Osnove podatkovnih baz

Iztok Savnik, FAMNIT

Osnovni podatki

- *Predavatelj:* *doc.dr.* Iztok Savnik
- *Asistent*: Uroš Sergaš
- *Vaje*: Avditorne in laboratorijske
- *Govorilne ure*: Po predavanju
- *Izpit*:
  - Pisni izpit
  - Domače naloge
  - Ustni izpit

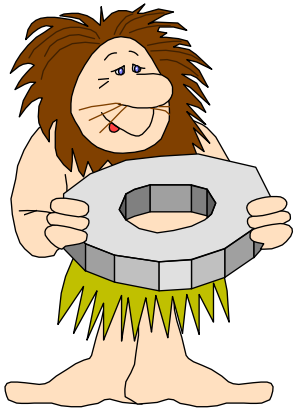
Literatura

- *Učbenik:*
  - Raghu Ramakrishnan, Johannes Gehrke, *Database Management Systems, McGraw-Hill, 3rd ed., 2007.*
- *Prosojnice:*
  - *Prosojnice so pripravljene na osnovi prosojnic in ostalega materiala „Cow Book“: R.Ramakrishnan, http://pages.cs.wisc.edu/~dbbook/*
- *Dodatna literatura:*
  - Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts,* 3rd ed., McGraw-Hill, 1997.

Ocenjevanje

- Pisni izpit – 40%
  - Štiri naloge iz vaj in predavanj
  - Čas 90-120 min
  - ≥ 50% !
- Domače naloge – 40%
  - 3 domače naloge
  - ≥ 50% !
- Ustni izpit - 20% !
  - Se izvede po uspešno opravljenih pisnem izpitu in domačih nalogah.
  - 3 vprašanja iz snovi predavanj in vaj
  - ≥50%
  - Končna ocena se tvori iz delnih ocen.

Kaj je SUPB?



- Sistem za Upravljanje s Podatkovnimi Bazami
- Velika, povezana zbirka podatkov.
- Modelira okolje v realnem svetu.
  - Entitete (e.g., osebe,studenti,predmeti...)
  - Razmerja (e.g., Tone je vpisan na PB)
- SUPB je programski sistem za shranjevanje podatkov in upravljanje s podatkovnimi bazami.

Datoteke vs. SUPB

- Aplikacije prenašajo velike količine podatkov med dinamičnim spominom in diskom.
- Posebna koda za posamezne poizvedbe.
- Zaščita podatkov pred nekonsistenco, ki je lahko posledica večih hkratnih uporabnikov.
- Zaščita pred izpadom sistema.
- Varnost in kontrola dostopa.

Zakaj uporabljati SUPB?

- Aplikacija mora brati/pisati velike količine podatkov iz diska
  - Vmesni pomnilnik, bloki, učinkovit dostop do podatkov.
  - Podatkovna neodvisnost.
- Zmanjšan čas razvoja aplikacije.
- Podatkovna integriteta in varnost.
- Uniformno administriranje podatkov.
- Hkraten dostop, transakcije, zaščita pred sistemskimi napakami.

Zakaj študij SUPB??

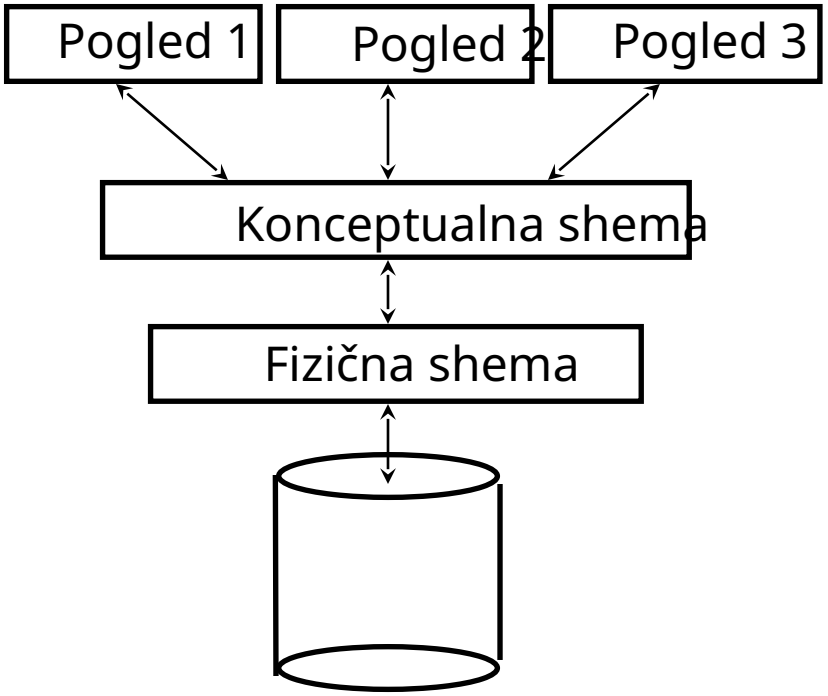
- Prehod iz računanja na informacije
  - “spodnja meja”: neurejene zbirke, podatkovna jezera, datotečni repozitorij (kaos!)
  - “zgornja meja”: iskanje na osnovi grafov znanja, znanstvene aplikacije, ...
- Podatkovne zbirke: narašča raznolikost in količina.
  - Digitalne knjižnice, interaktivni video, Genome projekt, grafi znanja, pregled neba (Sky survey), ...
  - ... potreba po SUPB zelo narašča.
- SUPB pokrivajo večino računalništva
  - OS, diski, jeziki, teorija, AI, multimedia, logika

Podatkovni modeli

- Podatkovni model je zbirka konceptualnih gradnikov za opis podatkov.
- Shema je opis konkretne zbirke podatkov z uporabo danega podatkovnega modela.
- Relacijski podatkovni model je najbolj pogosto uporabljan model danes.
  - Osnovni koncepti: relacija, ki je v osnovi tabela s stolpci in vrsticami.
  - Vsaka relacija ima shemo, ki opisuje stolpce in vrstice.

Nivoji abstrakcije

- Več pogledov, ena konceptualna shema in fizična shema.
  - Pogledi opisujejo kako uporabnik vidi podatke.
  - Konceptualna shema definira logično strukturo.
  - Fizična shema opisuje uporabljene datoteke in indekse.



- *Shema je definirana z uporabo DDL.*
- *Podatki se spreminjajo/poizvedujejo z DML.*

Primer: PB univerze

- Konceptualna shema:
  - *Študenti*(*sid: string, ime: string, login: string,*
    starost: integer, povprečje:real)
  - *Predmeti(pid: string, pime:string, točke:integer)*
  - *Vpis(sid:string, pid:string, ocena:string)*
- Fizična shema:
  - Relacije shranjene v neurejenih datotekah.
  - Indeks je definiran na prvem stolpcu relacije Študenti.
- Zunanja shema (Pogled):
  - *Predmet_Info(pid:string,vpisani:integer)*

Podatkovna neodvisnost *

- Aplikacije se ne ukvarjajo s tem kako so podatki strukturirani in shranjeni.
- *Logična podatkovna neodvisnost*: Zaščita pred spremembami v logični strukturi podatkov.
- *Fizična podatkovna neodvisnost*: Zaščita pred spremembami fizične podatkovne stukture podatkov.

☛ *Ena od najbolj pomembnih prednosti uporabe SUPB !*

Kontrola sočasnega dostopa

- Sočasno izvajanje uporabniških programov je bistveno za dobre performanse SUPB.
  - Dostop do diska je pogost in relativno počasen, zato je pomembno da je procesor ves čas zaposlen na večih sočasnih programih uporabnikov.
- Prekrivajoče se akcije različnih uporabniških programov lahko vodijo v nekonsistentnost podatkov. Na primer, ček se je vpisal, ni pa se še izračunalo stanje računa.
- SUPB zagotavlja, da do podobnih problemov ne bo prišlo: uporabniki lahko mislijo, da delajo na eno-uporabniškem sistemu.

Transakcija: Primer programa SUPB

- Osnovni koncept je *transakcija*, ki je *atomarna* sekvenca akcij SUPB (branje/pisanje).
- Vsaka transakcija, ki se izvrši celotna mora pustiti PB v konsistentnem stanju če je PB konsistentna ko se je transakcija začela izvajati.
  - Uporabniki lahko specificirajo enostavne *integritetne omejitve* nad podatki in podatkovna baza bo zagotovila veljavnost omejitev.
  - SUPB zares ne razume pomena podatkov (npr., ne razume kako se računajo obresti na bančnem računu).
  - Torej, zagotavljanje, da transakcija ohranja konsistentnost PB je odgovornost uporabnika !

Razvrščanje sočasnih transakcij

- SUPB zagotavlja, da je izvajanje množice trans. $\{T_1, \ldots, T_n\}$ ekvivalentno nekem *zaporednem* izvajanju $T_1' \ldots T_n'$.
  - Pred pisanjem/branjem transakcija:

    1) zahteva zaklepanje vseh objektov, ki se jih dotika in

    2) počaka, da SUPB omogoči zaklepanje teh objektov. Vsi zaklenjeni objekti se sprostijo na koncu izvjanja transakcije.

    (Strikten 2-fazni protokol zaklepanja.)

❖ Zaporedno izvajanje:

– Ideja: Če akcija T_i (pisanje X) vpliva na T_j (npr. bere X), bo ena izmed transakcij prva (npr. T_i) in bo druga (T_j) morala počakati dokler prva ne konča; s tem je definirana urejenost transakcij.

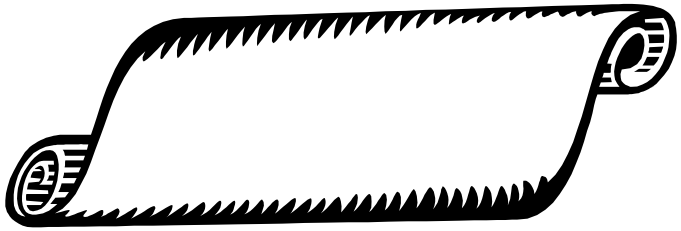
❖ Smrtni objem:

- Kaj če T_j že ima zaklenjen Y and T_i kasneje zahteva zaklepanje Y? (Smrtni objem - deadlock!) T_i ali T_j se ustavi in potem ponovno starta!

Zagotavljanje atomičnosti

- SUPB zagotavlja *atomičnost* (vse-ali-nič) četudi se zgodi sistemska napaka ali fizičen izpad sistema sredi transakcije.
- Ideja: Hranjenje dnevnika (zgodovine) vseh akcij, ki jih izvaja SUPB medtem, ko izvaja množico transakcij:
  - Pred vsako spremembo PB se najprej vpiše ustrezen zapis v dnevnik, pri tem pa zagotovimo, da se je zapis res fizično zapisal na disk. Podatkovna baza se spremini lahko kasneje. (*WAL protokol*; OS podpora često ni zadosti dobra)
  - Po sistemski napaki se vpliv delno izvršenih transakcij izniči tako, da se izničijo vse izvršene akcije transakcij. (WAL protokol zagotavlja, da ni zapisa v dnevniku, če ni sprememba že izvršena v PB!)

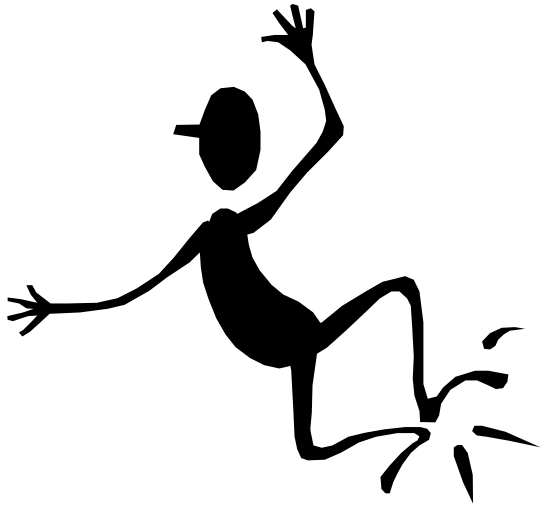
Dnevnik (Log)



- Naslednje akcije se zapisujejo v dnevnik:
  - T_i *popravi objekt*: Stara in nova vrednost objekta.
    - Zapis dnevnika mora biti shranjen na disku pred podatkovno stranjo, ki vsebuje podatke objekta!
  - T_i *zaključi/prekine*: Zapis dnevnika shrani akcijo.
- Zapisi dnevnika so povezani med sabo z uporabo Xact ID; enostavno je vzpostaviti stanje pred Xact (npr., da se reši smrtnega objema).
- Dnevnik se pogosto podvoji zaradi varnosti in se praviloma arhivira na persistentnem mediju.
- Vse aktivnosti dnevnika (aktivnosti za kontrolo sočasnega dostopa, zaklepanje/odklepanje, smrtni objemi, ...) izvaja SUPB transparentno.

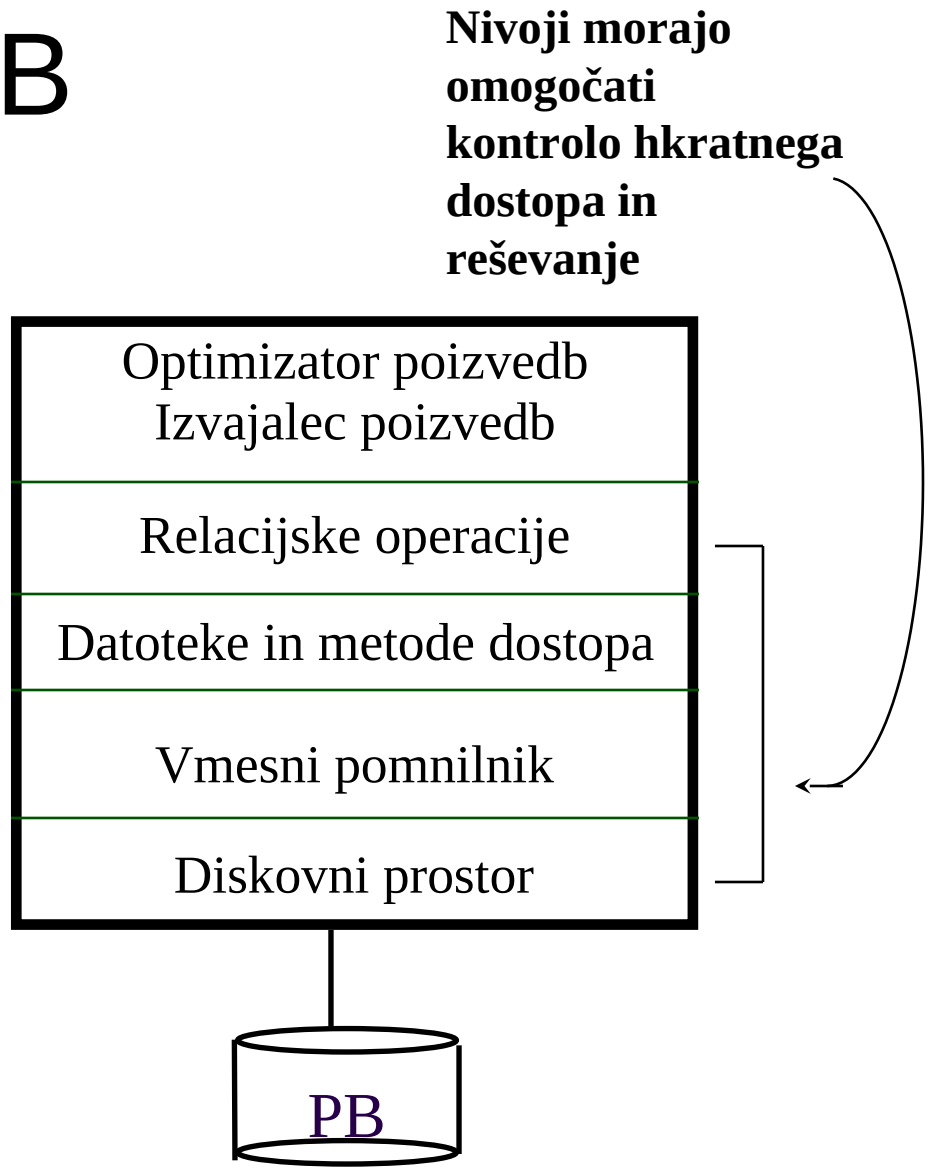
Baze osrečujejo sledeče ljudi ...

- Prodajalci SUPB.
- Razvijalci.
  - Načrtovalci informacijskih sistemov.
  - Programerji.
  - Webmaster.
- Končni uporabniki.
  - Uporabniki inf. Sistemov, vnašalci...
- Adminstrator SUPB (DBA)
  - Načrtovanje logiče/fizične sheme.
  - Varnost in autorizacija.
  - Dostopnost podatkov, vzpostavitev sistema po sistemski napaki.
  - Umerjanje SUPB po potrebi uporabnikov.

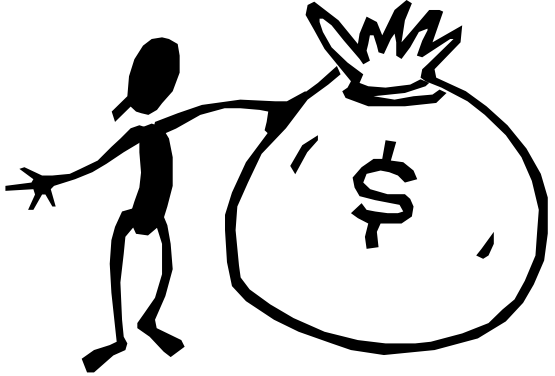


Struktura SUPB

- Tipični SUPB ima nivojsko arhitekturo.
- Slika ne prikazuje komponent za kontrolo hkratnega dostopa in recovery.
- To je samo ena izmed bolj pogostih arhitektur; vsak sistem ima svoje variacije.



Povzetek



- SUPB se uporablja za rokovanje (spreminjanje, poizvedovanje) velikih zbirk podatkov.
- Prednosti so ohranitev konsistentnega stanja PB po sistemskim napakam, hkraten dostop do SUPB, hiter razvoj aplikacij, podatkovna integriteta in zaščita.
- Nivoji abstrakcije nudijo podat. neodvisnost.
- SUPB tipično ima nivojsko arhitekturo.
- Adminstrator PB ima odgovorno službo in je dobro plačan ! ☺
- SUPB je eno od širših, bolj zanimivih področij računalništva.