

5/7

## : Razina izolacije SERIALIZABLE

Promjena izolacijske razine transakcije sprječava pojavu sablasnih n-torki i prljavog čitanja.

Zadana (defaultna) izolacijska razina je `READ COMMITED`, no možemo zatražiti korištenje druge izolacijske razine uz pomoć naredbe `SET TRANSACTION ISOLATION LEVEL` ako to učinimo odmah na početku transakcije. Također, možemo lako dohvatiti naziv trenutne izolacijske razine uz pomoć naredbe `SHOW TRANSACTION ISOLATION LEVEL`.

Uočite da nakon završetka transakcije iduća transakcija opet preuzima zadanu (defaultnu) izolacijsku razinu.

	userA	userB
1		<code>BEGIN TRANSACTION;</code>
2		<code>SELECT txid_current() txId;</code>
3		<code>SHOW TRANSACTION ISOLATION LEVEL;;</code>
4	<code>BEGIN TRANSACTION;</code>	
5	<code>SELECT txid_current() txId;</code>	
6	<code>INSERT INTO predmet2 VALUES (68, 'NoSQL baze podataka ', 4.0, 4);</code>	
7	<code>COMMIT;</code>	
8		<code>SELECT xmin, xmax, * FROM predmet2 WHERE sifpredmet &gt; 67;</code>
9		<code>COMMIT;</code>
10		<code>BEGIN TRANSACTION;</code>
11		<code>SELECT txid_current() txId;</code>
12		<code>SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;</code>
13		<code>SHOW TRANSACTION ISOLATION LEVEL;</code>
14	<code>BEGIN TRANSACTION;</code>	
15	<code>SELECT txid_current() txId;</code>	
16	<code>INSERT INTO predmet2 VALUES (69, 'Objektne baze podataka ', 4.0, 4);</code>	
17	<code>COMMIT;</code>	
18		<code>SELECT xmin, xmax, * FROM predmet2 WHERE sifpredmet &gt; 67;</code>
19		<code>COMMIT;</code>
20		<code>BEGIN TRANSACTION;</code>
21		<code>SELECT txid_current() txId;</code>
22		<code>SHOW TRANSACTION ISOLATION LEVEL;</code>
23		<code>SELECT xmin, xmax, * FROM predmet2 WHERE sifpredmet &gt; 67;</code>
24		<code>COMMIT;</code>

MVCC uz transakcijsku razinu *SERIALIZABLE* omogućuje da korisnik radi složene analize nad "svojom" instancom baze koja predstavlja svojevrsnu snimku baze u trenutku kada je korisnik započeo transakciju, bez potrebe da se podaci doslovno kopiraju i spremaju u zasebni repozitorij, a također i bez zaključavanja pristupa drugim korisnicima.

No ovakav način organizacije višekorisničkog pristupa i dalje ima svoju cijenu. Razmislite - koja je potencijalna negativna strana MVCC protokola ako zamislimo okruženje gdje se istovremeno provodi i veliki broj izmjena i veliki broj složenih analiza s transakcijama koje traju relativno dugo (iz perspektive računala)? Što biste preporučili da ublažite mogući negativni učinak ovakvog okruženja na performanse sustava?

Pretpostavimo da tablica **predmet2** ima 70 redaka te da su korisnici **userA** i **userB** oboje odabrali izolacijsku razinu `SERIALIZABLE`, a potom u tablicu **predmet2** unijeli svaki po jedan novi (različiti) redak. Nakon unos retka korisnik **userA** završava transakciju, dok **userB** ostavlja svoju transakciju otvorenom.

Svaki korisnik tada prebrojava koliko tablica **predmet2** ima redaka. Koji će rezultat dobiti?

- a** Korisnik **userA** vidi 72 retka (originalnih 70 i sve nove retke), korisnik **userB** samo originalnih 70.
- b** Oba korisnika vide 71 redak (originalnih 70 i "svoj" novi redak).
- c** Korisnik **userA** vidi 71 redak (originalnih 70 i "svoj" novi redak), dok korisnik **userB** vidi samo 70 budući da još nije potvrdio svoju transakciju.
- d** Korisnik **userA** vidi 71 redak (originalnih 70 i "svoj" novi redak), dok korisnik **userB** 72 (svoj novi redak i potvrđeni novi redak korisnika **userA**).
- e** Nijedan korisnik ne vidi ništa - paralelni unos dva retka u istu tablicu dok su oba korisnika u izolacijskoj razini `SERIALIZABLE` dovesti će do situacije "deadlock"-a, tj. obostranog smrzavanja transakcija.