**NMBP ZI skripta (nosql 1-3 teorija) + sitno o tokovima**

Skripta je skraćena teorija uvoda u noSql (prve tri prezentacije drugog ciklusa), namijenjena za ponavljanje gradiva. Za učenje bi trebalo koristiti prezentacije, u nedostatku boljih izvora 😊

1. Transakcije i obnova baze podataka

SUBP – zaštita podataka -> integritet, pristup podacima,

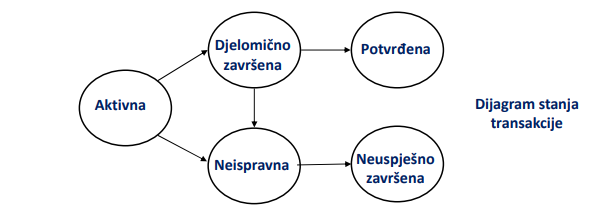
osigurava potporu za upravljanje transakcijama (obnova u slučaju greške, kontrola paralelnog pristupa)

TRANSAKCIJA – jedinica rada nad bazom podataka

Begin work -> Commit work (potvrda) / Rollback work (poništavanje svih izmjena)

* Upravljač transakcijama -> obavljanje transakcija i zadovoljavanje svih pravila integriteta
* ako granice transakcije nisu eksplicitno definirane s BEGIN/COMMIT/ROLLBACK, svaka SQL naredba smatra se transakcijom za sebe

STANJA TRANSAKCIJE



Djelomično završena – nakon što je obavljena zadnja operacija

Neuspješno završena – nakon što su poništeni svi efekti i baza vraćena u stanje prije početka transakcije

POTVRĐIVANJE TRANSAKCIJE

Točka potvrđivanja – sve izmjene koje je transakcija napravila postaju trajne, prije toga su tentativne (privremene)

**ACID**

Atomicity – transakcija se obavlja u cijelosti ili se ne obavlja

Consistency – transakcijom baza prelazi iz jednog konzistentnog stanja u drugo

Isolation – kada se paralelno obavljaju dvije ili više transakcija, učinak mora biti kao da su se obavljale jedna iza druge

Durability – ako je transakcija završila, njezini efekti ne smiju biti izgubljeni u slučaju kvara

OBNOVA BAZE PODATAKA

* dovesti bazu podataka u najnovije stanje za koje se pouzdano zna da je bilo ispravno
* korisnik periodički stvara *backup*

Redundancija – svaki podataka se mora moći rekonstruirati iz nekih drugih informacija redundantno pohranjenih negdje u drugdje u sustavu (dnevnici izmjena, zrcaljenje, sigurnosne kopije)

Postupak obnove:

1. Periodičko kopiranje baze podataka na arhivski medij
2. Svaka izmjena u bazi evidentira se u logičkog dnevniku izmjena

Write-Ahead Log rule – izmjena se prvo zapisuje u dnevnik, a tek onda se provodi

Tipovi pogrešaka – pogreške transakcija, računalnog sustava, kvar medija za pohranu

Zrcaljenje – dva jednaka područja (primarno i zrcalno), istovremeno se provode promjene

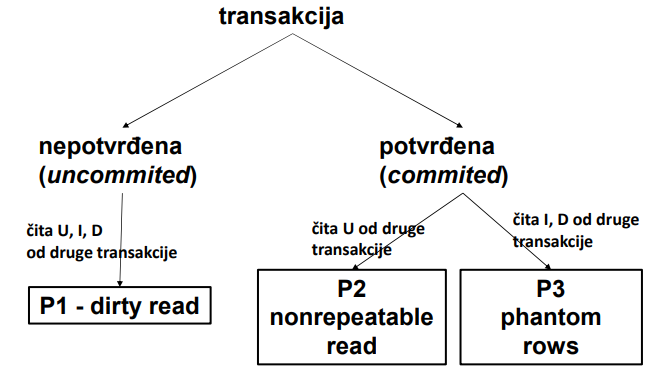
* ako se dogodi pogreška -> rad na ispravnog području, popravljanje zrcalnog, sinkronizacija
* ne pomaže pri poništavanju transakcija

Arhiviranje – obavlja se tijekom rada korisnika

* inkrementalno -> u razinama: kopija cijele baze / tjedna arhiva / dnevna arhiva

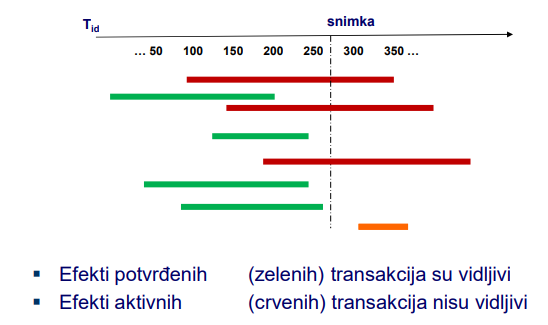
**IZOLACIJA (ISOLATION)**

* svaki redoslijed izvršavanja operacija koji ne narušava konzistentnost baze -> SERIJALIZABILAN
* problemi istodobnog pristupa: prljavo čitanje, neponovljivo čitanje, sablasne n-torke, izgubljena izmjena



Implementacija serijalizabilnosti:

1. Trivijalno – serijsko izvođenje
2. 2PL – Two Phase Locking (lose performanse)
3. Prije obavljanja operacije nad objektom, transakcija za objekt zatraži ključ (growing phase)
4. Nakon otpuštanja ključa transakcija više ne smije zatražit nikakav ključ (shrinking phase)
5. Multiverzijski protokol kontrole istodobnog pristupa – MVCC
   * + Transakciji se dodjeljuje identifikator
     + SUBP održava višestruke fizičke kopije jednog logičkog objekta u bazi
     + Kada transakcija čita iz objekta, SUBP čita najnoviju verziju zapisa koja je potvrđena, a kada piše u objekt, SUBP stvara novu fizičku verziju logičkog objekta
     + Pisanje ne blokira čitanje, čitanje ne blokira pisanje. Pisanje blokira pisanje
     + xmin – id transakcije koja je stvorila objekt (postgres)
     + xmax – id transakcije koja je obrisala objekt
     + snimka (snapshot)



------------------------------------------------------------------------------------------



* razine izolacije: kompromis između konzistentnosti i performansi
* odnose se na čitanje, kod pisanja podaci se zaključavaju kod svih protokola do kraja transakcije



Read Committed – snimka na početku svake SQL naredbe

Serializable – snimka na početku transakcije

2. NoSQL

Skalabilnost – mogućnost sustava da se nosi s rastućom količinom podataka (da zadrži prihvatljive performanse)

* + vertikalna -> dodavanje resursa jednom čvoru
  + horizontalna -> dodavanje čvorova u sustav

DISTRIBUIRANE BAZE PODATAKA

* skup logički povezanih baza podataka razmještenih u različitim čvorovima računalne mreže
* distribuiranost transparetna prema korisnicima
* mogućnost obavljanja i lokalnih i globalnih transakcija

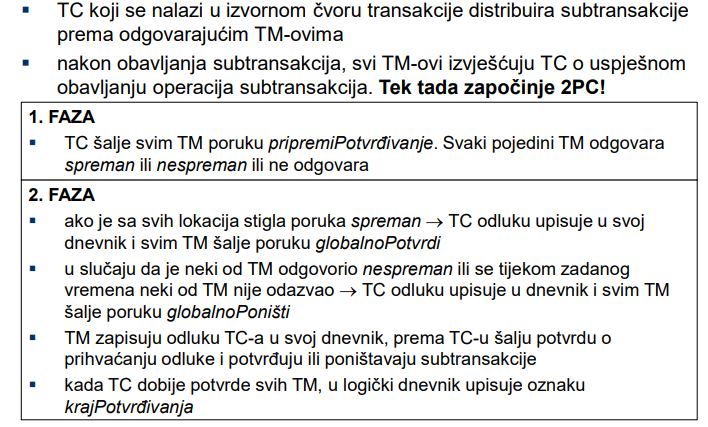
**oblikovanje distribucije = fragmentacija + alokacija**

* + shema fragmentacije – podjela baze u disjunktni skup fragmenata od kojih se može rekonstruirati baza bez gubitka informacije
  + shema alokacije – opisuje koji fragment je pridružen kojem čvoru distribuiranog sustava
    - stupanj replikacije fragmenta – broj čvorova u kojima je fragment alociran

1. Particionirana BP – svaki fragment alociran u točno jednom čvoru (stupanj = 1)
2. Potpuno replicirana BP – svaki fragment u svim čvorovima (stupanje = n, n – broj čvorova)
3. Parcijalno replicirana – svaki fragment u jednom, više ili svim čvorovima

Globalna transakcija – skup subtransakcija koje koordinirano izvršavaju SUBP-ovi u više čvovora

* atomarnost -> ili su sve subtransakcije obavljene ili nijedna
  + protokol: 2PC (Two Phase Commit) – u svakom čvoru menadžer i koordinator transakcija



* protokol je blokirajući ako postoji mogućnost da ispravan čvor neće moći završiti transakciju zbog prekida veze ili kvara nekog drugog čvora
* protokol je nezavisan s obzirom na mogućnost obnove ako svaki čvor (TC ili TM), nakon što se u njemu dogodio kvar sustava (ili medija) može samostalno, bez komunikacije s ostalim čvorovima, donijeti odluku o sudbini svih (sub)transakcija koje su se u tom čvoru odvijale u trenutku kvara
* kvarovi u DSUBPu -> prestanak rada jednog ili više čvora, gubitak veze, poruka, podjela mreže

REPLICIRANE BAZE PODATAKA

* fragment je repliciran ako je alociran u dva ili više čvorova
* prednosti: povećanje dostupnosti, smanjenje volumena prijenosa podataka, paralelno obavljanje dijelova istog upita
* nedostaci: problem konzistentnosti kopija istog elementa
* sinkroni protokoli -> sve kopije moraju se izmijeniti u okviru inicijalne transakcije
* asinkroni protokoli -> operacije inicijalne transakcije obavljaju se isključivo u inicijalnom čvoru, ne ovise o ostalima
* jednosmjerni protokoli -> za svaki logički element x određuje se samo jedan fizički element xp koji se proglašava primarnom kopijom, te se operacija izmjene koju nad elementom x obavlja inicijalna transakcija mora prvo obaviti nad kopijom xp
* dvosmjerni protokoli -> inicijalna transakcija može operaciju izmjene obaviti nad bilo kojom fizičkom kopijom

NOSQL

* specijalizirana rješenja za određene grupe problema
* agregatni model -> dozvoljava liste i gniježđenje drugih zapisa
  + jedan agregat – jedna transakcija
  + podatci u JSON formatu
* 4 glavna modela:

1. Ključ-vrijednost – CRUD operacije (Riak baza)
2. Ključ-dokument – CRUD + dohvat na temelju sadržaja dokumenta (JSON)
3. Column family – dvorazinska mapa (ključ retka, ključ stupca), svaki stupac član familije stupaca
4. Graf baze podataka – čvorovi imaju svojstva, bridovi oznaku, smjer, početni i odredišni čvor (+ mogu imati svojstva)

* Materijalizirane virtualne relacije -> upiti izračunati unaprijed, pohranjeni na disku
  + - Eager – ažuriranje kod unosa
    - Stale – ažuriranje nakon unosa, periodički, asinkrono
* noSql baze podataka nemaju shemu, implicitno postoji u aplikacijskom kodu

Tokovi podataka

Događaj - maleni, samostalni, nepromjenjivi zapis nečega što se dogodilo

Tok - slijed događaja koji pristižu dinamički tijekom vremena nepoznate duljine

- prijenos preko mreže:

1.) Sustavi poruka

Proizvođač - proizvodi događaje

Potrošač - prima događaje

- dva pristupa:

a) direktna komunikacija

- kad gubitak poruke nije jako važan

b) posrednički sustavi

- baze koje upravljaju prijenosom poruka

- posrednik poslužitelj, proizvođači i potrošači

klijenti, asinkrona komunikacija

- obrasci kod više potrošaća:

1. Load balancing

2. Fan out

dnevnički posrednički sustavi

- proizvodači dodaju u dnevnik, potrošaći čitaju

- dnevnik u particije

- anatomija teme -> više pretplatnika

- obrada događaja:

a) Event time / vrijeme događaja - trenutak kad se događaj dogodio

b) Processing time / vrijeme obrade - trenutak kad je uočen u sustavu

c) Ingestion time - vrijeme kad je događaj došao u sustav

- obrasci obrade podataka:

1.) Vremenski agnostičko - filtriranje (vrijeme nije bitno)

2.) Aproksimacije - sampling, sketching

Vremenski prozori - grupiranje zapisa po temporalnoj domeni u konačne skupine

a) fixed

b) hopping

c) sliding window

d) session window (w - dinamička)

3.) Pomoću prozora u vremenu obrade / processing time windowing

- grupiranje zapisa u vremenske prozore

4.) Pomoću prozora u vremenu događaja / event time windowing

Okidači (triggers) - kada generirati izlaz za neki prozor (u procesorskom vremenu), svaki izlaz

za neki prozor -> okno prozora

1.) Ponavljajući ažurirajući okidači - nakon N novih zapisa ili T procesorskog vremena

2.) Okidači dovršenosti -> Watermarks

Vodostaj (watermark) - monotono rastuća vremenska oznaka najstarijeg neobrađenog posla

a) idealni - ako imamo savršeno poznavanje ulaznih podataka

b) heuristički - postoje zakašnjeli podaci

- nedostaci: prespor ili prebrz

- rješenje: RU triggers + Watermark trigger (EOL) -> 0-N preuranjenih, 1 pravovremeno, 0-N

zakašnjelih okna

- dopušteno kašnjenje -> malo broj prozora besk., savršeni watermark 0

Akumulacija

1.) Odbacivanje - svako okno nezavisno od prethodnog

2.) Akumuliranje

3.) Akumuliranje i opozivanje

- otpornost na pogreške