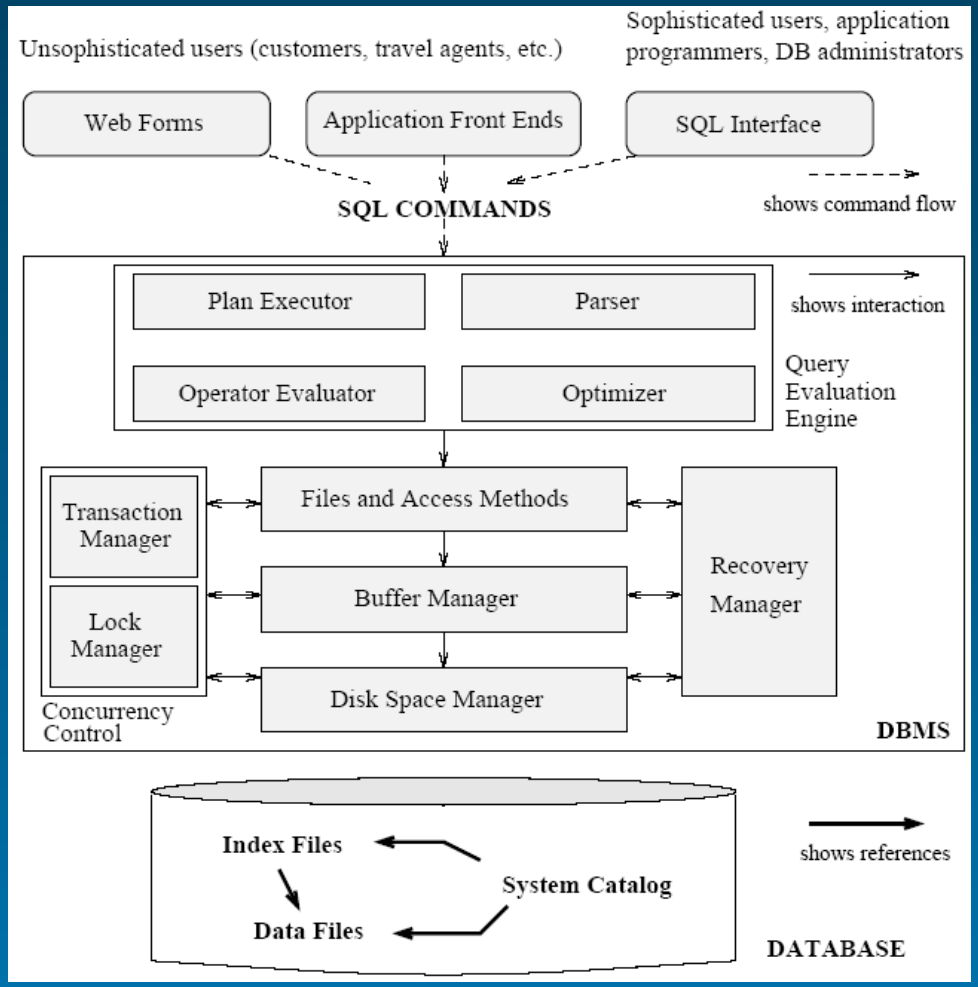
Poglavje 4

Diski in datoteke

Povzeto po [1]

Uvodna shema



Uvod..

- Podatki SUPB se hranijo na diskih (in trakovih)
- Disk Space Manager **upravlja** s prostorom na trdem disku in vodi evidenco o razpoložljivem prostoru
- File Manager **izdaja ukaze Disk Space Managerju** v zvezi z zaseganjem in sproščanjem prostora na disku
- File Manager predstavlja abstrakcijo datotek zapisov višjim nivojem SUPB

Uvod..

- File Manager je **odgovoren za upravljanje strani** znotraj datoteke in za urejanje zapisov znotraj strani
- File Manager **zasega in sprošča prostor na disku** v enotah – **straneh.** Velikost strani je eden od parametrov SUPB in je tipično od 4KB od 8 KB

Uvod

- Buffer manager **skrbi za prenos določene strani** iz diska v področje v glavnem pomnilniku, ki se imenuje vmesni pomnilnik (buffer pool)

Hierarhija pomnilnika..

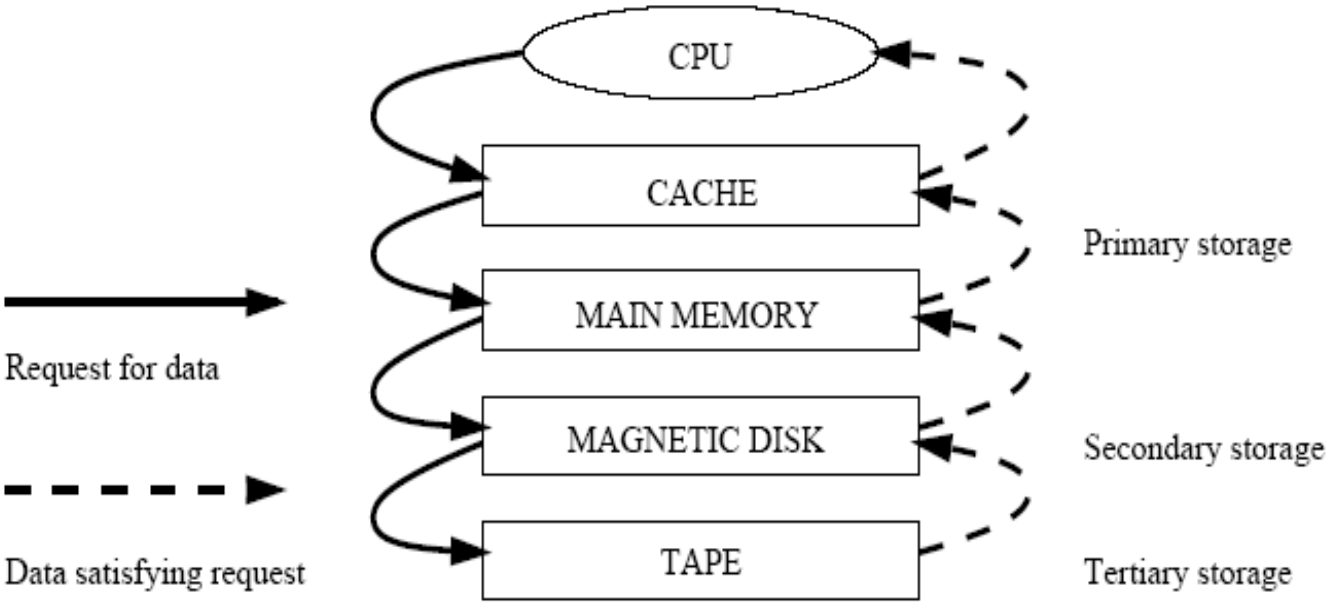


Figure 7.1 The Memory Hierarchy

Hierarhija pomnilnika..

- Dostop do primarnega pomnilnika je zelo hiter
- Stroški so za določeno količino glavnega pomnilnika so približno **100 krat večji** kot stroški za enako količino sekundarnega pomnilnika (disk)
- Počasnejše enote sekundarnega pomnilnika igrajo pomembno vlogo, saj je podatkov običajno zelo veliko
- Ker ne moremo vseh podatkov shraniti v glavni pomnilnik, jih shranjujemo na sekundarni (in terciarni pomnilnik-trakovi)

Hierarhija pomnilnika

- Obstajajo tudi **drugi razlogi** za shranjevanje podatkov na sekundarnem (in terciarnem pomnilniku):
  - 32 bitni naslovni prostor omogoča naslavljanje samo 4Gb podatkov....
  - podatki morajo biti obstojni, v primarnem pomnilniku podatki običajno niso obstojni.
- Stanje glede kapacitete in cene pomnilnikov se spreminja iz dneva v dan, a okvirno razmerje 100:1 se ne spreminja

Magnetni disk..

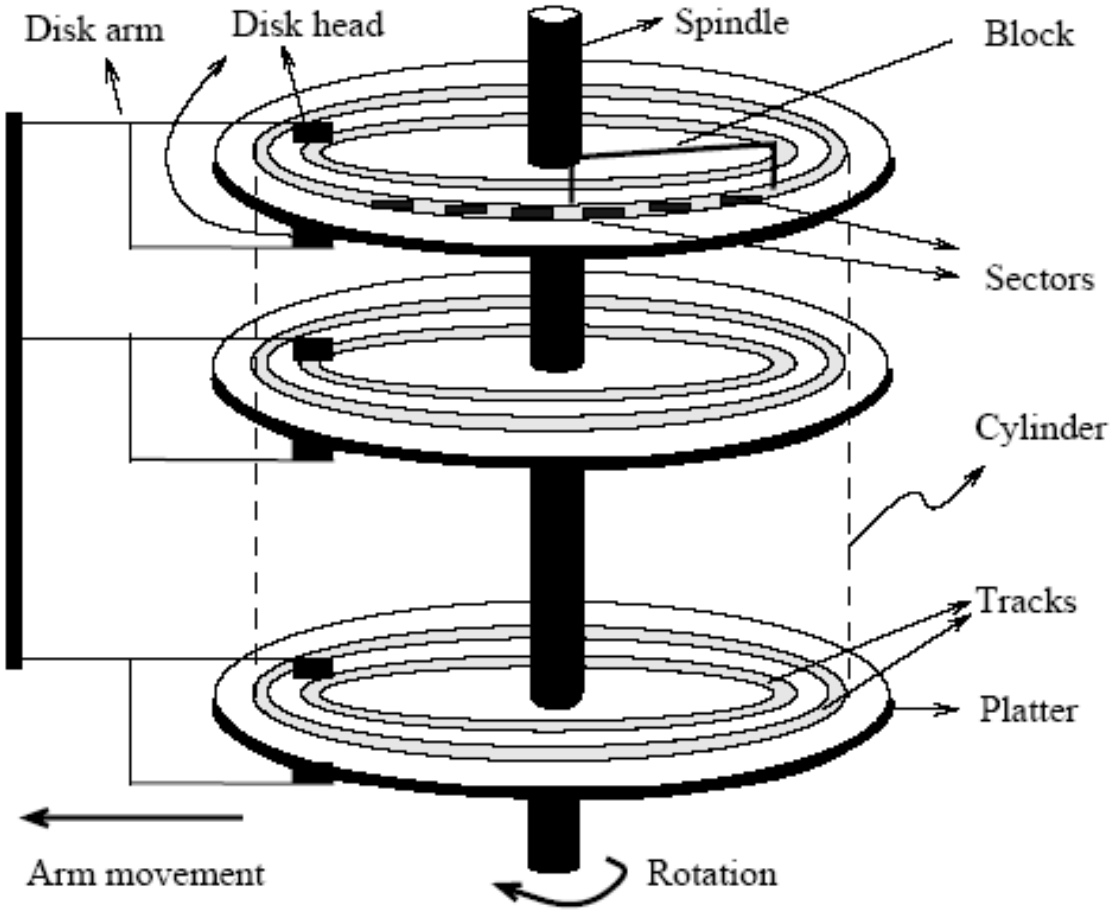


Figure 7.2 Structure of a Disk

Magnetni disk..

- Magnetni disk omogoča **neposreden dostop** do želene lokacije na njem
- SUPB omogoča **transparenten dostop** do podatkov, ki se nahajajo na disku
- Podatki na disku so shranjeni v enotah, ki se imenujejo BLOK-i
- BLOK predstavlja **zaporedje nizov** (byte) in je najmanjša enota, ki se jo lahko bere iz ali piše na disk

Magnetni disk..

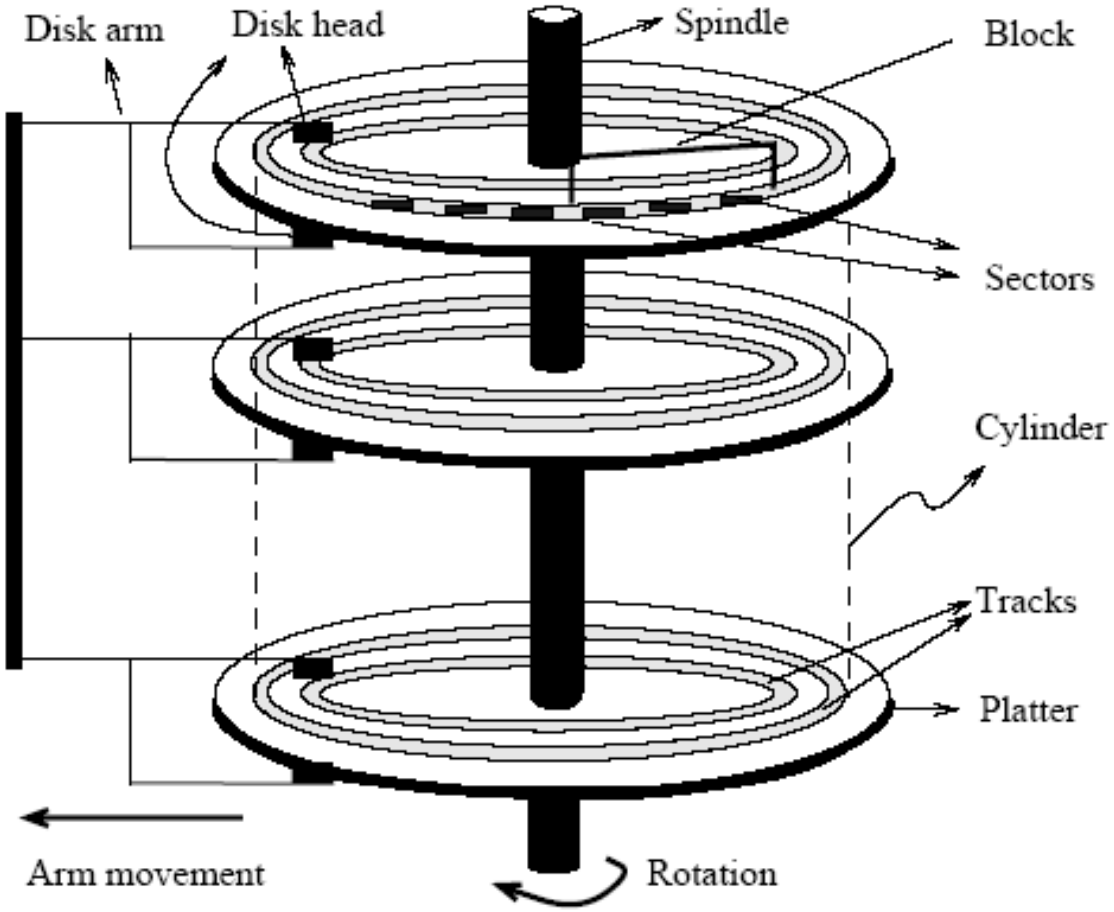


Figure 7.2 Structure of a Disk

Magnetni disk..

- BLOKI so organizirani v **koncentrične kroge** – SLEDI (track)
- SLEDI se nahajajo na eni ali obeh straneh magnetne plošče
- Množica vseh SLEDI, ki so **enako oddaljene** od središča, se imenuje CILINDER

Magnetni disk..

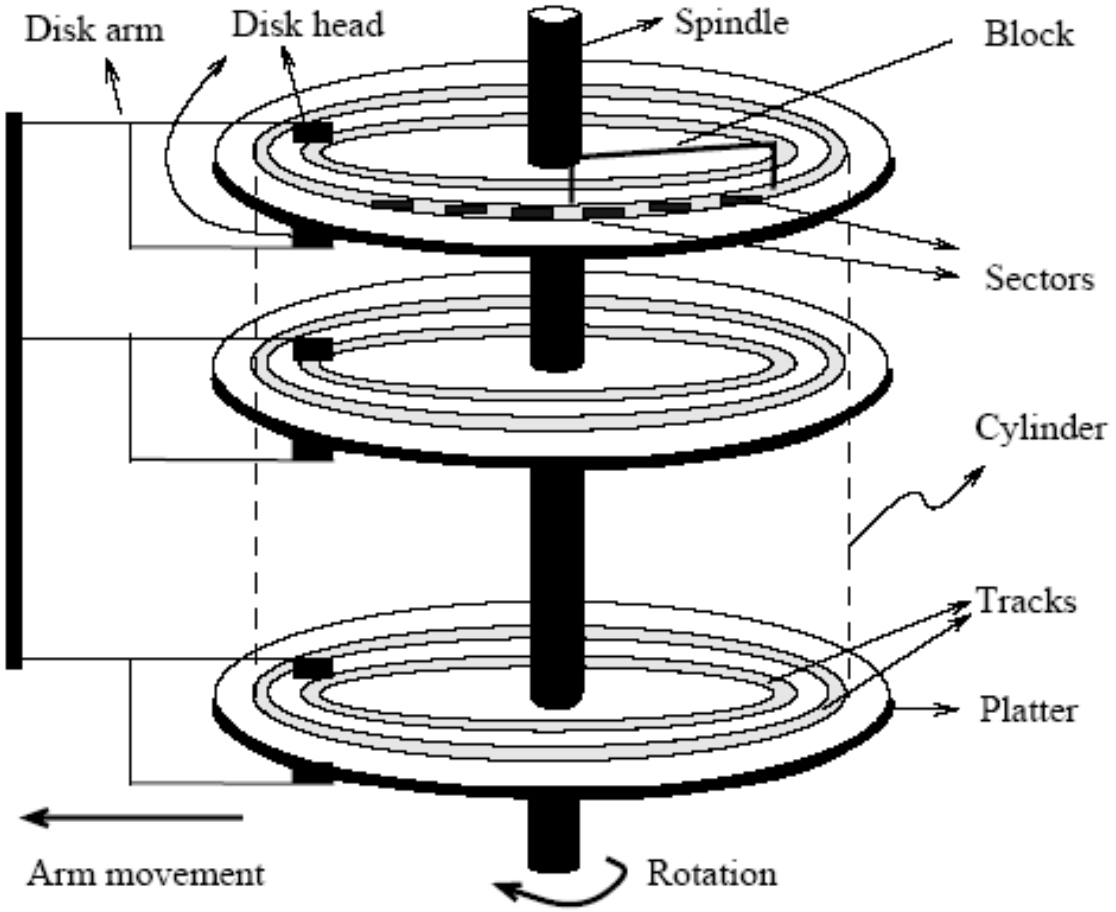


Figure 7.2 Structure of a Disk

Magnetni disk..

- Vsaka SLED **je razdeljena** na ODSEKE (sektorje) Velikost ODSEKA je določena z diskom in je ni mogoče spreminjati
- Velikost BLOKA se določi pri **formatiranju diska**. Njegova velikost je mnogokratnik velikosti ODSEKA
- V okviru diska je več glav, ki se premikajo hkrati. Za vsako ploščo ena glava (ali dve, če je vsebina na obeh straneh)

Magnetni disk..

- Disk je z računalnikom povezan preko krmilnika. Krmilnik **izvaja ukaze** za branje in pisanje na disk in zagotavlja pravilnost izvajanja teh ukazov

Magnetni disk..

- Čas, ki je **potreben za dostop** do določene lokacije na disku (**povprečni dostopni čas**) je sestavljen iz več komponent:
  - **iskalni čas** (premik glave na ustrezno sled)
  - **rotacijska zakasnitev** (čakalni čas, da se ustrezen blok na sledi zavrti do glave). Povprečno znaša polovico časa rotacije in je manjši od iskalnega časa.
  - **čas prenosa** (dejanski prenos bloka – branje ali pisanje).
- Primerjava: dostopni čas RAM-a (10ns) in diska (10ms).

Magnetni disk..

- Vpliv strukture diska na performanse:
  - podatki se morajo pred uporabo prenesti v glavni pomnilnik,
  - **najmanjša enota podatkov**, ki se bere ali piše na disk je blok. Če se potrebuje **samo en zapis** iz bloka, se prenese celotni blok.
  - čas za pisanje ali branje podatkov variira, odvisno od položaja podatkov na disku:
    - *dostopni čas=iskalni čas + rotacijska zakasnitev + čas prenosa*

Magnetni disk

- ZAKLJUČEK: Čas, ki ga SUPB porabi za obdelavo podatkov je močno odvisen od **lokacije podatkov na disku oz. razpršenosti podatkov po disku**. Čas, potreben za premikanje blokov iz diska oz. na disk je tako ponavadi večji kot čas potreben za obdelavo določenega podatka
- **Podatke je potrebno zato dobro razporediti po disku!!!.**

RAID – Uvod..

- Disk predstavlja **potencialno ozko grlo** za performanşe in vpliva na zanesljivost delovanja sistema
- Čeprav se performanse diskov zvišujejo, performanse CPU-jev rastejo veliko hitreje: perf. CPU 50% na leto, perf. diskov 10% na leto
- Diski vsebujejo mehanske elemente, zaradi česar je **verjetnost za napake** večja kot pri notranjem pomnilniku

RAID – Uvod..

- Če disk odpove, potem to v kontekstu podatkovnih baz pomeni **izgubo podatkov**
- REŠITEV: polje diskov (disk array), s katerim povečamo tako performanse kot zanesljivost delovanja

RAID – Uvod

- **Performanse se poveča** z razstavljanjem podatkov (data striping): podatke se distribuira po več diskih (občutek enega zelo hitrega diska)
- **Zanesljivost povečujemo** z redundandnostjo. Redundantne informacije so organizirane tako, da v primeru napake na disku omogočajo restavriranje podatkov na pokvarjenem disku
- Diskovna polja, ki implementirajo razstavljanje podatkov in redundantnost se imenujejo "**Redundant Arrays of Independent Disks**" – RAID
- Poznamo več RAID organizacij, ki predstavljajo **različne kompromise** (trade-offs) med performansami in zanesljivostjo

RAID – Razstavljanje podatkov..

- RAID – Data striping se kaže uporabniku kot zelo **velik disk**
- Podatki se razdelijo na enake particije (striping unit), ki se distribuirajo **na več diskov**
- Enote se po diskih distribuirajo po “round robin” algoritmu. Dva logično sosedni particiji tako nista na istem disku
- Če polje vključuje D diskov, se particija i zapiše na “i mod D” disk

RAID – Razstavljanje podatkov

- Primer:

Če je “striping unit” = blok in če so I/O operacije dolge po več blokov, potem se zahteva procesa paralelno na več diskih in tako povečamo pasovno širino prenosa podatkov tolikokrat, kolikor diskov imamo v polju

RAID – Redundantnost..

- S tem ko se z več diski povečajo performanse sistema, pa se **zmanjša** celotna zanesljivost sistema
- Predpostavimo: MTTF (mean-time-to-failure) znaša 50.000 ur (5,7 let). Pri 100 diskih v polju znaša MTTF 50.000/100=500ur (21 dni)
- Zanesljivost diskovnega polja se lahko poveča z **redundantnimi podatki**

RAID – Redundantnost..

- Redundantnost lahko **neizmerno poveča** MTTF
- Določiti oz. odločiti se je potrebno:
  - **kje bodo shranjeni redundantni podatki**:
    - na manjšem številu kontrolnih diskov ali
    - bodo porazdeljeni po vseh diskih.
  - **kako izračunati redundantne podatke**:
    - večina diskovnih polj shranjuje informacijo o pariteti (paritetna shema uporablja dodaten – redundanten kontrolni disk za obnovo (recovery) po nesreči)

RAID – Redundantnost..

- Če dodamo prejšnjemu polju 100-ih diskov 10 diskov z **redundantnimi podatki**, naraste MTTF na več kot 250 let!!!
- Velik MTTF pomeni **manjšo verjetnost** za napako

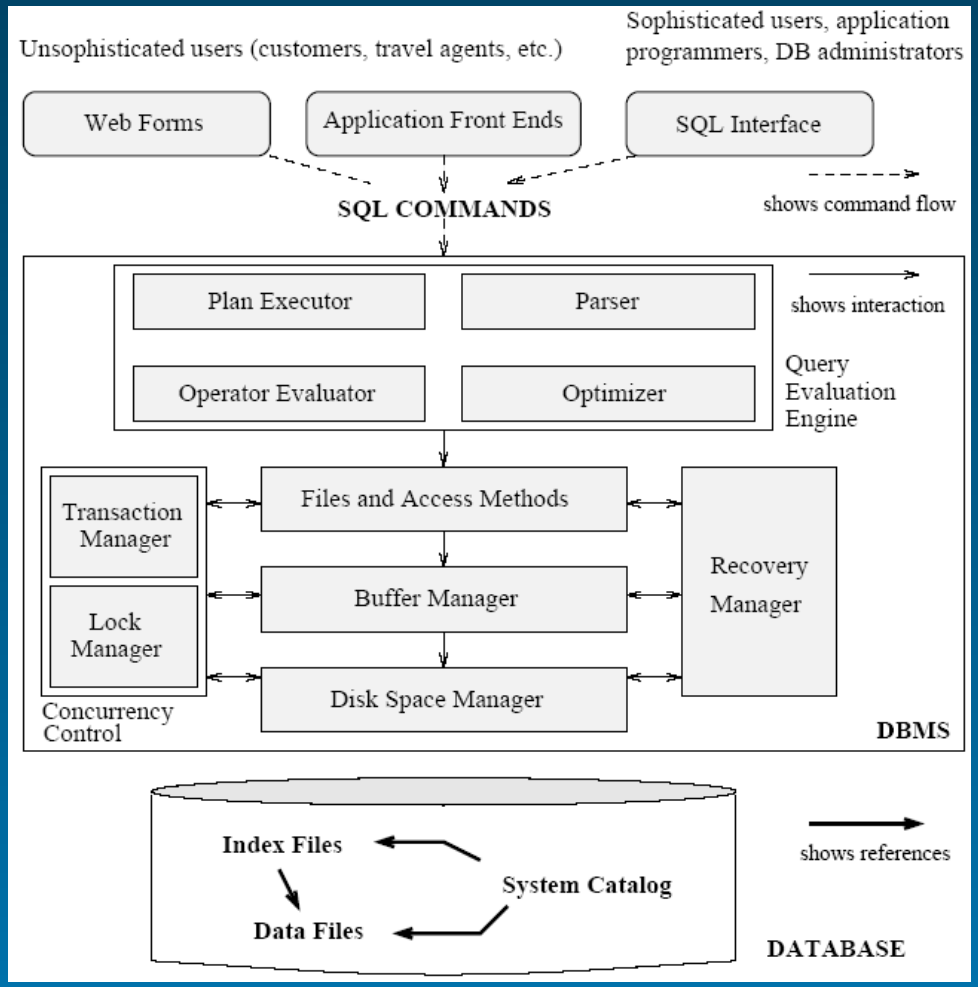
RAID – Stopnje redundance..

- Za primer predpostavimo, da imamo podatke, ki jih lahko spravimo na 4 diske. Od izbrane stopnje RAID, je odvisno dodatno število diskov (od 0 do 4).
- RAID 0: Nonredundant (data striping)
  - Uporablja data striping za povečanje pasovne širine
  - Ne vzdržuje nobene redundantne informacije
  - PROBLEM: MTTF pada linearno s številom diskov v polju
  - PREDNOSTI: najboljše performanse pisanja na disk, saj ni potrebno vzdrževati nobenih redundantnih podatkov
  - Efektivno uporabljeni prostor znaša vedno 100% prostora na disku. V našem primeru rabimo za svoje podatke 4 diske

RAID – Stopnje redundance

- RAID 1: Mirrored
  - Najdražja rešitev za polje diskov, ker se vzdržujeta dve kopiji podatkov na dveh diskih
  - Vsako pisanje bloka na disk vključuje pisanje na dva diska
  - Pisanje se ne izvede hkrati, ampak eno za drugo (zaradi primera nesreče med pisanjem)
  - Branje lahko vključuje paralelno branje dveh različnih blokov iz dveh diskov. Branje se lahko dodeli na disk, ki ima najmanjši dostopni čas
  - V našem primeru rabimo 4 diske + 4 diske za mirroring
  - efektivna uporaba prostora tako znaša 50%

Upravljanje prostora na disku..



Upravljanje prostora na disku..

- Za upravljanje z diskom skrbi **najnižji nivo** v SUPB arhitekturi – Disk Space Manager
- Disk Space Manager **podpira koncept** “strani” in podpira ukaze za dodeljevanje in sproščanje prostora na disku ter branje in pisanje strani

Upravljanje prostora na disku..

- Velikost strani je **enaka velikosti bloka** na disku, tako da se strani shranjujejo kot bloki na disku. Branje ali pisanje strani se tako lahko izvede v okviru ene I/O operacije
- Disk Space Manager **mora skriti podrobnosti** strojne opreme (in tudi operacijskega sistema) in omogočiti višjim plastem programske opreme (SUPB), da obravnava podatke kot zbirko strani

Upravljanje prostora na disku..

- Disk Space Manager **mora vzdrževati** stanje zasedenih in prostih blokov na disku (2 načina):
  - **seznam prostih blokov** (kazalec na prvi blok seznama se shrani na znano lokacijo na disku),
  - **vzdrževanje bitne mape** (za vsak blok je v bitni mapi bit, ki označuje, ali je blok zaseden ali ne)

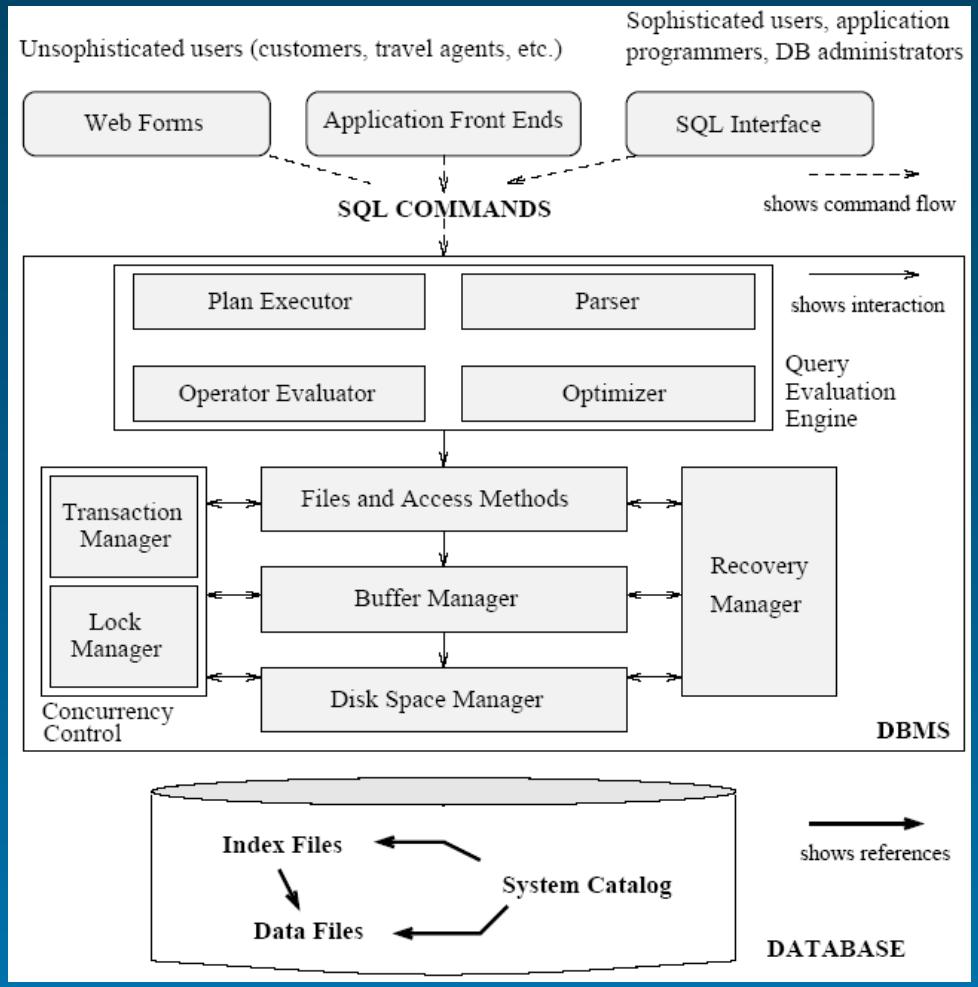
Upravljanje prostora na disku..

- Uporaba datotečnega sistema za upravljanje s prostorom:
  - Disk Space Manager lahko uporablja datoteke operacijskega sistema. Posledično se celotna PB se nahaja v eni ali več datotekah
  - V tem primeru je zadolžen za upravljanje prostora v teh datotekah

Upravljanje prostora na disku

- Nekateri SUPB **ne uporabljajo** datotečnega sistema, ampak svoj lastni sistem za upravljanje z diskom (ali pa obe možnosti):
  - Praktični razlog za to je, da bazo lahko uporabimo na več platformah (proizvajalec lažje portira SUPB na različne OS)
  - Tehnični razlog pa, da se pri 32 bitnem naslavljanju pojavi omejitev v velikosti datoteke

Buffer Manager..



Buffer Manager..

- Za razumevanje vloge in pomena Buffer managerja je nazoren naslednji PRIMER:
  - Kaj če hočemo izvesti poizvedbo nad PB, ki ima 1.000.000 strani, v pomnilnik pa jih lahko spravimo le 1.000?
- Buffer Manager je programska plast, ki skrbi za **prenašanje ustreznih strani** v pomnilnik
- Buffer Manager **upravlja z razpoložljivim pomnilnikom** (buffer pool)

Buffer Manager..

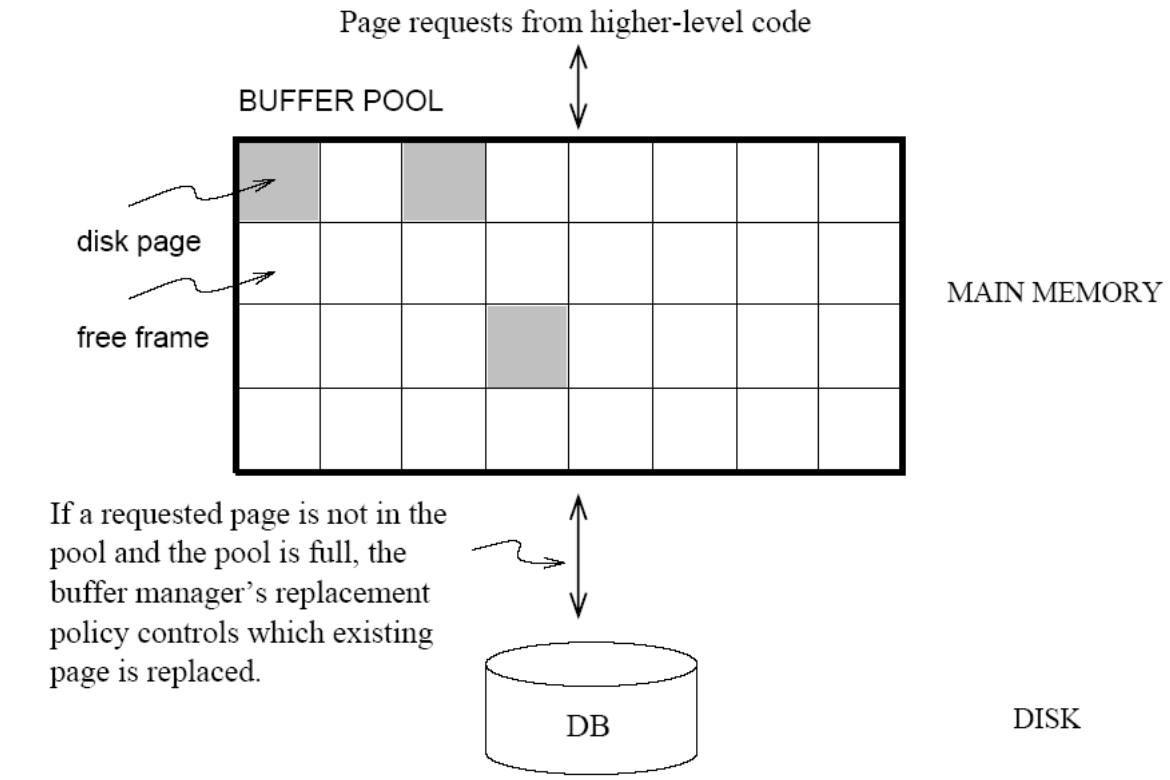


Figure 7.3 The Buffer Pool

Buffer Manager..

- Buffer Manager **zagotavlja** višjim plastem SUPB-ja strani, ki jih te rabijo za svoje delo
- Buffer Manager prenese v buffer pool tisto stran, katero je višja plast zahtevala za delo
- SUPB **mora obvestiti** Buffer Manager o tem, da je “sprostila” stran, ki je ne rabi več, ali pa da je stran ažurirala. Buffer manager je odgovoren za prenos na disk
- Pri odločanju o tem katere strani se bodo v pomnilniku zamenjale, se uporablja določena strategija (replacement strategy)

Buffer Manager..

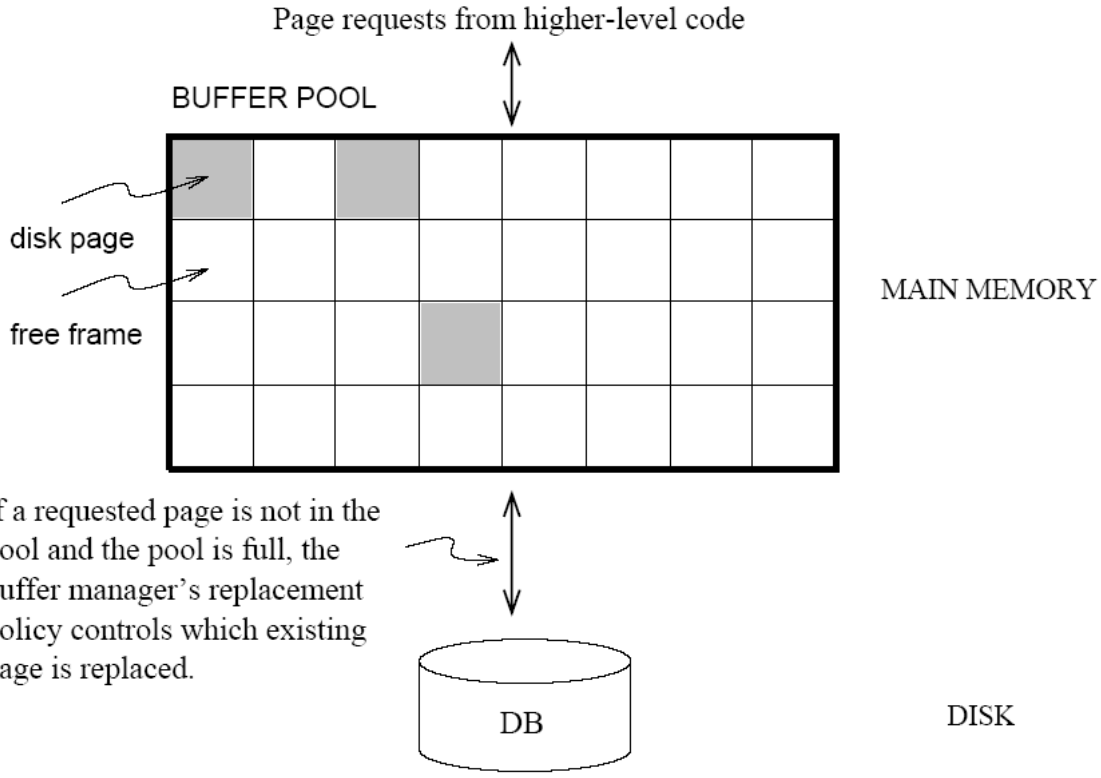


Figure 7.3 The Buffer Pool

Buffer Manager..

- Za vsak okvir v buffer pool-u se hrani 2 spremenljivki:
  - **pin_count**: kolikokrat je bila stran v okvirju zahtevana (+1) in kolikokrat sproščena (-1): število trenutnih uporabnikov strani
  - **dirty**: boolean vrednost, ki označuje, ali je bila stran spremenjena ali ne
- Na začetku je za vsak okvir pin_count=0 in dirty=off

Buffer Manager..

- Ko se pojavi zahteva po **določeni strani**, Buffer Manager izvede naslednje:
  - če se stran nahaja v kakšnem od okvirjev, vrne pomnilniški naslov okvirja in poveča pin_count za 1,
  - drugače pa izvede naslednje:
    - *izbere okvir za zamenjavo (z uporabo strategije za zamenjavo) in poveča pin_count*
    - *če je dirty bit okvirja, ki bo zamenjan "on", se stran prepiše na disk*
    - *stran se prenese iz diska v okvir, ki je določen za zamenjavo*

Buffer Manager..

- Če se zahtevana stran **ne nahaja** v buffer pool-u, in če so vsi okvirji zasedeni, se za zamenjavo izbere okvir, katerega pin_count=0. V primeru več takih okvirjev se izmed njih izbere okvir po določeni uporabljeni strategiji
- Če je dirty bit postavljen, potem je potrebno stran ob zamenjavi **zapisati nazaj na disk**, drugače pa se njena vsebina lahko prepiše z vsebino nove strani

Buffer Manager..

- Če v buffer pool-u ni nobene strani katera bi imela pin_count=0 in hkrati iskane strani ni v buffer pool-u, potem mora Buffer Manager počakati, da se sprosti zaseženje katere od strani in potem se lahko stran zamenja:
  - V praksi to pomeni, da je transakcija, ki zahteva tako stran lahko enostavno razveljavljena.

Buffer Manager

- Strategija zamenjevanja strani v buffer pool (buffer replacement policy. Najbolj uporabljena strategija je LRU – least recently used
- Buffer manager vodi seznam kazalcev na okvirje, ki imajo pin_count enak 0. Nov sproščeni okvir (sprosti se stran, ki je v okvirju) doda na seznam na koncu

Buffer Manager : Predpomnilnik OS..

- Rečemo lahko, da obstaja podobnost med navideznim pomnilnikom operacijskega sistema in upravljanjem s pomnilnikom pri SUPB
- Cilj obeh je zagotoviti dostop do več podatkov, kot pa jih lahko spravimo v pomnilnik. Ideja je, da se strani iz diska prenašajo v pomnilnik po potrebi in pri tem nadomeščajo strani, ki se jih v pomnilniku ne rabi več

Buffer Manager : Predpomnilnik OS..

- Zakaj ne uporabimo navideznega pomnilnika OS za potrebe SUPB?
  1. SUPB lahko bolj natančno predvidi zaporedje, v katerem se bo dostopalo do strani, kot pa tipičen OS
  2. SUPB rabi več nadzora nad stranmi, ki se zapisujejo na disk, kot pa ga omogoča tipičen OS

Buffer Manager : Predpomnilnik OS

- **Še bolj pomembno**: buffer manager uporablja strategijo “prefetching of pages”, ki omogoča predvidevanje več naslednjih zahtev in v skladu s tem se ustrezne strani prenesejo v pomnilnik, preden so dejansko zahtevane