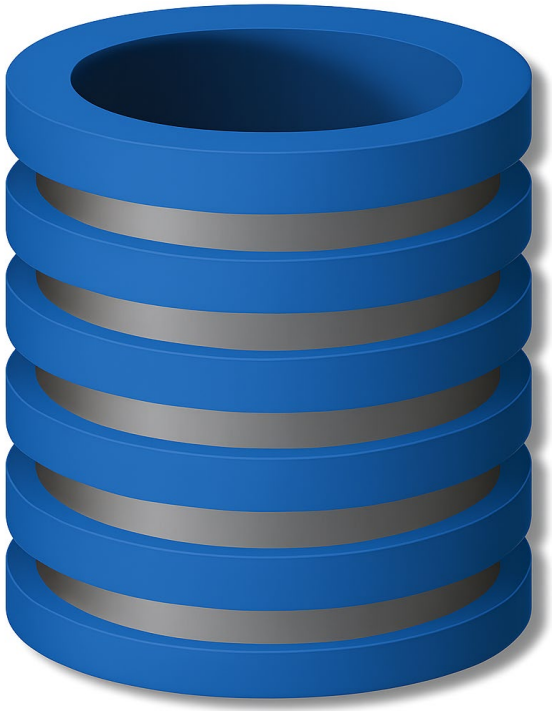


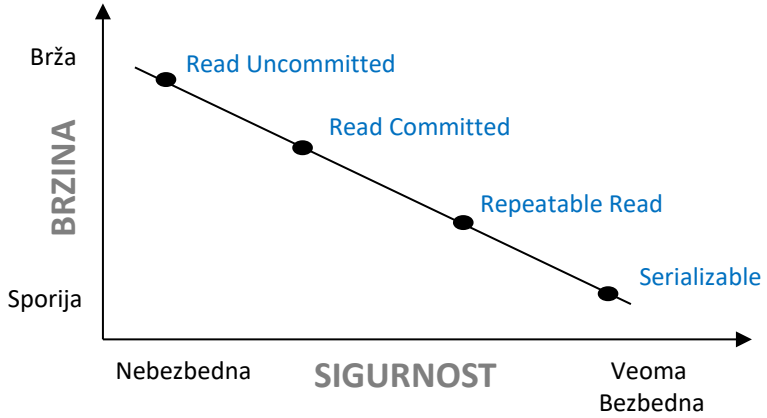
Transakcije II deo

Predmet: Administriranje Baze Podataka

Predavač: dr Dušan Stefanović

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

Nivo čitanja uncommitted read

Najbrži, ali najmanje siguran nivo izolacije

Nepotvrđeni podaci

Transakcije mogu čitati podatke koje druge transakcije nisu potvrdile

Dirty Read Problem

Kada se prosledi podatak korisniku koji još nije potvrđen(commited) na disku

Retka upotreba

Koristi se retko, najčešće samo za read-only analitiku gde je brzina bitnija od tačnosti.

Problem	Opis
Dirty Read	Transakcija čita podatke iz druge transakcije koja još nije potvrđena. Ako se rollback desi, čitani podaci su pogrešni.
Ne može se garantovati konzistentnost	Na primer, sistem za e-trgovinu može pokazati pogrešnu količinu zaliha.
Neprikladan za većinu aplikacija	Koristi se retko, najčešće samo za read-only analitiku gde je brzina bitnija od tačnosti.

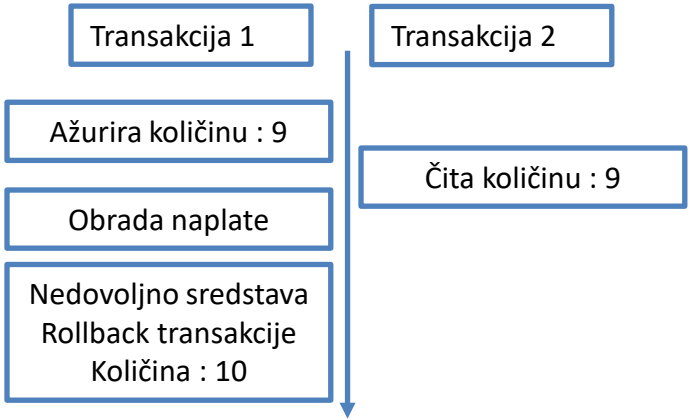
Kada koristiti Read Uncommitted	**Kada ne koristiti Read Uncommitted**
Radite sa velikim količinama podataka gde su performanse kritične	Donosite važne poslovne odluke na osnovu podataka
Možete tolerisati netačne/privremene podatke (npr. analitika u realnom vremenu)	Radite sa finansijskim ili inventarskim sistemima

PRIMER 1 - DIRTY READ

Korak	Transakcija 1	Transakcija 2	Napomena
1	Ažurira količinu proizvoda sa **10** na **9**		Promena je u toku, ali **nije potvrđena** (COMMIT)
2	Čeka potvrdu plaćanja (nema COMMIT)		Podaci su i dalje u **nepotvrđenom** stanju
3		Čita vrednost **9**	Čitanje "prljavih" podataka – **dirty read**
4	Plaćanje **neuspešno** → vrši se **ROLLBACK**		Vrednost se vraća na **10**
5		Transakcija 2 je koristila netačne podatke (vrednost 9)	Potencijalni problem za logiku aplikacije

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
```

PRIMER 2 - 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

Read Committed

Bezbedniji od read uncommited izolacionog nivoa i podrazumevan je u SQL DBMS

Nema Dirty Read

Transakcija će pročitati podatak samo ako je podatak commited tj. druga transakcija će pristupiti podatku tek kada podatak bude potvrđen a to znači dok se ne završi prva transakcija.

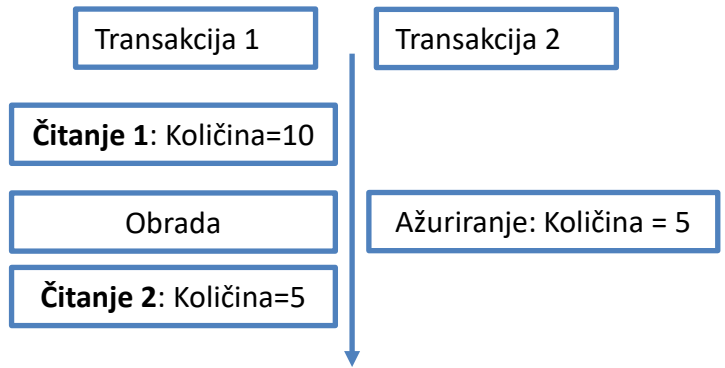
Repeatable Read Problem

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
```

KARAKTERISTIKE READ COMMITTED IZOLACIONOG NIVOA

Karakteristika	**Opis**
Izbegava dirty read	Da – podatak može biti pročitan **samo ako je commitovan**.
Non-repeatable read	Moguć – ako drugi commituje promenu između dva čitanja, vrednosti mogu biti različite.
Phantom read	Moguć – novi redovi koji zadovoljavaju uslov mogu se pojaviti pri ponovnom izvršavanju iste SELECT naredbe.
Učinak na performanse	Uravnotežen
Ponašanje pri čitanju podataka	Podaci se čitaju isključivo nakon što druga transakcija izvrši COMMIT.
Podrazumevani nivo u SQL DBMS	Da – u većini sistema (npr. SQL Server, PostgreSQL, Oracle).
Pogodno za	Većinu poslovnih aplikacija gde je **konzistentnost podataka važna**, ali nije potrebna stroga 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

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

Proizvod 1, 75
Proizvod 2, 80

COMMIT T1

BEGIN T2

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

COMMIT T2

REPEATABLE READ IZOLACIONI NIVO

Karakteristika	Opis
Izbegava dirty read	**Da** – transakcije **ne mogu čitati podatke** koje druge transakcije nisu commitovale.
Izbegava non-repeatable read	**Da** – transakcija **vidi iste vrednosti** prilikom svakog čitanja istog reda tokom trajanja transakcije.
Izbegava phantom read	**Ne** – moguće je da se **novi redovi pojave** u rezultatu istog upita ako drugi unesu podatke koji zadovoljavaju isti uslov.
Učinak na performanse	Veće zaključavanje – **više blokiranja** i čekanja jer redovi koji su pročitani ostaju zaključani do COMMIT-a.
Zaključavanje	**Shared locks** zadržavaju se na pročitanim redovima tokom cele transakcije – štiti od pisanja drugih transakcija.
Ponašanje pri čitanju podataka	Transakcija **uvek vidi istu verziju reda** – čak i ako druga transakcija pokuša da ga izmeni.
Pogodno za	Sisteme gde je **bitno da se vrednosti ne promene između više čitanja**, kao kod obračuna, fakturisanja, bankarstva.

Transakcija 1 čita podatak

Transakcija 1 očekuje da će podatak koji pročita ostati isti tokom trajanja transakcije

Transakcija 2 menja podatak

Transakcija 2 potvrđuje promenu

Transakcija 1 ponovo čita podatak

Repeatable Read sprečava promene

Nijedna druga transakcija ne može da menja red koji je već pročitan u okviru tekuće transakcije

PHANTOM READ

- Phantom read (fantomsko čitanje) se javlja kada transakcija dva puta izvrši isti upit, ali u drugom izvršavanju dobije dodatne redove koji nisu postojali u prvom rezultatu, jer ih je druga transakcija u međuvremenu unela (INSERT).
- Za razliku od non-repeatable read, koji se odnosi na promenu postojećih redova, phantom read se odnosi na pojavu novih redova.
- Ovaj problem je prisutan kod izolacionih nivoa:
  - read uncommited
  - read commited
  - repeatable read

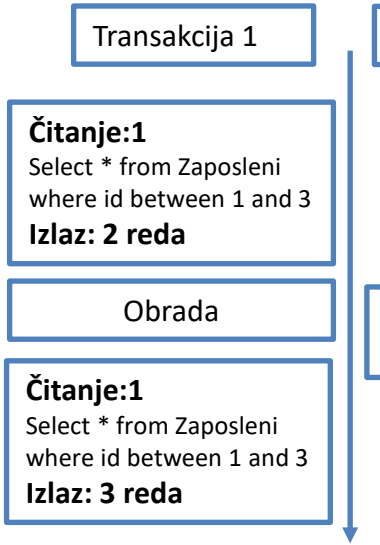


Tabela Zaposleni

Id	Ime
1	Ana
3	Vuk
50	Petar

```
-- Transaction 1
Begin Transaction

Select * from tblEmployees where Id between 1 and 3

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

Select * from tblEmployees where Id between 1 and 3
Commit Transaction
```

```
-- Transaction 2
Insert into tblEmployees
values(2, 'Marcus')
```

PHANTOM READ

	Transakcija 1 (T1)	Transakcija 2 (T2)	Rezultat / Napomena
1	SELECT * FROM Zaposleni WHERE Id BETWEEN 1 AND 3		Rezultat: 2 reda →**Ana (id=1), Vuk (Id=3)**
2	(Zadržava transakciju, npr. zbog obrade)	INSERT INTO Zaposleni VALUES (2, 'Lana')	T2 umeće novog zaposlenog koji **zadovoljava uslov upita T1**
3	SELECT * FROM Zaposleni WHERE Id BETWEEN 1 AND 3 (ponovljeni upit)		Rezultat: 3 reda → **Vuk (Id=3), Lana (Id=2), Ana (id=1)** – pojavio se **phantom red**

Tabela Zaposleni

Id	Ime
1	Ana
3	Vuk
50	Petar

Transakcija 1

Čitanje:1
Select * from Zaposleni where id between 1 and 3
Izlaz: 2 reda

Obrada

Čitanje:1
Select * from Zaposleni where id between 1 and 3
Izlaz: 3 reda

Transakcija 2

Insert novog zaposlenog:
Insert into Zaposleni values (2, 'Lana')

```
-- Transaction 1
Begin Transaction

Select * from tblEmployees where Id between 1 and 3

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

Select * from tblEmployees where Id between 1 and 3
Commit Transaction
```

```
-- Transaction 2
Insert into tblEmployees
values(2, 'Marcus')
```

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(Kolicina*Cena) **FROM** PRODAJA

$140

COMMIT T1

BEGIN T2

INSERT INTO PRODAJA
VALUES (‘Proizod 3’, 10, 1)

COMMIT T2

REŠENJE ZA PHANTOM READ

Problem:

Transakcija koja nije završena (COMMIT) u T1 iznenada vidi novi red koji nije postojao pri prvom čitanju.

To može uticati na obračune, validaciju, izveštavanje, jer podaci nisu konzistentni unutar iste transakcije.

Rešenje:

Serializable izolacioni nivo - Najviši nivo izolacije.

Osigurava da nema umetanja, ažuriranja ili brisanja redova koji bi mogli uticati na rezultate upita.

Efektivno zaključava opseg redova koji zadovoljavaju uslov (BETWEEN, LIKE, >, itd.).

Nedostatak: usporava sistem – više blokiranja i čekanja.

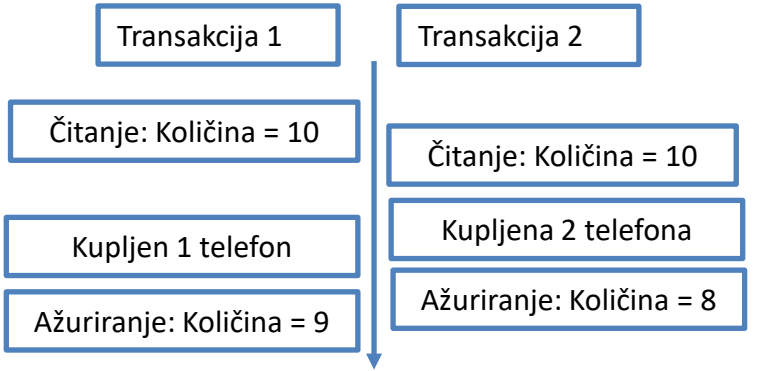
LOST UPDATE PROBLEM

Javlja se kada dve transakcije u isto vreme čitaju i ažuriraju isti podatak

	Transakcija 1 (T1)	Transakcija 2 (T2)	Napomena / Problem
1	Čita Količinu = 10	Čita Količinu = 10	Obe transakcije čitaju istu početnu vrednost
2	Kupuje 1 telefon → računa Količina = 9	Kupuje 2 telefona → računa Količina = 8	Obe nezavisno određuju novu količinu
3		Ažurira Količina = 8	T2 zapisuje rezultat prve
4	Ažurira Količina = 9		T1 **prepisuje rezultat T2** – izgubljena je informacija da su kupljena 2 telefona

Tabele Inventar

Id	Proizvod	Količina
1	iPhone	10




```
-- Transaction 1
Begin Tran
Declare @ItemsInStock int

Select @ItemsInStock = ItemsInStock
from tblInventory where Id=1

-- Transaction takes 10 seconds
Waitfor Delay '00:00:10'
Set @ItemsInStock = @ItemsInStock - 1

Update tblInventory
Set ItemsInStock = @ItemsInStock
where Id=1

Print @ItemsInStock
Commit Transaction
```

```
-- Transaction 2
Begin Tran
Declare @ItemsInStock int

Select @ItemsInStock = ItemsInStock
from tblInventory where Id=1

-- Transaction takes 1 second
Waitfor Delay '00:00:1'
Set @ItemsInStock = @ItemsInStock - 2

Update tblInventory
Set ItemsInStock = @ItemsInStock
where Id=1

Print @ItemsInStock
Commit Transaction
```

REŠENJE ZA LOST UPDATE

Izolacioni nivo	Da li rešava Lost Update?	Objašnjenje
Read Uncommitted	Ne	Dozvoljava čak i čitanje nepotvrđenih vrednosti – nesigurno
Read Committed	Ne	Ne garantuje da drugi neće u međuvremenu izmeniti podatak
Repeatable Read	Delimično	Štiti od izmene pročitanih redova, ali **ne štiti od istovremenog ažuriranja**
Serializable	Da	Sprečava sve paralelne izmene – garantuje da se transakcije izvršavaju sekvencijalno
Alternativa: Eksplicitno zaključavanje (SELECT ... FOR UPDATE)	Da	Blokira red koji je pročitan dok se transakcija ne završi

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