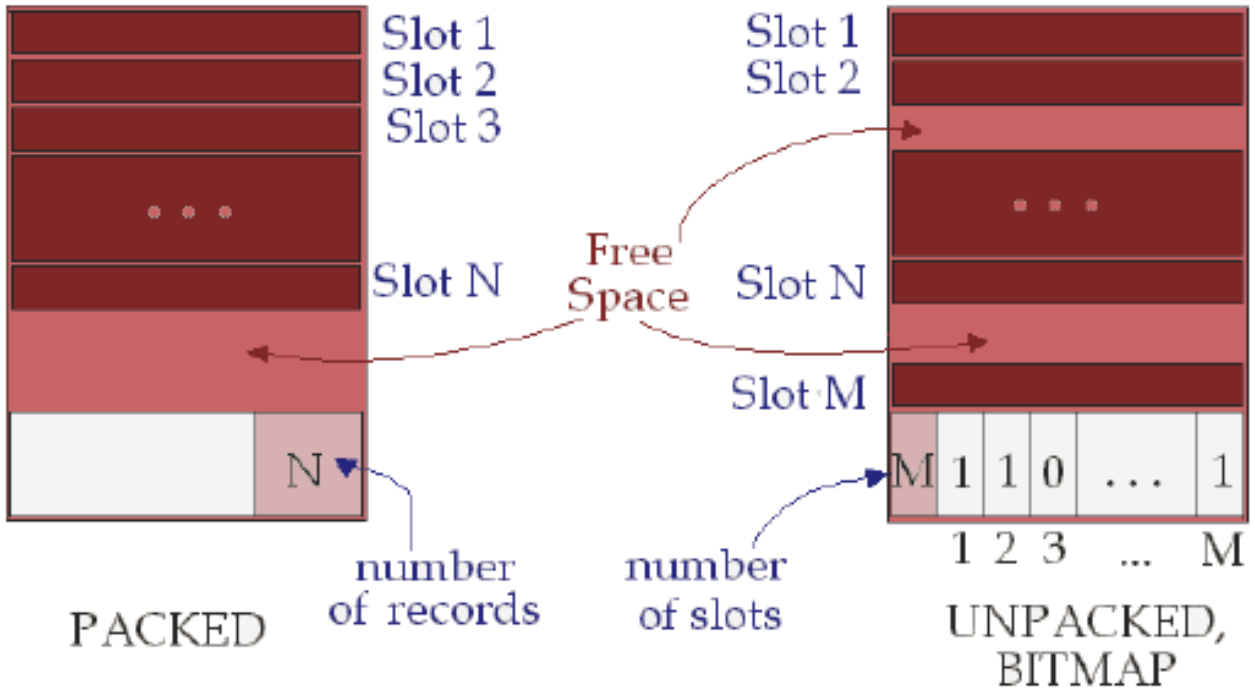
- Lungime variabilă
  - Câmpuri delimitate prin simboluri speciale



- Șir de referințe la câmpuri

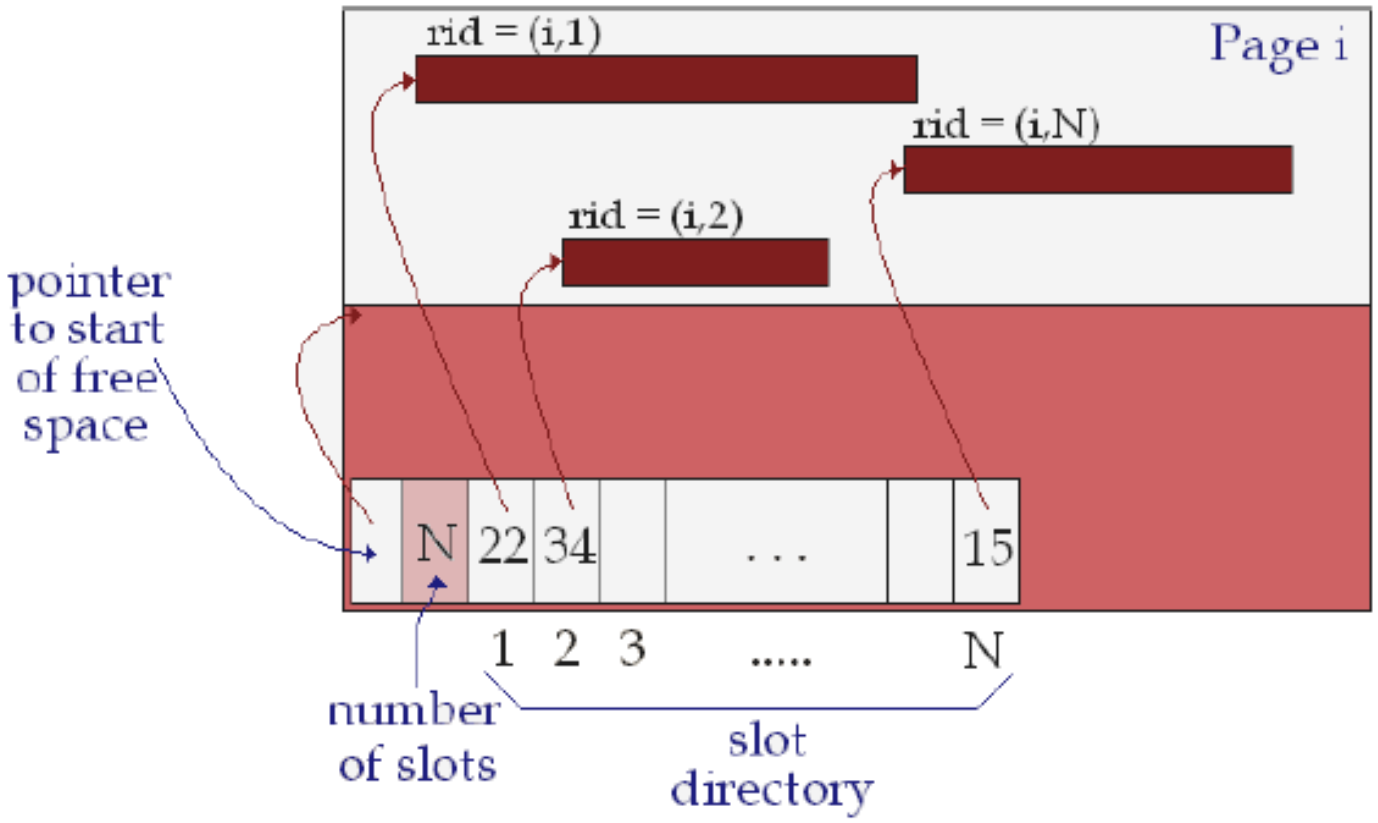


Formatarea paginilor: înregistrări cu lungime fixă



- Record id = <page id, slot no>.
- În varianta împachetată, mutarea înregistrărilor pentru gestionarea spațiului liber implică modificarea rid-ului, 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*