

Predavanja

Svibanj, 2021.



Zadaće sustava za upravljanje bazama podataka

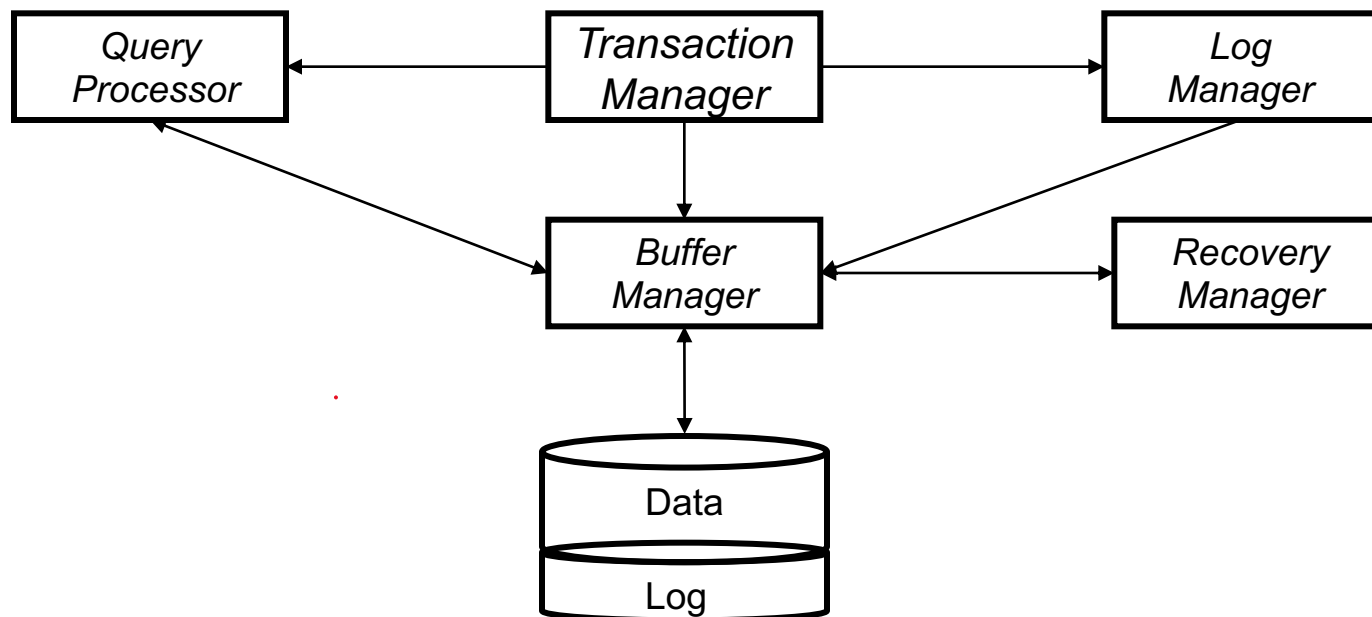
- skriva od korisnika detalje fizičke pohrane podataka
- omogućuje definiciju i rukovanje s podacima
- obavlja optimiranje upita
- obavlja funkciju zaštite podataka
 - integritet
 - pristup podacima - autorizacija, sigurnost
 - omogućuje pohranu pravila (integriteta i pristupa) u rječnik podataka – čime postaju nezaobilazna za sve korisnike
 - upravlja istodobnim pristupom podacima
 - omogućuje obnova u slučaju pogreške ili uništenja baze podataka
 - osigurava potporu za upravljanje transakcijama

1. TRANSAKCIJA

- jedinica rada nad bazom podataka
- sastoji se od niza logički povezanih izmjena
- početak transakcije - **BEGIN WORK**
- završetak transakcije:
 - **COMMIT WORK** - uspješan završetak - potvrđivanje transakcije
 - **ROLLBACK WORK** - neuspješan završetak - poništavanje transakcije - poništavanje svih izmjena koje je transakcija obavila

Upravljanje transakcijama

- *Transaction Management*
- Upravljač transakcijama (*transaction manager, transaction processing monitor – TP monitor*) - dio sustava koji brine o obavljanju transakcija i osigurava zadovoljavanje svih poznatih pravila integriteta.



Terminologija

- *Transaction Management*
- *Transaction Manager*
- *Query Processor*
- *Log*
- *Log Manager*
- *Recovery Manager*
- *Buffer Manager*
- Upravljanje transakcijama
- Upravljač transakcijama
- Procesor upita
- Dnevnik
- Upravljač dnevnica
- Upravljač obnovom
- Upravljač međuspremnicima

Primjer transakcije

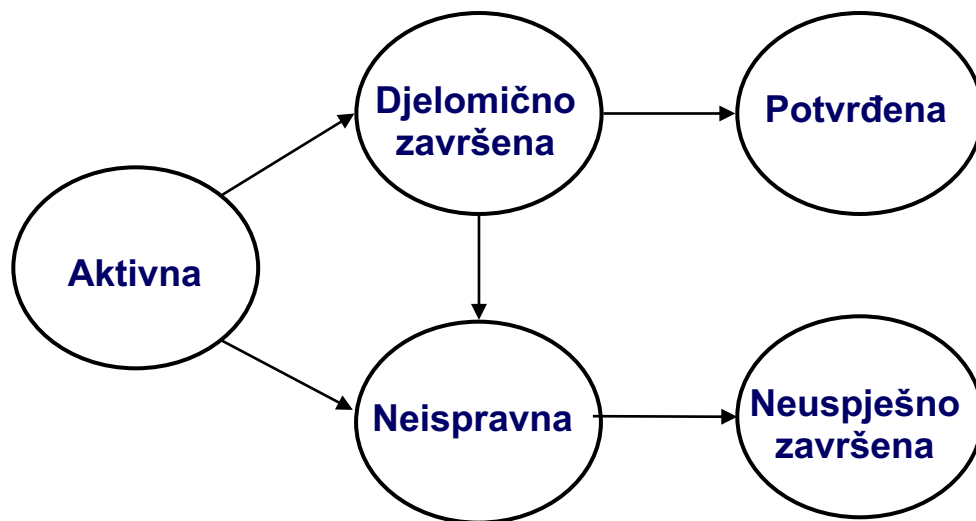
```
CREATE PROCEDURE prijenos (s_racuna INTEGER
                           , na_racun INTEGER
                           , iznos DECIMAL (8,2))
DEFINE pom_saldo DECIMAL (8,2);
BEGIN WORK;
    UPDATE racun SET saldo = saldo - iznos
        WHERE br_racun = s_racuna;
    UPDATE racun SET saldo = saldo + iznos
        WHERE br_racun = na_racun;
    SELECT saldo INTO pom_saldo FROM racun
        WHERE br_racun = s_racuna;
    IF pom_saldo < 0 THEN
        ROLLBACK WORK;
    ELSE
        COMMIT WORK;
    END IF
END PROCEDURE
```

Implicitne granice transakcija

- Ako granice transakcije nisu eksplicitno definirane naredbama BEGIN/COMMIT/ROLLBACK, tada se granice transakcije određuju implicitno:
 - svaka SQL naredba se smatra jednom transakcijom
npr. `UPDATE osoba SET ime = 'Janko' WHERE ime = 'Marko'`
 - naročito važno: UPDATE, DELETE, INSERT u slučajevima kada djeluju nad skupom n-torki
- Neki SUBP-ovi (npr. *Oracle*, *SQL Server*, ...) podržavaju način rada u kojem nije potrebno eksplicitno zadati početak transakcije
 - tada se početkom transakcije smatra prva naredba izvedena u okviru sjednice
 - potrebno je eksplicitno zadati COMMIT ili ROLLBACK - nakon čega opet implicitno počinje nova transakcija

Stanja transakcije

- 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



Dijagram stanja transakcije

Potvrđivanje transakcije

- Sve izmjene koje je transakcija načinila prije točke potvrđivanja mogu se smatrati tentativnima (*tentative* = privremeno, provizorno)
- Točka potvrđivanja (*commit point*) – trenutak u kojem sve izmjene koje je transakcija napravila postaju trajne
- U točki potvrđivanja otpuštaju se svi ključevi
- Potvrđena izmjena nikad ne može biti poništena - sustav garantira da će njezine izmjene biti trajno pohranjene u bazi podataka, čak i ako kvar nastane neposredno nakon njezinog potvrđivanja

Svojstva transakcije

▪ ACID

- Atomicity - nedjeljivost transakcije (atomarnost) - transakcija se mora obaviti u cijelosti ili se uopće ne smije obaviti
- Consistency - konzistentnost - transakcijom baza podataka prelazi iz jednog konzistentnog stanja u drugo konzistentno stanje
- Isolation - izolacija - kada se paralelno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge
- 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 desi neposredno nakon završetka transakcije

Nedjeljivost transakcije

```
CREATE PROCEDURE prijenos (s_racuna INTEGER, na_racun INTEGER
                        , iznos DECIMAL (8,2))
DEFINE pom_saldo DECIMAL (8,2);
BEGIN WORK;
  UPDATE racun SET saldo = saldo - iznos
    WHERE br_racun = s_racuna;
  UPDATE racun SET saldo = saldo + iznos
    WHERE br_racun = na_racun;
  SELECT saldo INTO pom_saldo FROM racun
    WHERE br_racun = s_racuna;
  ...
```


 **Kvar sustava**

Kvar se dogodio za vrijeme obavljanja druge UPDATE naredbe

- sustav mora osigurati poništavanje efekata prve UPDATE naredbe!

- Sa stanovišta krajnjeg korisnika transakcija je nedjeljiva
 - nije bitno što se moraju obaviti dvije ili više zasebnih operacija nad bazom podataka
- Korisnik mora biti siguran da je zadatak **obavljen potpuno i samo jednom** (ili ništa nije obavljeno)

Izdržljivost transakcije

```
...  
BEGIN WORK;  
    UPDATE racun SET saldo = saldo - iznos  
        WHERE br_racun = s_racuna;  
    UPDATE racun SET saldo = saldo + iznos  
        WHERE br_racun = na_racun;  
    SELECT saldo INTO pom_saldo FROM racun  
        WHERE br_racun = s_racuna;  
    IF pom_saldo < 0 THEN  
        ROLLBACK WORK;  
    ELSE  
        COMMIT WORK;  
         Kvar sustava  
    END IF
```

Kvar se dogodio nakon potvrđivanja transakcije

- efekti transakcije ne smiju biti izgubljeni
- Bez obzira u kojem se trenutku nakon potvrđivanja transakcije dogodio kvar, sustav mora osigurati da su njezini efekti trajno pohranjeni

Postizanje ACID svojstva transakcije

- Podsustav za upravljanje transakcijama i obnovu baze podataka u slučaju razrušenja brine o:
 - Atomarnosti
 - Izdržljivosti
- Podsustav za upravljanje istodobnim pristupom (concurrency control) omogućuje upravljanje:
 - Konzistentnošću
 - Izolacijom

2. OBNOVA BAZE PODATAKA (*Database Recovery*)

- dovesti bazu podataka u najnovije stanje za koje se pouzdano zna da je bilo ispravno
- Velike baze podataka – dijeljene, višekorisničke – nužno moraju posjedovati mehanizme obnove
- Male, jednokorisničke baze podataka obično imaju malu ili uopće nemaju potporu obnovi – obnova se prepušta korisnikovoj odgovornosti – podrazumijeva se da korisnik periodički stvara arhivsku (*backup*) kopiju pomoću koje u slučaju potrebe obnavlja bazu podataka

Uzroci pogrešaka

- pogreške opreme
- pogreške operacijskog sustava
- pogreške sustava za upravljanje bazama podataka
- pogreške aplikacijskog programa
- pogreške operatera
- kolebanje izvora energije
- požar, potres, sabotaza, ...

Općenito pravilo koje omogućuje obnovu

- Redundancija - svaki se podatak mora moći rekonstruirati iz nekih drugih informacija redundantno pohranjenih negdje drugdje u sustavu (na traci, na drugom disku, na zrcalnom disku, ...)
- Redundancija se postiže:
 - zrcaljenjem podataka (*mirroring*)
 - sigurnosnim kopijama (*backup*)
 - dnevnicima izmjena (*logical log*) koji služe za:
 - poništavanje transakcija
 - ponovno obavljanje transakcija

Općeniti opis postupka koji omogućuje obnovu

- ❶ Stvaranje arhivske kopije - periodičko kopiranje sadržaja baze podataka na arhivski medij (drugi disk, diskovni automat (*jukebox*) specijaliziranih sustava za sigurnosne kopije; nekad su to bile mag. trake)
(1 × dnevno, 1 × tjedno - ovisno o učestalosti promjena)
- ❷ Svaka izmjena u bazi podataka evidentira se u **logičkom dnevniku izmjena** (*logical log, journal, WAL—write ahead log*)
 - stara vrijednost zapisa, nova vrijednost zapisa
 - korisnik, vrijeme, ...
 - izmjena se **prvo zapisuje u dnevnik, a tek se onda provodi!** (*Write-Ahead Log Rule*)
 - dnevnici izmjena omogućuju
 - poništavanje transakcija (važno radi svojstva nedjeljivosti)
 - ponovno obavljanje transakcija (važno radi svojstva izdržljivosti)

Zašto?

Logički dnevnik izmjena

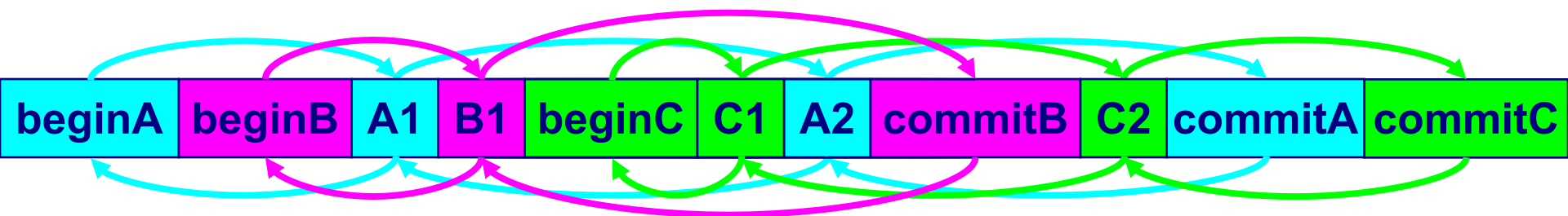
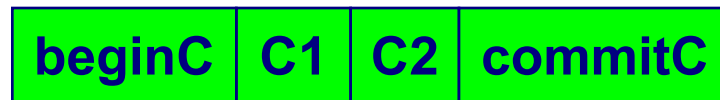
Transakcija A



Transakcija B



Transakcija C



Primjer sadržaja logičkog dnevnika u sustavu IBM IDS

osoba

oib	ime	prez
12345678901	Ivica	Horvat

Korisnici **miha** i **jure** obavljaju naredbe:

```
(m) BEGIN WORK;
(m) UPDATE osoba SET
    ime='Marko', prez='Ban'
    WHERE oib = '12345678901';
(j) BEGIN WORK;
(j) INSERT INTO osoba
    VALUES ('22233322233',
    'Mendo', 'Slavica');
(m) INSERT INTO osoba
    VALUES ('3334444555',
    'Ivica', 'Horvat');
(m) COMMIT WORK;
(j) COMMIT WORK;
```

```
# onlog -l -n 3360
addr    len  type    xid      id    link
48018   52    BEGIN    22       3360    12/22/2008 19:40:17 31  miha
34000000 200d0100 00000000 00000000 4... ..
[skraćeno]
4804c   104    HUPDAT    22       48018    200f5e 101 0 31 31 2
68000000 00004900 12000000 00000000 h.....I. ....
00000000 00000000 16000000 18800400 .....
7e16e100 5e0f2000 5e0f2000 01010000 ~...^.. ^. ....
00000000 1f001f00 02000000 00000000 .....
7075626c 000b0006 49766963 61204d61 publ.... Ivica Ma
726b6f20 00150007 486f7276 61742042 rko .... Horvat B
616e2020 20202020 an
480b4   52    BEGIN    24       3360    12/22/2008 19:40:29 41  jure
34000000 200d0100 00000000 00000000 4... ..
[skraćeno]
480e8   96    HINSERT   24       480b4    200f5e 201 31
[skraćeno]
32323233 33333232 3233334d 656e646f 22233322 233Mendo
20202020 20536c61 76696361 20202000 Sla vica .
48148   96    HINSERT   22       4804c    200f5e 102 31
[skraćeno]
33333334 34343435 35352049 76696361 33344445 55 Ivica
20202020 20486f72 76617420 20202000 Hor vat .
481a8   48    COMMIT    22       48148    12/22/2008 19:40:36
[skraćeno]
```

addr – Log record address

len – record length

type – record type name

xid – transaction number

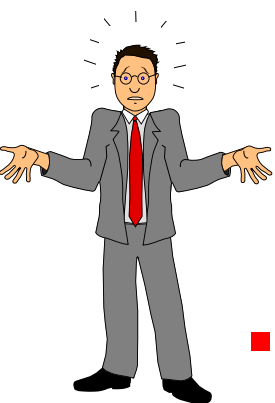
id – logical log number

link – link to the previous record in the transaction

Zapisivanje podataka iz radne memorije na disk

- Koriste se (među)spremnici (*buffer*):
 - Spremnik dnevnika
 - Spremnik baze podataka
- Sadržaj spremnika zapisuje se u dnevnik/bazu podataka:
 - Kad je spremnik popunjen ili
 - Kada SUBP izda nalog

Kad nastane kvar ...



- **Ako je baza je potpuno uništena:**
 - Učitava se najsvježija arhivska kopija
 - time se baza podataka dovodi u stanje kakvo je bilo u trenutku kad je napravljena posljednja arhivska kopija
 - Koristeći dnevnik izmjena ponovno se obavljaju izmjene koje su dogodile u međuvremenu - nakon izrade arhive
- **Baza nije uništena - sadržaj je nepouzdan** - program je prekinut tijekom obavljanja niza logički povezanih izmjena – potrebno je vratiti bazu podataka u ispravno stanje
 - pomoću podataka sadržanih u dnevniku izmjena poništavaju se sve izmjene koje su načinile nezavršene transakcije

Tipovi pogrešaka

- ❶ Pogreške transakcija (*transaction failure*) -
pogreške koje su posljedica (ne)planiranog prekida transakcije (npr. ROLLBACK WORK)
 - ❶ pomoću dnevnika izmjena poništavaju se efekti transakcije, kao da transakcija nikada nije započela s radom
- ❷ Pogreška računalnog sustava (*system failure*) -
baza podataka nije fizički uništena
 - ❷ transakcije koje su se obavljale u trenutku prekida se nakon ponovnog pokretanja poništavaju – koriste se **kontrolne točke** (eng. *checkpoint*)
- ❸ Kvar medija za pohranu (*media failure*) - baza podataka je fizički uništena
 - ❸ baza podataka se obnavlja pomoću arhivske kopije i pripadnog dnevnika izmjena

❶ Pogreške transakcija

- U slučaju pogreške transakcije SUBP će poništiti sve efekte transakcije
- Dovedi bazu u stanje kao da transakcija nije nikada započela s radom
- **Poništavanje izmjena** - pretragom dnevnika unatrag - nova vrijednost zapisa zamjenjuje se sa starom vrijednošću - sve dok se ne dođe do početka transakcije, odnosno do zapisa BEGIN WORK

Poništavanje transakcije pomoću dnevnika izmjena

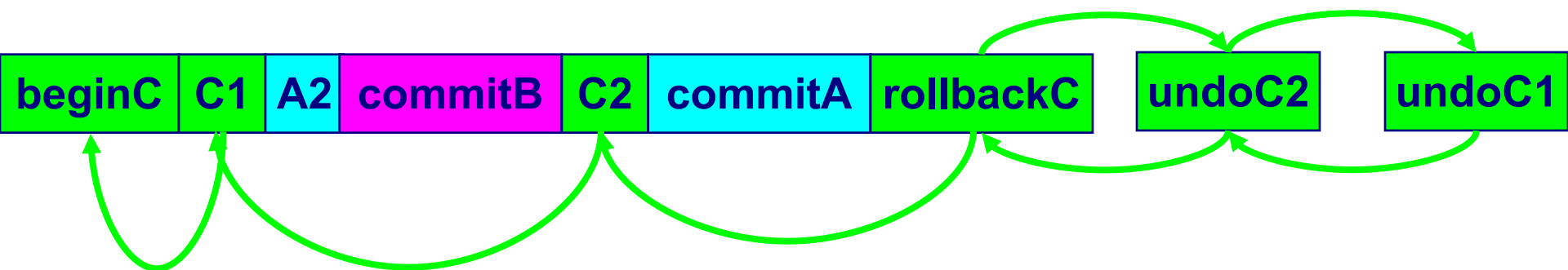
Transakcija A



Transakcija B



Transakcija C



❶ a) Pogreške koje otkriva aplikacija

- Slučajevi u kojima aplikacija predviđa obavljanje naredbe **ROLLBACK WORK**

```
...  
  IF pom_saldo < 0 THEN  
    ROLLBACK WORK;  
  ELSE  
    COMMIT WORK;  
  END IF  
...
```

1 b) Pogreške koje ne otkriva aplikacija

- Ako se dogodi pogreška za koju nema pretpostavljene reakcije, odnosno, transakcija završi na neplanirani način (u programu ne postoji eksplicitna naredba ROLLBACK WORK)
 - program se prekida i SUBP poništava transakciju
 - ako se naredbe izvode u interaktivnom sučelju, tada će daljnji tijek ovisiti o klijentu –automatsko poništavanje, ili se čeka da korisnik pokrene poništavanje, npr. PostgreSQL/pgAdmin automatski obavlja ROLLBACK

Primjer: pokušaj unosa zapisa čiji ključ već postoji
u bazi:

početak programa
BEGIN WORK;

```
CREATE TABLE osoba (  
    mbr    INTEGER PRIMARY KEY,  
    prezime VARCHAR(20),  
    ime     VARCHAR(20));
```

Pogreška!

```
INSERT INTO osoba VALUES (1,'Djetlić', 'Pero');  
INSERT INTO osoba VALUES (2,'Marić', 'Maro');  
INSERT INTO osoba VALUES (1,'Katić', 'Kata');  
INSERT INTO osoba VALUES (4,'Matić', 'Mato');  
COMMIT WORK;
```

neće se obaviti

završetak programa

② Pogreške računalnog sustava

- Baza nije uništena
 - **sve transakcije koje su se odvijale u trenutku kvara moraju biti poništene** jer nisu završene!
 - pretraživanjem dnevnika od početka identificiraju se transakcije za koje postoji **BEGIN** i ne postoji **COMMIT** ili **ROLLBACK**
 - takav postupak bi predugo trajao
 - u određenim intervalima (obično svakih 5 minuta) određuje se **kontrolna točka** (*checkpoint*)

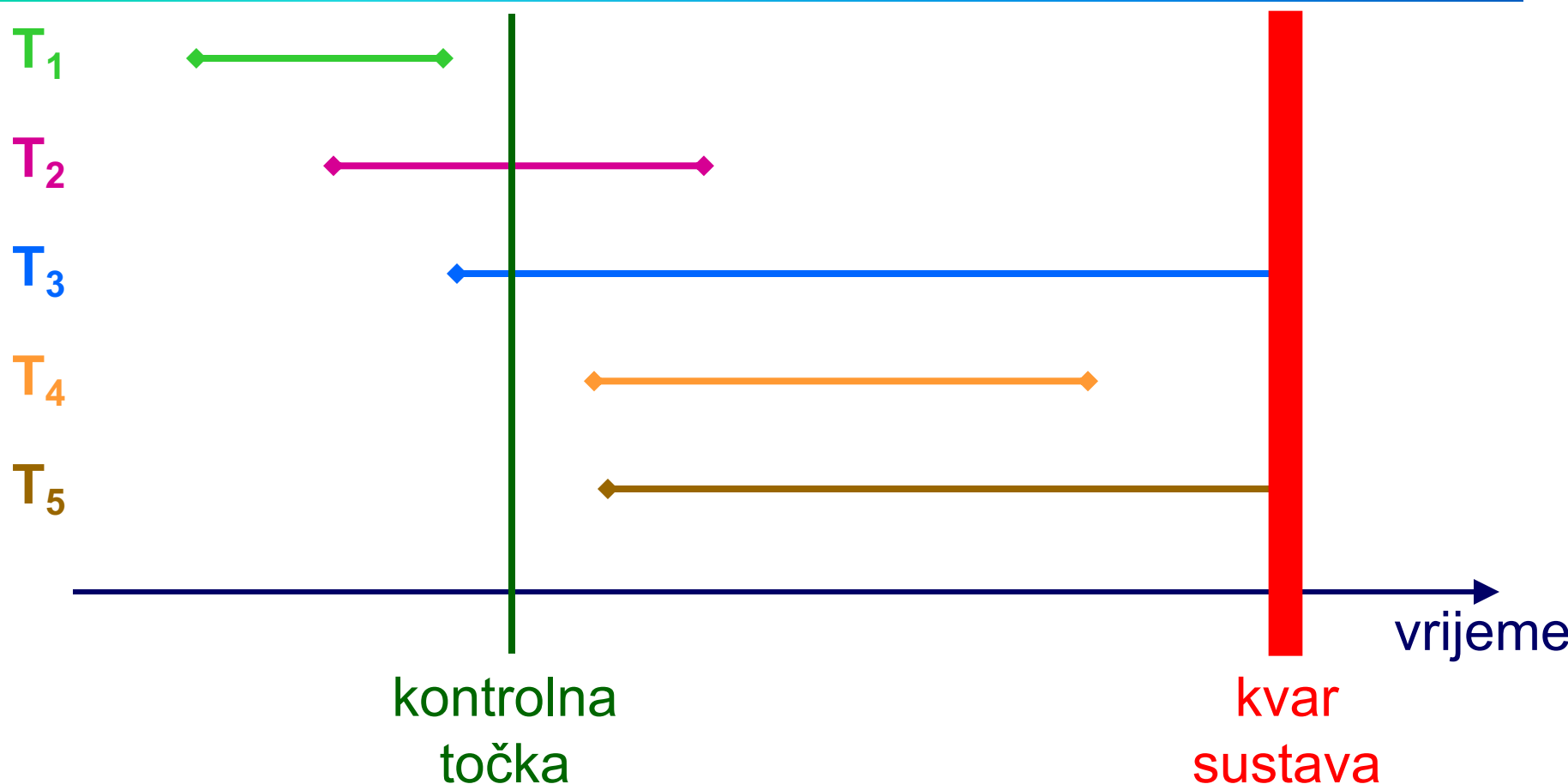
Aktivnosti u kontrolnoj točki

- ➊ pohrana sadržaja **spremnika dnevnika** (*log buffer*) u datoteku dnevnika
- ➋ zapisivanje **zapisa kontrolne točke** u datoteku dnevnika
- ➌ zapisivanje **adrese zapisa kontrolne točke** iz datoteke dnevnika u datoteku za ponovno pokretanje (*restart file*)
- ➍ pohrana sadržaja **spremnika baze podataka** (*database buffer*) u bazu podataka

Zapis kontrolne točke sadrži:

- listu svih aktivnih transakcija
- za svaku transakciju - adresu najnovijeg zapisa u datoteci dnevnika

Primjer:



Transakcije T_3 i T_5 treba poništiti

Transakcije T_2 i T_4 treba ponovo obaviti

Zašto?

Proces obnove

- Iz datoteke za ponovno pokretanje pročitava se adresa posljednje kontrolne točke
- Iz datoteke dnevnika pročitava se zapis kontrolne točke - lista transakcija koje su bile aktivne u kontrolnoj točki i adrese njihovih zadnjih zapisa
- Stvara se:
 - **lista za poništavanje** - na početku sadrži transakcije iz zapisa kontrolne točke
 - **lista za ponovo 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 ponovo obavljanje
- Ponovo se obavljaju transakcije iz liste za ponovo obavljanje
- Poništavaju se transakcije iz liste za poništavanje

SUBP ne može prihvatiti niti jedan zahtjev dok se ne završi proces obnove!

Očuvanje izmjena u slučaju razrušenja neposredno nakon završetka transakcije

- Postizanje izdržljivosti transakcije (*durability*)

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, ALI:

- 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

- Sadržaj spremnika dnevnika mora biti zapisan u dnevnik na disku prije nego što završi procedura potvrđivanja transakcije (*Write-Ahead Log Rule*)

Ponovno obavljanje i poništavanje transakcija

- Ako dođe do pogreške u samom procesu obnove, pa ga treba ponovo pokrenuti:
 - Učinak ponovnog obavljanja, bez obzira koliko se puta obavljalo mora biti isti kao da je operacija obavljena **točno jednom**
$$\text{redo}(\text{redo}(\text{redo}(\dots\text{redo}(x)))) = \text{redo}(x)$$
 - Učinak poništavanja, bez obzira koliko se puta obavljalo mora biti isti kao da je operacija obavljena **točno jednom**
$$\text{undo}(\text{undo}(\text{undo}(\dots\text{undo}(x)))) = \text{undo}(x)$$
- Ponovno obavljanje = Obnova unaprijed (*forward recovery*)
- Poništavanje = Obnova unatrag (*backward recovery*)

③ Kvar medija za pohranu

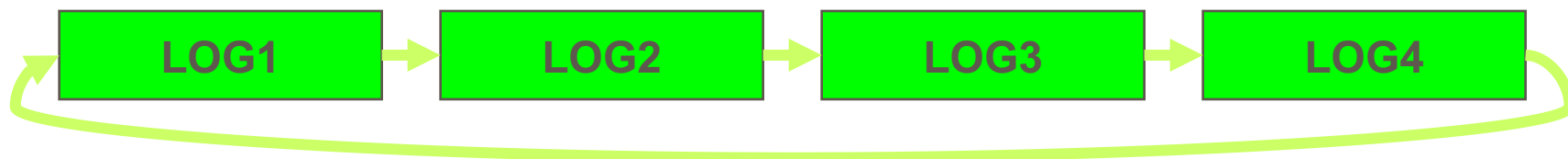
- baza je fizički uništena - **npr. zbog kvara diska**
- obnova sadržaja baze pomoću najnovije arhivske kopije
- pomoću najnovijeg dnevnika obavljaju se transakcije koje su bile provedene od trenutka arhiviranja
 - ako je najnovija arhivska kopija “pokvarena”
 - uzima se predzadnja arhivska kopija
 - dnevnik izmjena od predzadnje arhive do zadnje arhive
 - dnevnik izmjena nastalih nakon zadnje arhive

PREPORUKE:

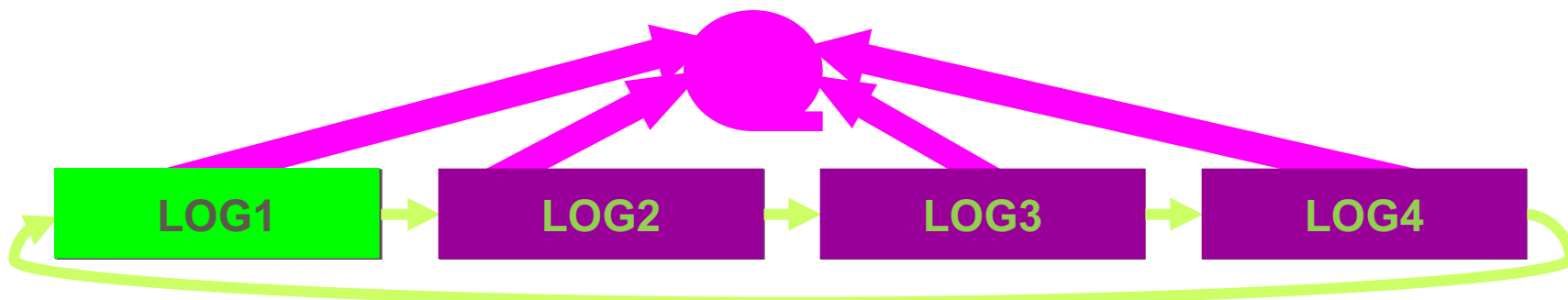
- čuvati najmanje tri posljednje arhive i pripadne dnevnike
- dnevnik se ne nalazi na istom disku na kojem je baza podataka
 - što ako je dnevnik “pokvaren”?

Ciklička izmjena logičkih dnevnika

- Dnevnik može biti vrlo velik – započinje u času pokretanja arhiviranja i aktivan je do sljedećeg arhiviranja
- Dnevници su ključni za obnovu - kako ih očuvati?
 - ➔ Čim prije treba njihov sadržaj pohraniti na „sigurno mjesto”
- Dnevnik se dijeli na manje odsječke koji se ciklički izmjenjuju



- Čim se jedan dnevnik popuni - kopira se na arhivski medij
- Dnevnik se mora nalaziti na disku sve dok su transakcije sadržane u njemu aktivne - da bi se omogućilo poništavanje (**ROLLBACK**)



- Čim se završe sve transakcije iz LOG1, on se oslobađa i može se ponovo koristiti

Vremenski preduge i prevelike transakcije

- Vremenski preduge transakcije
 - Korisnik započne transakciju i „ode na kavu”
 - Podaci koje je transakcija zaključala nedostupni su ostalim korisnicima
 - Nepotrebno se zadržavaju dnevnici
 - ➔ U programskom sustavu (aplikaciji) ograničiti trajanje neaktivnih transakcija (pažljivo odabrati vremensku jedinicu!)
 - ➔ upozoriti korisnika (*time-out warning*)
 - ➔ ako korisnik ne reagira - prekid i poništavanje transakcije
- Prevelike transakcije
 - Transakcija stvara vrlo mnogo izmjena
 - U slučaju prekida rada mnogo će izmjena biti izgubljeno
 - ➔ Velike transakcije gdje god je to moguće treba podijeliti na manje

Mehanizmi za postizanje visoke dostupnosti

1. Zrcaljenje podataka (*mirroring*)
2. Arhiviranje tijekom obavljanja transakcija (*on-line backup*)
3. Inkrementalno arhiviranje (*incremental backup*)
4. Replikacija

1. Zrcaljenje

- Postoje dva jednaka područja – primarno područje i zrcalno područje
- **Promjene se provode istovremeno u primarnom i zrcalnom području**
- U slučaju pogreške u jednom od područja
 - ➔ nastavak rada na ispravnom području
 - ➔ nalog za ponovo kreiranje (“popravljanje”) zrcalnog područja
 - ➔ nakon „popravka” područja se sinkroniziraju
- Zrcaljenje se može provoditi pod kontrolom sklopovlja
 - RAID - Redundant Arrays of Independent (Inexpensive) Disks
- Zrcaljenje pod kontrolom SUBP-a – na razini sustava se određuje što zrcaliti (npr. SQL Server zrcali samo cijelu bazu podataka)

Zrcaljenje na razini baze podataka

- Primarna namjena je **visoka dostupnost** – u slučaju kvara jednog područja dostupni su podaci u drugom
- Skalabilnost - podjela opterećenja, npr. zrcaljeni dio se možda može koristiti za izvještajni sustav
- Zrcaljenje ne može pomoći pri poništavanju transakcija niti kod kvara čitavog sustava
- Ne može zamijeniti arhiviranje i vođenje dnevnika

2. Arhiviranje tijekom rada korisnika

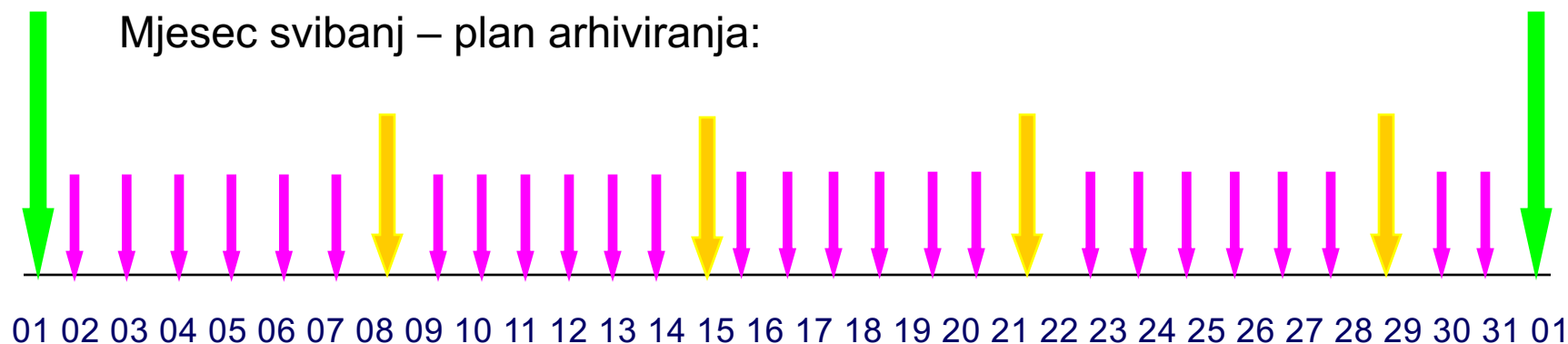
- *On-line backup*
- U klasičnim sustavima arhiviranje se obavljalo na sustavu na kojem nije bilo korisnika (npr. u petak na kraju radnog vremena)
- Današnji sustavi omogućuju da se arhiviranje obavlja tijekom rada korisnika, odnosno izvođenja transakcija
- Stanje baze podataka pohranjeno u arhivskoj kopiji je konzistentno i odgovara stanju kakvo je bilo u bazi podataka u času pokretanja arhiviranja.

3. Inkrementalno arhiviranje

- Arhiviranje baze podataka dodatno opterećuje sustav i usporava rad korisnika
 - želi se skratiti trajanje i obim arhiviranja
- Inkrementalno arhiviranje omogućuje stvaranje arhiva različitih razina
- Na primjer (u različitim SUBP-ovima se koristi različita terminologija):
- **razina 0** – kopija čitave baze podataka – npr. jednom mjesečno
- **razina 1** – tjedna arhiva – sadrži promjene nastale nakon arhive razine 0
- **razina 2** – dnevna arhiva - sadrži promjene nastale nakon arhive razine 1

... Inkrementalno arhiviranje

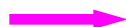
Mjesec svibanj – plan arhiviranja:



arhiva razine 0

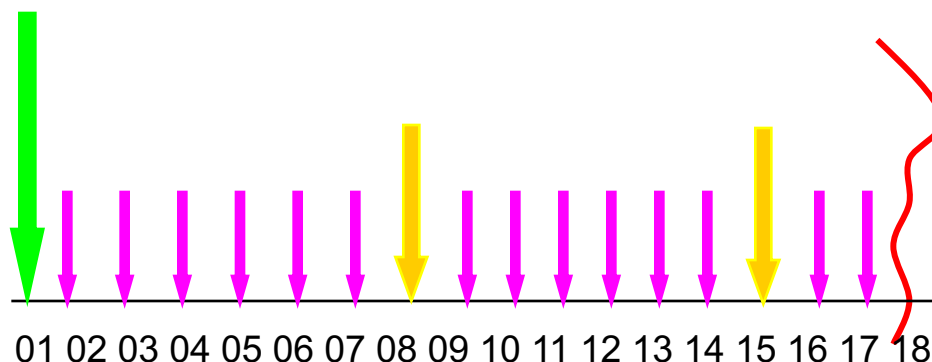


arhiva razine 1



arhiva razine 2

Ako se 18. svibnja dogodi kvar diska:



Pri obnovi se koriste se:

1. Arhiva razine 0 od 1. svibnja
2. Arhiva razine 1 od 15. svibnja
3. Arhiva razine 2 od 17. svibnja
4. Logički dnevnik koji je započeo nakon arhive razine 2 od 17. svibnja

4. Replikacija

- Baza podataka se replicira na istovjetni poslužitelj (isti OS, ista verzija SUBP-a)
 - Sinkrona replikacija
 - Asinkrona replikacija
- Različiti SUBP-ovi omogućuju različitu granulaciju replikacije (npr. PostgreSQL replicira sve baze na instanci, a SQL Server omogućuje odabir objekata koji će se replicirati)
- Primarna namjena je **visoka dostupnost** – u slučaju kvara jednog područja dostupni su podaci u drugom

PostgreSQL mogućnosti replikacije

- Sinkrona ili asinkrona replikacija (*streaming replication*)

PostgreSQL mogućnosti – postizanje izdržljivosti

- Arhiviranje tijekom rada korisnika – **On line backup**
(*Continuous Archiving and Point-in-Time Recovery (PITR)*):
 - **Arhiviranje baze podataka tijekom rada korisnika**
`pg_basebackup()` - logički dnevnici omogućavaju konzistentnost:
 - U času pokretanja arhiviranja baze podataka stvara se novi dnevnik izmjena u koji se pohranjuju izmjene koje su nastale nakon tog trenutka,
 - Odsjecci dnevnika mogu se pohranjivati u arhivsku kopiju kontinuirano (default), ili po završetku arhiviranja
 - **Kontinuirano arhiviranje dnevnika izmjena** (*WAL Archiving*) – ciklička izmjena logičkih dnevnika (prikaznica 33.)
- Uz pomoć arhive baze podataka i dnevnika izmjena moguće je obnoviti razrušenu bazu podataka bez gubitaka – na stanje neposredno prije kvara, ili na stanje u nekoj vremenskoj točki
- U slučaju kvara računalnog sustava baza podataka se dovodi u konzistentno stanje pomoću dnevnika izmjena