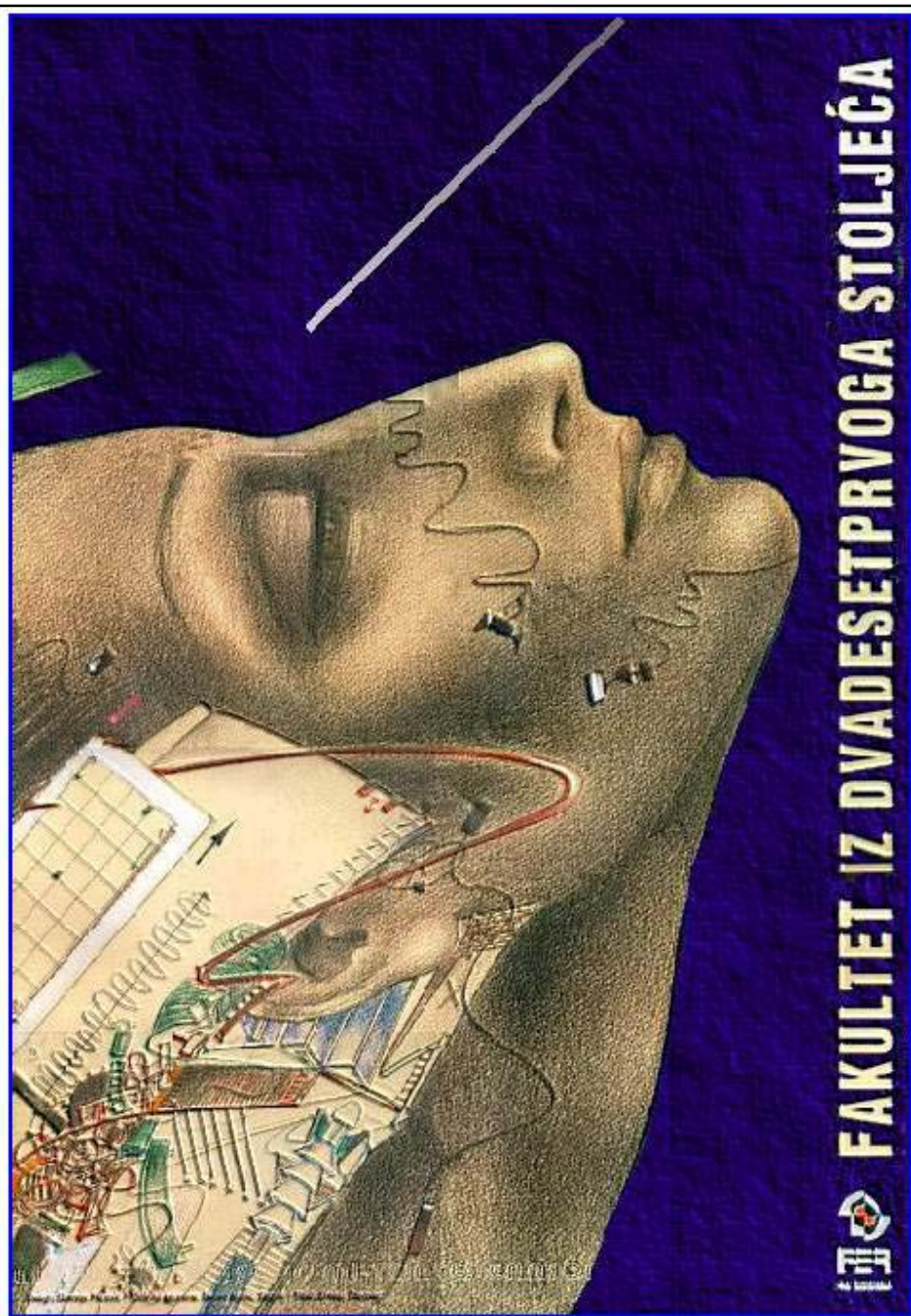


Baze podataka

Predavanja
lipanj 2008.

17. Sustav za upravljanje bazama podataka



Sustav za upravljanje bazama podataka

- *Database Management System*
 - skriva od korisnika detalje fizičke pohrane podataka
 - omogućuje definiciju i rukovanje s podacima
 - obavlja optimiranje upita
- obavlja funkciju zaštite podataka
 - integritet podataka
 - pristup podacima - autorizacija, sigurnost
 - **osigurava potporu za upravljanje transakcijama**
 - obnova u slučaju pogreške ili uništenja baze podataka
 - kontrola paralelnog pristupa

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

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

- Ukoliko granice transakcije nisu eksplicitno definirane naredbama BEGIN/COMMIT/ROLLBACK, tada se granice transakcije određuju implicitno:
 - početkom transakcije smatra se početak programa
 - uspješan završetak programa - potvrda transakcije
 - neuspješan završetak programa - poništavanje transakcije

ili

- svaka SQL naredba se smatra transakcijom za sebe
 - naročito važno: UPDATE, DELETE, INSERT u slučajevima kada djeluju nad skupom n-torki

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 - ukoliko 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

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 "*backup*" kopiju pomoću koje u slučaju potrebe obnavlja bazu podataka

Uzroci pogrešaka

- pogreške opreme
- pogreške operacijskog sustava
- pogreške sustava za upravljanje bazama podataka
- pogreške operatera
- kolebanje izvora energije
- požar, sabotaža, ...

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, ...)

Općeniti opis postupka koji omogućuje obnovu

- ❶ Periodičko kopiranje sadržaja baze podataka na arhivski medij (traka)
(1 × dnevno, 1 × tjedno - ovisno o učestalosti promjena)
- ❷ Svaka izmjena u bazi podataka evidentira se u **logičkom dnevniku izmjena** (*logical log, journal*)
 - stara vrijednost zapisa, nova vrijednost zapisa
 - korisnik, vrijeme, ...
 - izmjena se **prvo zapisuje u dnevnik, a tek se onda provodi!**
 - **dnevnici izmjena omogućuju**
 - poništavanje transakcija (važno radi svojstva nedjeljivosti)
 - ponovno obavljanje transakcija (važno radi svojstva izdržljivosti)

Dnevnik izmjena

Transakcija A

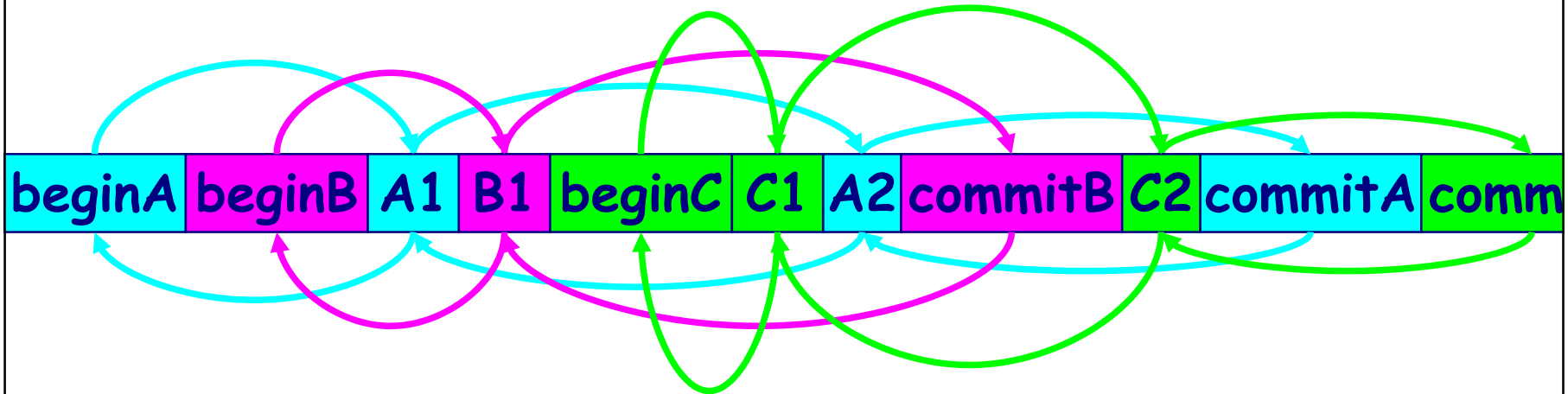
beginA	A1	A2	commitA
--------	----	----	---------

Transakcija B

beginB	B1	commitB
--------	----	---------

Transakcija C

beginC	C1	C2	commitC
--------	----	----	---------



Poništavanje transakcije pomoću dnevnika izmjena

Transakcija A

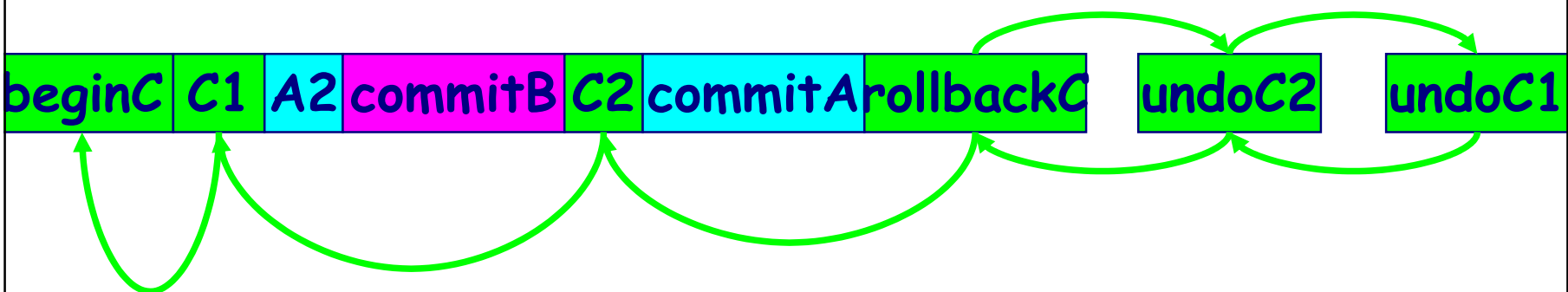
beginA	A1	A2	commitA
--------	----	----	---------

Transakcija B

beginB	B1	commitB
--------	----	---------

Transakcija C

beginC	C1	C2	rollbackC
--------	----	----	-----------



Tipovi pogrešaka

- ❶ Pogreške transakcija (*transaction failure*) - pogreške koje otkriva sama aplikacija ili pogreške koje su posljedica neplaniranog prekida transakcije
 - ❷ Pogreška računalskog sustava (*system failure*) - baza podataka nije fizički uništena
 - ❸ Kvar medija za pohranu (*media failure*) - baza podataka je fizički uništena
-

Slučaj ❶ - pomoću dnevnika izmjena poništavaju se efekti transakcije, kao da transakcija nikada nije započela s radom

Slučaj ❷ - transakcije koje su se obavljale u trenutku prekida se nakon ponovnog pokretanja poništavaju

Slučaj ❸ - baza podataka se obnavlja pomoću arhivske kopije i pripadnog dnevnika izmjena

Pogreške transakcija koje otkriva aplikacija

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

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


Pogreške transakcija koje ne otkriva aplikacija

- ako se dogodi pogreška za koju program nema pretpostavljenu reakciju, program završava na neplanirani način, SUBP automatski obavlja **ROLLBACK WORK**
- Primjer: pokušaj unosa zapisa čiji ključ već postoji u bazi:

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

početak programa

```
BEGIN WORK;
```

```
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;
```

završetak programa

 Pogreška!

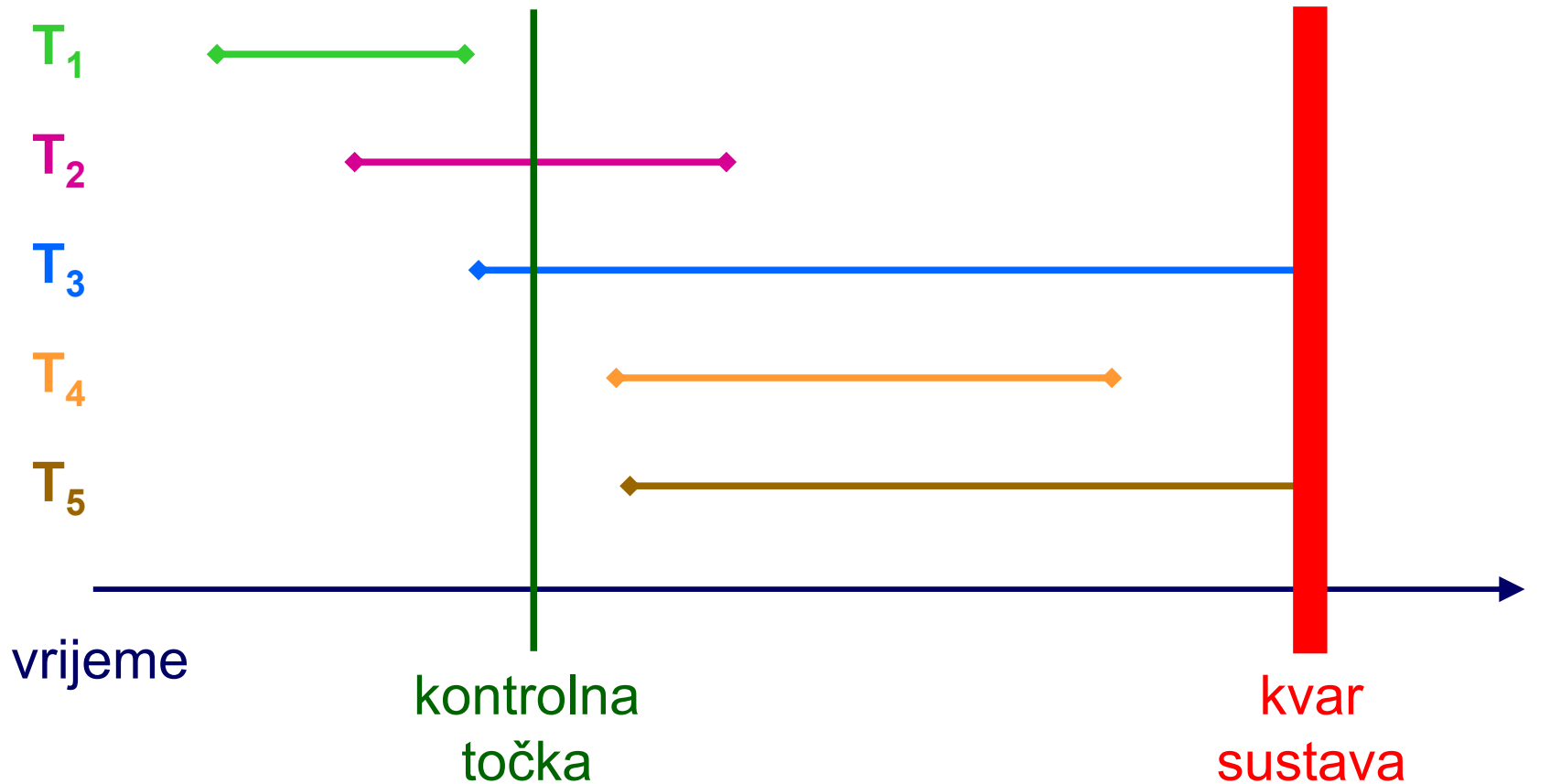
Pogreške računalskog sustava

- baza nije uništena
 - sve transakcije koje su se odvijale u trenutku 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 bi predugo trajao
 - u određenim intervalima (obično svakih 5 minuta) određuje se **kontrolna točka** (*checkpoint*)

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**

Proces obnove

- Stvara se:
 - lista za poništavanje - na početku sadrži sve transakcije koje su bile aktivne u kontrolnoj točki
 - 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!

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

3. KONTROLA PARALELNOG PRISTUPA

- u višekorisničkom radu **više programa može istovremeno pristupiti istim podacima**
- Rezultat transakcije ne smije ovisiti o tome da li se istodobno odvijaju i neke druge transakcije!!!
- SUBP mora spriječiti sljedeće:
 - dva (ili više) programa "istovremeno" mijenjaju isti podatak – problem: **izgubljene izmjene (*lost update*) - posljednji pobjeđuje**
 - neki programi pregledavaju podatak dok ga neki drugi program mijenja – problemi: **prljavo čitanje (*dirty read*), neponovljivo čitanje (*nonrepeatable read*), sablasti (*phantoms*)**

Primjer: Izgubljena izmjena (*Lost update*)

Rezervacija zrakoplovnih karata

Prodavač A



Pročitaj broj
slobodnih mjesta
 $BR = 20$

Rezerviraj 3 mjesta
 $BR = BR - 3$

Let OU 660
25.03.2007.
Slob. mjesta
20

Let OU 660
25.03.2007.
Slob. mjesta
17

Let OU 660
25.03.2007.
Slob. mjesta
19

Prodavač B



Pročitaj broj
slobodnih mjesta
 $BR = 20$

Rezerviraj 1 mjesto
 $BR = BR - 1$

Prljavo čitanje (*Dirty read*)

osoba

mbr	prez	ime
1111	Novak	Ivan
2222	Kolar	Iva

Transakcija A

```
.  
INSERT INTO osoba  
VALUES (3333,  
       'Jurić',  
       'Ana');  
.br/>.br/>.br/>.br/>ROLLBACK WORK;
```

Transakcija B

```
.  
.br/>SELECT * FROM osoba;
```

1111	Novak	Ivan
2222	Kolar	Iva
3333	Jurić	Ana

→ n-torka koja
nikad nije
stvarno postojala
u bazi podataka

Neponovljivo čitanje i sablasne n-torke

- Ista transakcija obavljanjem istog upita mora dobiti uvijek isti rezultat (osim ako sama nije promijenila podatke čije čitanje ponavlja)

Transakcija A

```
SELECT saldo FROM racun  
WHERE brRacun = 2;
```

Rezultat: 400.00

```
.  
-- A ne mijenja saldo za  
-- racun s brojem 2
```

```
.  
SELECT saldo FROM racun  
WHERE brRacun = 2;
```

Rezultat mora (opet) biti:
400.00

Primjer: Neponovljivo čitanje (*Nonrepeatable read*)

Transakcija A

```
SELECT saldo FROM racun  
WHERE brRacun = 2;
```

Rezultat: 400.00

```
-- A ne mijenja saldo za  
-- racun s brojem 2
```

```
SELECT saldo FROM racun  
WHERE brRacun = 2;
```

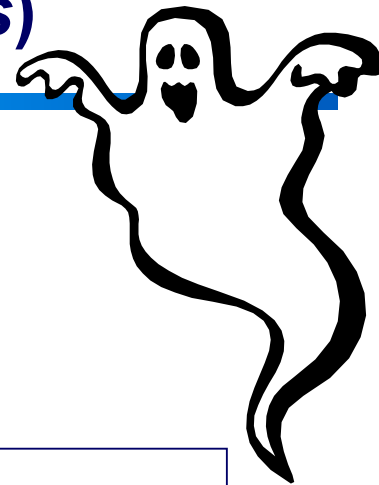
Rezultat: 440.00

Transakcija B

```
UPDATE racun  
SET saldo = saldo * 1.1  
WHERE brRacun = 2
```

- **Ista transakcija** obavljanjem **istog upita** dobije drugačiji rezultat

Primjer: Sablasne n-torke (*Phantom rows*)



Transakcija A

```
SELECT COUNT(*)  
  FROM racun  
 WHERE saldo > 100
```

Rezultat: 2

-- A ne mijenja racun

```
SELECT COUNT(*)  
  FROM racun  
 WHERE saldo > 100
```

Transakcija B

```
INSERT INTO racun  
  VALUES (3, 400.00)
```

Rezultat: 3 !!!

- **Ista transakcija** obavljanjem **istog upita** dobije drugačiji rezultat - zbog toga što je u međuvremenu transakcijom B unesena n-torka koja zadovoljava kriterij upita

3.1. Zaključavanje (*Locking*)

- transakcija može zaključati podatak (podatke)
 - sprečava druge transakcije da pristupe podatku dok ga ona ne otključa
- podaci koji su se mijenjali tijekom transakcije **ostaju zaključani do kraja transakcije**
- dio SUBP (*locking manager*) zaključava zapise i prosuđuje u slučajevima kad postoji više zahtjeva za zaključavanjem istog podatka

Zaključavanje

Rezervacija zrakoplovnih karata

Prodavač A



Zaključaj n-torku: OK

Pročitaj broj slob.mjesta

$BR = 20$

Rezerviraj 3 mjesta

$BR = BR - 3$

Otključaj n-torku

Let OU 660
25.03.2007.
Slob. mjesta

20

Let OU 660
25.03.2007.
Slob. mjesta

17

Let OU 660
25.03.2007.
Slob. mjesta

16

Prodavač B



Zaključaj n-torku: čekaj

čekaj → OK

Pročitaj broj slob.mjesta

$BR = 17$

Rezerviraj 1 mjesto

$BR = BR - 1$

Otključaj n-torku

3.2. Potpuni zastoј (*Deadlock*)

Transakcija 1

Zaključaj A

Zaključaj B

Transakcija 2

Zaključaj B

Zaključaj A

----- **POTPUNI ZASTOJ** -----

- izbjegavanje potpunog zastoja:
 - transakcija zatraži sva zaključavanja odjednom (npr. na početku) - **zaključa sve ili ništa!**
 - zahtijeva se da transakcije zaključavaju podatke u nekom **određenom poretku**
- u slučaju da se dogodi potpuni zastoј:
 - barem jedna transakcija se mora prekinuti - poništavaju se njezini efekti

3.3. Vrste zaključavanja

- ključ za pisanje/izmjenu
 - **WRITE LOCK, EXCLUSIVE LOCK**
- ključ za čitanje
 - **READ LOCK, SHARED LOCK**

3.4. Granulacija zaključavanja

- Veličina objekta koji se zaključava
 - baza podataka
 - tablica/relacija
 - memorijska stranica
 - n-torka

- Zaključavanje većih objekata
 - manji broj ključeva \Rightarrow manji utrošak proc. vremena i memorije
 - manja dostupnost podataka (često se zaključa više nego što je potrebno)

- Zaključavanje manjih objekata
 - veći broj ključeva \Rightarrow veći utrošak proc. vremena i memorije
 - veća dostupnost podataka (zaključavaju se samo objekti koje je zaista potrebno zaključati)

4. Zaključak

- Zaštita integriteta i sigurnost baze podataka temelji se na pravilima pohranjenim u rječniku podataka
- Pravila pohranjena u rječniku podataka
 - nezaobilazna su za sve korisnike
 - ne opterećuju aplikacijske programe
- Obnova baze podataka bez gubitka informacija moguća je jedino ako se:
 - redovito izrađuju arhivske kopije
 - vodi briga o dnevnicima izmjena