Napredne baze podataka

Transakcije

Tehničko veleučilište u Zagrebu
Informatičko-računarski odjel

Sadržaj predavanja

- Što su transakcije
- Svojstva transakcija
- Pogreške u transakciji
- Obnova u slučaju razrušenja

•Uvod

- Situacija (MySQL)
  - Naredbe nad bazom najčešće se izvode slijedno
  - Pridodavanje male pažnje onome što se u bazi trenutno dogodilo ili će se dogoditi
  - MySQL izvršava svaku naredbu kao samostalnu jedinicu
- Cilj
  - Ako se određena naredba ne izvrši, sljedeću nema smisla izvršavati
  - Robusnost aplikacija
- Potreba
  - Transakcijski model
  - Mogućnost grupiranja niza SQL naredbi koje će se izvršavati zajedno

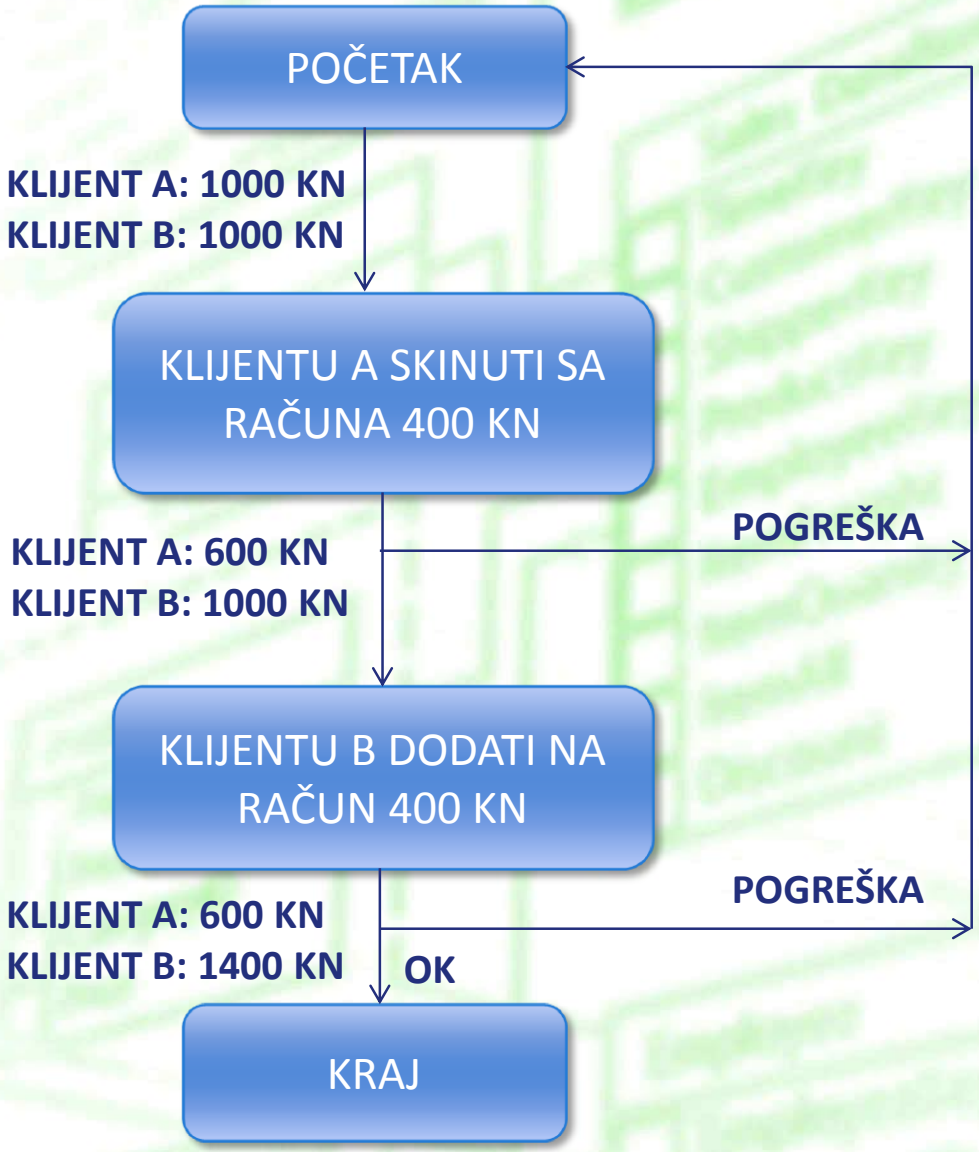
Transakcija

- *Slijed operacija (čitanja i pisanja podataka) koje jedan korisnik ili aplikacija provodi nad bazom podataka*
- Jedinica rada nad bazom podataka
- Logički se mora provesti kao **nedjeljiva cjelina**
  - Sastoji se od niza logički povezanih izmjena
- „Svi za jednog, jedan za sve”
  - Pojedinačne operacije unutar transakcije nisu bitne same za sebe
- Svaka transakcija unosi promjenu u bazi

•Primjer

Dijagram toka

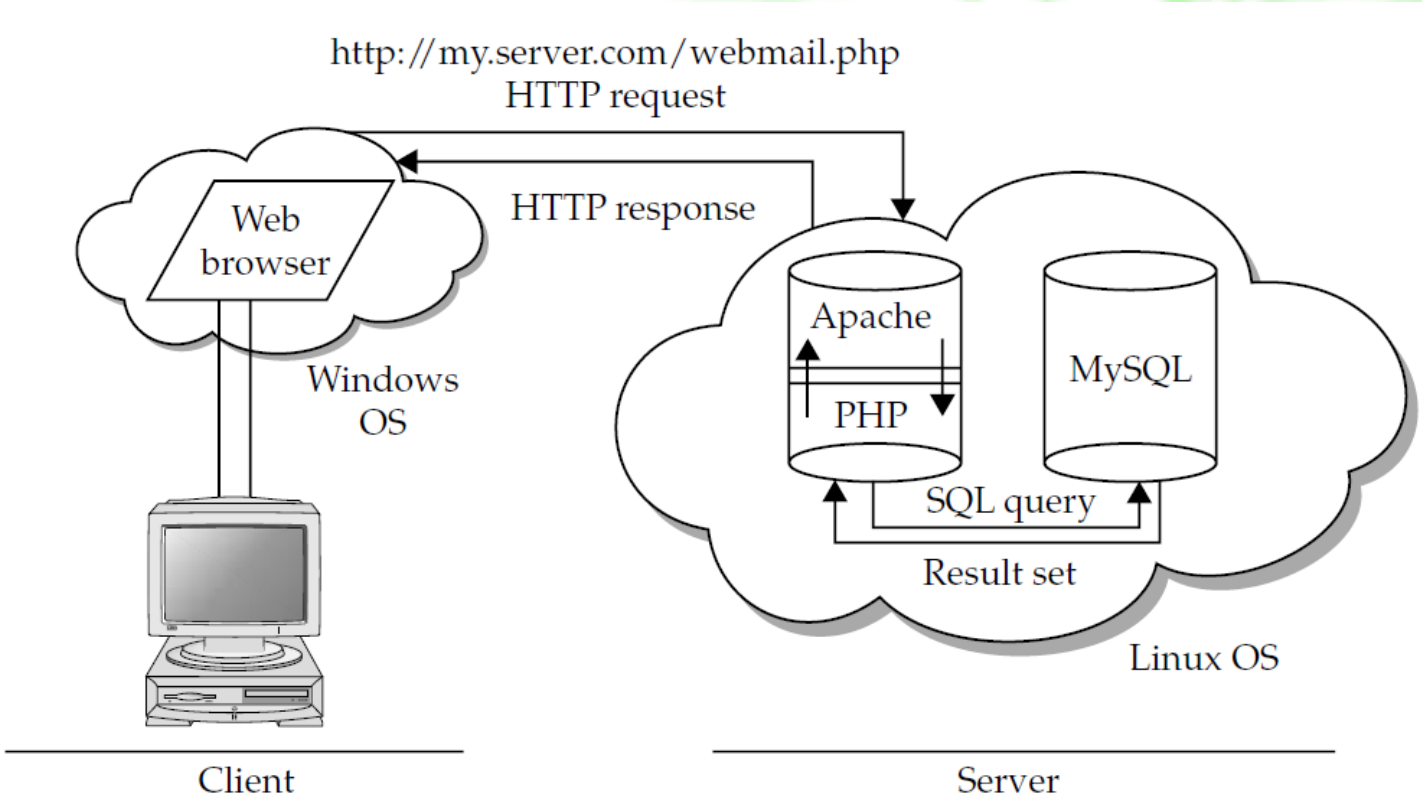
- klijent A uplaćuje klijentu B 400 kn na bankovni račun
- ne smije se dogoditi da se novac ‘izgubi’ ili da se ‘pojavi višak’



•Web aplikacije

MySQL u funkciji podrške web stranicama i aplikacijama na webu

- Unapređenje transakcijske učinkovitosti
- Reduciranje vremena odaziva
- Poboljšanje korisničke pristupačnosti
- Transakcijske aplikacije vs. aplikacije za potporu poslovnom odlučivanju



•Načelo “sve ili ništa”

- Svaka jedinica posla provedena nad bazom podataka mora biti:
  - **provedena u cijelosti**

    ILI
  - **ne smije biti provedena uopće**

DA BI SE OČUVAO INTEGRITET BAZE

- Transakcija koja iz bilo kojeg razloga nije do kraja bila obavljena morala bi biti ***neutralizirana*** *– svi podaci koje je ona do trenutka prekida promijenila morali bi natrag dobiti* ***svoje polazne vrijednosti***

- Neizvršavanje transakcije u potpunosti i neizvršavanje neutralizacije -> *jedan od najčešćih slučajeva pogreške u bazi!*

Granice transakcije

- Početak transakcije
  - **`START TRANSACTION ili BEGIN WORK`**
- Završetak transakcije
  - **`COMMIT WORK`**
    - Uspješan završetak, pohrani promjene
    - Sve promjene učinjene prilikom transakcije mogu se spremiti i tada postaju vidljive
  - **`ROLLBACK WORK`**
    - Neuspješan završetak, ne snimaj promjene i poništi sve do sada napravljeno od riječi START ili BEGIN
    - Kao da se promjene nisu ni dogodile
    - Koristi se za prekid transakcije unutar koje je uočena pogreška
    - Slično se obavlja automatski kada je transakcija prekinuta izvana

Granice transakcije

- Eksplicitno
  - BEGIN WORK
  - ….
  - COMMIT WORK ili ROLLBACK WORK
- Implicitno
  - **Svaka operacija izmjene** u bazi podataka predstavlja transakciju, njezin uspješan završetak predstavlja COMMIT, a neuspješan ROLLBACK
- Transakcije se **ne mogu** ugnježđivati
  - Jedna aplikacija u jednom trenutku može imati aktivnu samo jednu transakciju
- Transakcije ne podržavaju opoziv (rollback) **DDL** naredbi

•Preduvjeti

- Ako se želi da više operacija (SQL naredbi) predstavlja jednu transakciju, potrebno je:
  - Postaviti varijablu **AUTOCOMMIT** na 0 (CJELOVITOST)
    - Vrijednost **AUTOCOMMIT inicijalno je postavljena na 1 (MySQL)**
    - Pretpostavljeni (*defaultni*) način rada je takav da se svaka naredba poziva i izvršava posebno!
  - Koristiti storage engine koji podržava transakcije
    - Alternativa InnoDB mehanizmu pohranjivanja su PSEUDOTRANSAKCIJE – imitacija transakcija u tablicama koje ne podržavaju ACID transakcije koristeći zaključavanje tablica/stranica/redaka

SET AUTOCOMMIT=0

InnoDB

Implicitno pohranjivanje ili opoziv

- Svaka sesija pokreće se s postavljenim *autocommit* načinom rada
- U slučaju kada se onemogući *autocommit* način (SET AUTOCOMMIT=0;) – ako se sesija nenadano prekine, MySQL automatski napravi opoziv transakcije
- Neke naredbe ne mogu biti opozvane jer se njihovim izvođenjem izvodi implicitno potvrđivanje promjena (commit)
  - CREATE DATABASE, CREATE TABLE, DROP DATABASE, DROP TABLE, ALTER TABLE, CREATE FUNCTION, ALTER FUNCTION,…
  - Pravilo koje nije univerzalno ako se radi o ključnoj riječi TEMPORARY
  - Mnoge od njih implicitno završavaju i transakciju koja je u tijeku (kao da je napravljen commit prije izvršavanja naredbe)
    - http://docs.oracle.com/cd/E17952_01/refman-5.1-en/implicit-commit.html

•Primjer

Potrebno je radniku Petru Kruljcu povećati koeficijent plaće za 0.5, te istovremeno osigurati smanjenje koeficijenta za 0.5 radniku Dini Parlovu.

```
SET AUTOCOMMIT = 0;
BEGIN;
UPDATE radnik
      SET koefPlaca=koefPlaca+0.5
      WHERE prezimeRadnik = 'Kruljac' AND
      imeRadnik='Petar';
UPDATE radnik
      SET koefPlaca=koefPlaca-0.5
      WHERE prezimeRadnik = 'Parlov' AND
      imeRadnik='Dino';
COMMIT; /*(ili ROLLBACK ako je potrebno)*/
SET AUTOCOMMIT = 1;
```

Točke pohranjivanja

- **`SAVEPOINT imeTockePohranjivanja`**
  - Postavljanje točke pohranjivanja kako bi se kasnije mogao napraviti opoziv do te točke
- **`ROLLBACK TO SAVEPOINT imeTockePohranjivanja`**
  - Napravi opoziv svih naredbi nakon imenovane točke pohranjivanja
- **`RELESE SAVEPOINT`**
  - Briše postavljenu točku pohranjivanja

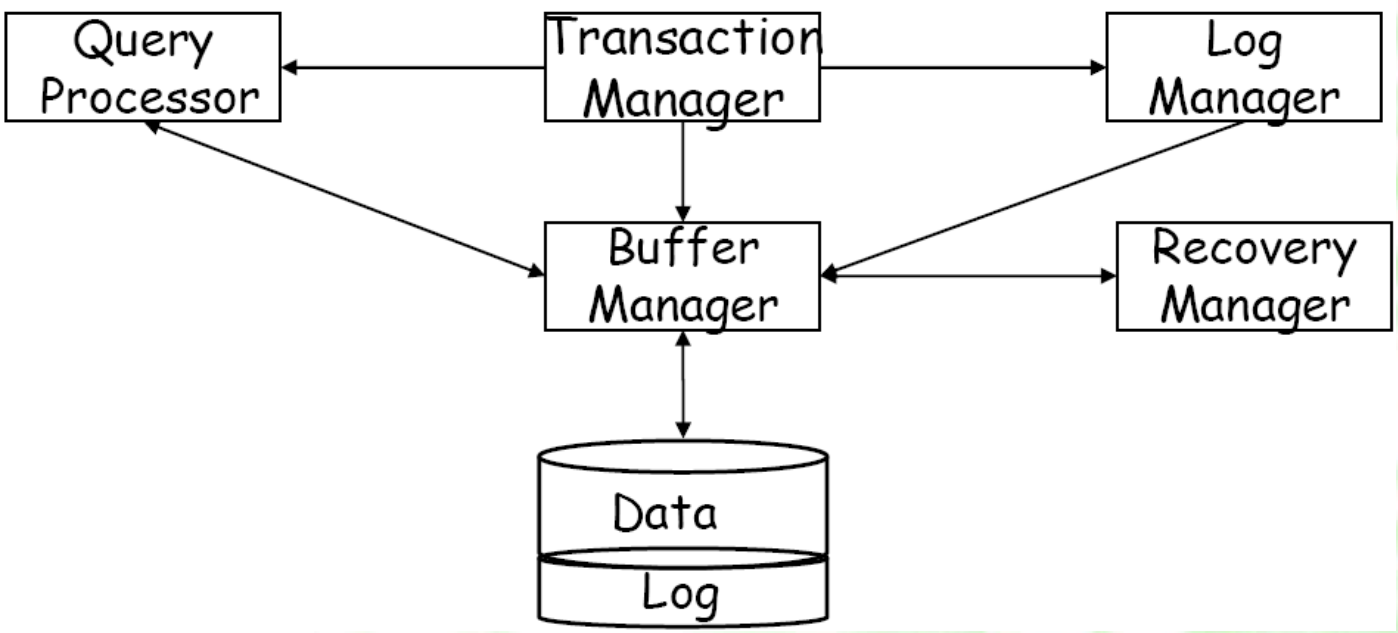
•Primjer

```
SET AUTOCOMMIT = 0;
BEGIN;
UPDATE radnik
       SET koefPlaca=koefPlaca+0.5
       WHERE prezimeRadnik = 'Kruljac' AND imeRadnik='Petar';
SAVEPOINT tocka;
UPDATE radnik
       SET koefPlaca=koefPlaca-0.5
       WHERE prezimeRadnik = 'Parlov' AND imeRadnik='Dino';
ROLLBACK TO SAVEPOINT tocka;
SELECT * FROM radnik WHERE (prezimeRadnik = 'Kruljac' AND
  imeRadnik='Petar')
       OR (prezimeRadnik = 'Parlov' AND imeRadnik='Dino');
COMMIT;
SET AUTOCOMMIT = 1;
```

•Obavljanje transakcija

- Dio sustava koji brine o obavljanju transakcija -> **transaction manager** (transaction processing monitor – TP monitor)
- Osigurava zadovoljavanje svih poznatih pravila integriteta

osigurava da su akcije koje korisnici pokušavaju izvesti ispravne



•Stanja transakcija

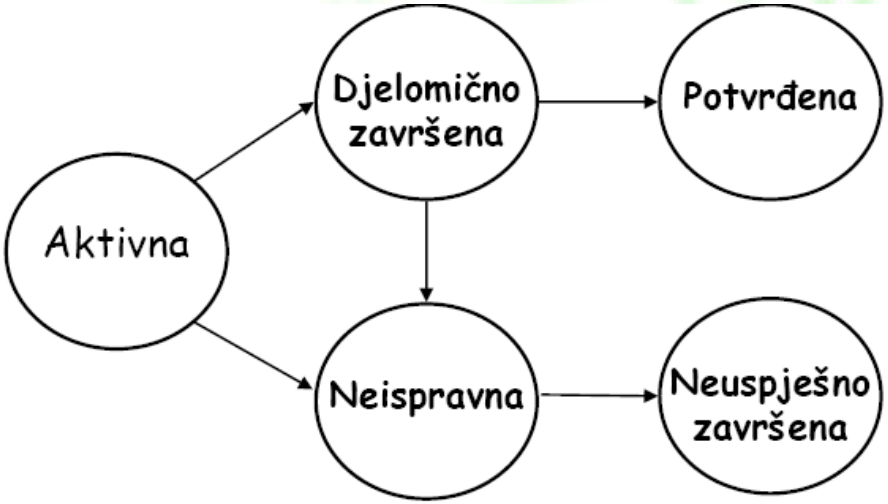
- Jedna transakcija s korisničkog stanovišta predstavlja jednu nedjeljivu cjelinu koja se obično realizira kao niz od nekoliko elementarnih zahvata u samoj bazi

- Glavno svojstvo -> prevodi **bazu iz jednog konzistentnog stanja u drugo**!

- **Međustanja** koja nastaju nakon pojedinih operacija unutar transakcije mogu biti **nekonzistentna**

Stanja transakcija

- **Aktivna (active)** – tijekom izvođenja
- **Djelomično završena (partially committed)** – nakon što je obavljena njezina posljednja operacija
- **Neispravna (failed)** – nakon što se ustanovi da nije moguće nastaviti njezino normalno izvođenje
- **Neuspješno završena (aborted)** – nakon što su poništeni njezini efekti i baza podataka vraćena u stanje kakvo je bilo prije nego što je započela
- **Potvrđena (committed)** – uspješno završena



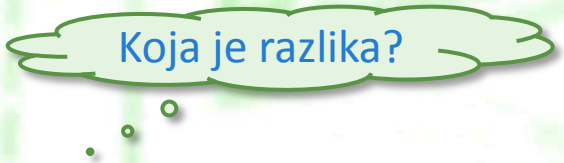
Stanja transakcija

- **Točka potvrđivanja** (commit point)
  - Sve izmjene koje je transakcija napravila postaju permanentne
  - Sve izmjene koje je transakcija načinila prije točke potvrđivanja mogu se smatrati pokušajima
  - U točki potvrđivanja otpuštaju se svi ključevi
- Kod ulaska u transakciju *otpuštaju se sva zaključavanja* ako su pokrenuta (vidi kasnije predavanja koja se bave zaključavanjima n-torke ili relacije)
- **Potvrđena izmjena nikad ne može biti poništena**
  - sustav garantira da će njezine izmjene biti permanentno pohranjene u bazi podataka, čak i ako kvar nastane već u sljedećem trenutku

Svojstva transakcija

Svojstva transakcija

- ACID
  - Atomicity - atomarnost
  - Consistency - konzistentnost
  - Isolation - izolacija
  - Durability - izdržljivost
- Svojstva koja osigurava DBMS (RDBMS)
- Odnose se na pravila održavanja integriteta i konzistentnosti baze podataka



•ACID

Atomicity - atomarnost

- **Transakcija se mora obaviti u cijelosti ili se uopće ne smije obaviti**

- Transakcija može biti neizvršena u cijelosti u tri slučaja:

1. Prekinuta ili neuspješno završena
   - Zbog pogrešaka pri izvođenju
   - Ako je prekinuta od strane DBMS-a, bit će resetirana i pokrenuta ispočetka
2. Prilikom pada sustava
   - Primjerice prilikom nestanka struje u vrijeme izvođenja transakcije
3. Nepredviđene situacije
   - Neočekivane vrijednosti podataka koji se unose
   - Nemogućnost pristupa memoriji (disku)

•ACID

Consistency - konzistentnost

- **Transakcijom, baza podataka prelazi iz jednog konzistentnog stanja u drugo konzistentno stanje**

- Korisnik koji potvrđuje transakciju mora osigurati da će transakcija ostaviti bazu podataka u ispravnom stanju

- Primjer
  - Ako definiramo prebacivanje određenog iznosa s računa klijenta A klijentu B, nakon što oduzmemo iznos s računa A, moramo ISTI iznos dodati računu B (DBMS se neće sam za to pobrinuti)

•ACID

Isolation - izolacija

- **Kada se paralelno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge**

- Kako bi to osigurao, DBMS koristi posebne algoritme za organizaciju
  - Serial Scheduling

•ACID

Durability - izdržljivost

- **Ako je transakcija obavila svoj posao, njezini efekti ne smiju biti izgubljeni ako se dogodi kvar sustava, čak i u situaciji kada se kvar dogodi neposredno nakon završetka transakcije**

- Ako je do kvara došlo za vrijeme izvođenja transakcije, DBMS se mora pobrinuti za vraćanje u konzistentno stanje pomoću logova
  - Prije nego se odradi, svaka promjena na disku evidentirana je u logovima

•ACID primjer

Račun u banci (A i B)

- ostali upiti ne mogu vidjeti A ili B sve dok transakcija ne završi

-> Isolation

- jednom obavljen transfer, nema povratka (novac ostaje kod B)

-> Durability

- ne može se skinuti iznos s A ako se ne stavi na B

-> Atomicity

- novac se ne dobiva ni gubi

-> Consistency

•Serijabilnost

- Neka se u višekorisničkoj bazi podataka izvodi nekoliko transakcija paralelno tako da se pojedini dijelovi tih transakcija izvode vremenski izmiješano

- Ako je konačni učinak njihovog izvođenja isti kao da su se one izvršavale serijski (sekvencijalno), kažemo da se radi o **serijabilnom** (ili serijalizabilnom) izvršavanju transakcija

- Lokoti i protokol dvofaznog zaključavanja (kasnija predavanja)

Pogreške u transakciji

•Pogreška u transakciji?

```
BEGIN WORK;
   INSERT INTO mjesto VALUES(2, 'naziv1', 1);
   INSERT INTO mjesto VALUES(2, 'naziv2', 1);
COMMIT WORK;
```

- (1 row(s)affected)
- (0 ms taken)

- Error Code : 1062
- Duplicate entry '2' for key 'pbrMjesto‘
- (0 ms taken)

- **Izvršit će se sve SQL naredbe koje su izvedive!**
  - potrebno mu je deklarirati željeno ponašanje u slučaju neuspjeha

Tipovi pogrešaka

1. Pogreške koje otkriva sama aplikacija
2. Pogreške unutar transakcije kojima aplikacija ne rukuje na eksplicitan način (npr. dijeljenje s nulom)
3. Kvar računalskog sustava - baza nije fizički uništena
4. Kvar medija za pohranu - fizički uništena baza

- Slučajevi 1 i 2 ne smatraju se razrušenjem

- U tim slučajevima potrebno je osigurati metodu obnove

Uzroci razrušenja

- Kako spriječiti kvarove i greške koje se mogu dogoditi?

- Pogreške opreme
- Pogreške operacijskog sustava
- Pogreške DBMS-a
- Pogreške aplikacijskog programa
- Pogreške operatera
- Kolebanje izvora energije
- Sabotaža
- Prirodne katastrofe

Obnova u slučaju razrušenja

- Dovesti bazu podataka u najnovije stanje za koje se pouzdano zna da je bilo ispravno
- Velike baze podataka, dijeljene i *višekorisničke* moraju obavezno posjedovati mehanizme obnove
- Male *jednokorisničke* baze često nemaju mehanizme obnove već se u nekom vremenskom intervalu radi sigurnosna kopija podatka na čije se stanje vraća sustav u slučaju razrušenja

•Pogreške

- Pogreške koje otkriva aplikacija
  - Slučajevi u kojima aplikacija predviđa obavljanje naredbe ROLLBACK WORK
  - Poništavanje efekata transakcije
    - kao da transakcija nikad nije započela s radom
  - Poništavanje izmjena - pretragom dnevnika unatrag
    - nova vrijednost zapisa zamjenjuje se sa starom vrijednošću - sve dok se ne dođe do početka transakcije - BEGIN WORK

- Pogreške koje ne otkriva aplikacija
  - ako se javi pogreška za koju nismo pretpostavili akciju – program završi na neplanirani način
  - u tim slučajevima ne postoji naredba ROLLBACK WORK !
    - Poništavanje efekata transakcije, poništavanje poruka
  - Primjer: pokušaj unosa zapisa čiji ključ već postoji u bazi !

Oporavak nakon razrušenja

Način zapisivanja podataka u bazu

- Način zapisivanja
  - Koriste se međuspremnici (buffer)
    - Međuspremnik dnevnika
    - Međuspremnik baze podataka
- Sadržaj međuspremnika zapisuje se u dnevnik/bazu podataka
  - kad je popunjen
  - kada SUBP izda nalog
- Ako kvar nastane nakon potvrđivanja i prije nego što su izmjene iz memorijskih spremnika prebačene u bazu podataka, izmjene bi u času kvara bile izgubljene. Međutim:
  - procedura za ponovno pokretanje provest će promjene u bazi podataka
  - vrijednosti koje je potrebno zapisati u bazu podataka pronalaze se u odgovarajućim zapisima u dnevniku izmjena
  - dnevnik mora biti zapisan prije nego što završi procedura potvrđivanja – write-ahead log rule

Dnevnik baze podataka
Žurnal podataka
Transaction log

- baza nije razrušena - sve transakcije koje su se odvijale u vrijeme kvara moraju biti poništene jer nisu kompletne!
- pretraživanjem dnevnika od početka identificiraju se transakcije za koje postoji BEGIN i ne postoji COMMIT ili ROLLBACK
- takav postupak predugo bi trajao -> uvode se **kontrolne točke (checkpoint)**
  - MySQL baze nazivaju ih **broj dnevnika**
- u određenim intervalima povećava se broj dnevnika; moguće ga je postaviti naredbom **flush logs** ili će ga sustav sam promijeniti

Proces obnove

- Stvara se lista za poništavanje - na početku sadrži sve transakcije koje su bile aktivne u kontrolnoj točki
  - Lista za ponovno obavljanje - na početku je prazna
  - Pretražuje se dnevnik od kontrolne točke
    - transakcija za koju se pronađe BEGIN dodaje se u listu za poništavanje
    - transakcija za koju se pronađe COMMIT prebacuje se iz liste za poništavanje u listu za ponovno obavljanje
  - Ponovno se obavljaju transakcije iz liste za ponovno obavljanje
  - Poništavaju se transakcije iz liste za poništavanje
- Redo logs & undo logs
- DBMS ne može prihvatiti ni jedan zahtjev dok se ne završi proces obnove!