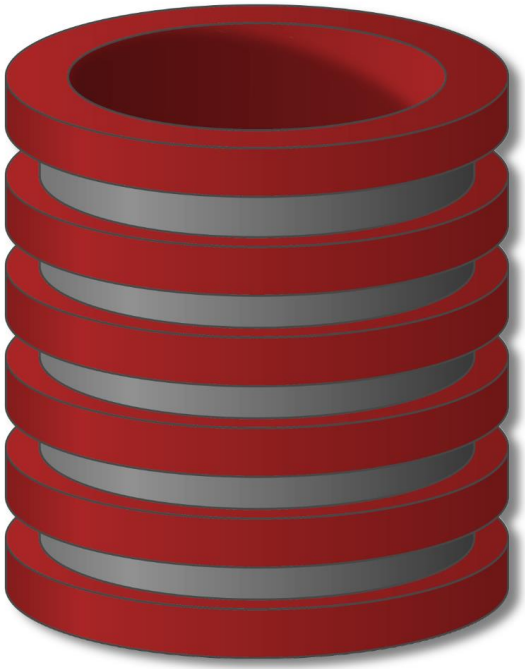


Transakcije

Predmet: Administriranje Baze Podataka

Predavač: dr Dušan Stefanović

TRANSAKCIJE - POJAM

- Transakcije su vrlo bitne u svetu baza podataka
- Transakcija u realnom svetu
  - Ukoliko prodavcu knjiga platimo, očekujemo da dobijemo knjigu.
  - Transakcija je ukoliko su se obe stvari desile
  - Ukoliko smo dali novac očekujemo knigu, ukoliko prodam knigu očekujem novac za nju
- U svetu računara primer transakcije je bankarski sistem
  - Ulogujemo se na naš račun preko Weba i želimo da prebacimo novac sa jednog računa na drugi.
  - Ova akcija zahteva dve promene nad podacima, sa jednog računa oduzimamo novac a na drugom računu dodajemo novac.
  - Transakcija je uspešna ukoliko su se obe stvari izvršene, ukoliko jedna stvar ne uspe, sistem vraća podatke na početno stanje

TRANSAKCIJE - POJAM

- Kada želimo da prodamo knjigu nekoliko stvari je potrebno da se ispuni
  - Proveriti da li je knjiga na stanju
  - Kreirati porudžbinu
  - Dodati proizvod u tabelu za porudžbine
  - Ažurirati tabelu za proizvodima sa novim stanjem
- Šta će se desiti ukoliko u isto vreme dve osobe žele da kupe istu knjigu?
- Transakcije se koriste da grupišu više povezanih iskaza tako da se svi iskazi izvršavaju ili se ni jedan ne izvršava (nema sredine).

TRANSAKCIJE - POJAM

- Kolekcija upita
- Predstavlja jedinicu posla
- Npr. Prebacivanje novca sa jednog računa na drugi račun (SELECT, UPDATE, UPDATE)

BEGIN T1

```
SELECT IZNOS FROM ACCOUNT WHERE RACUN_ID = 1

IZNOS > 100

UPDATE ACCOUNT SET IZNOS = IZNOS - 100 WHERE RACUN_ID = 1

UPDATE ACCOUNT SET IZNOS = IZNOS + 100 WHERE RACUN_ID = 2
```

COMMIT T1

RACUN_ID	IZNOS
1	$900
2	$600

TRANSAKCIJE - OSOBINE

- Termin koji se vrlo često koristi u radu sa transakcijama je **ACID** i ugrađen je DBMS
  - Atomic
    - Zahteva da se sve akcije u transakciji izvrše ili sistem vraća na originalno stanje.
    - Razlog zašto se sve akcije u transakciji ne izvrše je nestanak električne energije ili nedovoljno prostora ili ...
    - Atomic znači sve ili ništa
  - Consistent
    - Transakcijom baza podataka iz jednog validnog stanja prelazi u drugo na osnovu pravila u bazi
  - Isolated
    - Odnosi se na to da podatak koji je uključen u transakciji bude zaključan za vreme trajanja transakcije, tj. nesme da se dozvoli drugom sistemu da menja isti podatak
  - Durable
    - Garantuje izvršenje transakcije čak i ukoliko se desi neki otkaz

TRANSAKCIJE - OSOBINE

Da bi transakcija bila uspešna sledeće osobine moraju da se ispune

Atomičnost (Atomicity)

- Transakcija ne može da se izvrši parcijalno.
- Transakcija je uspešna ili neuspešna samo kao celina
- Princip sve ili ništa
- Svi upiti u transakciji moraju da se izvrše u suprotnom se ni jedan ne izvršava
- Sistem se vraća na stanje pre izvršenja transakcije (rollback)

OSOBINE TRANSAKCIJE – ATOMIČNOST

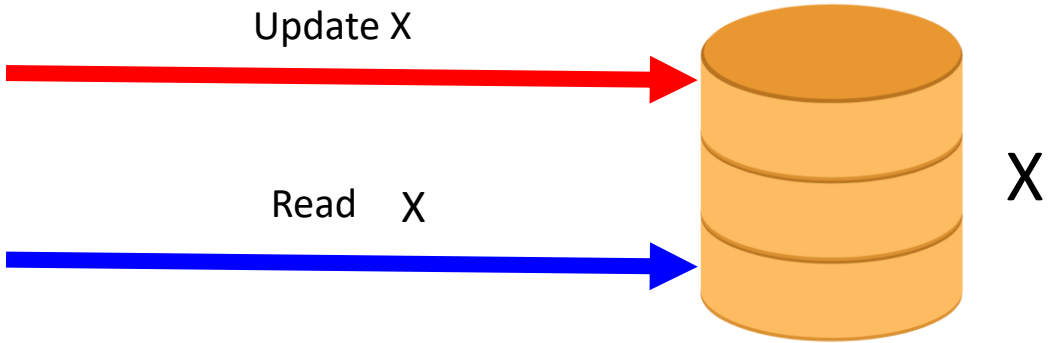
OSOBINE TRANSAKCIJE – ATOMIČNOST

RACUN_ID	IZNOS
1	$900
2	$500

- Nakon oporavka računara, sa računa je umanjen iznos ali na drugom računu nije dodata razlika
- Atomična transakcija vratiće se na stanje pre izvršenja prvog upita u transakciji (**rollback**)

OSOBINE TRANSAKCIJE – KONZISTENTNOST

- **Konzistentnost u podacima (Consistency in data)**
  - Definiše korisnik kreiranjem šeme i na osnovu:
    - Referencijalni integritet (strani ključ)
    - Izolacija
    - Atomičnost
- **Konzistetnost u čitanju (Consistency in read)**



Lajkovi

ID **(PK)**	BLOB	LIKES
1	xx	2
2	xx	1

Fotografije

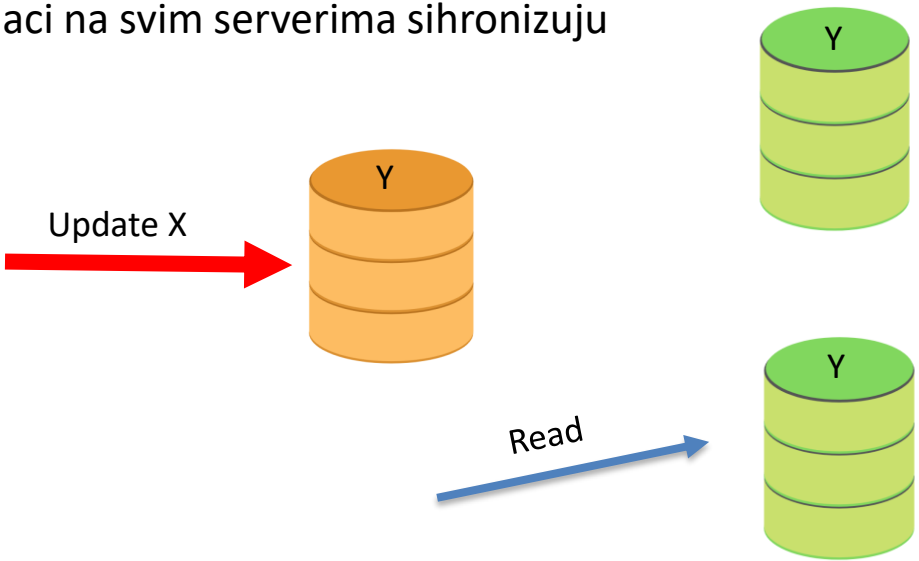
USER **(PK)**	PICTURE_ID **(PK)(FK)**
Jon	1
Edmond	1
Jon	2

KONZISTETNOST U ČITANJU

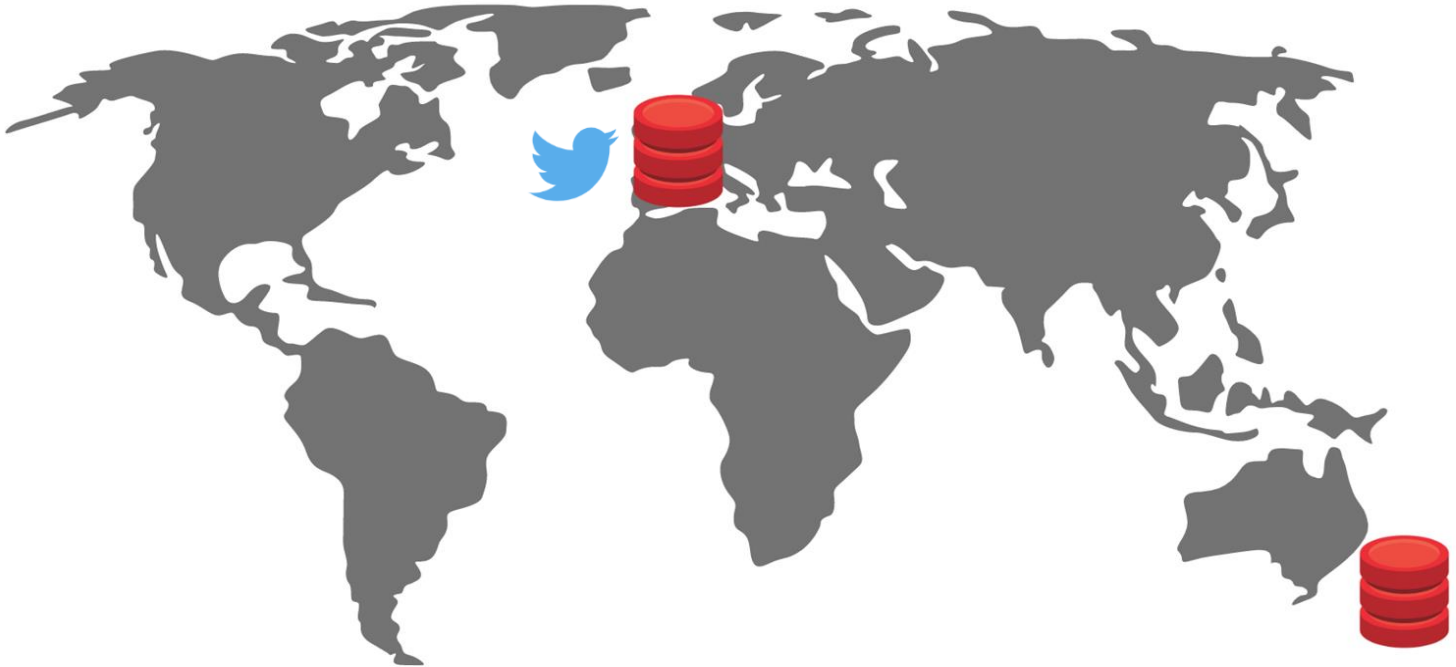
- Ukoliko transakcija potvrdi (commit) update da li će nova transakcija odmah da vidi tu promenu
  - Relacione i No Sql baze podatake mogu da imaju problem sa konzistetnošću u čitanju podataka
    - Horizontalna skalabilnost – potrebno je podatak replicirati na više servera u klasteru a za to je potrebno vreme, neko vreme podaci na serverima nisu konzistetni
  - Eventualna konzistentnost
    - Na račun konzistetnosti su poboljšane performasne i skalabilnost
    - Karakteristika NoSql

EVENTUALNA KONZISTETNOST

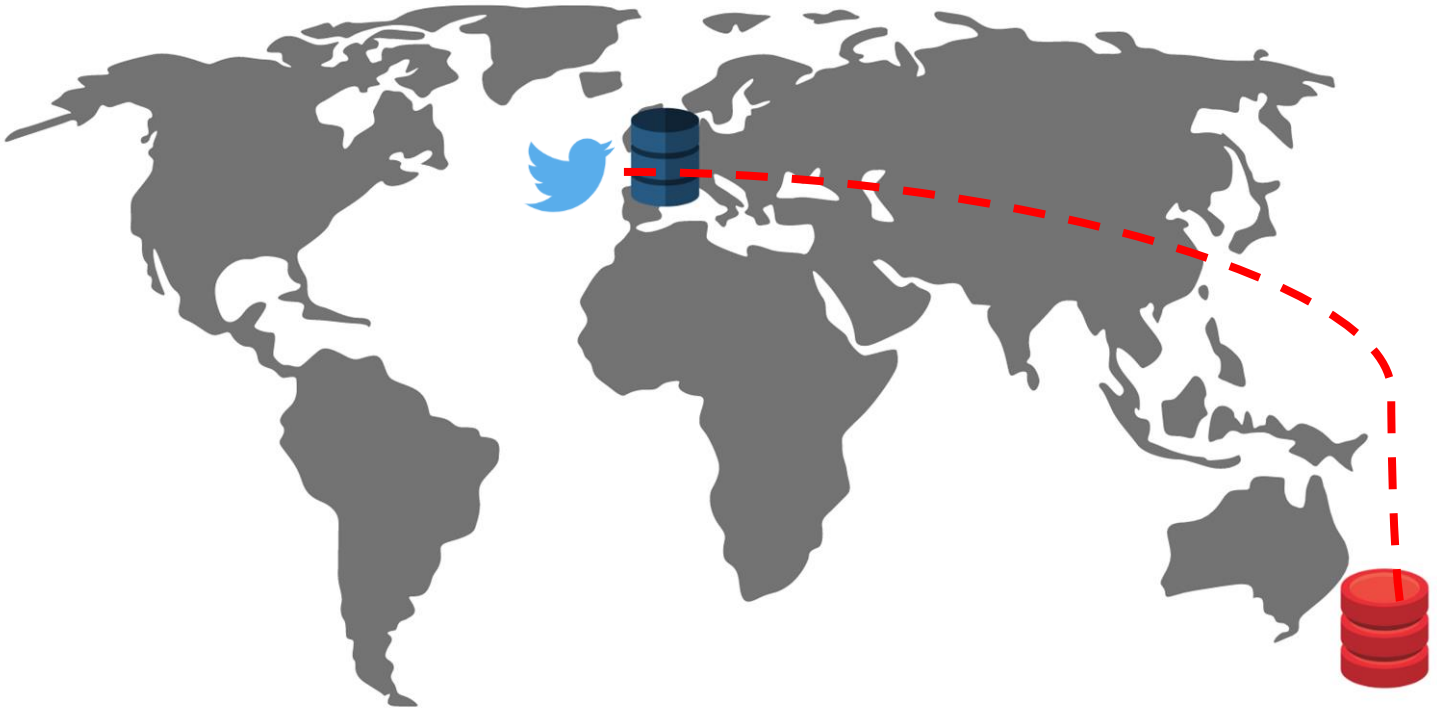
- U početku imamo konzistetnu vrednost y u sve tri baze
- Naranđasta baza je Master baza koja prima promene i propagira ih do svojih replika
- Promena u master bazi se ne propagira odmah tako da može da se desi nekonzistetnost u podacima ali istovremeno da je zadovoljena eventualna konzistetnost jer će u jednom trenutku da se podaci na svim serverima sihronizuju



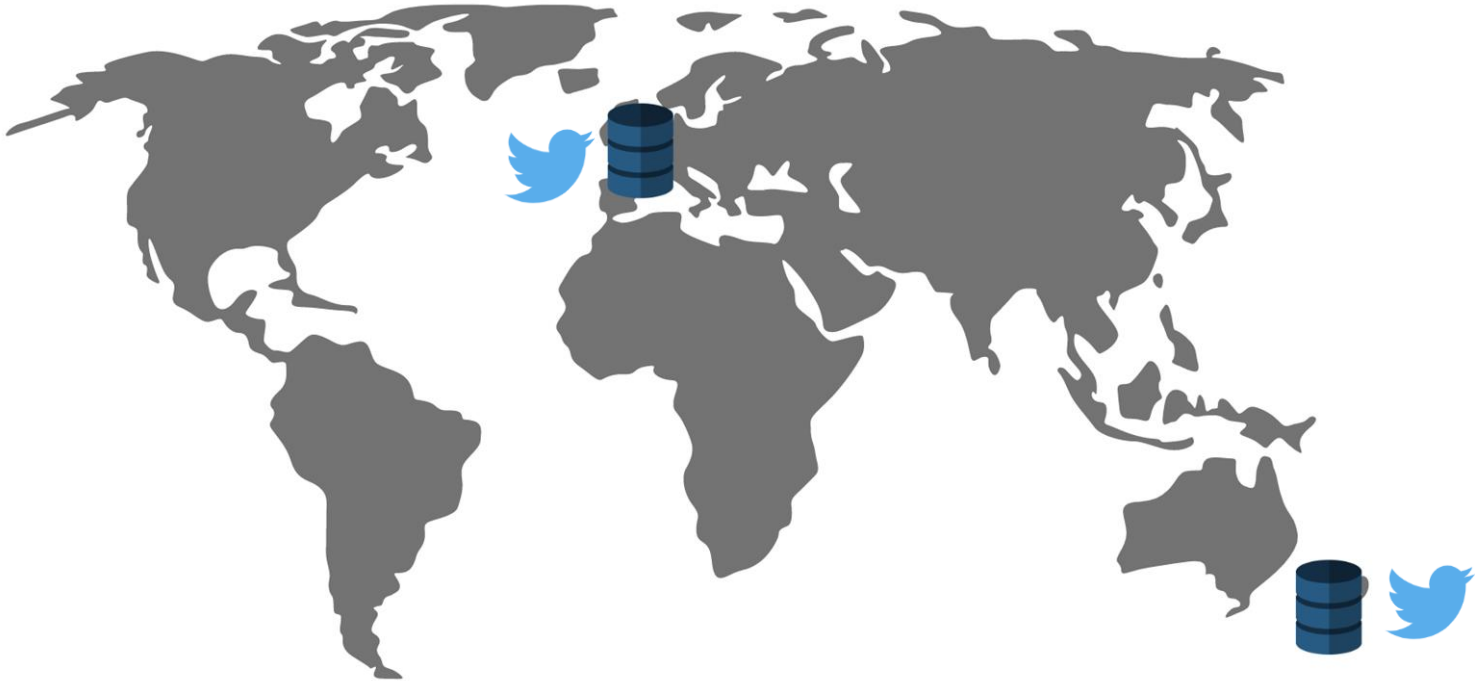
EVENTUALNA KONZISTETNOST



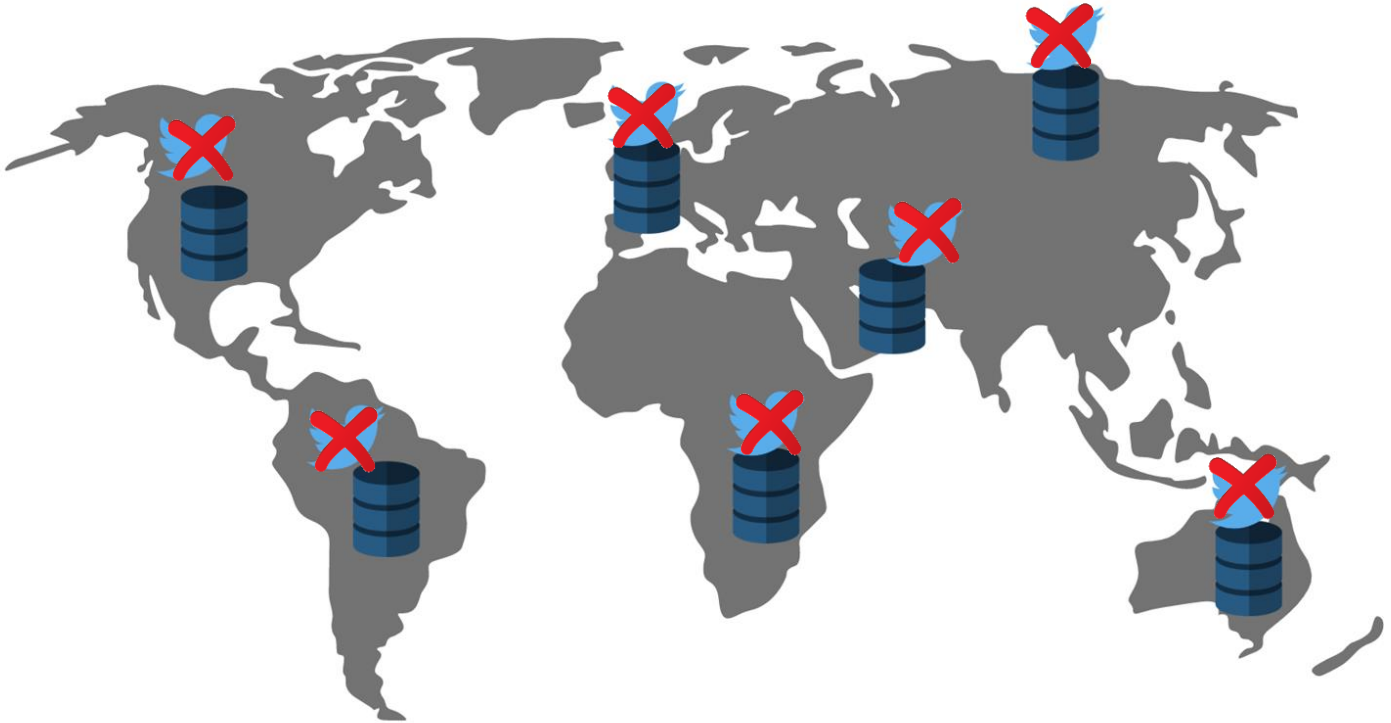
EVENTUALNA KONZISTETNOST



EVENTUALNA KONZISTETNOST



EVENTUALNA KONZISTETNOST



IZDRŽLJIVOST (DURABILITY)

- Transakcija koja je potvrđena (committed) mora da bude obrađena i u slučaju hardverskih otkaza u sistemu
  - Redis baza podataka nije *durable,* već in memory baza podataka
  - Uvek kada se koristi keš sto je slučaj sa *in memory* bazama podataka nismo konzistetni

TRANSAKCIJE - IZOLACIJA

Izolacija (Isolation)

- Transakcije su nezavisne od spoljnjeg sveta
- Odnosi se na to da li jedna transakcija može da vidi promene koje pravi druga transakcija nad istim setom podataka u bazi
- Javlja se zbog konkuretnog pristupa istim podacima
- **Problem** (fenomen) čitanja podataka - nedostatak adekvatne izolacije
- **Rešenje** - Izolacioni nivoi

TRANSAKCIJE – IZOLACIJA

ID	IME	SALDO	...
1	ZAJEDNIČKI	$10000	...
2	ANA	$50	
3	VUK	$45	
...	...	...	

- Tri računa
  - Zajednički - zajednički račun
  - Ana i Vuk - odvojeni računi
- Posmatraju se koraci u transferu novca između Zajedničkog računa i Aninog računa

Prikaži saldo Zajedničkog računa ($10000)

Prikaži saldo za Anin račun ($50)

Promeni saldo na Zajedničkom računu ($10000)- $1000

Promeni saldo na Aninom računu ($50) + $1000

TRANSAKCIJE – IZOLACIJA

Javiće se konflikt jer dva programa istovremeno rade nad istim podatkom, na Joint računu treba da bude $8000

ID	IME	SALDO	
1	ZAJEDNIČKI	$9000	...
2	ANA	$1050	
3	VUK	$1045	
...	...	...	

Slučaj kada Ana i Vuk istovremeno rade operacije sa svog i Zajedničkog računa.

Vuk veruje da na Zajedničkom računu i dalje ima $1000 jer je u prvoj iteraciji prilikom prikazivanja dobijo taj podatak

Ana

Prikaži saldo Zajedničkog računa ($10000)

Prikaži saldo za Anin račun ($50)

Promeni saldo na Zajedničkom računu ($10000)- $1000

Promeni saldo na Aninom računu ($50) + $1000

Vuk

Prikaži saldo Zajedničkog računa ($10000)

Prikaži saldo na Vukovom računu ($45)

Promeni saldo na Zajedničkom računu ($10000)- $1000

Promeni saldo na Vukovom računu ($45) + $1000

TRANSAKCIJA – REŠENJE PROBLEMA

- Potrebno je nekoliko koraka učiniti jedinstvenim tj grupisati u jedan blok koji je nedeljiv praveći transakciju.
- Ukoliko se u toku transakcije javi problem(nestanak el. energije) sistem se vraća na stanje pre transakcije.
- Transakcije i dalje ne rešavaju *race condition* problem tj. transakcije koje se izvršavau u isto vreme.
    - Slučaj kada je Vuk u fazi čitanja podataka a Ana u fazi izmene podataka (*dirty read*)

Ana
BEGIN TRANSACTION
Prikaži saldo Zajedničkog računa ($10000)
Prikaži saldo Aninog računa ($50)
Promeni saldo na Zajedničkom računu ($10000)- $1000
Promeni saldo na Aninom računu ($50) + $1000
COMMIT

Vuk
BEGIN TRANSACTION
Prikaži saldo Zajedničkog računa ($10000)
Prikaži saldo Vukovog računa ($45)
Promeni saldo na Zajedničkom računu ($10000)- $1000
Promeni saldo na Vukovom računu ($45) + $1000
COMMIT

TRANSAKCIJA – PESIMISTIČNO ZAKLJUČAVANJE

- Transakcija nije dovoljna za rešenje problema dirty read, već je potrebno obezbediti i *zaključavanje* od istovremene promene istog podatka.
- Ideja je da čim transakcija startuje podatak sa kojim transakcija radi se zaključava sve dok se ne završi transakcija (commit).

ID	IME	SALDO	...
1	ZAJEDNIČKI	$8000	...
2	ANA	$1050	
3	VUK	$1045	
...	...	...	

Automatski se zaključava Zajednički račun

Vuk čeka dok se Zajednički račun(ćelija) ne otključa, dok se transakcija sa leve strane ne završi koja traje 10 deo sekunde

Ana

BEGIN TRANSACTION
Prikaži saldo Zajedničkog računa ($10000)
Prikaži saldo Aninog računa ($50)
Promeni saldo na Zajedničkom računu ($10000)- $1000
Promeni saldo na Aninom računu ($50) + $1000
COMMIT

Vuk

BEGIN TRANSACTION
Prikaži saldo Zajedničkog računa ($10000)
Prikaži saldo Vukovog računa ($45)
Promeni saldo na Zajedničkom računu ($10000)- $1000
Promeni saldo na Vukovom računu ($45) + $1000
COMMIT

TRANSAKCIJA – OPTIMISTIČNO ZAKLJUČAVANJE

- Pesimistično zaključavanje u našem slučaju je dobro za Anu ali ne i za Vuka koji je morao da čeka
- Ovakav način zaključavanja može dovesti do toga da veliki broj korisnika treba da čeka ili do pojave greške.
- Međutim sama transakcija ne mora da znači da će doći do menjanja podataka već samo do čitanja.
- Optimistično zaključavanje dozvoljava da se više transakcija izvršavaju istovremeno sa ciljem da neće doći do konflikta

Ana

BEGIN TRANSACTION

Prikaži saldo na Zajedničkom računu ($10000)

Prikaži saldo na Aninom računu ($50)

Promeni saldo na na Zajedničkom računu ($10000)- $1000

Promeni saldo na Aninom računu ($50) + $1000

COMMIT

DBMS je otkrio konflikt sa drugom transakcijom, vraća se na početak transakcije

Vuk

BEGIN TRANSACTION

Prikaži saldo na Zajedničkom računu ($10000)

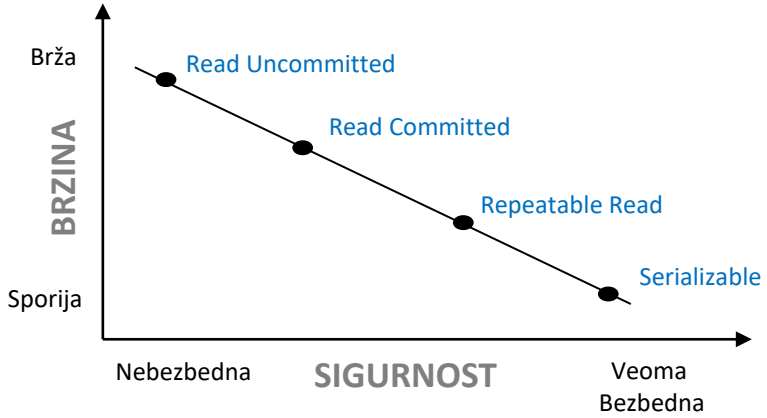
Prikaži saldo na Vukovom računu ($45)

Promeni saldo na Zajedničkom računu ($10000)- $1000

Greška – *dirty read* je otkrivena - **Rollback**

TRANSAKCIJE – IZOLACIONI NIVOI

- Četri primarna izolaciona nivoa koja su podržana od strane DBMS.



.

Izolacioni Nivoi	Dirty Read	Lost Update	Nonrepeatable Reads	Phantom Reads
Read Uncommitted	Da	Da	Da	Da
Read Committed	Ne	Da	Da	Da
Repeatable Read	Ne	Ne	Ne	Da
Snapshot	Ne	Ne	Ne	Ne
Serializable	Ne	Ne	Ne	Ne

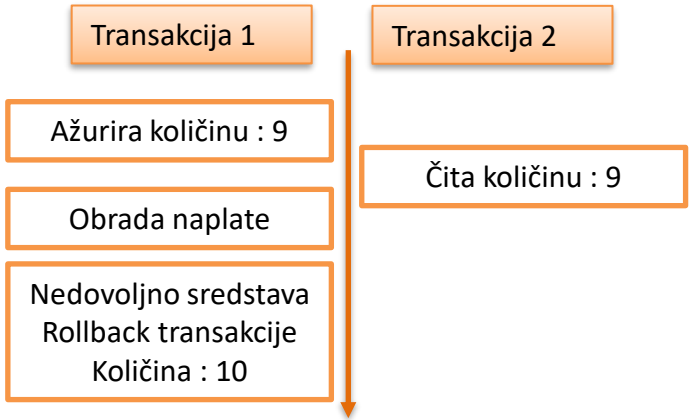
J

IZOLACIONI NIVOI – READ UNCOMMITTED

- Četri primarna izolaciona nivoa koja su podržana od strane DBMS.
  - Read uncommited
    - Najbrži ali najmanje bezbedan nivo.
    - Ne garantuje se nezavisnost transakcija u odnosu na druge transakcije
    - Dozvoljeno je tzv. *dirty read* a to je *situacija kada baza prosledi podatak korisniku koji još nije potvrđen(commited) na disku*
    - *Ovaj izolacioni level se koristi retko u situacijama kada ne brinemo o ispravnosti podatka.*

Tabele Inventar

Id	Proizvod	Količina
1	iPhone	10




```
--Transaction 1 :
Begin Tran
Update tblInventory set ItemsInStock = 9 where Id=1

-- Billing the customer
Waitfor Delay '00:00:15'
-- Insufficient Funds. Rollback transaction
Rollback Transaction
```

```
--Transaction 2 :
Set Transaction Isolation Level Read Uncommitted
Select * from tblInventory where Id=1
```

ili

```
-- Transaction 2
Set transaction isolation level read committed
Select * from tblInventory (NOLOCK) Where Id=1
```

IZOLACIONI NIVOI – DIRTY READ

PRODAJA

PID	Kolicina	Cena
Proizvod 1	10	$5
Proizvod 2	20	$4

BEGIN T1

SELECT PID, Kolicina*Cena **FROM** PRODAJA

Proizvod 1, 50
Proizvod 2, 80

SELECT SUM(Kolicina*Cena) **FROM** PRODAJA

$155

COMMIT T1

BEGIN T2

UPDATE PRODAJA **SET** Kolicina= Kolicina+5
WHERE PID =1

ROLLBACK T2

IZOLACIONI NIVOI – READ COMMITTED

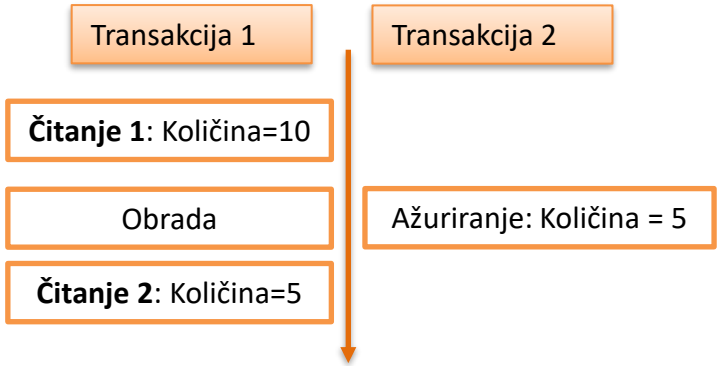
Četri primarna izolaciona nivoa koja su podržana od strane DBMS.

- Read commited
  - Bezbedniji od read uncommited izolacionog nivoa i dafault je u SQL DBMS
  - Ne dozvoljava pojavu dirty read podatka
  - Transakcija će pročitati podatak samo ako je podatak *commited* tj. druga transakcija će pristupiti podatku tek kada podatak bude commitovan a to znači dok se ne završi prva transakcija.
  - **Problem - non repeatable read:** Ukoliko transakcija čita isti podatak dva puta a druga transakcija ažurira taj podatak između dva čitanja rezlutat neće biti isti između dva čitanja u prvoj transakciji

```
--Transaction 1 :
Begin Tran
Update tblInventory set ItemsInStock = 9 where Id=1

-- Billing the customer
Waitfor Delay '00:00:15'
-- Insufficient Funds. Rollback transaction
Rollback Transaction
```

```
-- Transaction 2
Select * from tblInventory Where Id=1
```




```
-- Transaction 1
Begin Transaction
Select ItemsInStock
from tblInventory where Id = 1

-- Do Some work
waitfor delay '00:00:10'

Select ItemsInStock
from tblInventory where Id = 1
Commit Transaction
```

```
-- Transaction 2
Update tblInventory
set ItemsInStock = 5 where Id = 1
```

IZOLACIJA- NON REPEATABLE READ

PRODAJA

PID	Kolicina	Cena
Proizvod 1	**15**	$5
Proizvod 2	20	$4

BEGIN T1

SELECT PID, Kolicina*Cena **FROM** PRODAJA

Proizvod 1, 50
Proizvod 2, 80

BEGIN T2

UPDATE PRODAJA **SET** Kolicina = Kolicina+5
WHERE PID =1

COMMIT T2

SELECT SUM(Kolicina*Cena) **FROM** PRODAJA

$155

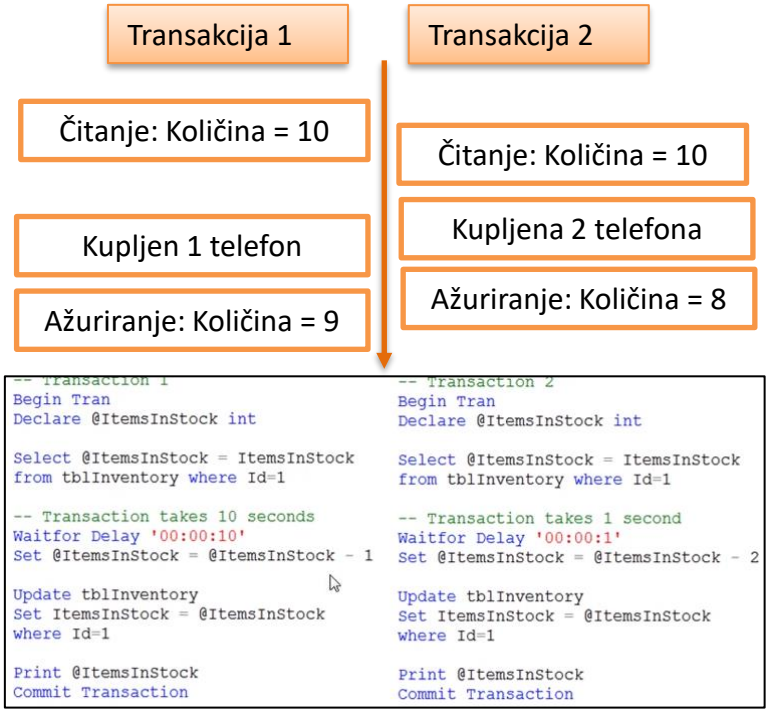
COMMIT T1

IZOLACIONI NIVOI – LOST UPDATE

- Lost update problem
  - Javlja se kada dve transakcije u isto vreme čitaju i ažuriraju isti podatak
  - Na kraju obe transakcije, umesto da je za količinu upisan broj 7 biće upisan 9
  - Ovaj problem je prisutan kod izolacionih nivoa:
    - **read uncommited**
    - **read commited**

Tabele Inventar

Id	Proizvod	Količina
1	iPhone	10



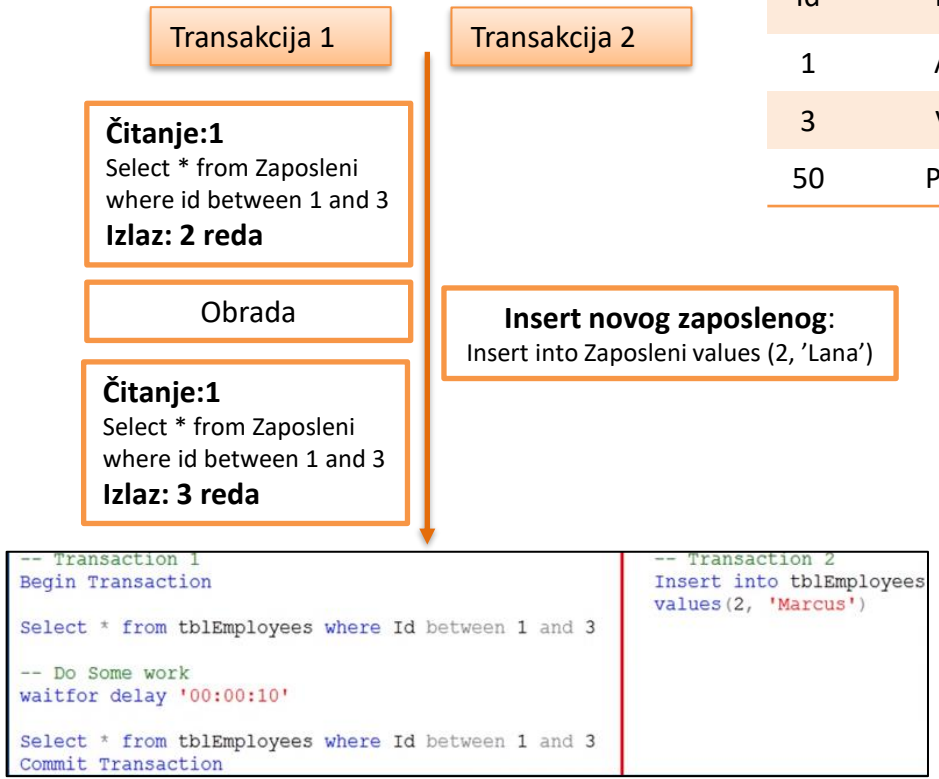
IZOLACIONI NIVOI – PHANTOM READ

Phantom reads problem

- Javlja se kada se jedan read upit izvrši dva puta u transakciji i dobije se različit broj redova
- To će se desiti ukolio druga transakcija upiše novi red koji odgovara uslovima pretrage prve transakcije
- Ovaj problem je prisutan kod izolacionih nivoa:
  - read uncommited
  - read commited
  - repeatable read

Tabela Zaposleni

Id	Ime
1	Ana
3	Vuk
50	Petar



IZOLACIJA- PHANTOM READ

SALES

PID	Kolicina	Cena
Proizvod 1	10	$5
Proizvod 2	20	$4
Proizvod 3	10	$1

BEGIN T1

SELECT PID, Kolicina*Cena **FROM** PRODAJA

Proizvod 1, 50
Proizvod 2, 80

SELECT SUM(QNT*PRICE) **FROM** SALES

$140

COMMIT T1

BEGIN T2

INSERT INTO SALES
VALUES (‘Product 3’, 10, 1)

COMMIT T2

TRANSAKCIJE - REPEATABLE READ VS SERIALIZABLE

- Repeatable read (default u MySql)
  - Izolacioni nivo obezbeđuje da podatak koji čita jedna transakcija sprečava brisanja ili ažuriranja od strane druge transakcije
  - Ne sprečava da novi red bude ubačen od strane druge transakcije što dovodi do tzv phantom read concurrency problem
- Serializable
  - Izolacioni nivo najveće bezbednosti, obezbeđuje da podatak koji čita jedna transakcija sprečava brisanja ili ažuriranja od strane druge transakcije
  - Sprečava i ubacivanje novog reda od strane druge transakcije dok se završi prva transakcija
  - U većini slučajeva ovaj izolacioni nivo će usporiti aplikaciju jer da bi se izvršila naredna transakcija mora da se sačeka prva čak i kad se radi čitanje tj. Read baze podataka.

Izolacija– izolacioni nivoi

- **Read uncommitted**
  - Nema Izolacije (Svi navedeni problemi mogu da se jave)
  - Bilo koja promena od spolja je vidljiva u transakciji i u slučaju da se poništi transakcija
- **Read committed**
  - Svaki upit u transakciji vidi samo potvrđene (commited) stvari
  - U većini slučajeva
- **Repeatable Read**
  - Svaki upit u transakciji vidi samo potvrđene (commited) podatke na početku transakcije (verzionisanje)
- **Serializable**
  - Transakcije su serijalizovane, dok se ne obradi prva, druga po redu transakcija čeka

TRANSAKCIJE - NAPOMENE

- Sve relacione baze podataka podržavaju Atomičnost, Konzistetnost i Izdržljivost po automatizmu.
- Osobine koje su vezane za izolaciju zavise od korisnika i obično su podešene na bezbedan nivo ali mogu da se podignu ili smanje.
- MySql ne podržava uvek ACID
  - Zavisi od engin-a MySql-a koji se koristi i od načina na koji je podešena baza podataka
  - InnoDB engine podržava ACID
  - MyISAM engine ne podržava ACID