

KONTROLA PARALELNOG PRISTUPA

IX predavanje

Dr.sc. Emir Mešković

Kontrola paralelnog pristupa

- ▶ **Višekorisnički SUBP**
 - ▶ Ispravni i maksimalno dostupni podaci u uslovima paralelnog pristupa velikog broja korisnika
 - ▶ Komponenta SUBP-a zadužena za kontrolu paralelnog pristupa
- ▶ **Jednostavan ali neefikasan način korištenja višekorisničkog SUBP-a**
 - ▶ Korisnici obavljaju transakciju jednu za drugom (serijsko izvršavanje transakcija) – sistem počinje izvršavati sljedeću transakciju tek kad je u potpunosti završio prethodnu
 - ▶ Slaba ukupna iskoristivost sistema (npr. CPU čeka završetak U/I operacije)
 - ▶ Korisnik koji pokreće relativno kratku transakciju može (nepredvidivo) dugo čekati na završetak neke relativno duge transakcije

Zbog čega je istovremeni pristup važan?

- ▶ Budući da transakcije obuhvaćaju U/I i CPU operacije, njihovo istovremeno obavljanje bi omogućilo istovremeno korištenje različitih resursa računara
 - ▶ Povećava se broj transakcija obavljen u jedinici vremena (*throughput*), čime se povećava ukupna iskoristivost sistema (*utilization*)
 - ▶ Prosječno vrijeme koje protekne između aktiviranja i završetka transakcije (*average response time*) se smanjuje
- ▶ Današnji sistemi su (većinom) višekorisnički, te serijsko izvršavanje transakcija (izvršavanje jedne po jedne transakcije) predstavlja neracionalno raspolaganje računarskim resursima
 - ▶ Potrebno je omogućiti istovremeno (ili prividno istovremeno) izvršavanje transakcija

Transakcija

- ▶ Kontrola paralelnog pristupa (kao i postupak obnove baze podataka) usko su povezani sa pojmom transakcije
- ▶ Jedinica rada nad bazom podataka
- ▶ Sastoji se od niza logički povezanih izmjena
- ▶ Početak transakcije – **START TRANSACTION (BEGIN WORK)**
- ▶ Završetak transakcije:
 - ▶ **COMMIT WORK** - uspješan završetak - potvrđivanje transakcije
 - ▶ **ROLLBACK WORK** - neuspješan završetak - poništavanje transakcije - poništavanje svih izmjena koje je obavila transakcija

Određivanje granica transakcije

- ▶ eksplicitno:

START TRANSACTION

.....

.....

COMMIT WORK ili **ROLLBACK WORK**

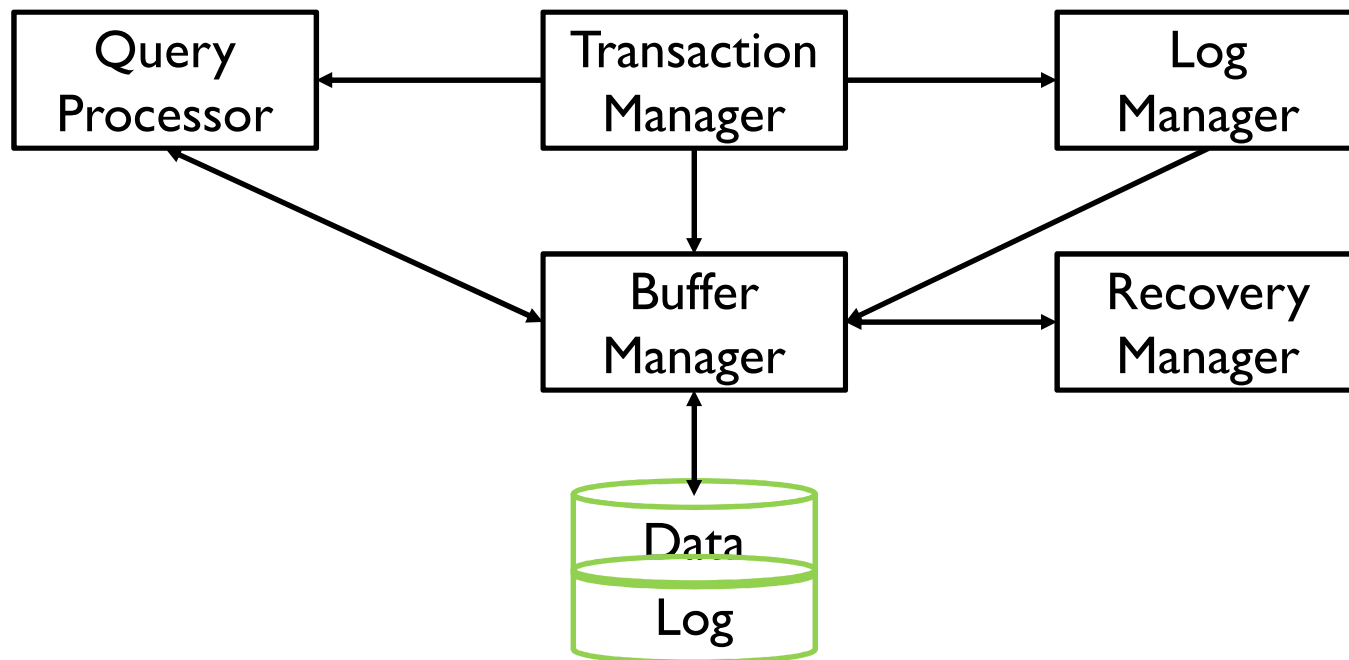
- ▶ implicitno:

- ▶ svaka operacija izmjene u bazi podataka (npr. **UPDATE** naredba) predstavlja transakciju, njezin uspješan završetak **COMMIT WORK**, neuspješan završetak predstavlja **ROLLBACK WORK**

- ▶ transakcije se ne mogu “ugnježdjivati” - jedna aplikacija u jednom času može imati samo jednu aktivnu transakciju
- ▶ sve operacije obnove moraju se obavljati unutar granica transakcije

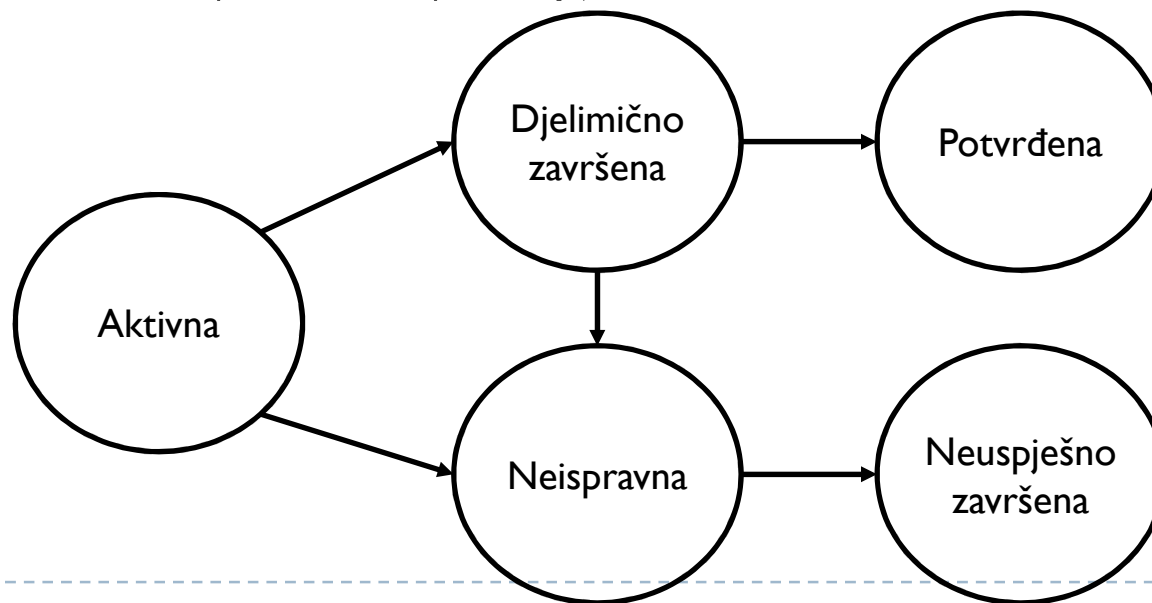
Obavljanje transakcija

- ▶ Dio sistema koji brine o obavljanju transakcija (*transaction manager, transaction processing monitor – TP monitor*) osigurava zadovoljavanje svih poznatih pravila integriteta



Stanja transakcije

- ▶ Aktivna (*active*) – tokom izvođenja
- ▶ Djelimično 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

Primjer transakcije

```
CREATE PROCEDURE prijenos (s_racuna INTEGER, na_racun INTEGER,  
                           iznos DECIMAL (8,2))  
  
BEGIN  
    DECLARE pom_saldo DECIMAL (8,2);  
    START TRANSACTION;  
        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
```

Potvrđivanje transakcije

- ▶ Tačka potvrđivanja (*commit point*) – sve izmjene koje je transakcija napravila postaju permanentne (trajne)
- ▶ Sve izmjene koje je transakcija napravila prije tačke potvrđivanja mogu se smatrati tentativnima (privremenim – *tentative*)
- ▶ U tački potvrđivanja otpuštaju se svi ključevi
- ▶ Potvrđena izmjena nikad ne može biti poništena – sistem garantuje da će njezine izmjene biti trajno pohranjene u bazi podataka, čak i ako kvar nastane već u sljedećem trenutku

Osobine transakcije

▶ ACID

- ▶ *Atomicity* - atomarnost - transakcija se mora obaviti u potpunosti ili se uopšte 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 efekat mora biti isti 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 sistema, 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))
```

```
DECLARE pom_saldo DECIMAL (8,2);
```

```
START TRANSACTION;
```

```
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 se dogodio za vrijeme obavljanja druge UPDATE naredbe

- Sistem mora osigurati poništavanje efekata prve UPDATE naredbe

- ▶ Sa stanovišta krajnjeg korisnika transakcija je nedjeljiva
 - ▶ ne zanima ga činjenica da se moraju obaviti dvije operacije nad bazom podataka
- ▶ Korisnik mora biti siguran da je zadatak obavljen potpuno i samo jednom!

 Kvar sistema

Izdržljivost transakcije

...

START TRANSACTION;

```
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
```



Kvar sistema

Kvar se dogodio nakon potvrđivanja transakcije

- Efekti transakcije ne smiju biti izgubljeni

- ▶ Bez obzira u kojem se trenutku nakon potvrđivanja transakcije dogodio kvar, sistem mora osigurati da su njeni efekti trajno pohranjeni

Paralelni pristup i transakcija

- ▶ Transakcija je niz logički povezanih operacija koje se izvršavaju kao cjelina i prevode bazu podataka iz jednog u drugo konzistentno stanje
- ▶ Rezultat transakcije ne smije ovisiti o tome odvijaju li se istovremeno i neke druge transakcije
- ▶ ACID svojstva transakcije
 - ▶ Atomarnost, konzistentnost i izdržljivost
 - ▶ Nisu ugroženi istovremenim pristupom
 - ▶ Izolacija – kada se istovremeno obavljaju dvije ili više transakcija, njihov efekat mora biti jednak kao da su se obavljale jedna iza druge
 - ▶ Problem kada više korisnika pristupa istom podatku/podacima njihove se aktivnosti (čitanje i/ili pisanje) isprepliću

Paralelni pristup i transakcija

- ▶ Dva objekta u bazi podataka ($x = 100, y = 100$)
- ▶ Integritetsko ograničenje: $x = y$
- ▶ Operacije čitanja ili pisanja (*database operations*)
 - ▶ Pročitaj(x, p) u varijablu p učitaj vrijednosti elementa x
 - ▶ Zapiši(y, p) u element y upiši vrijednost varijable p

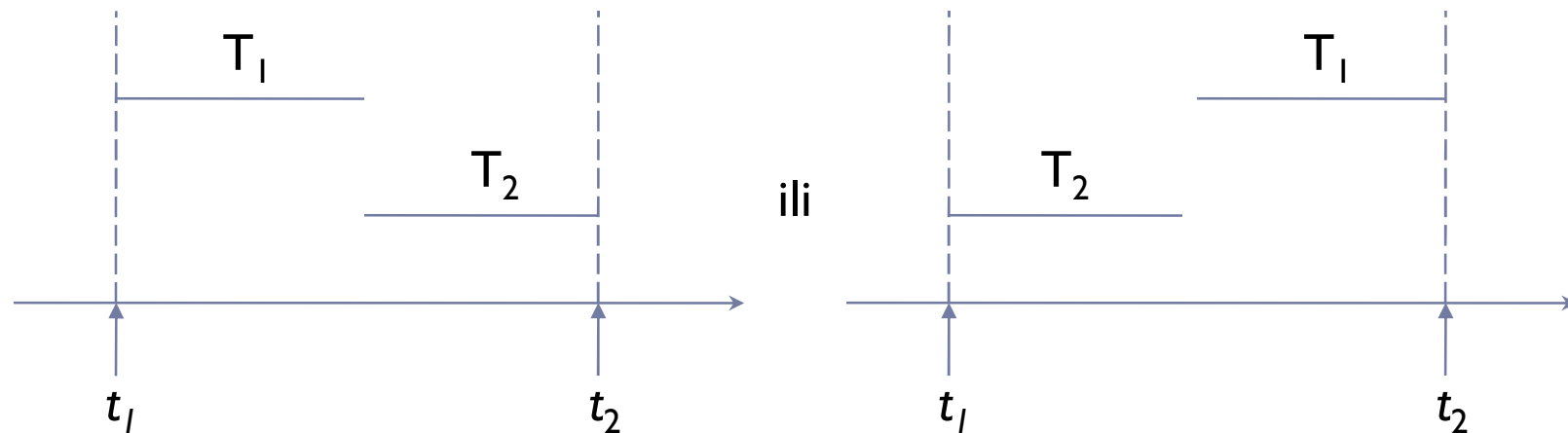
T_1
pročitaj(x, p)
 $p \leftarrow p + 100$
zapiši(x, p)
pročitaj(y, p)
 $p \leftarrow p + 100$
zapiši(y, p)

T_2
pročitaj(x, p)
 $p \leftarrow p * 2$
zapiši(x, p)
pročitaj(y, p)
 $p \leftarrow p * 2$
zapiši(y, p)

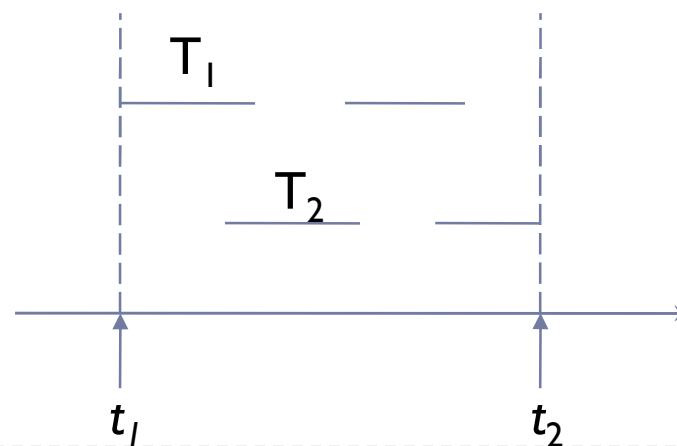
- ▶ Transakcije su korektne
 - ▶ Korektna transakcija prevodi bazu podataka iz jednog konzistentnog u drugo konzistentno stanje
 - ▶ Ako se korektne transakcije izvršavaju međusobno izolirano (jedna iza druge, serijski), neće narušiti konzistentnost baze podataka

Istovremeno izvršavanje transakcija

- ▶ Serijsko izvršavanje transakcija



- ▶ Istovremeno izvršavanje transakcija – jedan od mogućih redoslijeda

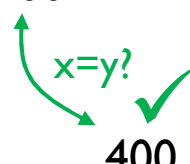


Serijsko izvršavanje transakcija

► Primjer:

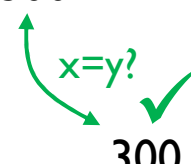
a) redoslijed T_1, T_2

T_1	T_2	x	y
		100	100
pročitaj(x, p)			
$p \leftarrow p + 100$			
zapiši(x, p)		200	
pročitaj(y, p)			
$p \leftarrow p + 100$			
zapiši(y, p)			200
	pročitaj(x, p)		
	$p \leftarrow p * 2$		
	zapiši(x, p)	400	
	pročitaj(y, p)		
	$p \leftarrow p * 2$		
	zapiši(y, p)		400



b) redoslijed T_2, T_1

T_1	T_2	x	y
		100	100
	pročitaj(x, p)		
	$p \leftarrow p * 2$		
	zapiši(x, p)	200	
	pročitaj(y, p)		
	$p \leftarrow p * 2$		
	zapiši(y, p)		200
pročitaj(x, p)			
$p \leftarrow p + 100$			
zapiši(x, p)			
pročitaj(y, p)			
$p \leftarrow p + 100$			
zapiši(y, p)			300



Istovremeno izvršavanje transakcija

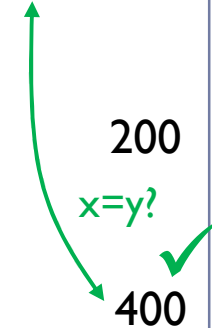
c) redoslijed izvršavanja koji narušava konzistentnost baze podataka

T_1	T_2	x	y
pročitaj(x, p) $p \leftarrow p + 100$ zapiši(x, p)		100	100
	pročitaj(x, p) $p \leftarrow p * 2$ zapiši(x, p)	200	
	pročitaj(y, p) $p \leftarrow p * 2$ zapiši(y, p)	400	
pročitaj(y, p) $p \leftarrow p + 100$ zapiši(y, p)			200
		$x=y?$ $x \neq y$	300

Istovremeno izvršavanje transakcija

d) redoslijed izvršavanja koji ne narušava konzistentnost baze podataka

T_1	T_2	x	y
pročitaj(x, p) $p \leftarrow p + 100$ zapiši(x, p)		100	100
	pročitaj(x, p) $p \leftarrow p * 2$ zapiši(x, p)	200	
pročitaj(y, p) $p \leftarrow p + 100$ zapiši(y, p)		400	
	pročitaj(y, p) $p \leftarrow p * 2$ zapiši(y, p)		200
			400



- ▶ Redoslijed izvršavanja nije serijski ali je efekat izvršavanja jednak efektu serijskog izvršavanja
- ▶ Svaki takav redoslijed ne narušava konzistentnost baze podataka – za njega se kaže da je serijalizibilan

Kontrola paralelnog pristupa - pregled

- ▶ ne dozvoliti istovremenu izmjenu podatka
 - ▶ problemi:
 - ▶ izgubljene izmjene (*lost update*) - posljednji pobjeđuje
- ▶ ne dozvoliti čitanje podataka koje netko drugi mijenja
 - ▶ Problemi:
 - ▶ prljavo čitanje (*dirty read*),
 - ▶ neponovljivo čitanje (*nonrepeatable read*),
 - ▶ sablasne n-torke (*phantom rows*)

Izgubljene izmjene (*lost updates*)

Prodavač A



Pročitaj broj
slobodnih mjesta

$BR = 20$

Rezervši 3 mjesta

$BR = BR - 3$

Let OU 660
21.03.06.
Slob. mjesta
20

Let OU 660
21.03.06.
Slob. mjesta

17

Let OU 660
21.03.06.
Slob. mjesta

Prodavač B



Pročitaj broj
slobodnih mjesta

$BR = 20$

Rezervši 1 mjesto

$BR = BR - 1$

► 20

19

Prljavo čitanje (*dirty read*)

Transakcija A:

SELECT * FROM osoba;

rezultat:		
1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

INSERT INTO osoba VALUES
(4567, "Mešić", "Ema");

ponišćavanje efekata
transakcije A

rezultat:		
1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino
4567	Mešić	Ema

Transakcija B:

SELECT * FROM osoba;

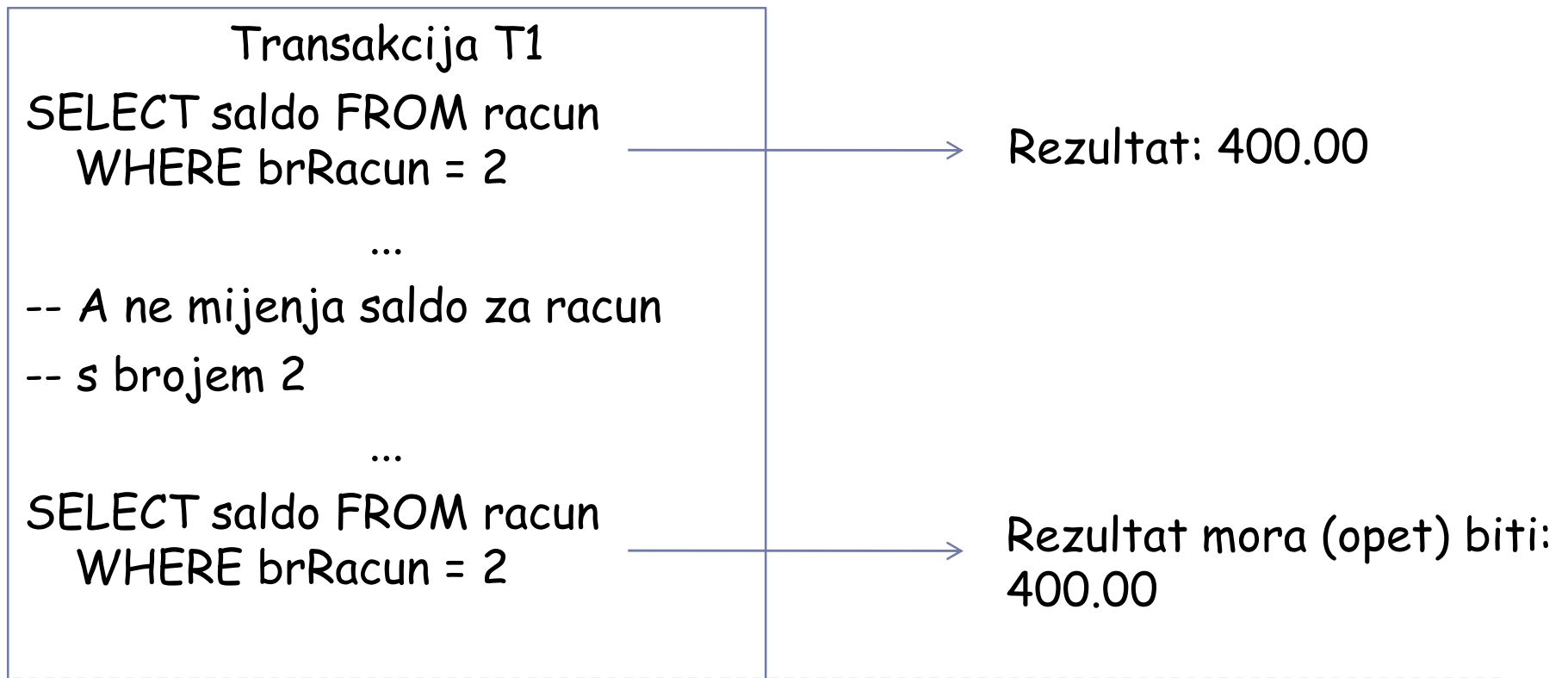
→ n-torka koja nikad nije stvarno postojala u bazi podataka

rezultat:		
1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

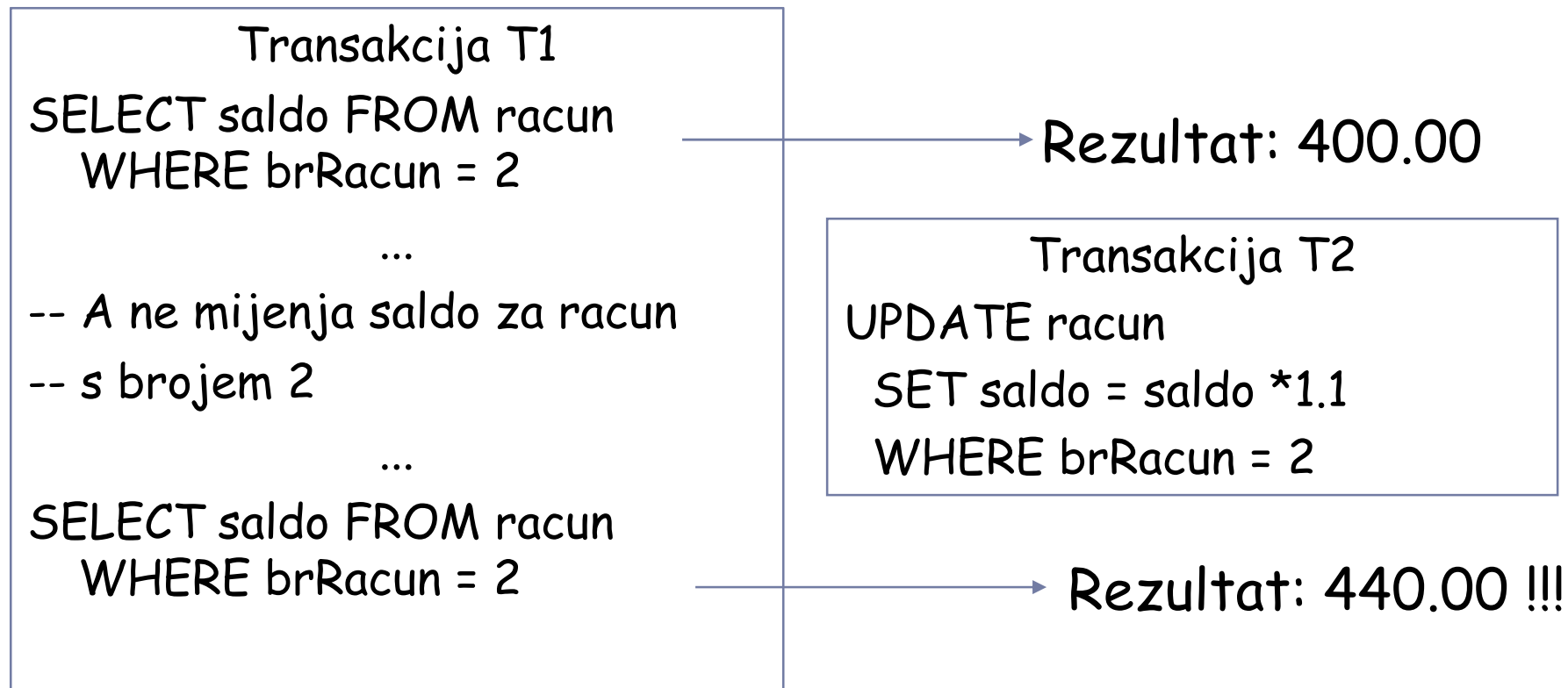
SELECT * FROM osoba;

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)

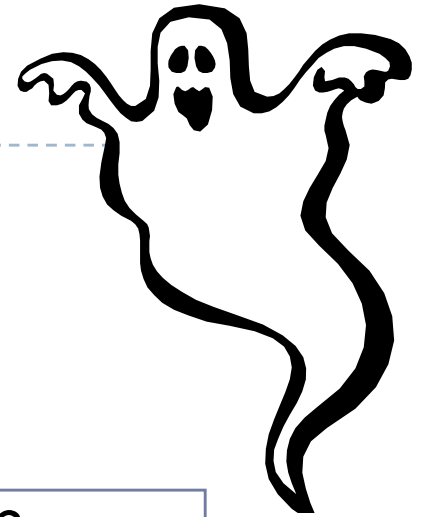


Neponovljivo čitanje (*nonrepeatable read*)



- ➡ Unutar iste transakcije za isti upit dobije se različiti rezultat

Sablasne n-torke (*phantom rows*)



Transakcija T1

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

Rezultat: 2

.
. .
. .

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

Transakcija T2
INSERT INTO racun
VALUES (4, 500.00)

Rezultat: 3 !!!

- Unutar iste transakcije za isti upit dobije se različiti rezultat - zbog toga što je u međuvremenu transakcija T2 ubacila n-torku koja zadovoljava kriterij upita

Kontrola paralelnog pristupa - rješenja

- ▶ Protokol zasnovan na zaključavanju
- ▶ Protokol korištenja vremenskih oznaka
- ▶ Protokol zasnovan na validaciji
- ▶ Protokol temeljen na grafovima
- ▶ ...

Zaključavanje (*Locking*)

- ▶ transakcija može zaključati podatak (podatke)
 - ▶ sprečava druge transakcije da pristupe podatku dok ga ona ne otključa
- ▶ podaci koji su se mijenjali tokom 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

Prodavač A



Zaključaj BR

Pročitaj broj
slobodnih mjesta

$BR = 20$

Rezervši 3 mjesta

$BR = BR - 3$

Otključaj BR

Let OU 660
21.03.06.
Slob. mjesta
20

Let OU 660
21.03.06.
Slob. mjesta
17

Let OU 660
21.03.06.
Slob. mjesta
16

Prodavač B



Zaključaj BR

Pročitaj broj
slobodnih mjesta

$BR = 17$

Rezervši 1 mjesto

$BR = BR - 1$

Otključaj BR

Vrste zaključavanja

- ▶ **ključ za pisanje/izmjenu** - WRITE LOCK, EXCLUSIVE LOCK
 - ▶ transakcija T_i zaključa objekat za pisanje
 - ▶ niti jedna druga transakcija ga ne može zaključati (niti za čitanje niti za pisanje) dok ga T_i ne otključa
 - svaka operacija izmjene (SQL naredbe INSERT, UPDATE, DELETE) postavlja ključ za pisanje!!!
- ▶ **ključ za čitanje** - READ LOCK, SHARED LOCK
 - ▶ transakcija T_i (SQL naredbom SELECT) zaključa objekat za čitanje
 - ▶ bilo koja druga transakcija ga također može zaključati za čitanje
 - ▶ niti jedna ga transakcija ne može zaključati za pisanje

Vrste zaključavanja

Proces 2
pokušava
postaviti na isti
objekt ključ:

Proces 1 postavio je na objekt ključ:

	READ	WRITE	NO LOCK
READ	OK	ERROR	OK
WRITE	ERROR	ERROR	OK

Vrste zaključavanja (MySQL)

- ▶ ključ namjere – **INTENTION LOCK**
 - ▶ Postavlja ključ nad kompletnom tabelom
 - ▶ Indicira koji tip ključa (za čitanje ili pisanje) će transakcija zahtjevati kasnije
 - ▶ Intention shared (IS) – SELECT ... LOCK IN SHARE MODE
 - ▶ Intention exclusive (IX) – SELECT ... FOR UPDATE
 - ▶ Prije nego transakcija zatraži ključ za čitanje nad zapisom tabele t mora prethodno zatražiti IS ključ ili snažniji ključ nad t
 - ▶ Prije nego transakcija zatraži ključ za pisanje nad zapisom, mora najprije zatražiti IX ključ nad tabelom t

Vrste zaključavanja (MySQL)

Proces 2
pokušava
postaviti na
isti objekt
ključ:

Proces 1 postavio je na objekt ključ:

	READ	INTENTION SHARED	WRITE	INTENTION EXCLUSIVE
READ	OK	OK	ERROR	ERROR
INTENTION SHARED	OK	OK	ERROR	OK
WRITE	ERROR	ERROR	ERROR	ERROR
INTENTION EXCLUSIVE	ERROR	OK	ERROR	OK

Serijalizabilnost – da li je zaključavanje dovoljno?

T_1	T_2	x	y
zaključaj(x)		100	100
pročitaj(x, p)			
$p \leftarrow p + 100$			
zapiši(x, p)		200	
otključaj(x)			
	zaključaj(x)		
	pročitaj(x, p)		
	$p \leftarrow p * 2$		
	zapiši(x, p)	400	
	otključaj(x)		
	zaključaj(y)		
	pročitaj(y, p)		
	$p \leftarrow p * 2$		
	zapiši(y, p)		200
	otključaj(y)		
zaključaj(y)			
pročitaj(y, p)			
$p \leftarrow p + 100$			
zapiši(y, p)			
otključaj(y)			

400

200

x=y?

x≠y

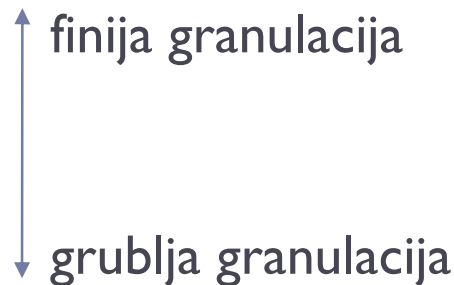
300

Protokol dvofaznog zaključavanja (*Two-phase locking protocol (2PL)*)

- ▶ serijalizacija je osigurana ako sve transakcije poštuju protokol dvofaznog zaključavanja:
 - ▶ prije obavljanja operacije nad objektom (npr. n-torkom iz baze), transakcija mora za taj objekat zatražiti ključ
 - ▶ nakon otpuštanja ključa transakcija ne smije više zatražiti nikakav ključ
- ▶ transakcije koje poštuju 2PL protokol imaju 2 faze - fazu pribavljanja ključeva (faza rasta - *growing phase*) i fazu otpuštanja ključeva (fazu sužavanja - *shrinking phase*)
- ▶ faza otpuštanja ključeva najčešće je stješnjena u jednu operaciju (COMMIT ili ROLLBACK na kraju transakcije)

Granulacija podataka

- ▶ Granulacija podataka je određena relativnom veličinom objekta koji će biti zaključan
 - ▶ n-torka
 - ▶ fizička stranica
 - ▶ relacija
 - ▶ baza podataka
- ▶ Granulacija podataka pri zaključavanju utiče na performanse sistema
 - ▶ Odabirom finije granulacije povećava se konkurentnost i troškovi postavljanja ključeva
 - ▶ Odabirom grublje granulacije smanjuje se konkurentnost i troškovi postavljanja ključeva
- ▶ koja je granulacija “najbolja”?
 - ▶ Ovisi o konkretnim operacijama transakcije



Primjer: granulacija podataka

rokovi	sifIspit	nazPred	sala	datIspit
	21224	Razvoj softvera	Stelegt	14.01.2017

	21253	Baze podataka	Stelegt	07.07.2017

100 000 n-torki

T_1

```
SELECT nazPred, sala FROM rokovi  
WHERE sifIspit = 21253
```

- ▶ Uz granulaciju određenu relacijom \Rightarrow slaba konkurentnost, nepotrebno se ograničava pristup svim n-torkama relacije
- ▶ Koristiti granulaciju određenu n-torkom!
- ▶ Očigledno, SUBP mora podržavati zaključavanje na više nivoa granulacije

T_2

```
SELECT nazPred, sala FROM rokovi
```

- ▶ Uz granulaciju određenu n-torkom \Rightarrow loše performanse, pojedinačno se postavlja 100 000 ključeva
- ▶ Koristiti granulaciju određenu relacijom!

Granulacija zaključavanja

- ▶ Baza podataka – DATABASE
 - ▶ Nije podržano u MySQL-u
- ▶ Relacija – RELATION, TABLE
 - ▶ SELECT ... FOR SHARE
 - ▶ SELECT ... FOR UPDATE
- ▶ Memorijska stranica – MEMORY PAGE
- ▶ n-torka – ROW
- ▶ Indeks – INDEX, KEY
 - ▶ *Gap locks* u MySQL-u – npr. čuva mjesto za ključ čiji je zapis obrisao – za slučaj poništavanja transakcije

Nepotpuni zastoј (*Live lock*)



- ➡ moguće je da Transakcija 2 čeka zauvijek, iako je mnogo puta imala priliku zaključati podatak A
- ➡ rješenje - strategija **FIRST-COME-FIRST-SERVED**

Potpuni zastoј (*Deadlock*)

Transakcija 1

Zaključaj A

Zaključaj B

Transakcija 3

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

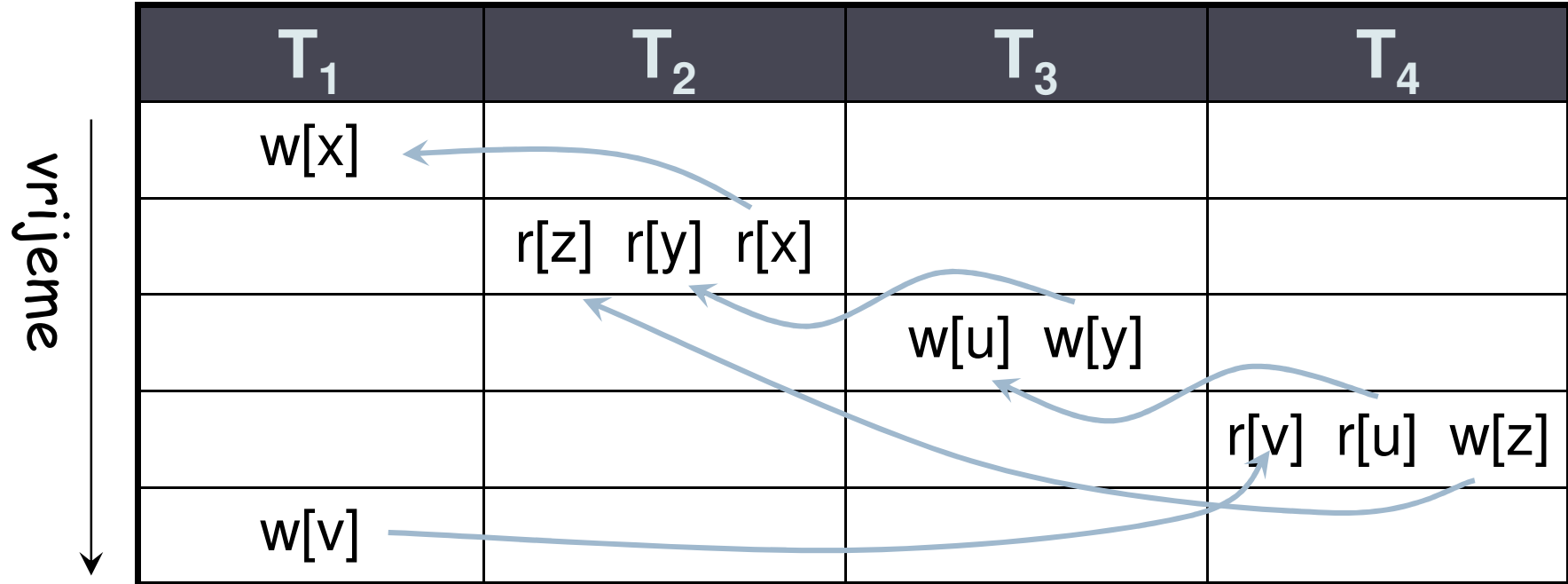
Potpuni zastoј (*Deadlock*)

- ▶ Izbjegavanje potpunog zastoja (nastavak):
 - ▶ *timeout-based deadlock prevention*:
 - ▶ transakcija čeka na dobijanje ključa
 - ▶ ako ga ne dobije tokom zadanog vremenskog perioda, menadžer transakcija pretpostavlja da se ključ ne može postaviti upravo zbog pojave potpunog zastoja
 - ▶ poništava transakciju koja čeka na dobijanje ključa
 - ▶ moguće je da će se poništiti transakcija koja ustvari ne sudjeluje u potpunom zastoju

Detekcija potpunog zastoja

- ▶ Menadžer transakcija održava wait-for-graf
 - ▶ usmjereni graf čiji su čvorovi označeni identifikatorima transakcija, T_i, T_j, T_k, \dots ,
 - ▶ lûk usmjeren od čvora T_i prema čvoru T_j povlači se ako i samo ako transakcija T_i čeka na postavljanje ključa zbog nekog ključa kojeg je postavila transakcija T_j .
- ▶ Potpuni zastoje detektuje se otkrivanjem petlji u *wait-for* grafu.
- ▶ Petlja se iz grafa uklanja izbacivanjem jednog ili više čvorova, T_i, T_j, T_k, \dots , njima pripadnih lûkova, uz istovremeno poništavanje pripadnih transakcija T_i, T_j, T_k, \dots

Primjer

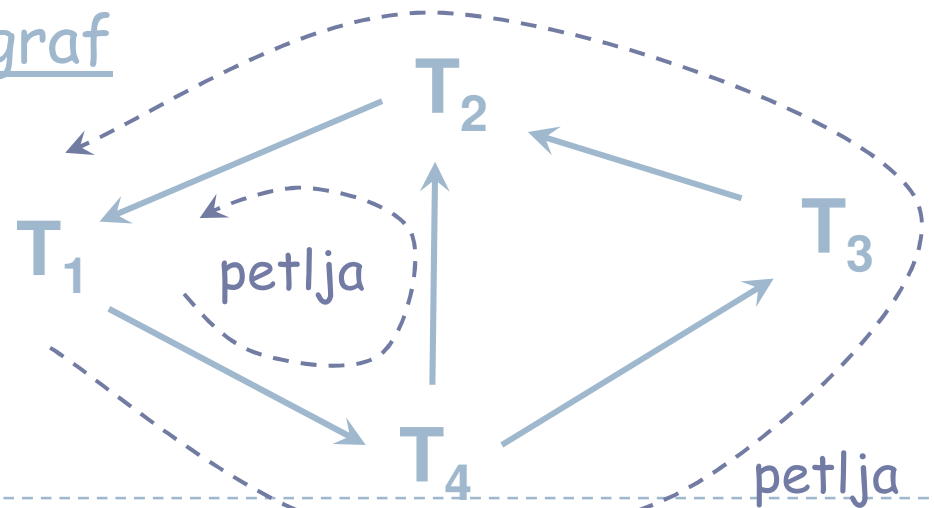


wait-for graf

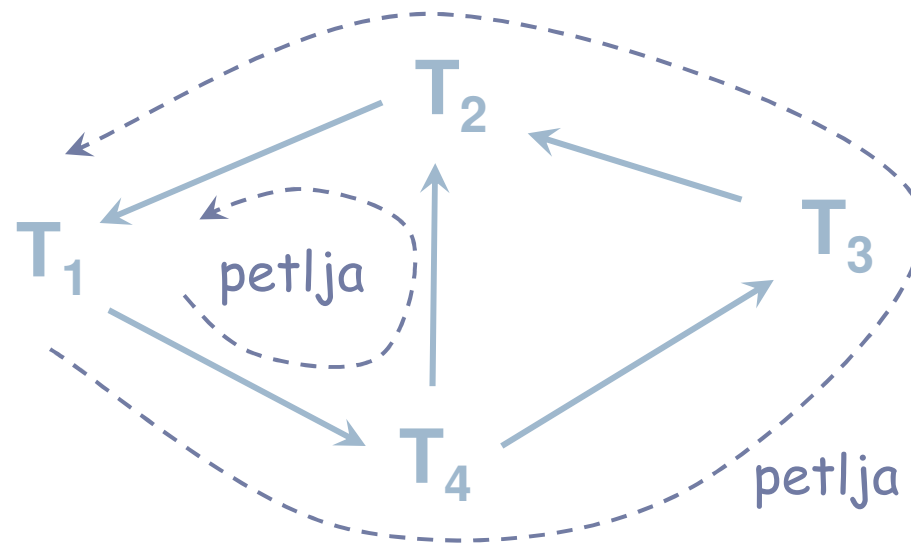
Notacija:

w[x] - zaključavanje
zapisa x za pisanje;

r[y] - zaključavanje
zapisa y za čitanje.



- ispitivanje postoje li petlje u wait-for grafu se provodi ili permanentno (kod dodavanja svakog novog lûka u graf) ili u određenim vremenskim razmacima
- u slučaju da se dogodi potpuni zastoј:
 - barem jedna transakcija se mora prekinuti - poništavaju se njezini efekti



Poništavanjem T_3 eliminisala bi se samo jedna petlja. Poništavanjem bilo koje od preostalih transakcija eliminisale bi se obje petlje u grafu.

Nivo izolacije (*Isolation level*)

- ▶ Određuje način pristupa podacima u višekorisničkom okruženju
- ▶ Nivoi izolacije prema standardu SQL-92
- ▶ prljavo čitanje – **READ UNCOMMITTED**
 - ▶ Čitaju se svi traženi podaci bez zaključavanja i bez provjere da li su možda zaključani!
 - ▶ mogu se pojaviti n-torke koje nikada stvarno nisu postojale u bazi podataka!
- ▶ čitanje potvrđenih n-torki - **READ COMMITTED**
 - ▶ Proces koji čita provjerava da li je trenutno pročitani podatak zaključan za pisanje - podaci se ne zaključavaju!

Nivo izolacije (*Isolation level*)

▶ Ponovljivo čitanje – REPEATABLE READ

- ▶ Osigurava ponovljivo čitanje podataka u okviru transakcije
- ▶ Podatak se zaključava i ostaje zaključan ključem za čitanje do kraja transakcije
- ▶ Ne spriječava pojavu sablasnih n-torki
- ▶ U MySQL-u:
 - ▶ dohvat n-torke iz radnog skupa ne uzrokuje zaključavanje (za čitanje) odgovarajuće n-torke u bazi podataka
 - ▶ drugi proces može obavljati izmjene nad n-torkama u bazi podataka
 - ▶ kod ponovnog dohvata istog radnog skupa **unutar iste transakcije** izmjene koje je proveo drugi proces nisu vidljive

Nivo izolacije (*Isolation level*)

- ▶ Serijalizabilnost – **SERIALIZABLE**
 - ▶ dohvat n-torke iz radnog skupa uzrokuje zaključavanje (za čitanje) odgovarajuće n-torke u bazi podataka
 - ▶ dohvaćanjem nove n-torke **prethodna ostaje zaključana**
 - ▶ kod ponovnog dohvaćanja istog radnog skupa **unutar iste transakcije** prethodno zaključane n-torke i dalje su zaključane
 - ▶ druge transakcije ne mogu unositi nove podatke
 - ▶ zapisi/stranice ostaju zaključane do kraja transakcije!

Definisanje nivoa izolacije - SQL

SQL-92:

- ☞ READ UNCOMMITTED
- ☞ READ COMMITTED
- ☞ REPEATABLE READ
- ☞ SERIALIZABLE

MySQL:

SET TRANSACTION
ISOLATION LEVEL

- ☞ READ UNCOMMITTED
- ☞ READ COMMITTED
- ☞ REPEATABLE READ
- ☞ SERIALIZABLE

Primjer: READ UNCOMMITTED – prljavo čitanje

SELECT * FROM osoba;

Program A:

START TRANSACTION;

UPDATE osoba SET prezime='Jurić'
WHERE sifOsoba=2345;

.
. .

ROLLBACK;

rezultat:		
1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Program B:
SET TRANSACTION
ISOLATION LEVEL
READ UNCOMMITTED;

SELECT * FROM osoba;

rezultat:		
1111	Pirić	Damir
2345	Jurić	Maja
3456	Pejić	Dino

SELECT * FROM osoba;

rezultat:		
1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Primjer: READ UNCOMMITTED – sablasne n-torke

SELECT * FROM osoba;

Program A:

START TRANSACTION;

INSERT INTO osoba VALUES
(4567, "Mešić", "Ema");
.
.
.

ROLLBACK;

rezultat:

1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Program B:

SET TRANSACTION
ISOLATION LEVEL
READ UNCOMMITTED;

SELECT * FROM osoba;

rezultat:

1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino
4567	Mešić	Ema

SELECT * FROM osoba;

rezultat:

1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Primjer: READ COMMITTED – sprječavanje prljavog čitanja

SELECT * FROM osoba;

rezultat:		
1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Program A:

```
START TRANSACTION;  
INSERT INTO osoba VALUES  
  (4567, "Mešić", "Ema");  
  .  
  .  
  .  
ROLLBACK;
```

Program B:

```
SET TRANSACTION  
ISOLATION LEVEL  
READ COMMITTED;  
SELECT * FROM osoba;
```

GREŠKA: Zapis zaključan!

Modalitet zaključavanja (Lock mode)

- ▶ Pomoću različitih modaliteta zaključavanja korisnik definiše šta će se dogoditi sa procesom koji pokušava zaključati već zaključani objekat
- ▶ bez čekanja - **SET LOCK MODE TO NOT WAIT**
 - ▶ proces odmah dojavljuje poruku o grešci!
 - ▶ program mora ispitivati da li je došlo do greške (SQLCODE) te predviđa akcije u slučaju greške
- ▶ čekanje - **SET LOCK MODE TO WAIT**
 - ▶ proces čeka otključavanje podataka
 - ▶ proces može čekati zauvijek! → MOŽE DOVESTI DO POTPUNOG ZASTOJA!

Modalitet zaključavanja (Lock mode)

- ▶ čekanje kroz zadano vrijeme – **SET LOCK MODE TO WAIT (n-secs)**
 - ▶ čeka se najviše zadani broj sekundi (**n-secs**)
 - ▶ ukoliko tokom zadanog vremena objekat nije otključan → proces dojavljuje poruku o grešci!
 - ▶ program mora ispitivati da li je došlo do greške (SQLCODE) te predviđa akcije u slučaju greške

Primjer: Čitanje potvrđenih n-torki

`SELECT * FROM osoba;`

Program A:

`BEGIN WORK;`

`INSERT INTO osoba VALUES
(4567, "Mešić", "Ema");`

.

.

.

`ROLLBACK WORK;`

rezultat:

1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Program B:

`SET LOCK MODE
TO WAIT;`

`SET ISOLATION TO
COMMITTED READ;`

`SELECT * FROM osoba;`

..... čeka na otključavanje

zapis otključan,
obavlja se naredba

rezultat:

1111	Pirić	Damir
2345	Đurić	Maja
3456	Pejić	Dino

Problemi koji javljaju kod različitih nivoa izolacije (SQL-92)

▶ READ UNCOMMITTED

- ▶ n-torke koje nikad stvarno nisu postojale u bazi podataka – dirty read problem
- ▶ neponovljivo čitanje
- ▶ sablasne n-torke

▶ READ COMMITTED

- ▶ neponovljivo čitanje
- ▶ sablasne n-torke

▶ REPEATABLE READ

- ▶ sablasne n-torke

▶ SERIALIZABLE

- ▶ -