TRANSAKCIJE

TRANSAKCIJE

ŠTA JE TRANSAKCIJA?

- Kolekcija operacija (čitanja i pisanja) koje želimo zajedno da obavimo nad bazom podataka u cilju obavljanja nekog posla definisanog na višem nivou (aplikativnom).

TRANSAKCIJE U SQL-u

- Počinju sa BEGIN
- Završavaju se sa COMMIT ili ROLLBACK (ABORT)
    - Ako se završe sa COMMIT, baza ili sačuva sve promene ili radi ROLLBACK
    - Ako se završe sa ROLLBACK, sve promene se odbacuju i podaci se vraćaju u prvobitno stanje

SVOJSTVA TRANSAKCIJA

KOJA SVOJSTVA TRANSAKCIJA TREBA DA POSEDUJE?

- A - Atomičnost skupa operacija (Atomicity)
- C - Konzistentnost podataka (Consistency)
- I - Izolovanost operacija (Isolation)
- D - Izdržljivost podataka (Durability)

ATOMIČNOST

ŠTA JE ATOMIČNOST?

- Svojstvo koje garantuje da će se operacije u transakciji obaviti kao jedinstvena jedinica posla – sve ili ništa

KOJA SU DVA MOGUĆA SCENARIJA?

- Sve operacije su se uspešno izvršile i urađen je COMMIT
- Neka od operacija se nije uspešno izvršila i urađen je ROLLBACK

KAKO SE PRATI DA LI SU SVE OPERACIJE IZVRŠENE?

- Sve izmene se čuvaju u logovima kako bi se podaci mogli vratiti u prvobitno stanje
- Prave se posebne kopije originalnih stranica gde su podaci nalaze nad kojima transakcije obavljaju operacije

KONZISTENTNOST

ŠTA JE KONZISTENTNOST?

- Svojstvo koje sa logičkog aspekta obezbeđuje da se baza posle izvršenja transakcije nalazi u konzistentnom stanju i da su prilikom operacija ispoštovana sva ograničenja (nullable, unique, foreign key constraint, ...)

IZOLOVANOST

ŠTA JE IZOLOVANOST?

- Svojstvo koje omogućava da se svaka transakcija izvršava kao da je jedina, nezavisno od drugih transakcija

DA LI SE SVAKA TRANSAKCIJA ZAISTA TAKO IZVRŠAVA?

- Ne, baze postižu konkurentnost tako što “žongliraju” operacijama različitih transkacija da izazovu takav efekat

KAKO BAZE UPRAVLJAJU KONFLIKTIMA?

- Tako što ih izbegavaju :) (pesimistički pristup, ne dozvoljavaju da se uopšte dese - npr. 2PL – Two-Phase Commit)
- Tako što ih detektuju (optimistički pristup, pretpostavljaju da će se retko destiti, pa kada se dese onda će ih rešavati - npr. MVCC- - Multi-Version Concurrency Control)

SQL ANOMALIJE

ŠTA PRESTAVLJAJU OVE ANOMALIJE?

- Greške koje mogu nastati kada se transakcije ne izvršavaju jedna po jedna, jedna za drugom

KOJE ANOMALIJE POSTOJE?

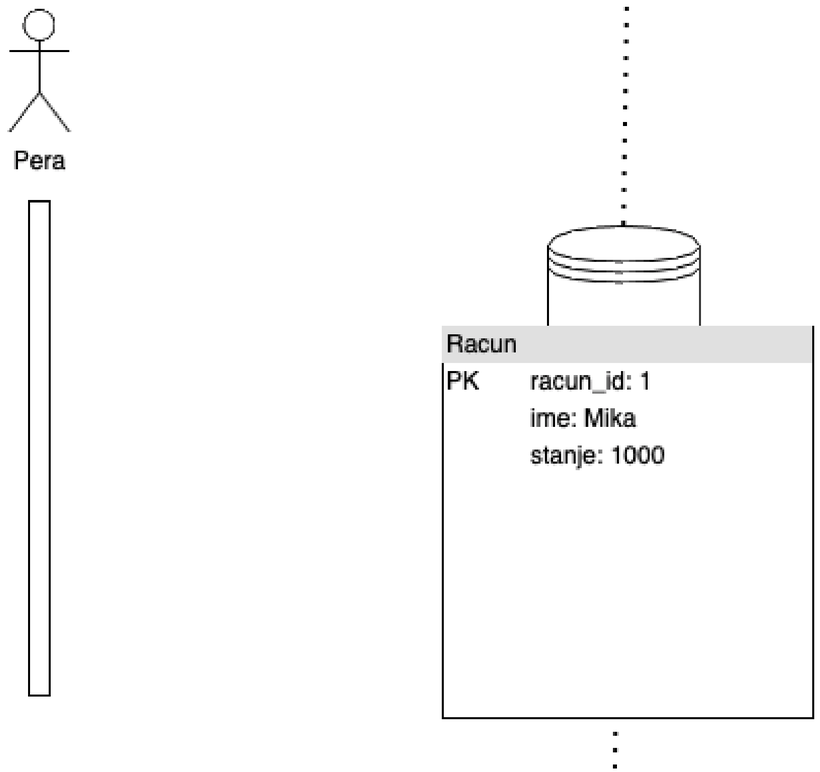
- Dirty reads[1]
- Non-repeatable reads[1]
- Phantom reads[1]
- Read skews
- Write skews
- Lost updates

[1]SQL-92 standard http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

DIRTY READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

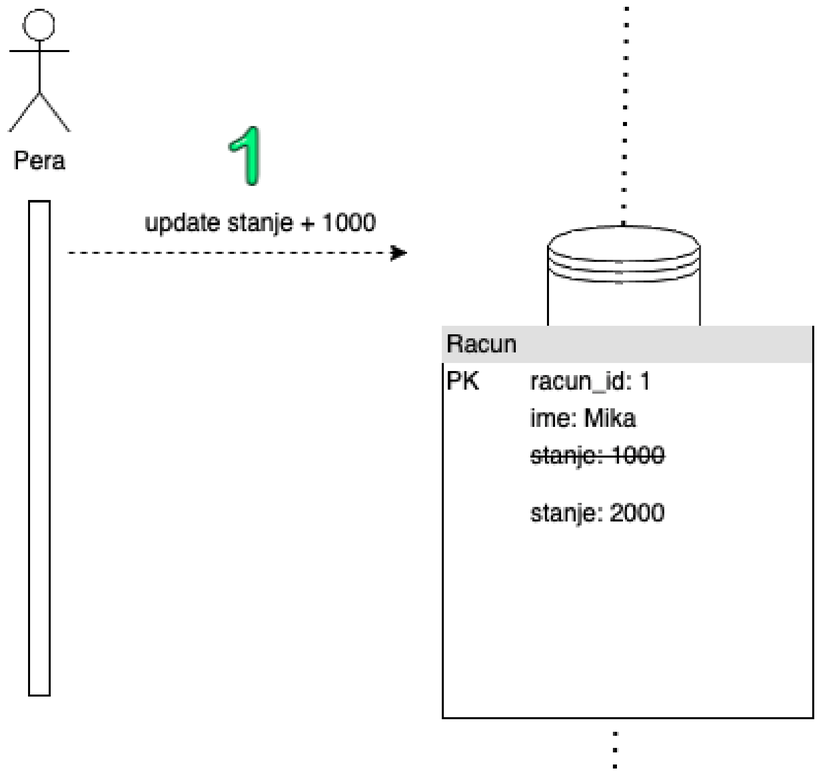
- Jedna transakcija čita nekomitovane modifikacije koje druga konkurentna transakcija napravi a na kraju odradi rollback



DIRTY READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

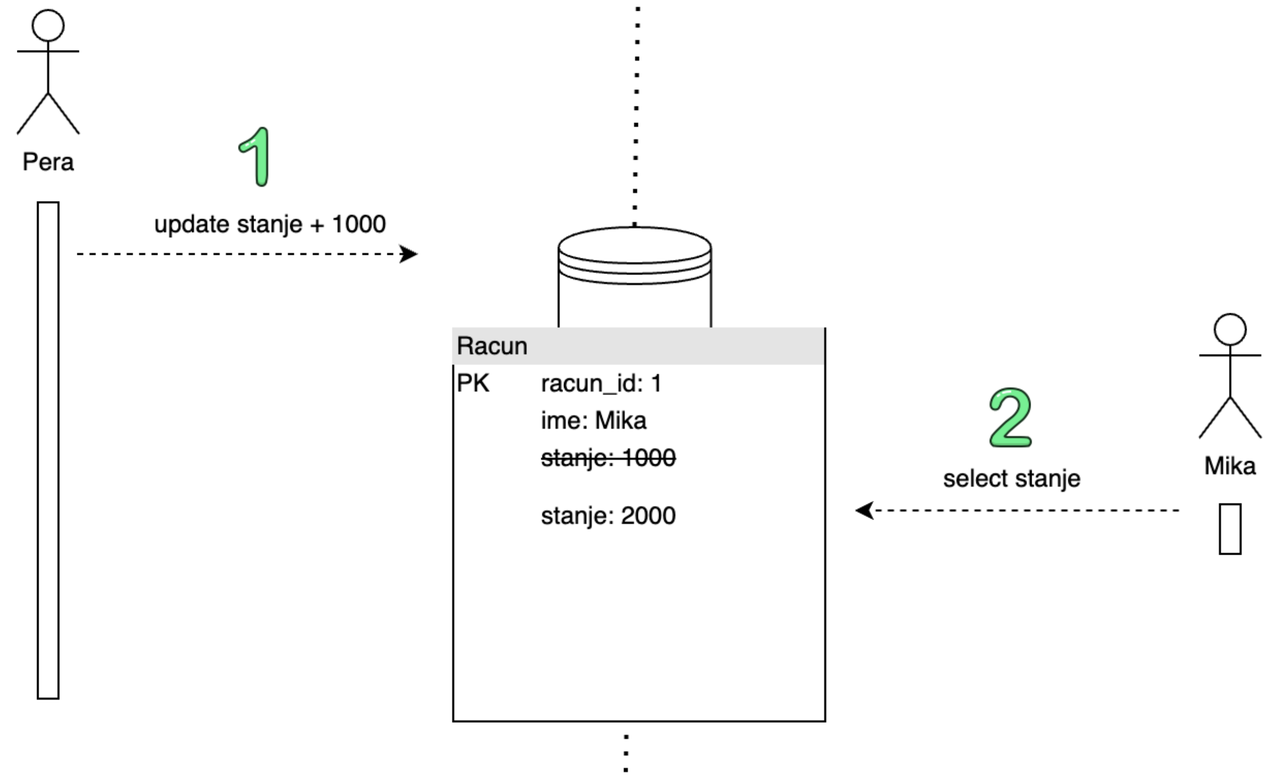
- Jedna transakcija čita nekomitovane modifikacije koje druga konkurentna transakcija napravi a na kraju odradi rollback



DIRTY READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

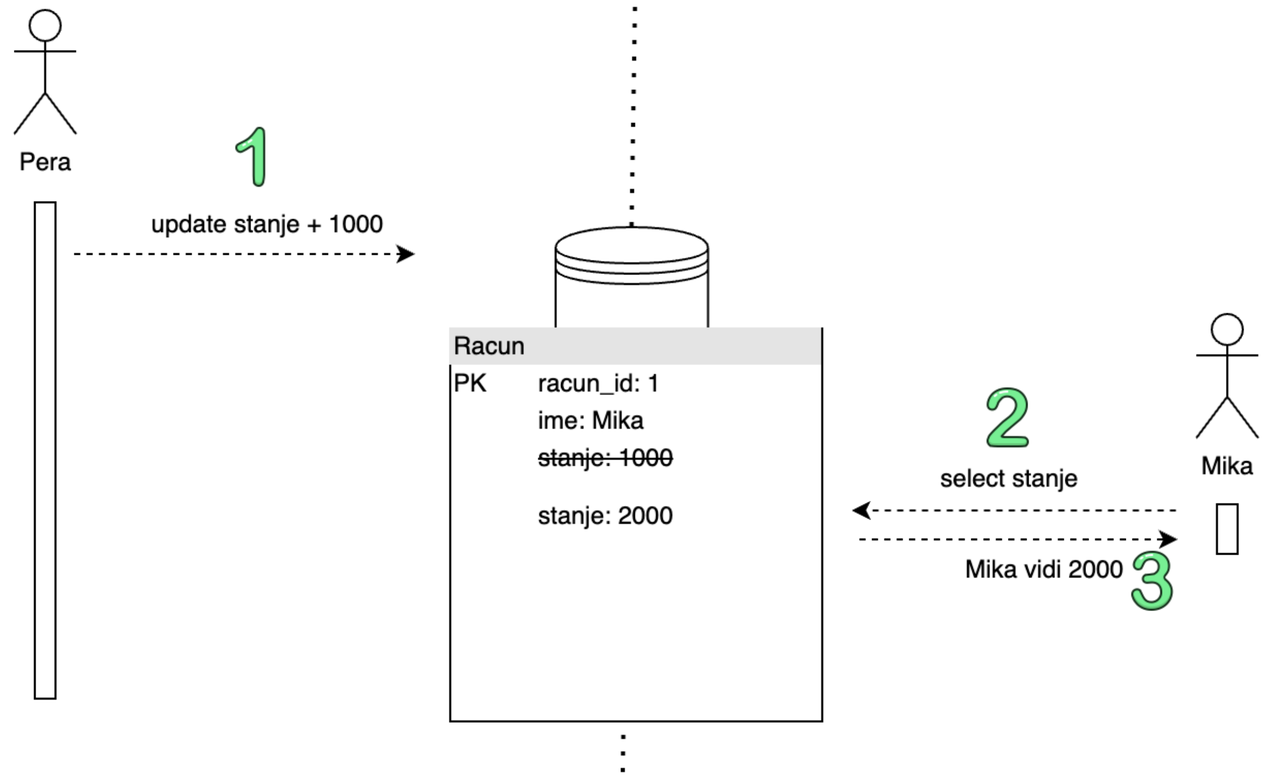
- Jedna transakcija čita nekomitovane modifikacije koje druga konkurentna transakcija napravi a na kraju odradi rollback



DIRTY READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

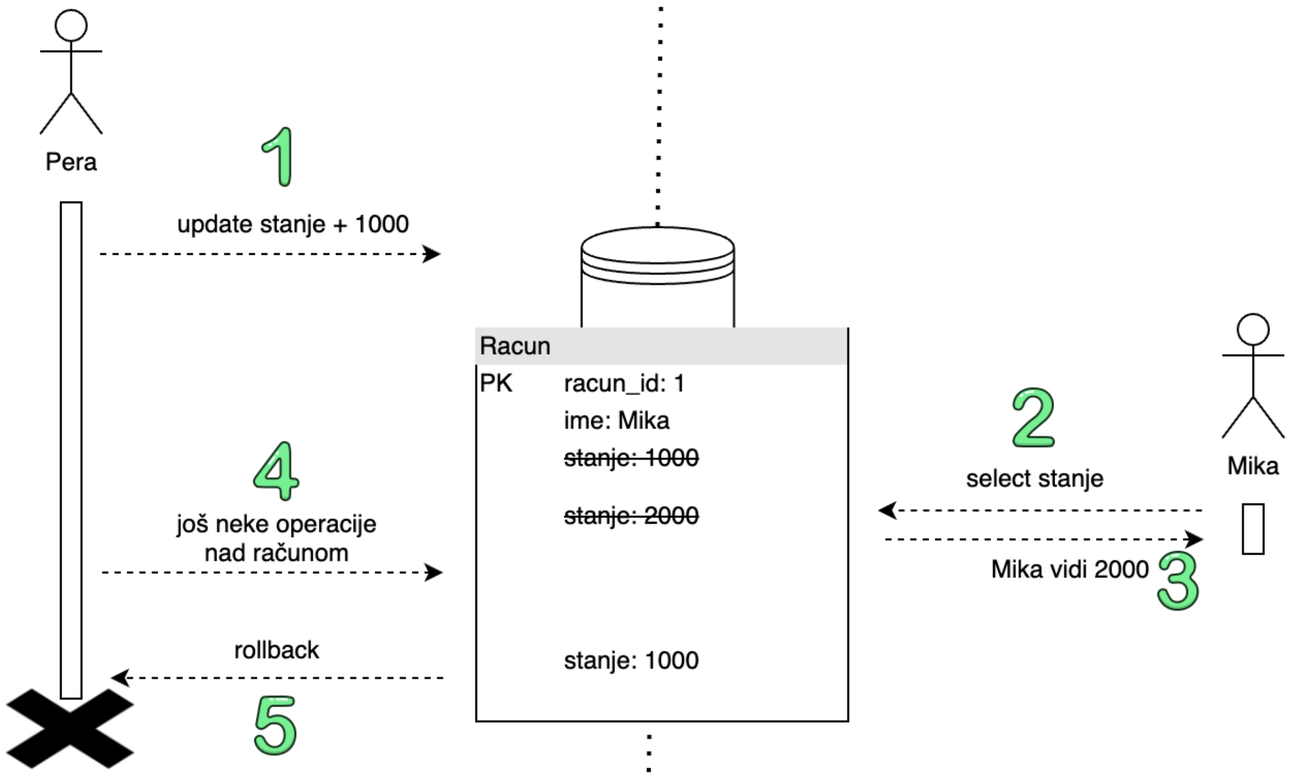
- Jedna transakcija čita nekomitovane modifikacije koje druga konkurentna transakcija napravi a na kraju odradi rollback



DIRTY READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

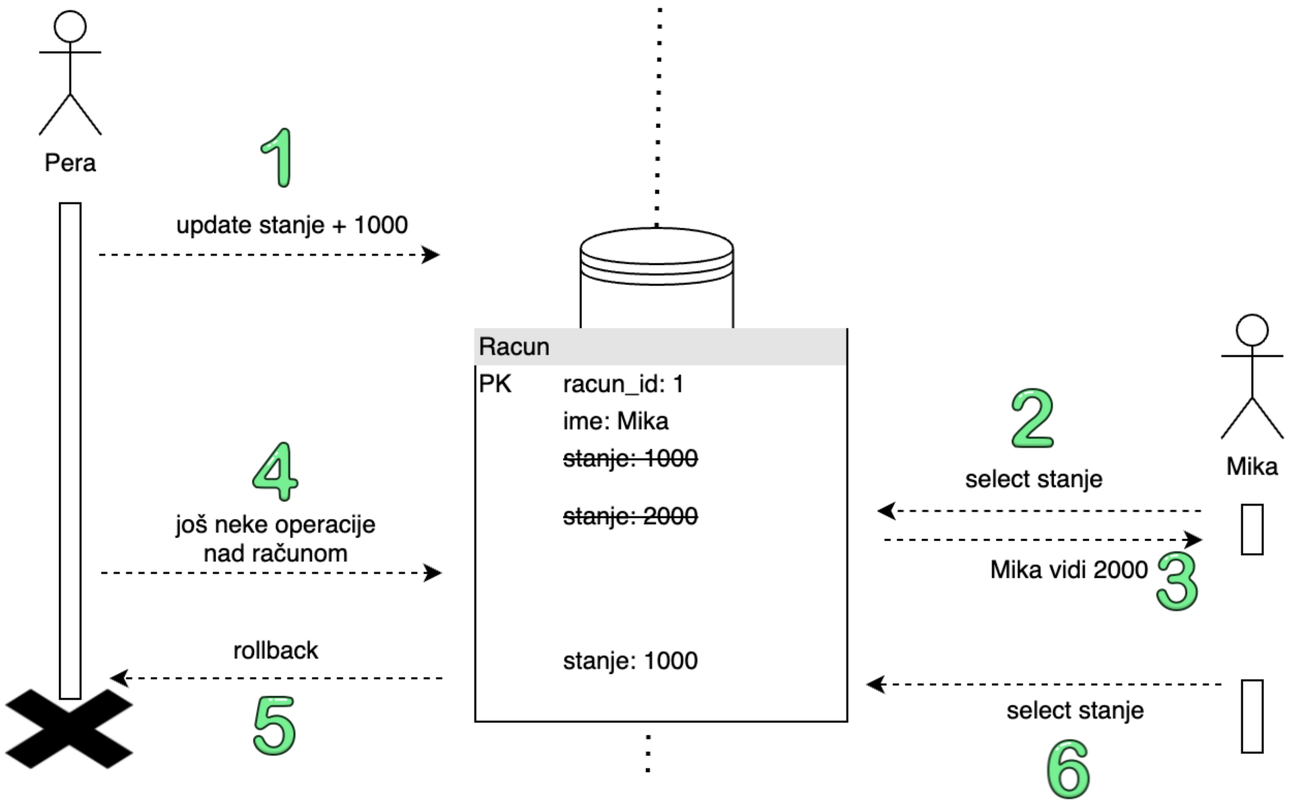
- Jedna transakcija čita nekomitovane modifikacije koje druga konkurentna transakcija napravi a na kraju odradi rollback



DIRTY READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

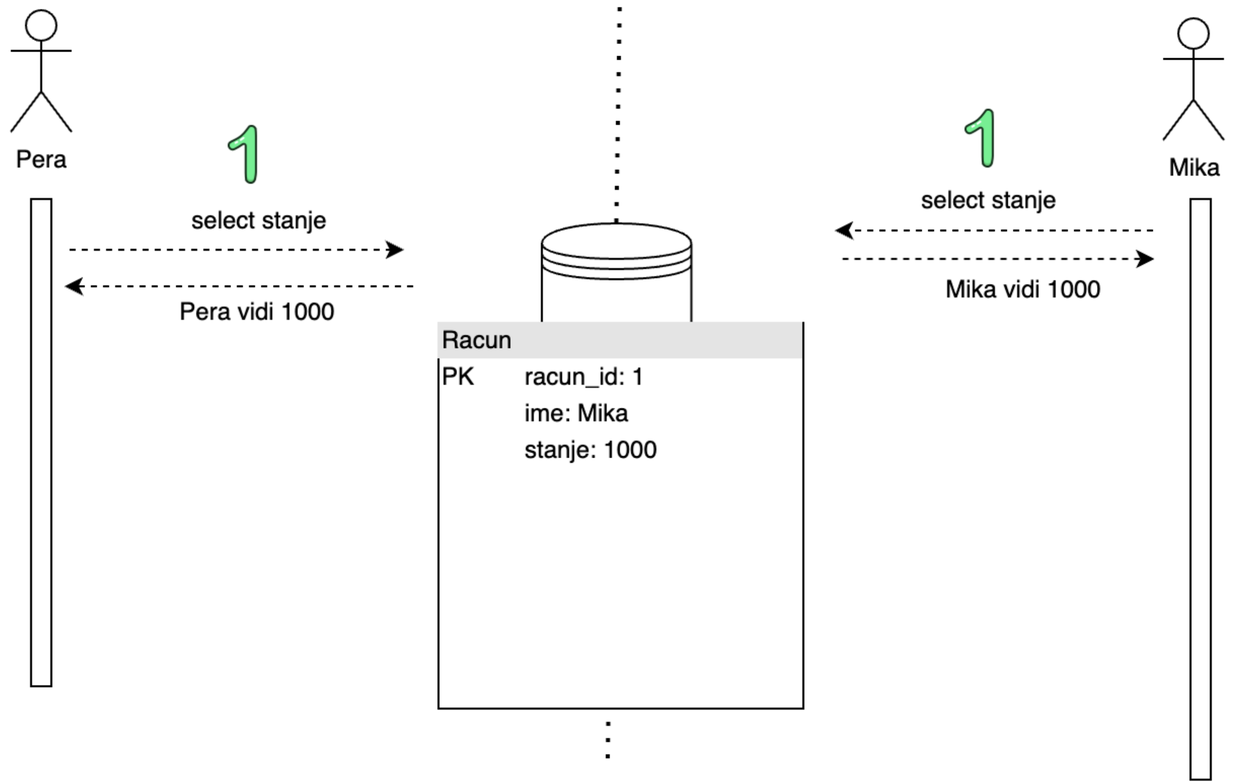
- Jedna transakcija čita nekomitovane modifikacije koje druga konkurentna transakcija napravi a na kraju odradi rollback



NON-REPEATABLE READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

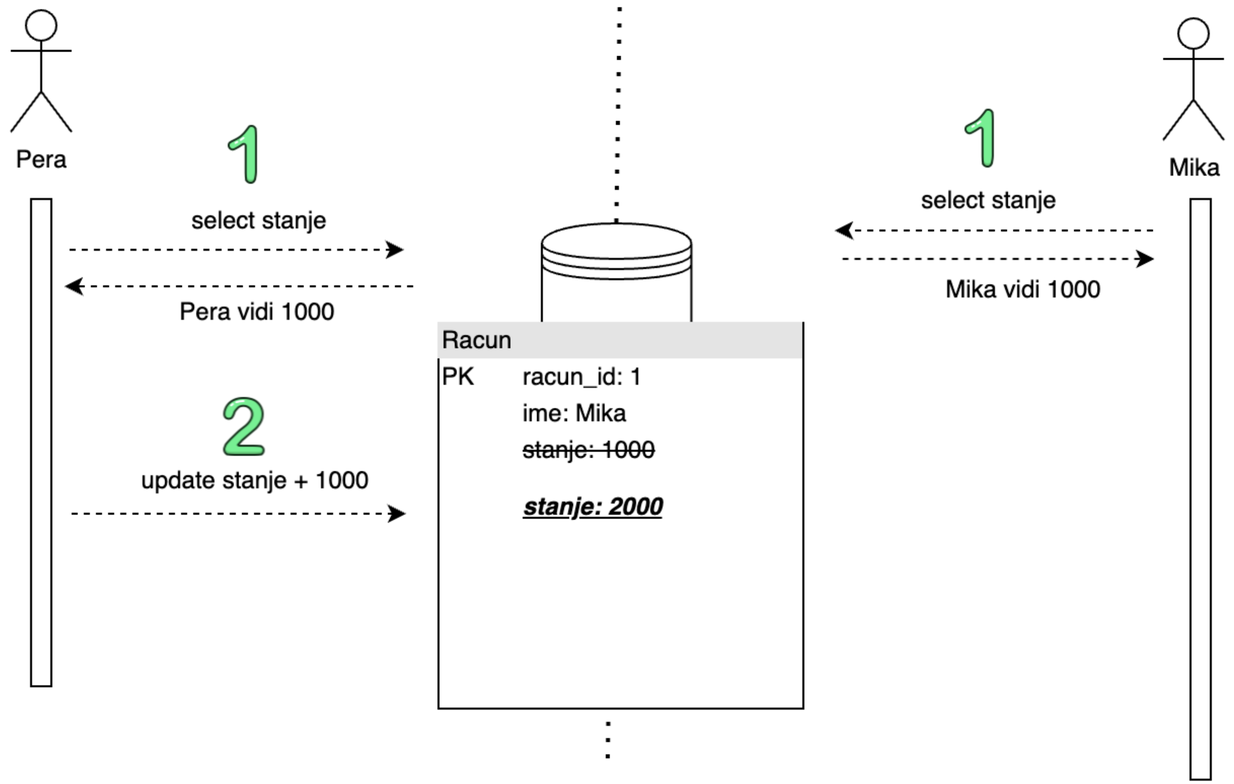
- Prva transakcija čita podatke koje druga konkurentna transakcija menja i radi commit. Prva transakcija ponovo čita podatke i vidi drugačiji rezultat



NON-REPEATABLE READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

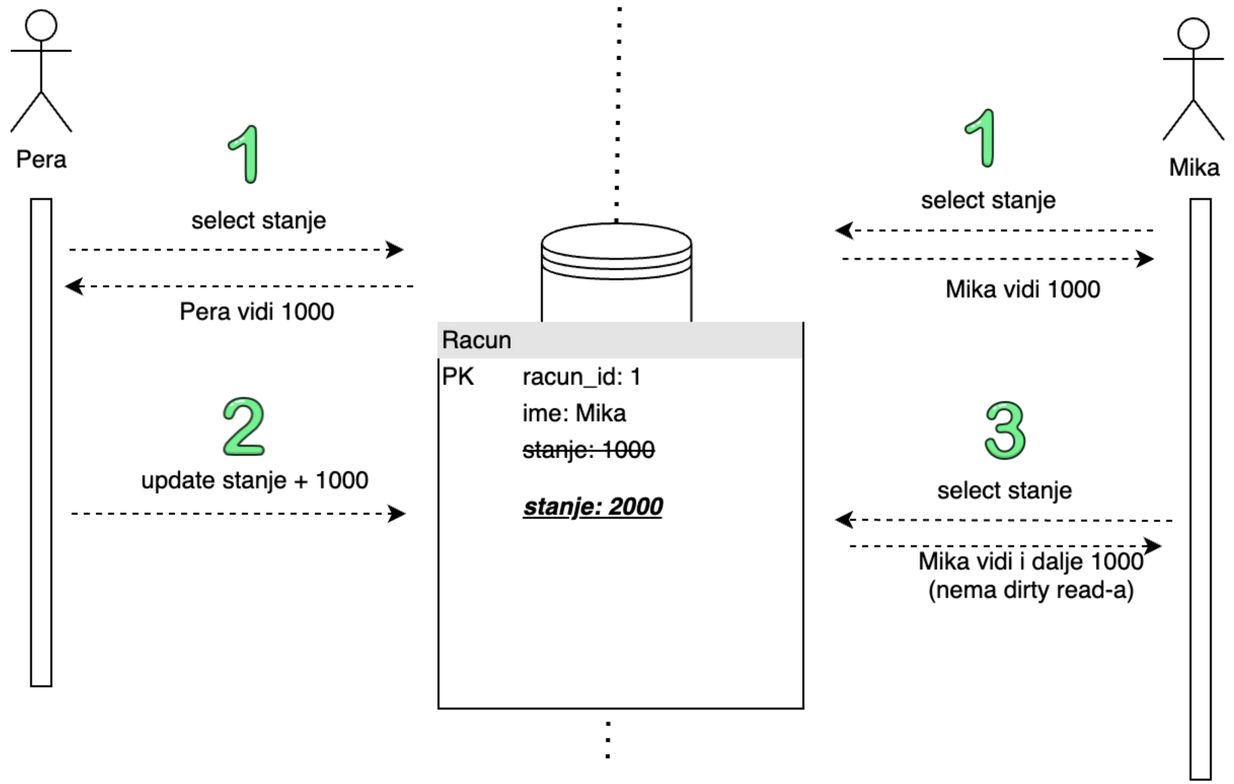
- Prva transakcija čita podatke koje druga konkurentna transakcija menja i radi commit. Prva transakcija ponovo čita podatke i vidi drugačiji rezultat



NON-REPEATABLE READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

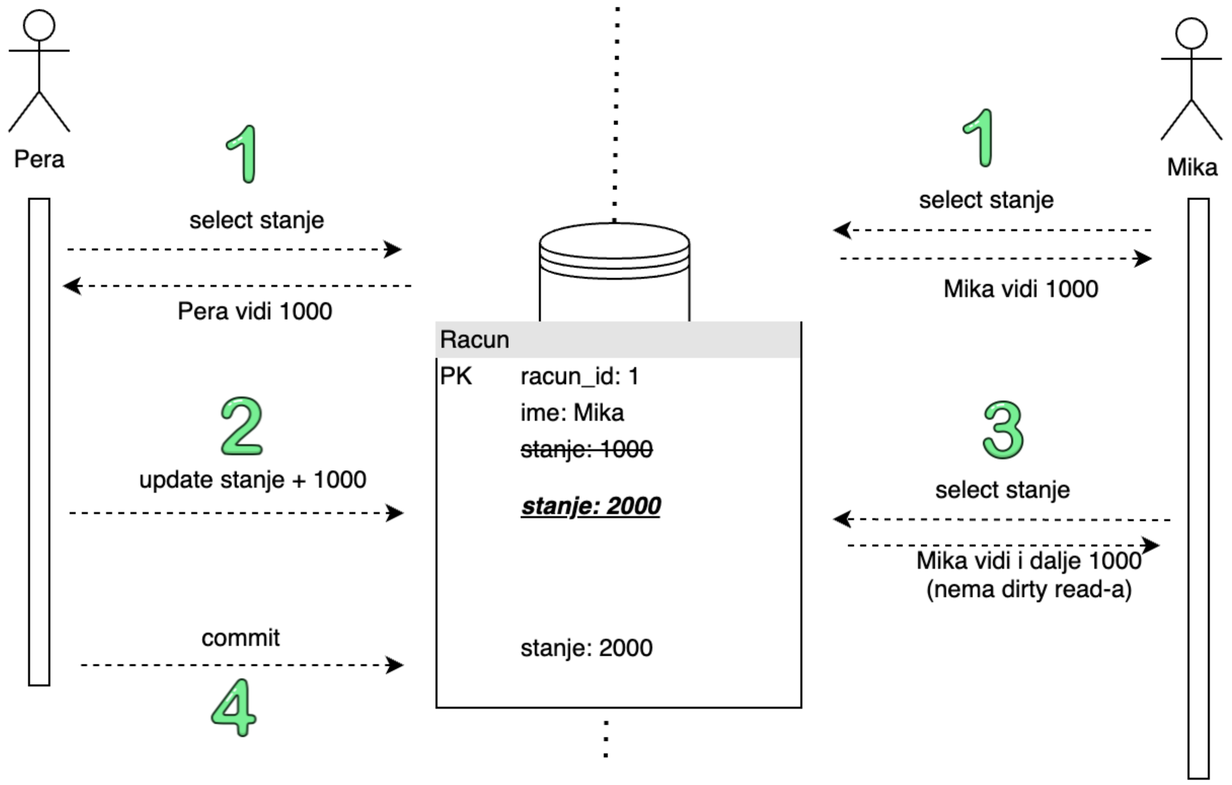
- Prva transakcija čita podatke koje druga konkurentna transakcija menja i radi commit. Prva transakcija ponovo čita podatke i vidi drugačiji rezultat



NON-REPEATABLE READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

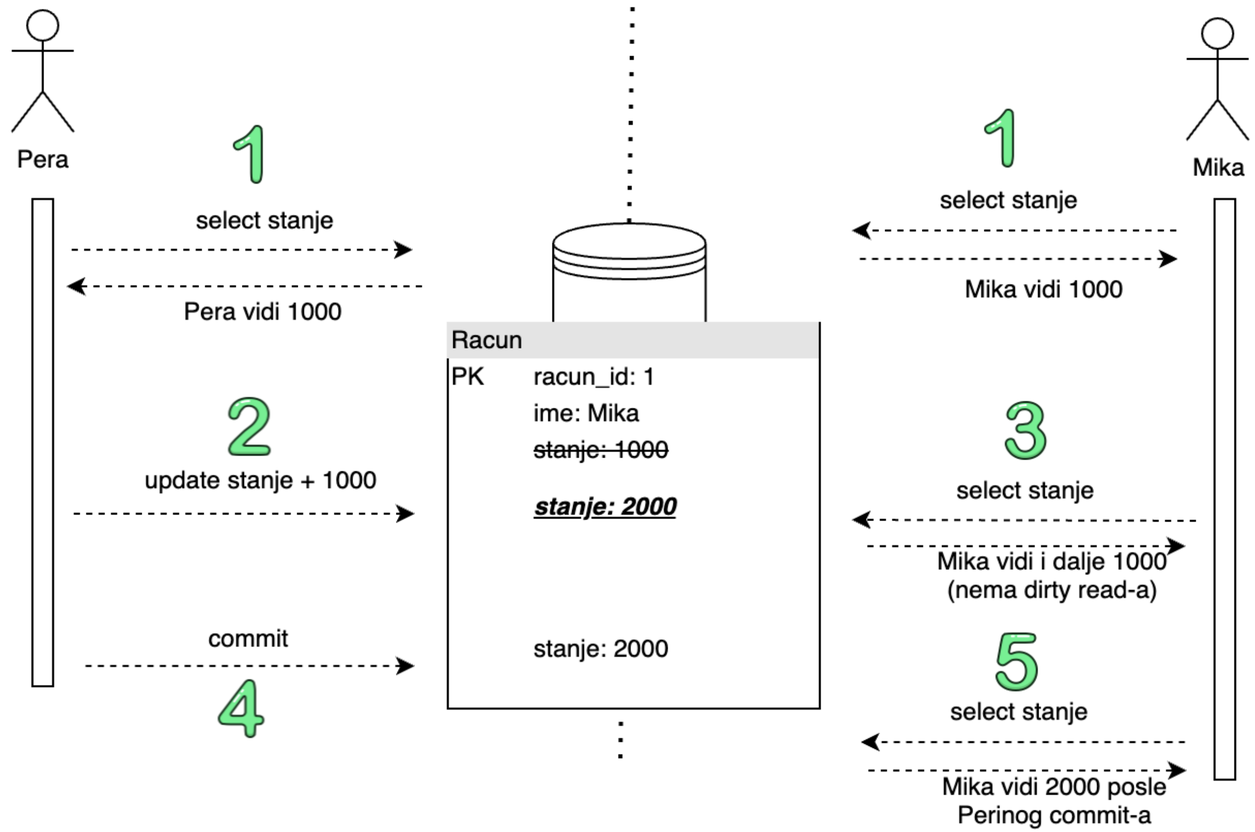
- Prva transakcija čita podatke koje druga konkurentna transakcija menja i radi commit. Prva transakcija ponovo čita podatke i vidi drugačiji rezultat



NON-REPEATABLE READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

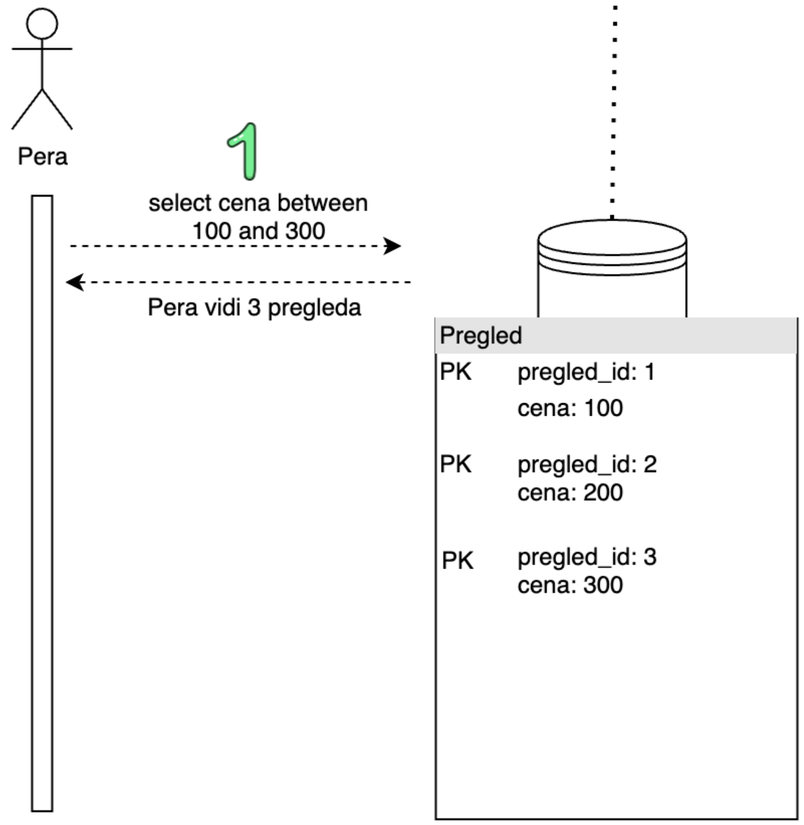
- Prva transakcija čita podatke koje druga konkurentna transakcija menja i radi commit. Prva transakcija ponovo čita podatke i vidi drugačiji rezultat



PHANTOM READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

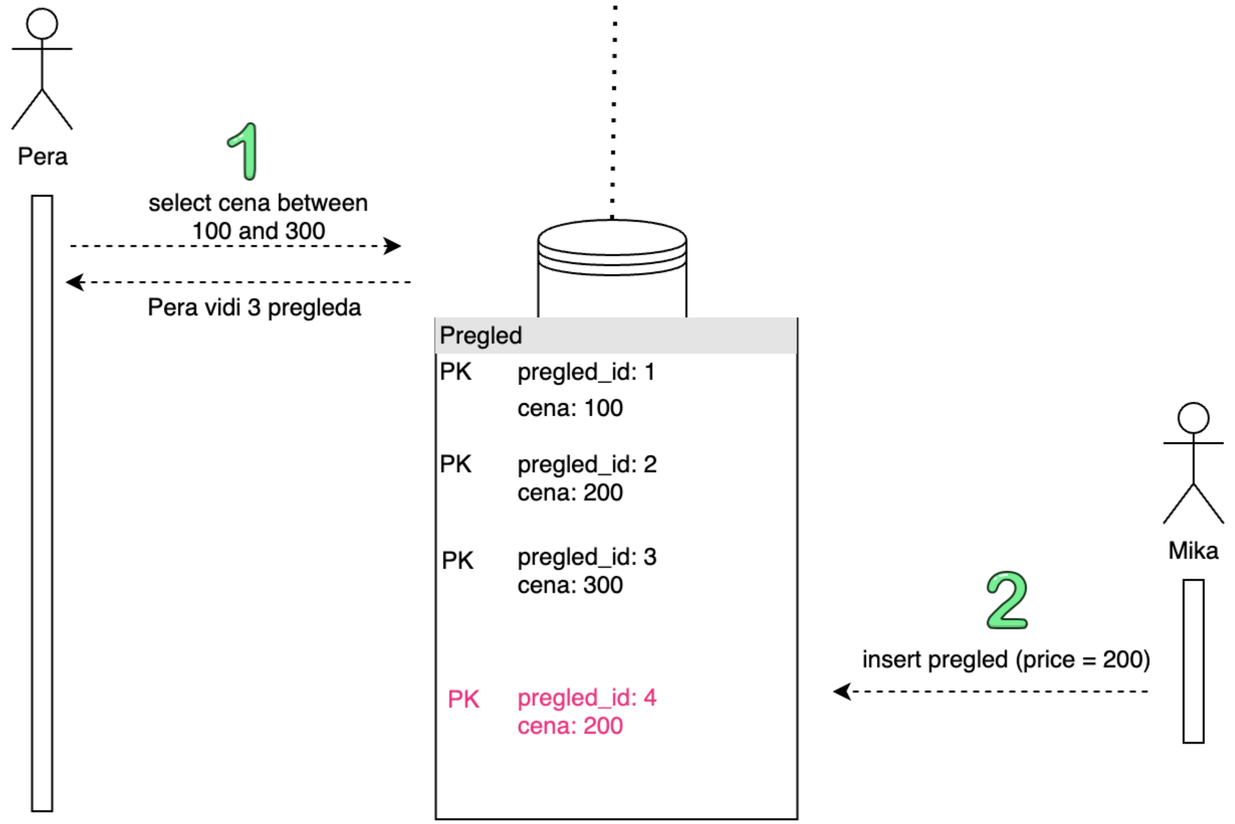
- Jedna transakcija čita torke iz nekog opsega, druga transakcija radi insert nove torke u isti opseg i radi commit. Prva transakcija ponovo čita torke iz istog opsega i vidi ”fantomsku torku”



PHANTOM READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

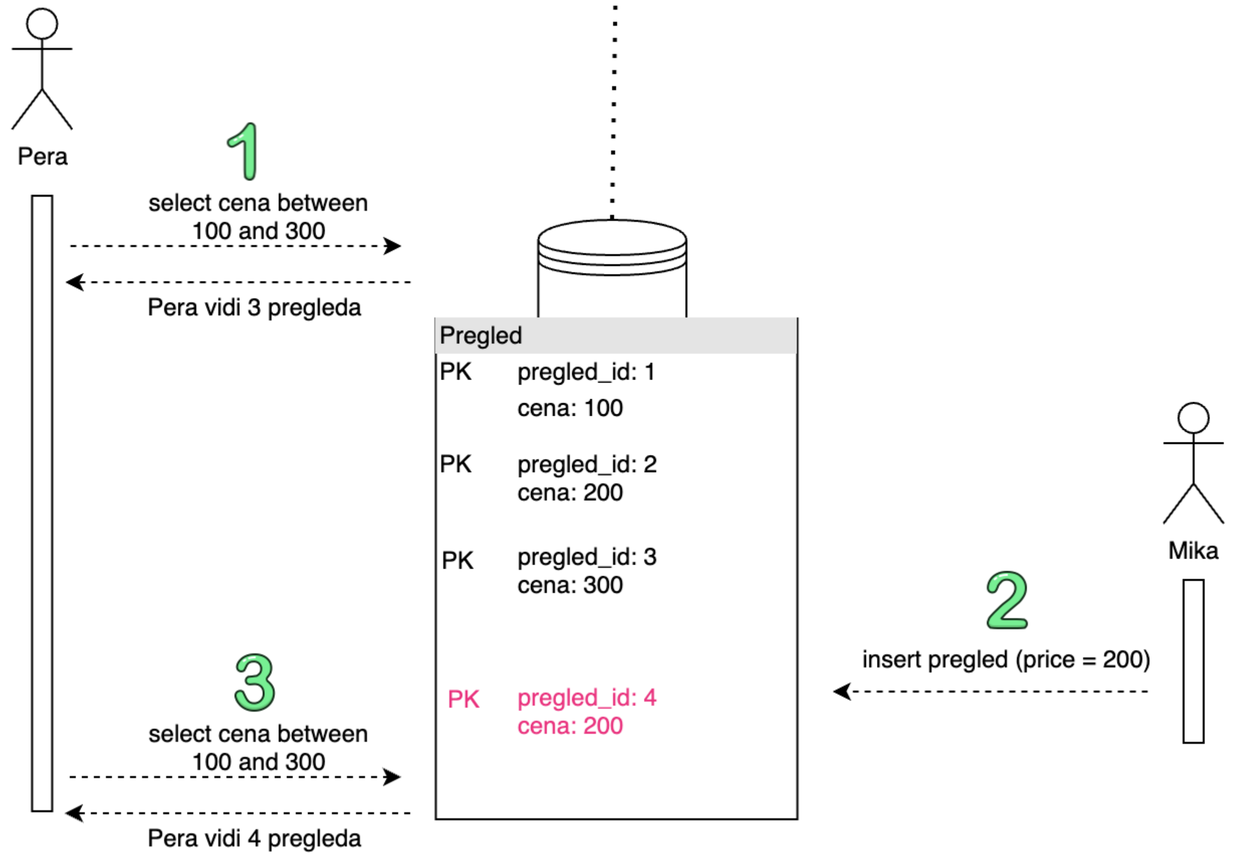
- Jedna transakcija čita torke iz nekog opsega, druga transakcija radi insert nove torke u isti opseg i radi commit. Prva transakcija ponovo čita torke iz istog opsega i vidi ”fantomsku torku”



PHANTOM READS

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

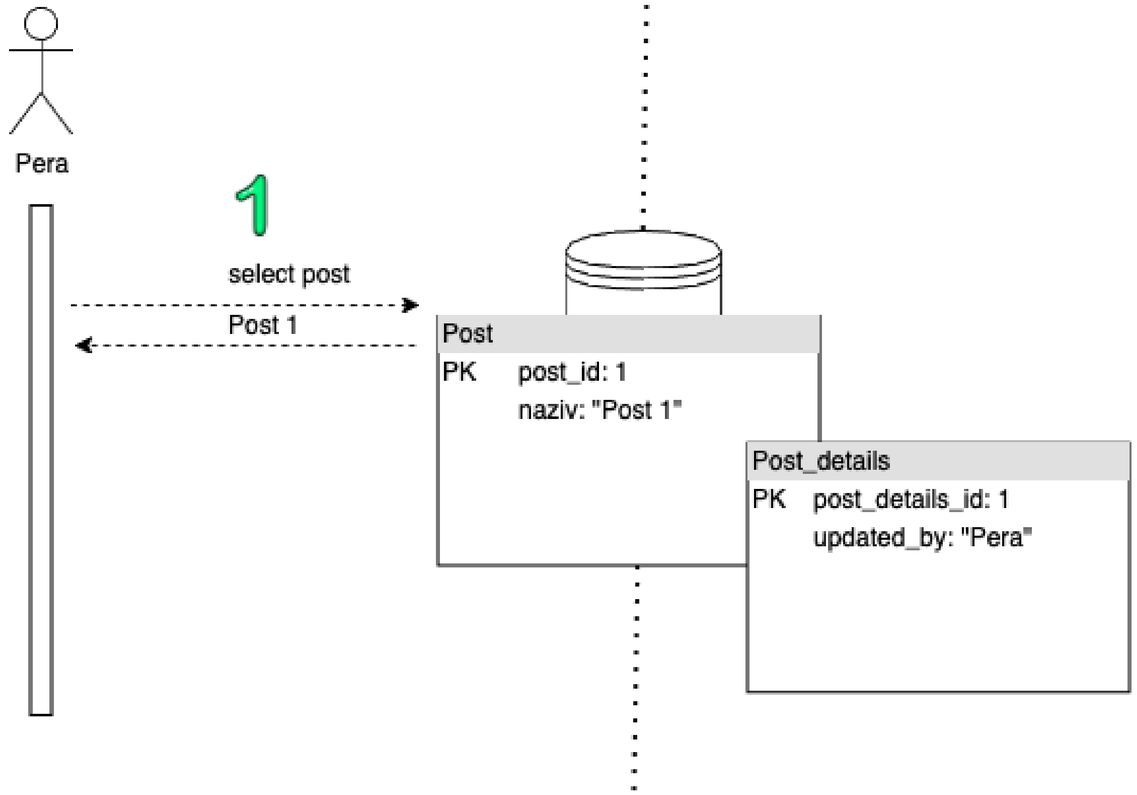
- Jedna transakcija čita torke iz nekog opsega, druga transakcija radi insert nove torke u isti opseg i radi commit. Prva transakcija ponovo čita torke iz istog opsega i vidi ”fantomsku torku”



READ SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

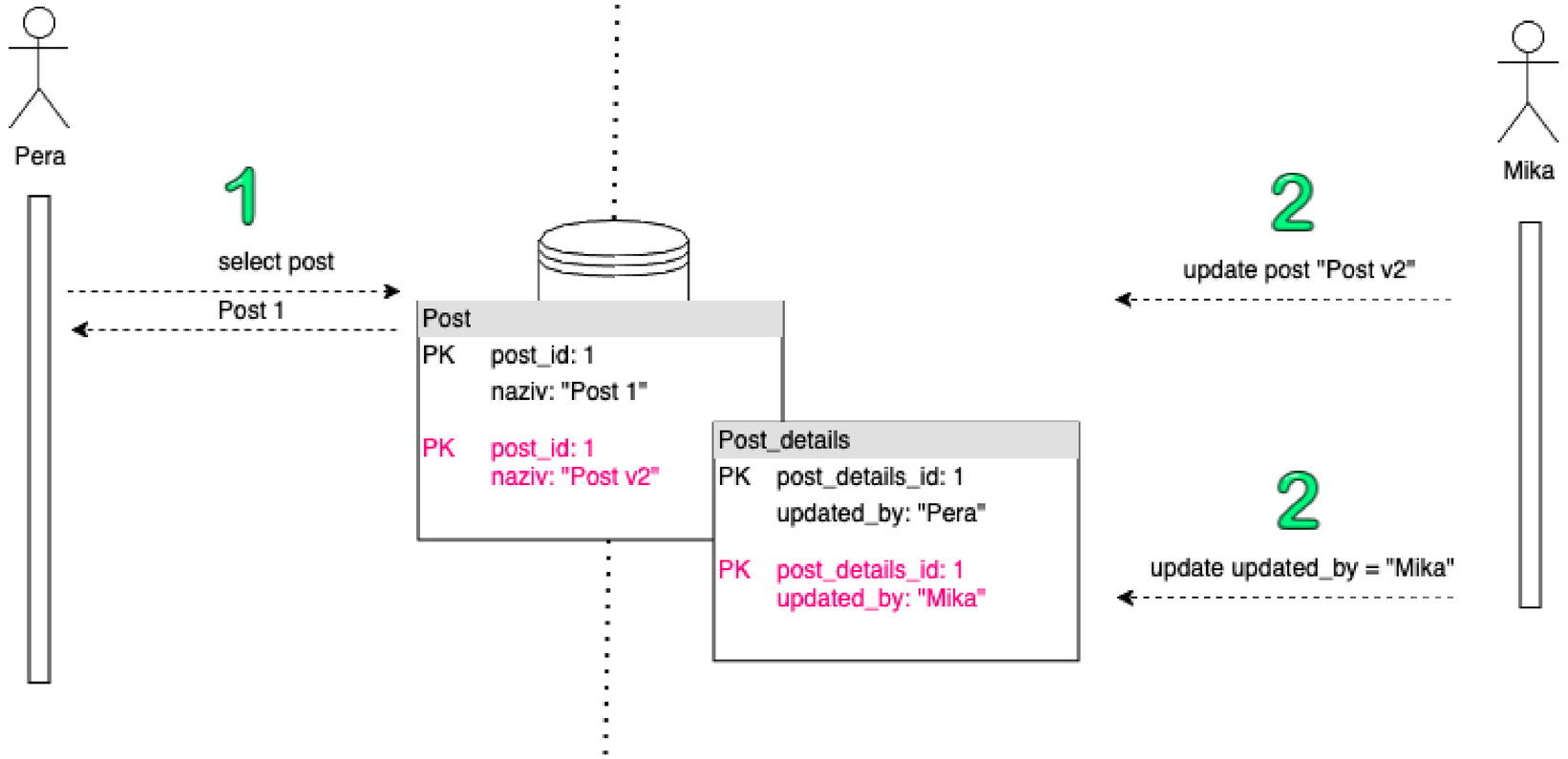
- Moraju biti uključene bar dve tabele u scenario. Prva transakcija čita iz prve tabele, druga transakcija ažurira obe tabele. Kada su obe tabele ažurirane, prva transakcija čita iz druge tabele na osnovu podatka iz prve tabele i vidi staru verziju podatka iz prve tabele i novu verziju podatka iz druge tabele.



READ SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

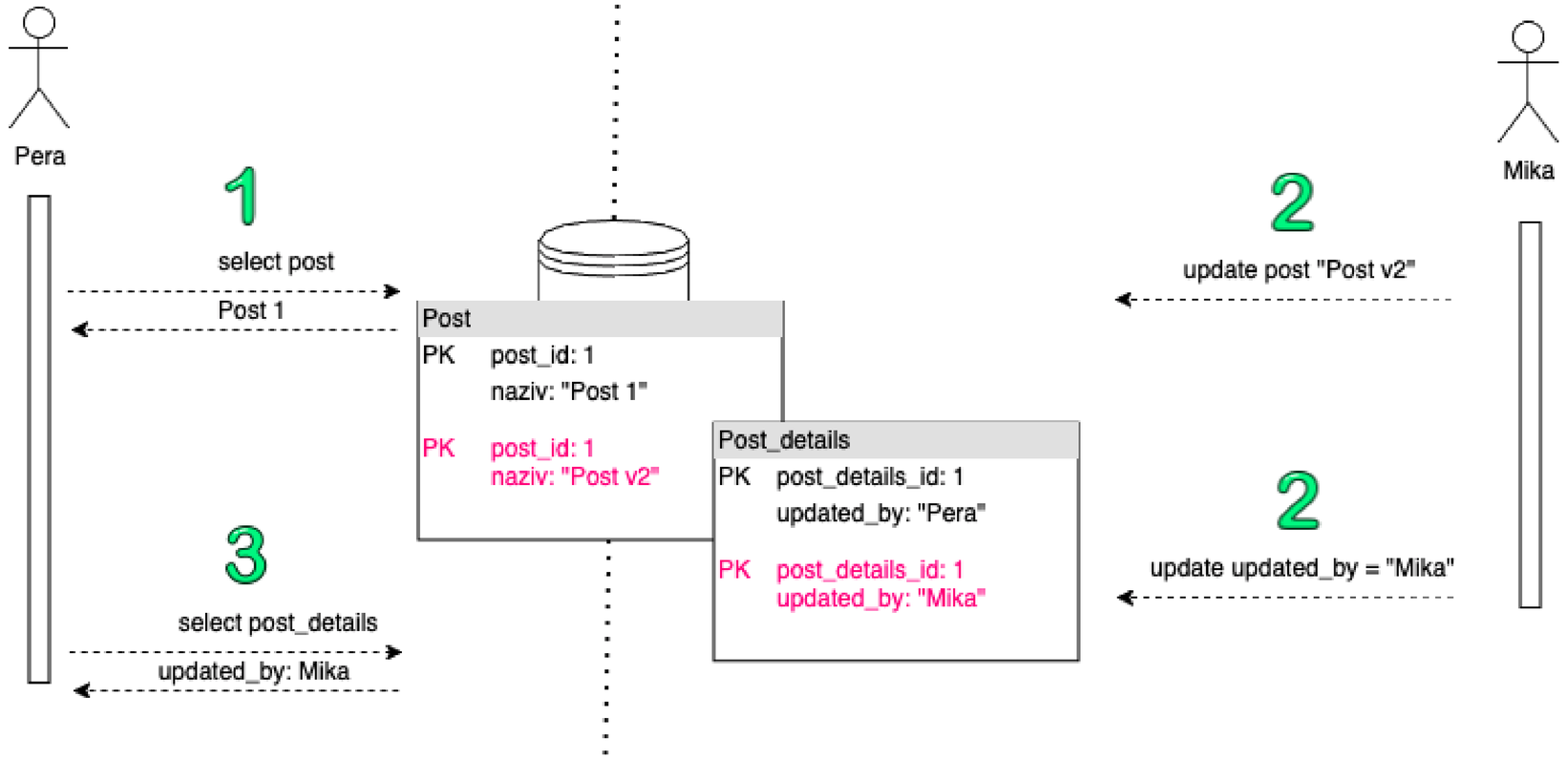
- Moraju biti uključene bar dve tabele u scenario. Prva transakcija čita iz prve tabele, druga transakcija ažurira obe tabele. Kada su obe tabele ažurirane, prva transakcija čita iz druge tabele na osnovu podatka iz prve tabele i vidi staru verziju podatka iz prve tabele i novu verziju podatka iz druge tabele.



READ SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

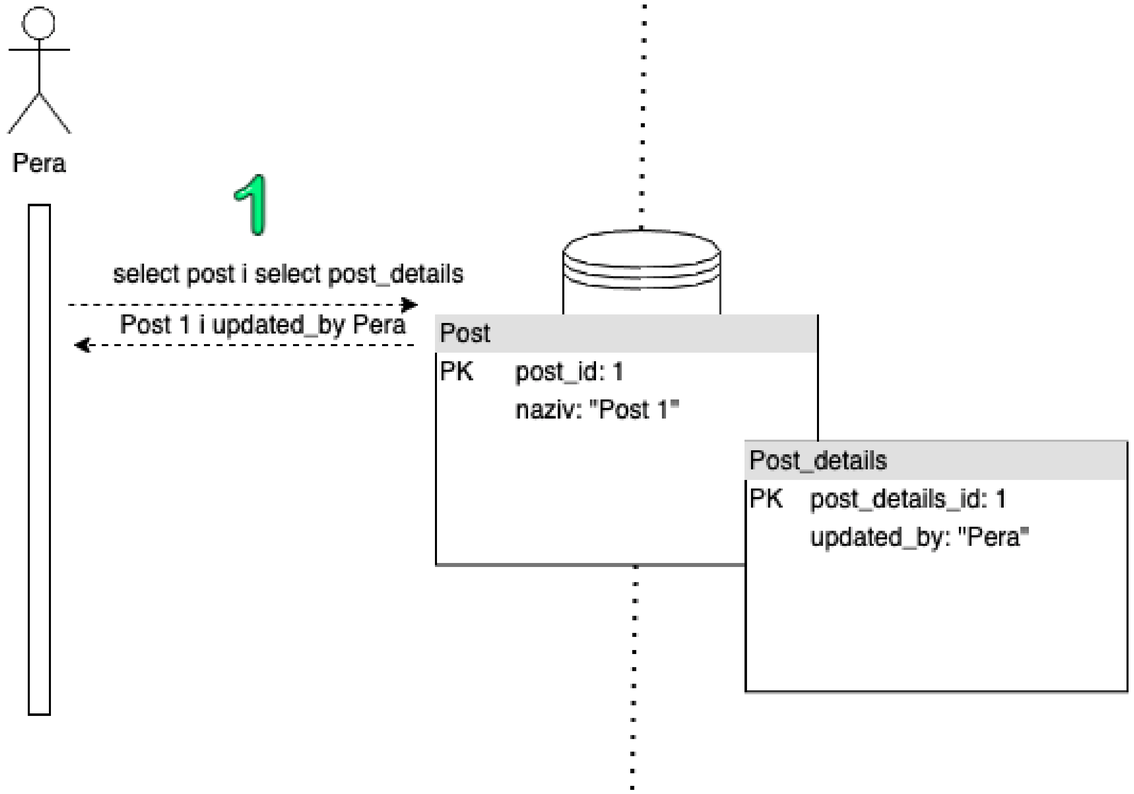
- Moraju biti uključene bar dve tabele u scenario. Prva transakcija čita iz prve tabele, druga transakcija ažurira obe tabele. Kada su obe tabele ažurirane, prva transakcija čita iz druge tabele na osnovu podatka iz prve tabele i vidi staru verziju podatka iz prve tabele i novu verziju podatka iz druge tabele.



WRITE SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

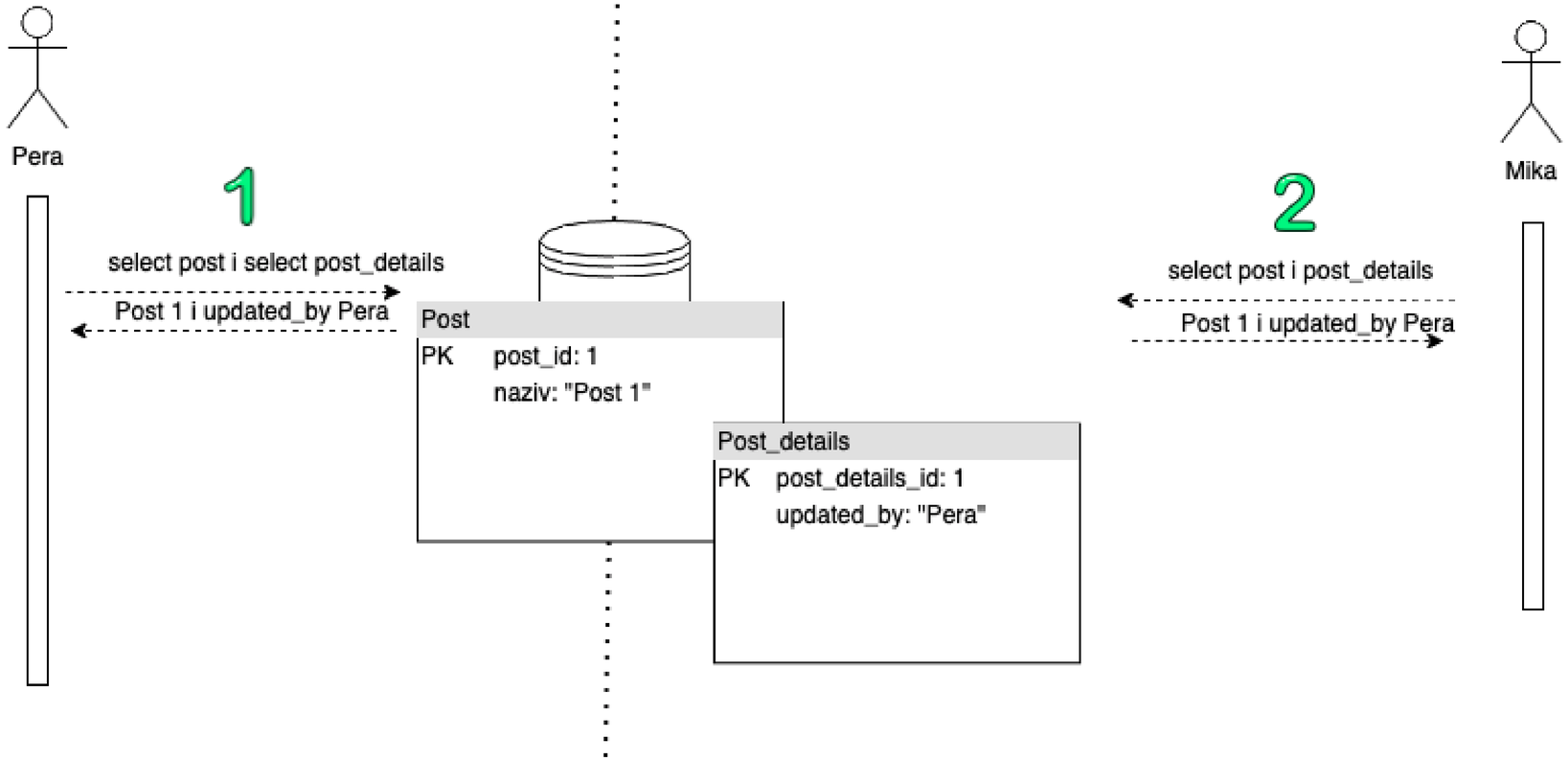
- Moraju biti uključene bar dve tabele u scenario. Obe transakcije čitaju iste podatke iz obe tabele, prva transakcija na osnovu očitanih podataka menja podatke u prvoj tabeli, druga transakcija menja podatke u drugoj tabeli.



WRITE SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

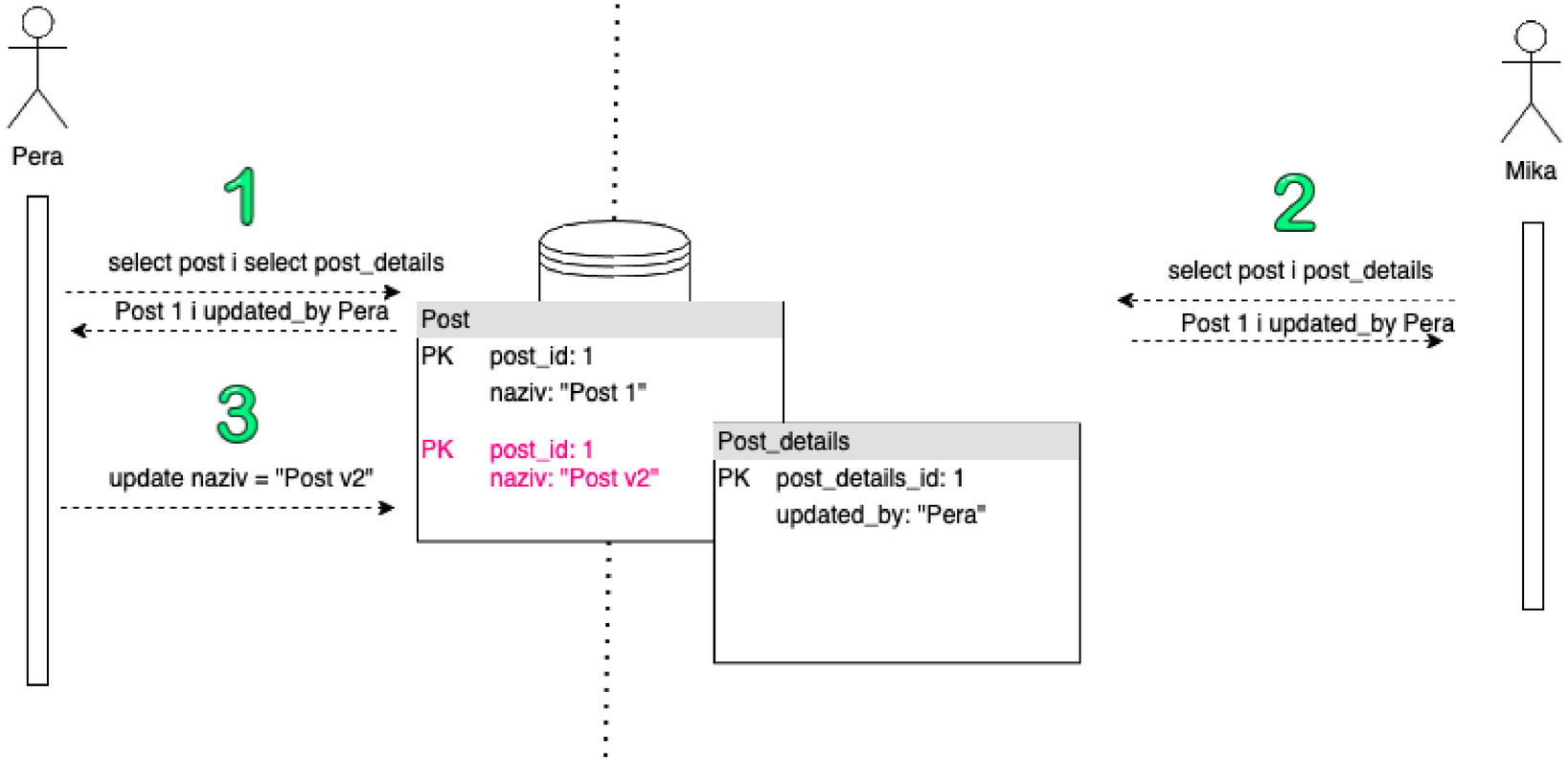
- Moraju biti uključene bar dve tabele u scenario. Obe transakcije čitaju iste podatke iz obe tabele, prva transakcija na osnovu očitanih podataka menja podatke u prvoj tabeli, druga transakcija menja podatke u drugoj tabeli.



WRITE SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

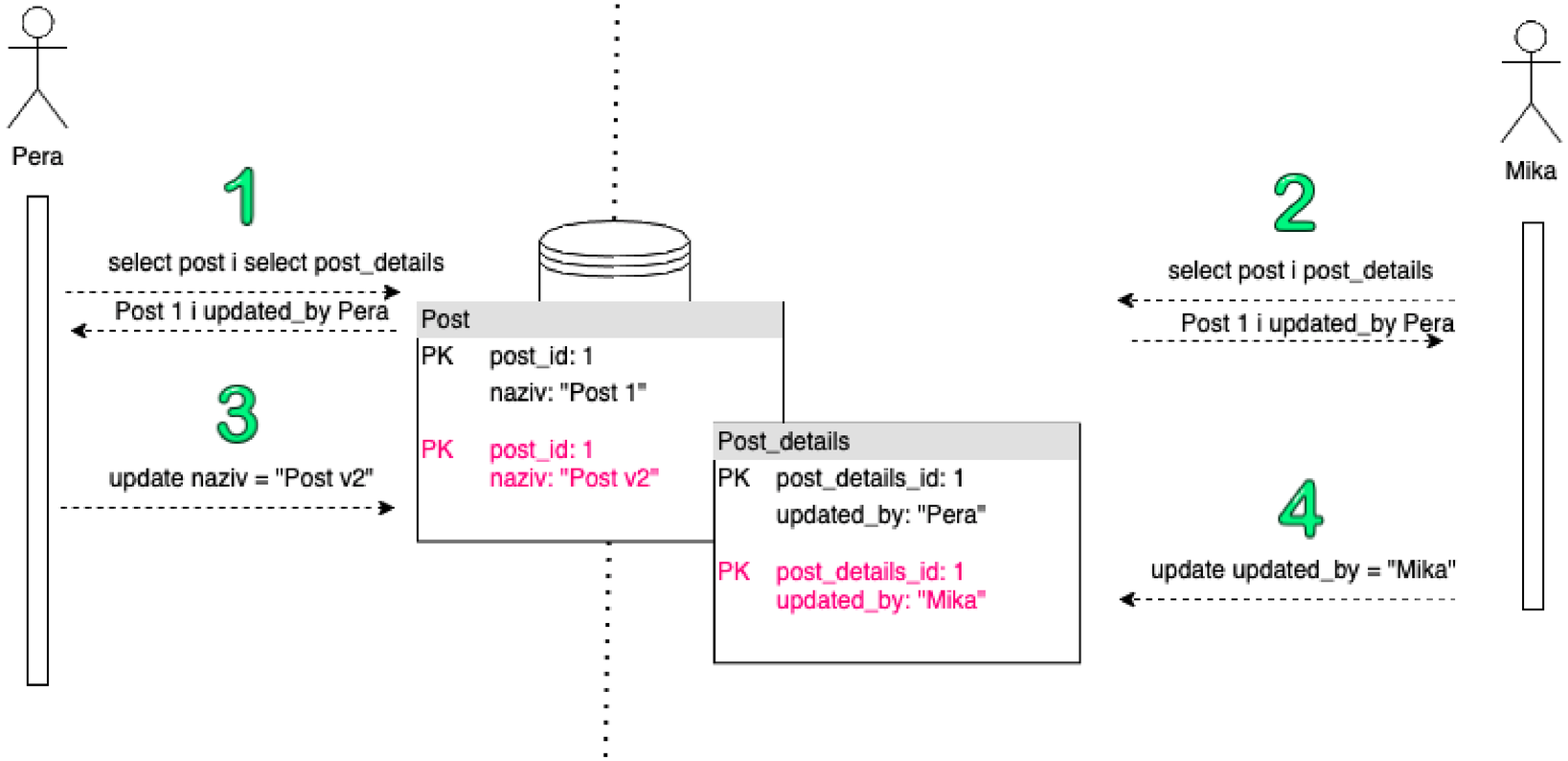
- Moraju biti uključene bar dve tabele u scenario. Obe transakcije čitaju iste podatke iz obe tabele, prva transakcija na osnovu očitanih podataka menja podatke u prvoj tabeli, druga transakcija menja podatke u drugoj tabeli.



WRITE SKEW

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

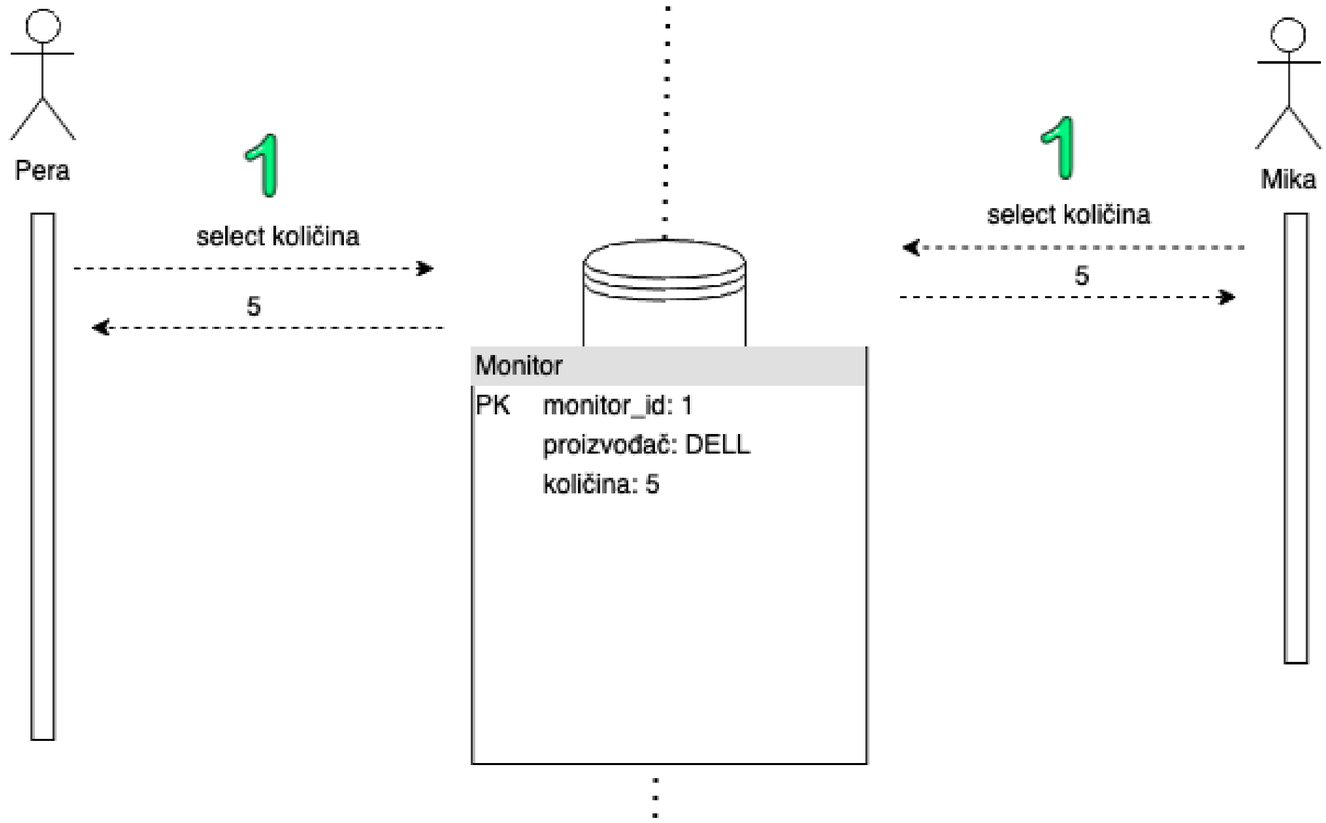
- Moraju biti uključene bar dve tabele u scenario. Obe transakcije čitaju iste podatke iz obe tabele, prva transakcija na osnovu očitanih podataka menja podatke u prvoj tabeli, druga transakcija menja podatke u drugoj tabeli.



LOST UPDATE

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

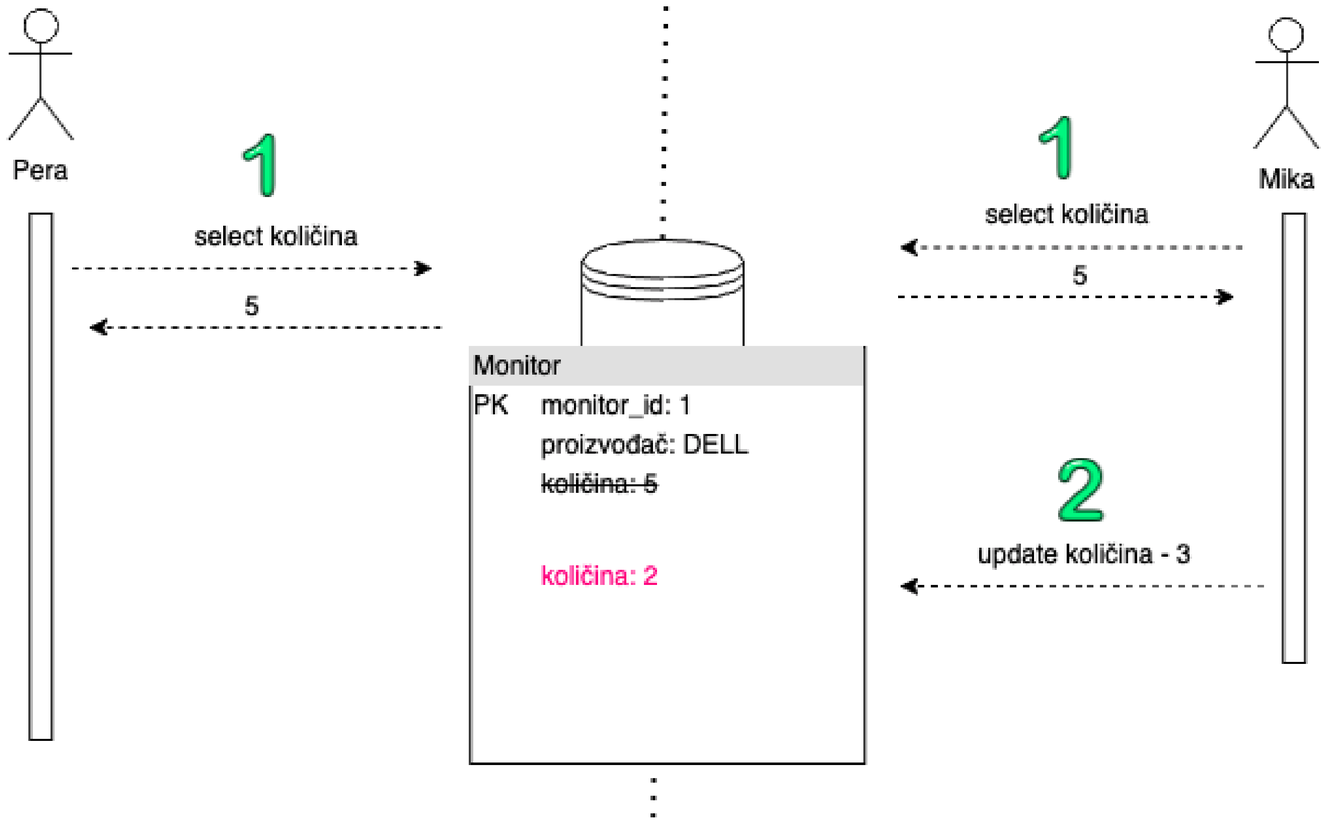
Jedna transakcija čita podatke iz tabele na osnovu koje treba da se napravi neka odluka a nije svesna da je druga transakcija već modifikovala te iste podatke i komitovala ih.



LOST UPDATE

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

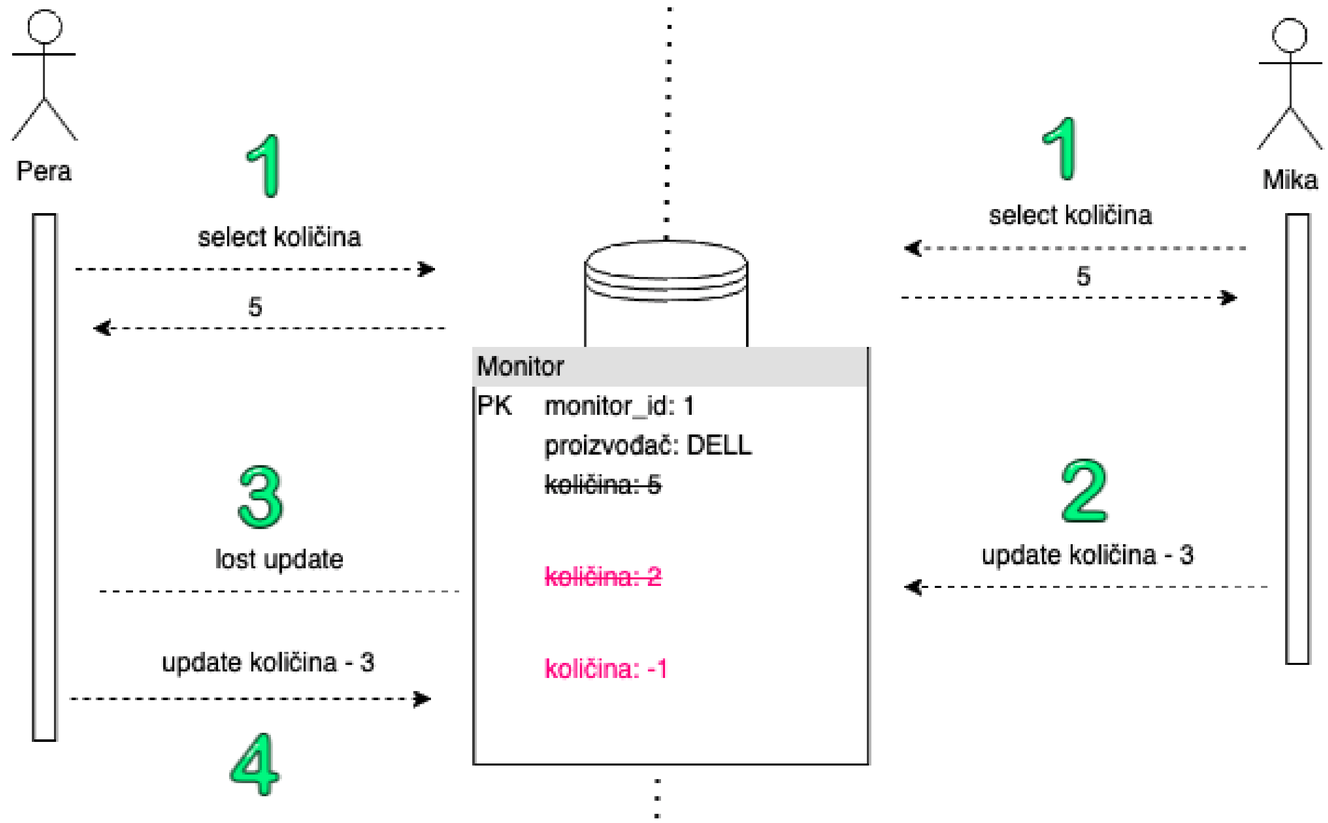
Jedna transakcija čita podatke iz tabele na osnovu koje treba da se napravi neka odluka a nije svesna da je druga transakcija već modifikovala te iste podatke i komitovala ih.



LOST UPDATE

ŠTA SE DOGAĐA KOD OVE ANOMALIJE?

Jedna transakcija čita podatke iz tabele na osnovu koje treba da se napravi neka odluka a nije svesna da je druga transakcija već modifikovala te iste podatke i komitovala ih.



NIVOI IZOLACIJE

Da li je dozvoljena anomalija? / Nivo izolacije	Dirty read	Non-repeatable read	Phantom read
Read Uncommited	DA	DA	DA
Read Commited	NE	DA	DA
Repeatable Read	NE	NE	DA
Serializable	NE	NE	NE

IZDRŽLJIVOST PODATAKA

ŠTA JE IZDRŽLJIVOST PODATAKA?

- Svojstvo koje omogućava da ako se transakcija izvrši, njeni efekti (izmene) budu trajno sačuvani

KAKO SE PRATI DA LI SU SVE OPERACIJE IZVRŠENE?

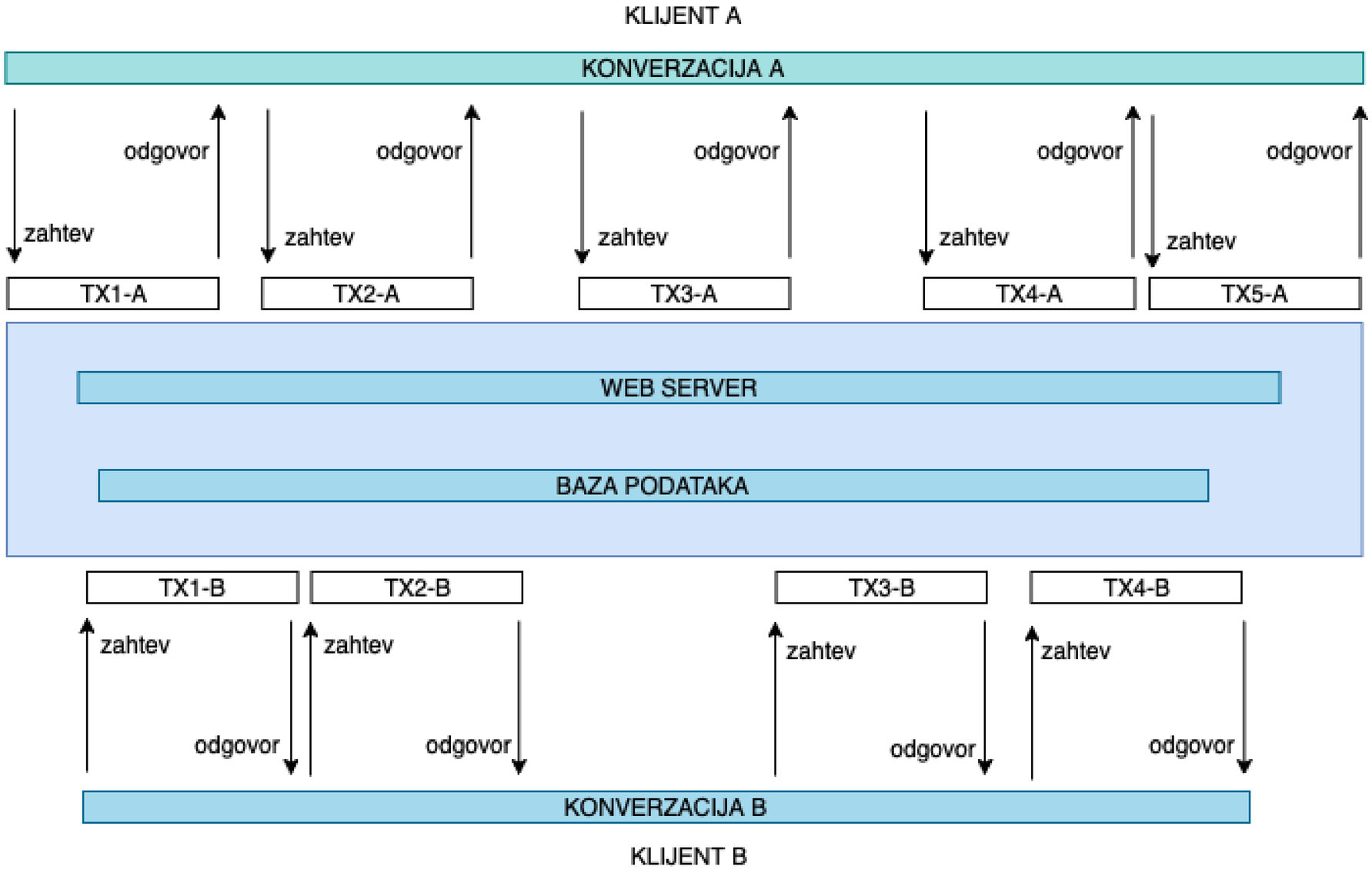
- Sve izmene se čuvaju u logovima kako bi se podaci mogli vratiti u prvobitno stanje
- Prave se posebne kopije originalnih stranica gde su podaci nalaze nad kojima transakcije obavljaju operacije

TRANSAKCIJE NA APLIKATIVNOM NIVOU

ŠTA SU DUGE KONVERZACIJE?

- Sa stanovišta aplikacije, cela komunikacija sa bazom se ne mora uvek obaviti u jednoj fizičkoj transakciji
- “Logička transakcija” se može sastojati od više korisničkih zahteva iz pretraživača ka serveru koji zavise od odgovora koje dobiju iz baze kao međurezultat na osnovu kojeg prave naredni zahtev (i razmišljaju :)) itd.
- Tokom duže zahtev-odgovor komunikacije, izvršiće se više transakcija
- Celu tu komunikaciju možemo zvati dugom konverzacijom

TRANSAKCIJE NA APLIKATIVNOM NIVOU

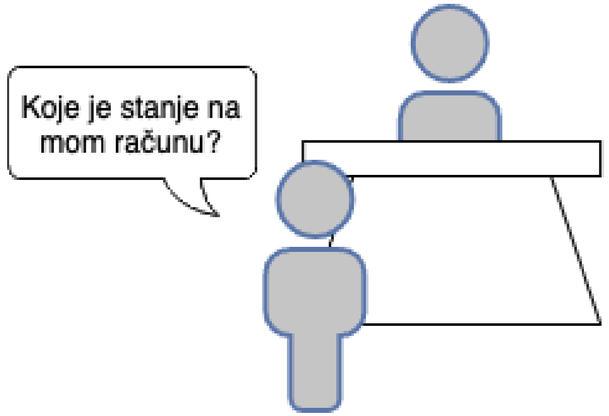


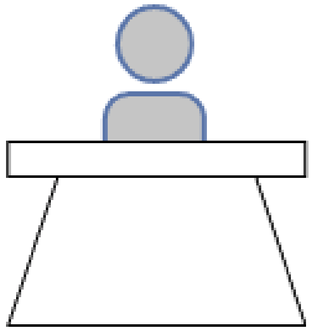
TRANSAKCIJE NA APLIKATIVNOM NIVOU

- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**

TRANSAKCIJE NA APLIKATIVNOM NIVOU

- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**

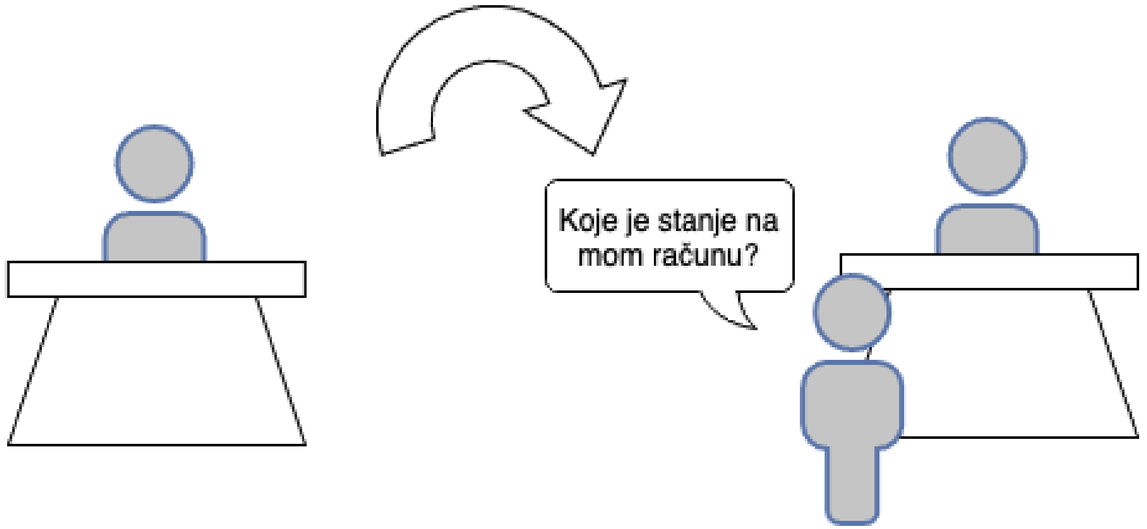




```
SELECT stanje FROM racuni
WHERE br_racuna = 1234567;
```

TRANSAKCIJE NA APLIKATIVNOM NIVOU

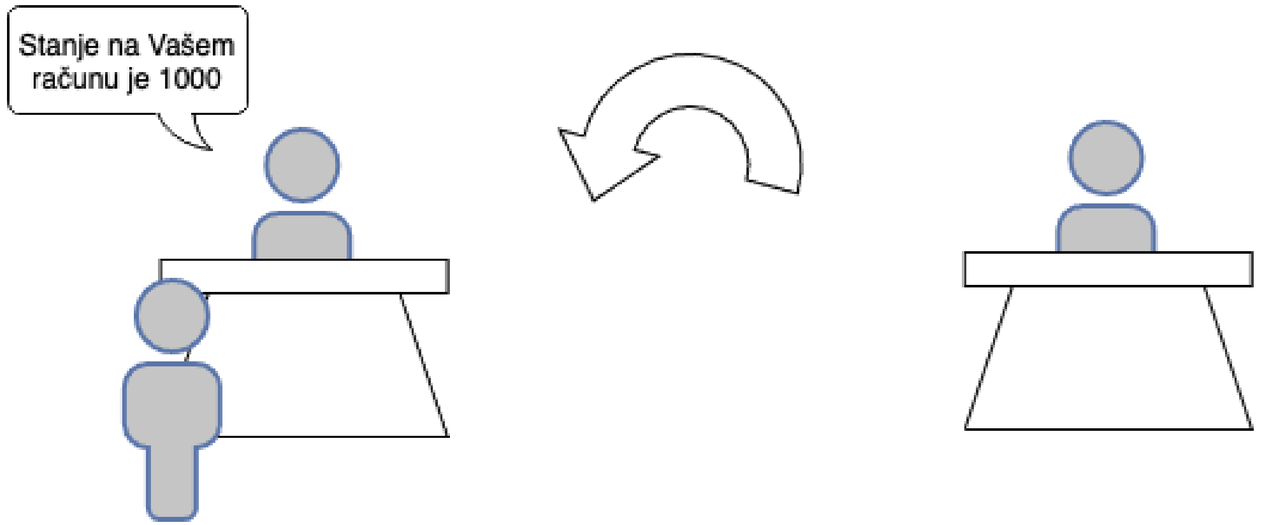
- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**




```
SELECT stanje FROM racuni
WHERE br_racuna = 1234567;
```

TRANSAKCIJE NA APLIKATIVNOM NIVOU

- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**

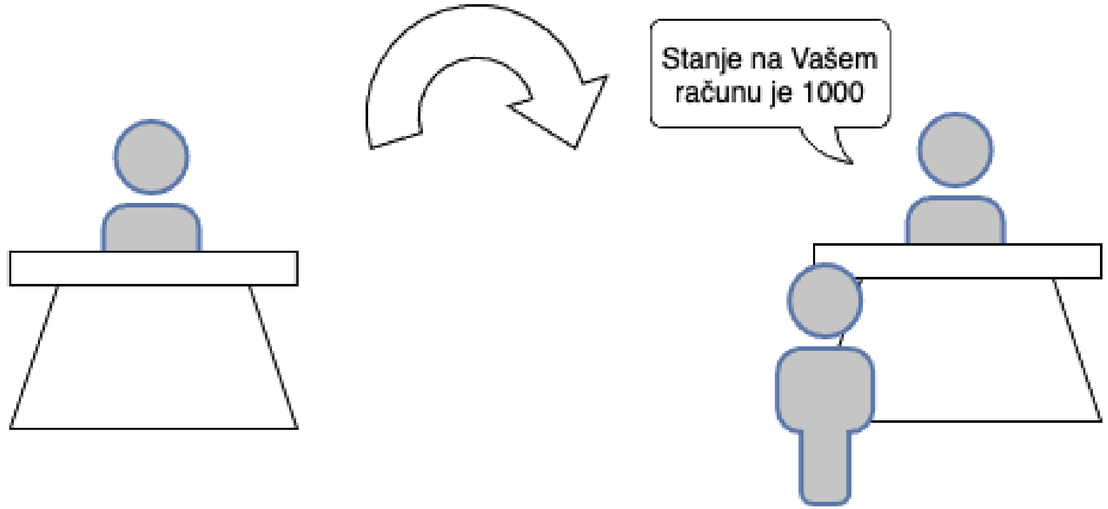



```
SELECT stanje FROM racuni
WHERE br_racuna = 1234567;
```

1000

TRANSAKCIJE NA APLIKATIVNOM NIVOU

- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**

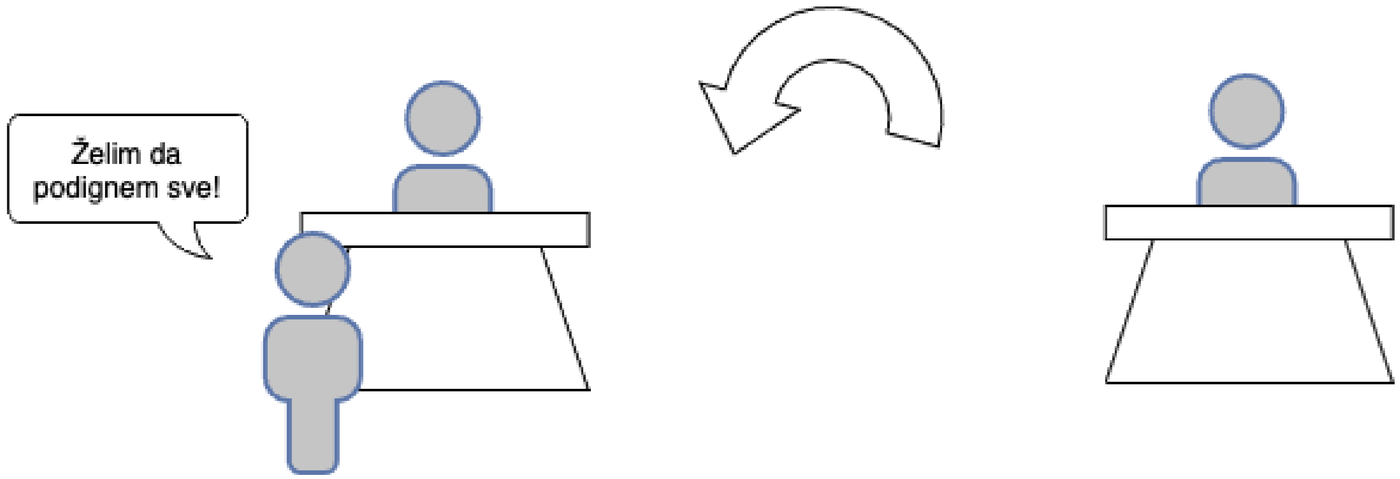



```
SELECT stanje FROM racuni
WHERE br_racuna = 1234567;
```

1000

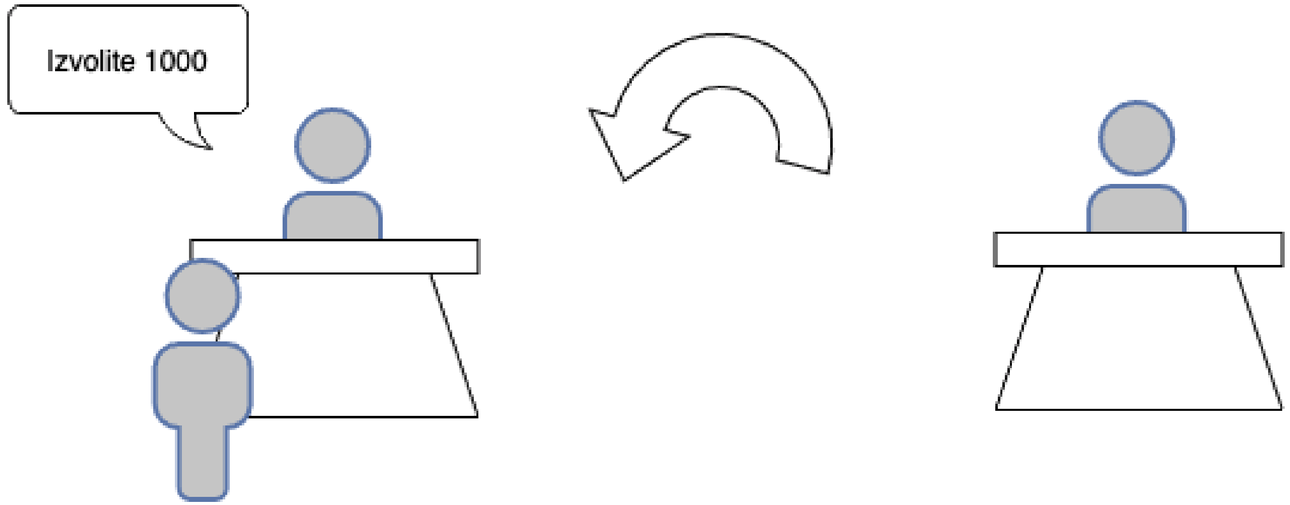
TRANSAKCIJE NA APLIKATIVNOM NIVOU

- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**



TRANSAKCIJE NA APLIKATIVNOM NIVOU

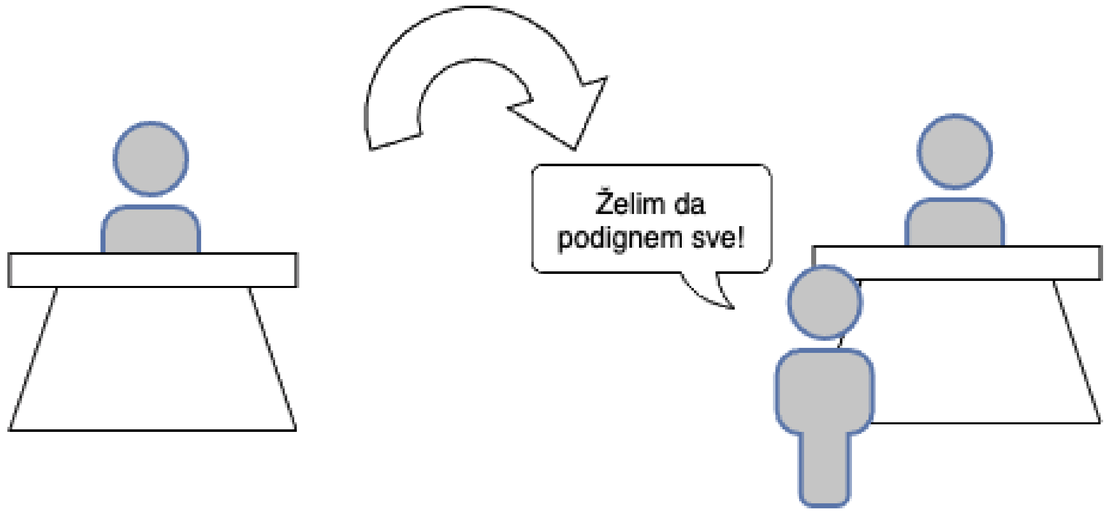
DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...




```
UPDATE racuni SET
stanje=stanje-1000 WHERE
   br_racuna=1234567;
      COMMIT;
```

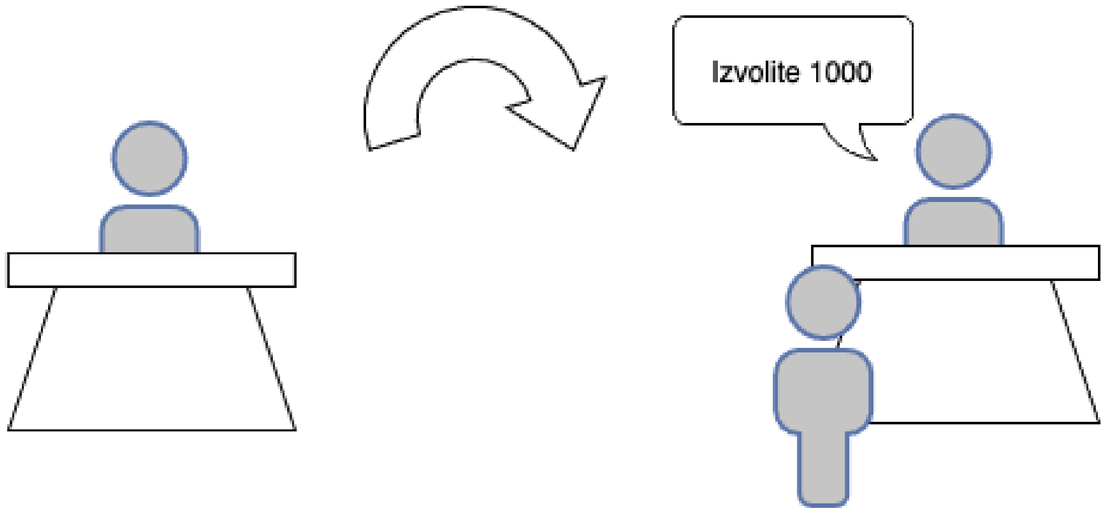
TRANSAKCIJE NA APLIKATIVNOM NIVOU

- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**



TRANSAKCIJE NA APLIKATIVNOM NIVOU

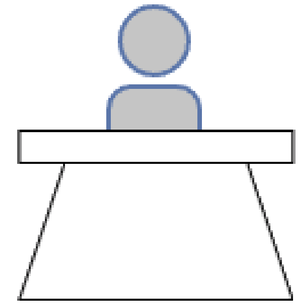
- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**



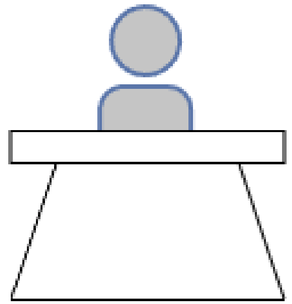

```
UPDATE racuni SET
stanje=stanje-1000 WHERE
br_racuna=1234567;
COMMIT;
```

TRANSAKCIJE NA APLIKATIVNOM NIVOU

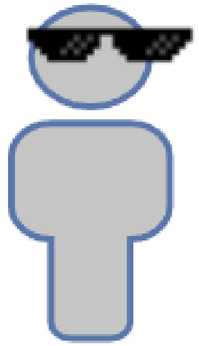
- **DVA ISTOVREMENO DOSTUPNA ŠALTERA U BANCI, DOLAZI JEDAN KLIJENT...**



Prethodno stanje: 1000
Novo stanje: 0



Prethodno stanje: 1000
Novo stanje: 0



Klijent izašao sa 2000

TRANSAKCIJE NA APLIKATIVNOM NIVOU

DVA PRISTUPA ZAKLJUČAVANJU RESURSA NA APLIKATIVNOM NIVOU

- Nivoi izolacije implicitno zaključavaju resurse na nivou baze da se obezbedi konzistentnost čitanja
- Na aplikativnom nivou moraju se eksplicitno zaključavati resursi
    - Pesimističko zaključavanje
    - “Optimističko” zaključavanje

PESIMISTIČKO ZAKLJUČAVANJE

KAKO SE OSTVARUJE?

- Baze nude naredbe za eksplicitno zaključavanje resursa
- Dve vrste *lock*-a postoje
    - Shared lock – dozvoljene su operacije čitanja svima, operacije pisanja nisu dozvoljene ostalim transakcijama (obično klauzula na kraju upita FOR SHARE)
    - Exclusive lock – nisu dozvoljene ni operacije čitanja ni pisanja ostalim transakcijama (pogodne za poslednju transakciju u dugoj konverzaciji, obično klauzula na kraju uputa FOR UPDATE)

OPTIMISTIČKO ZAKLJUČAVANJE

KAKO SE OSTVARUJE?

- Zapravo i nije zaključavanje u pravom smislu te reči
- Ideja je da se doda u tabelu kolona koja će predstavljati poslednju verziju torke (timestamp ili increment integer)
- Pri svakoj izmeni torke, ažurira se verzija
- Samo jedna transakcija u jednom trenutku može da menja jednu torku

KAKO FUNKCIONIŠE?

- Kada korisnik pročita torku dobiće i verziju iste
- Pri ažuriranju torke će se porediti i verzija, npr.

```
UPDATE monitor SET (kolicina, version) = (20, 2) WHERE monitor_id = 1 AND version = 1
```

- Ako naredba vrati 0, verzija je u međuvremenu ažurirana, a to znači da je cela torka ažurirana i klijent koji je pokušao ažuriranje barata nevalidnim podacima

REFERENCE

- PRIMERI I SLAJDOVI PO UZORU NA https://github.com/mbranko/isa19/tree/master/08-tx
- GRAY J. THE TRANSACTION CONCEPT: VIRTUES AND LIMITATIONS. http://jimgray.azurewebsites.net/papers/thetransactionconcept.pdf
- DATABASE LANGUAGE SQL 92. http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt
- BERNSON ET AL. A CRITIQUE OF ANSI SQL ISOLATION LEVELS. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-95-51.pdf
- POSTGRESQL EXPLICIT LOCKING. https://www.postgresql.org/docs/current/explicit-locking.html
- MIHALCEA V. HIGH-PERFORMANCE JAVA PERSISTENCE. https://vladmihalcea.com/books/high-performance-java-persistence/
- SILBERSCHATZ A, KORTH H, SUDARSHAN S. DATABASE SYSTEM CONCEPTS. https://db-book.com/

KOJA SU VAŠA PITANJA?