Obrada transakcija, planovi izvršavanja transakcija, izolacija i zaključavanje MS SQL baze podataka

Seminarski rad

Strahinja Laktović
br. indeksa 1089

Sistemi za upravljanje bazama podataka
Elektronski fakultet u Nišu

1. Uvod

Transakcija je niz čitanja i upisivanja u bazu koji sistem za upravljanje bazom podataka tretira kao jednu celinu (eng. *unit of work*) i predstavlja važan koncept u konkurentnom izvršenju poslova nad bazom podataka i oporavljanju od grešaka. Ovaj rad će se detaljno baviti transakcijama, planovima njihovog konkurentnog izvršavanja i mehanizmima kojima se takav rad omogućava.

Stoga, u drugom poglavlju se obradjuju ACID svojstva transakcija a treće poglavlje uvodi nas u konkurentno izvršavanje više transakcija.

Biće reči o njihovom planu izvršavanja (eng. *schedule*), pojmu plana koji se može serijalizovati ali i o animalijama koje se mogu javiti pri konkurentnom radu.

Mehanizmi zaključavanja transakcija kojima se omogućava izvršavanje samo onih planova koji se mogu serijalizovati i problemi koji se mogu javiti prilikom njihove implementacije tema su četvrtog poglavlja.

Anomalije pri konkurentnom izvršavanju transakcija se mogu izbeći podešavanjem nivoa izolacije transakcija koje se obradjuju u petom poglavlju.

Peto poglavlje se bavi implementacijom transakcija u MS SQL Server-u i specifičnim rešenjima problema vezanih za preplitanje rada transakcija, njihovo zaključavanje resursa i nivoa izolacije.

U šestom poglavlju će biti demonstrirano preplitanje transakcija, problemi do kojih preplitanje može dovesti i predočena rešenja koje rešavaju te probleme na primeru MS SQL Server-a.

2. ACID

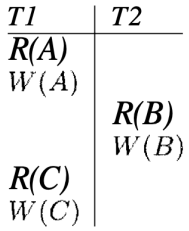
ACID svojstva transakcija garantuju validnost podataka u bazi u konkurentom okruženju i pri oporavku od grešaka. ACID je akronim reči (eng. *Atomicity, Consistency, Integrity, Durability*) i odnosi se na atomičnost, konzistentnost, integritet i održivost.

- *Atomičnost* - sve operacije unutar transakcije moraju biti izvršene, ili nijedna. Korisnici ne treba da brinu o transakcijama koje su na pola prekinute usled pada sistema ili nečeg sličnog
- *Konzistentnost* - transakcije treba da ostave bazu u konzistentnom stanju i time ne prekrše pravila izmene podataka i njihova ograničenja. Odgovornost je programera da ovo pravilo bude do kraja ispoštovano
- *Integritet* - iako sistem za upravljanje bazama podataka prepliće izvršavanje transakcija, na transakcije ne smeju uticati druge transakcije koje su u toku
- *Održivost* - svojstvo transakcija koje garantuje da će transakcije koje su završene biti održive čak i u slučaju pada sistema

3. Planovi izvršavanja

Sistem za upravljanje bazom podataka vidi transakciju kao niz akcija, akcija koje mogu biti čitanje ili upisivanje i konačno *potvrdi* (eng. *commit*) ili *obustavi* (eng. *abort*).

Plan izvršavanja čini lista akcija (čitanje, upisivanje, potvrdi ili obustavi) više transakcija koje se izvršavaju konkurentno. Niz akcija koji „vidi" svaka transakcija treba biti isti kod svih koje su deo istog plana. Ukoliko plan sadrži akcije potvrdi ili obustavi svih transakcija koje su deo plana kažemo da je kompletan, a ukoliko se transakcije ne prepliću već izvršavaju sukcesivno onda je serijski.



Slika 3.1 *Plan izvršavanja dveju transakcija*

Sistem za upravljanje bazama podataka može da isprepletе akcije više transakcija i time znatno ubrza rad celokupne baze. Naravno, ne mogu svi koraci biti isprepleteni, jer mora biti ispoštovana izolacija transakcija. Sa druge strane, pokretanje druge transakcije dok se jedna blokira na U/I radnju je sasvim moguće zato što CPU i U/I mogu da rade paralelno. Npr. kratka transakcija može da se izvrši dok duga čeka na U/I operaciju, dok bi u serijskom radu možda morala da bude blokirana nepotrebno dugo iako se radi o jednostavnoj transakciji.

Plan koji se može serijalizovati je plan izvršavanja nad skupom transakcija koji je jednak nekom serijskom planu izvršavanja nad podskupm potvrdjenih transakcija unutar tog skupa.

Redosled izvršavanja potvrdjenih transakcija ne mora nužno da bude isti, pri ponavljanju izvršenja, ali će krajnji rezultat biti isti.

T1	T2
R(A)	
$W(A)$	
	R(A)
	W(A)
R(B)	
$W(B)$	
	R(B)
	W(B)
	Commit
Commit	

Slika 2.2 *Plan izvršavanja dveju transakcija koji je moguće serijalizovati*

3.1 Anomalije pri preplitanju

Zbog preplitanja više transakcija sasvim je realna sitacija gde prepletene transakcije žele da obave neku od akcija čitanja ili pisanja nad istim objektom. To može dovesti do sledećih anomalija:

- *Čitanje nepotvrdjenih podataka* (eng. *Reading uncommited data*) - transakcija jedan upisuje novu vrednost nekog objekta koji zatim treba da pročita druga transakcija pre nego što je prva potvrdjena čime ona radi sa novom vrednošću koja možda na kraju neće ni biti sačuvana. Čitanje ovakvih podataka, zbog kojih je baza u nekonzistentnom stanju, zove se *dirty read*.
- *Neponovljivo čitanje* (eng. *Unrepeatable read*) - dešava se kada jedna transakcija pročita neku vrednost, zatim je druga izmeni, zbog čega prva, kad pročita opet ne dobija isti rezultat iako nije menjala tu vrednost u medjuvremenu.
- *Prepisivanje nepotvrdjenih podataka* (eng. *Overwriting uncommited data*) - anomalija koja se dešava kada jedna transakcija izmeni vrednost nekog objekta a zatim nje i druga čime gubimo prvu izmenu.

4. Kontrola preplitanja zaključavanjem

Da bi se omogućilo izvršavanje samo planova koje je moguće serijalizovati i da se ne bi izgubili rezultati potvrdjenih akcija usled obustavljanja drugih sistem za upravljanje bazama podataka koristi sistem zaključavanja. Protokol zaključavanja je skup pravila koje transakcije moraju ispoštovati da bi se omogućio pravilan rad sistema usred njihovog preplitanja. Najšire korišćen protokol zaključavanja je Striktno dvo-fazno zaključavanje (eng. *Strict Two-Phase Locking*)

Striktno dvo-fazno zaključavanje sastoji se od dva pravila:

• Ukoliko transakcija želi da čita podatak mora da zahteva *zajedničko* zaključavanje nad podatkom, a ukoliko želi da upisuje novu vrednost onda *ekskluzivno* zaključavanje

• sve podatke koje je transakcija zaključala bivaju oslobodjeni nakon njenog potvrdjivanja

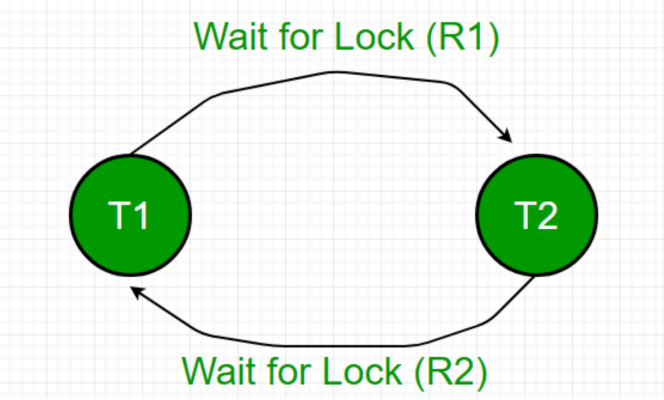
Protokol zaključavanja omogućava bezbedno preplitanja transakcija. Ukoliko transakcije zaključavaju različite podatke izvršiće se skroz nezavisno, a ukoliko obavljaju akcije nad istim podatkom sistem zaključavanja izvršiće serijalizaciju njihovog rada, odnosno dok jedna transakcija ne završi sa zaključanim podatkom druga ne može izvršiti akcije nad njim.

T1	T2
8(A)	
R(A)	
	8(A)
	R(A)
	X(B)
	R(B)
	W(B)
	Conllnit
X(C)	
R(C)	
W(C)	
Commit	

Slika 4.1 *Primer plana sa striktnim dvo-faznim zaključavanjem i prepletenim akcijama dveju transakcija*

4.1 Potpuni zastoj

U radu sa striktnim dvo-faznim zaključavanjem može doći do situacije potpunog zastoja (eng. *deadlock*) gde se jedna transakcija blokira tražeći resurs druge, dok je druga blokirana jer joj je potreban resurs prve.



Slika 4.2 *Potpuni zastoj*

Jedan od jednostavnijih načina za detekciju potpunog zastoja je pesimističko zaključivanje da je, ukoliko transakcija predugo čeka na resurs, vrlo verovatno došlo do zastoja i ta transakcija treba biti obustavljena.

Drugi način bi bilo postojanje procesa koji povremeno traži potpune zastoje i razrešava ih obustavljajući jedne od transakcija (ovakav mehanizam koristi MS SQL Server, o čemu će kasnije biti više reči).

Kada transakcije zatraže zajedničko zaključavanje nad nekim podacima ona može kasnije zatražiti i ekskluzvno nad odredjenim podskupom podataka. Recimo, upit koji želi da promeni vrednost kolone samo odredjenih redova u nekoj tabeli može zatražiti deljeno zaključavanje nad celom tabelom i onda nakon filtriranja ekskluzivno zaključavanje samo onih redova čije kolone trebaju biti izmenjene. Ovakav pristup *pojačavanja nivoa zaključavanja* može dovesti do povećanja broja potpunih zastoja.

Sa druge strane pristup gde se, ukoliko iskoristimo predjašnji primer, zatraži ekskluzivno zaključavanje tabele isprva pa se onda ono *spusti* na zajedničko zaključavanje redova koji zadovoljavaju uslov nekog filtriranja smanjuje konkurentnost izvršavanja transakcija ali na duže staze povećava ukupan broj izvršenih transakcija smanjivanjem mogućnosti dolaska u situaciju potpunog zastoja.

4.2 Nivoi zaključavanja

Sistem za upravljanje bazama podataka može vršiti zaključavanje resursa na nivou celih tabela ili odredjenih redova. Zaključavanje na nivou redova je daleko optimalnije od zaključavanja celih tabela, medjutim može dovesti do novih problema.

Na primer, ukoliko jedna transakcija ima zajedničko zaključavanje svih redova tabele koje se pojača na ekskluzivno zaključavanje filtriranih redova po nekom kriterijumu i u tom trenutku druga transakcija kreira novi red koji zadovoljava kriterijum filtriranja i ekskluzivno zaključa taj red dolazi do *fantomskog* problema. Prva transakcija može dva puta da pročita redove koji zadovoljavaju odredjen kriterijum i da dobije različite rezultate iako sama nije izvršila nikakve izmene. Često aplikacije koje imaju transakcije koje mogu da dovedu do fantomskog problema mogu povremeno da dozvole ovako nešto. Na programeru je odgovornost da li će dozvoliti ovako nešto i time povećati konkurentnost transakcija ili povećati nivo izolacije.

5. Nivoi izolacije transakcija

Da bi korisnici imali kontrolu nad transakcijama, u smislu pojačavanja restrikcija kada su u pitanju neke ili povećanja konkurentnosti kada su u pitanju druge transakcije SQL dozvoljava podešavanje *način rada*, *veličinu dijagnostike* i *nivo izolacije* pri kreiranju transakcije. Veličina dijagnostike predstavlja broj greški koje mogu biti snimljene, dok način rada može biti *samo čitanje* (eng. *read-only*) gde je dozvoljeno samo čitanje podataka *ili čitanje i pisanje* (eng. *read-write*), što omogućava transakciji da čita, menja, briše i dodaje nove podatke. Transakcije koje samo čitaju podatke mogu samo deljeno da zaključavaju resurse.

Nivo izolacije podešava nivo izloženosti transakcija akcijama drugih transakcija. Izolacioni nivoi su

- *čitanje nepotvrdjenog* (eng. *read uncommited*) - osnovni nivo izolacije koji ne pruža nikakvu zaštitu u preplitanju transakcija
- *čitanje potvrdjenog* (eng. *read commited*) - nivo zaštite koji sprečava problem *dirty read* zato što tekuća transakcija ne može da pročita objekte koje menja neka druga
- *ponovljivo čitanje* (eng. *repeatable read*) - transakcije sa ovim nivoom izolacije čita samo potvrdjene podatke kao i prethodni, ali pročitani podaci ne mogu biti menjani dok se transakcija ne potvrdi
- *serijalizabilno* (eng. *serializable*) - najjači nivo izolacije - sprečava dešavanje svih glavnih problema kod transakcija koje se preplitano izvršavaju - *dirty read, neponovljivo čitanje* i *fantomski* problem

Čitanje nepotvrdjenog je nivo izolacije gde praktično nema zaštite i zaključavanja. Transakcije ,,vide" objekte koje menjaju druge transakcije i mogu ih pročitati i same menjati. Zbog toga je *način rada* po kom transakcije sa ovim nivoom izolacije mogu raditi *read-only.*

Transakcije sa nivoom zaštite *čitanje potvrdjenog* čitaju promene isključivo potvrdjenih transakcija, ipak vrednost koja je pročitana druga transakcija može da izmeni, a takodje može doći i do *fantomskog* problema. Ove transakcije ekskluzivno zaključavaju podatke koje menjaju i otključavaju ih tek nakon potvrdjivanja. Zajedničko zaključavanje kod čitanja otključavaju odmah nakon.

Ponovljivo čitanje daje sigurnost transakciji da ne može pročitati objekte koje su druge transakcije menjale kao i da podatke koje ona čita druge transakcije ne mogu menjati. Efektivno, zaključavanja se obezbede pre čitanja i pre pisanja i objekti se otključavaju nakon potvrdjivanja transakcije.

Ono što razlikuje nivo ponovljivog čitanja od *serijalizabilnog* nivoa je to da se kod serijalizabilnog zaključavaju i grupe objekata kojima objekti nad kojima se radi pripadaju, po odredjenom načinu filtriranja, te se ne mogu desiti *fantomski* problemi.

6. Transakcije kod MS SQL Server-a

Sistem za upravljanje bazama podataka kao što je MS SQL Server dizajniran je tako da ima sve potrebne komponente kojim bi se poštovala ACID svojstva transakcija.

Programeri su zaduženi za kreiranje transakcija koje obuhvataju celinu koja treba da obezbedi logičnu postojanost podataka. Oni definišu niz akcija (čitanje i upis) nakon kojih podaci ostaju u postojanom stanju relativnom u odnosu na biznis pravila.

Sa druge strane, sistem za upravljanje podataka je zadužen za obezbedjivanje fizičkog integriteta transakcije. Integritet se postiže metodama koje pospešuju izolovanost transakcija, kao što je zaključavanje i metode logovanja.

Metode logovanja obezbedjuju svojstvo održivosti time što se akcije pre izvršavanja upisuju na disk koji je vrlo dobro očuvan i repliciran što omogućava, u slučaju pada sistema, SQL Server-u da poništi sve transakcije koje su bile prekinute na pola.

6.1 Aplikativna kontrola transakcija

Kontrola transakcija kod SQL Servera vrši se korišćenjem *Transact-SQL* nardbi ili pozivom API funkcija baze podataka. Transakcijama se upravlja na nivou konekcije. Kada se pokrene transakcija, uspostavljanjem konekcije, sve naredbe do kraja su deo nje. Razlika je jedino kod MARS (eng. *Multiple Active Result Sets*) sesije gde se transakcije izvršavaju na nivou celine (eng. *batch*).

SQL Server, inicijalno, koristi a*utocommit* transakcije kod kojih se svaka Transact-SQL naredba nakon izvršavanja odmah potvrdi ili obustavi ukoliko je bilo grešaka u izvršavanju.

Kod *implicitnih* transakcija dovoljno je samo izvršavanje naredbe potvrdjivanja ili obustavljanja nakon odredjenog seta akcija i nova transakcija će se implicitno kreirati nakon toga.

Distribuirane transakcije su poseban tip transakcija koje se protežu na dva ili više servera - *menadžera resursa*. Upravljanje transakcijom mora da koordiniše *menadžer transakcija* kao što je MS DTC (eng. *Microsoft Distributed Transaction Coordinator*). Svaka instanca SQL servera može biti menadžer resursa u distribuiranoj transakciji. Transakcije koje se protežu na više baza u okviru jednog SQL Server-a se takodje tretiraju ko distribuirane transakcije, s tim što korisniku izgledaju isto kao i regularne.

Distribuirane transakcije u cilju koordinisanja celokupnog rada koriste protokol dvofaznog potvrdjivanja (eng. T*wo-phase commit*).

Za vreme *faze pripreme* transakcioni menadžer dobije zahteva za potvrdjivanje od nekog menadžera resursa i šalje svim resurs menadžerima naredbu da se pripreme za potvrdjivanje. Resurs menadžeri obave sve što je neophodno i zatim se logovi akcija upišu na disk. Svaki od resurs menadžera zatim šalje odgovor o uspešnosti priprema.

U *fazi potvrdjivanja* ukoliko menadžer transakcija primi od svih resurs menadžera odgovor da su pripreme bile uspešne šalje aplikaciji potvrdan odgovor i završava transakciju. Ako bar jedan resurs menadžer pošalje negativan odgovor onda menadžer transakcija šalje naredbu obustave svim ostalima.

Naredba *commit* koja se piše na kraju transakcija svojim izvršenjem garantuje da su sve modifikacije unutar transakcije sačuvane i otključava sve zaključane resurse.

Ukoliko se desi greška unutar transakcije ili programer odluči da obustavi transakciju, naredba *rollback* poništava sve promene nastale unutar nje i otključava sve resurse.

6.2 Zaključavanje i izolacija

SQL Server koristi zaključavanje i verzioniranje redova za očuvanje integriteta transakcije i konzistentnosti baze.

Zaključavanje resursa podrazumeva nemogućnost drugih transakcija da ih koriste dok jedna ne završi sa njima. Mogu se zaključavati redovi, stranice, tabele, u zavisnosti od potreba transakcije.

Verzioniranje redova je tehnika koja se koristi kada se za izolacioni nivo transakcije izabere nivo koji koristi verzioniranje. Time se za čitanje koristi verzija koja je bila aktuelna na početku

transakcije, izbegava zajedničko zaključavanje i značajno smanjuje šansa za medjusobno blokiranje.

Vrste zaključavanja kod SQL Server-a su:

- *zajedničko* (eng. *shared*) - koristi se za čitanje podataka
- za *ažuriranje* (eng. *update*) - koristi se za ažuriranje resursa
- *ekskluzivno* (eng. *exclusive*) - operacije kao što su dodavanje, ažuriranje i brisanje koriste ovaj tip zaključavanja. Onemogućava istovremeno ažuriranje nad istim resursom.
- *zaključavanje s namerom* (eng. *intent*) - koristi se za uspostavljanje hijerarhije zaključavanja. Tipovi zaključavanja s namerom mogu biti zajedničko, ekskluzivno ili zajedničko sa ekskluzivnom namerom.
- *grupno ažuriranje* (eng. *bulk update*) - koristi se kod grupnog ažuiranja kada je specificiran TABLOCK hint
- *zaključavanje raspona ključeva* (eng. *key-range*) - štiti niz redova koji su pročitani kod serijalizabilne transakcije i onemogućava dodavanje novih, sprečava *fantomski* problem
- *zaključavanje šeme* (eng. *schema lock*) - zaključavanje na nivou tabele kada se brišu tabele ili dodaju kolone

Zaključavanje za ažuriranje se koristi da bi se izbegli potpuni zastoji u radu kod sistema koji imaju samo zajedničko i ekskluzivno zaključavanje. Često dve transakcije zajednički zaključaju redove koje čitaju i onda pojačaju nivo zaključavanja na ekskluzivni kada žele da ažuriraju neki od podataka što može dovesti do potpunog zastoja. U slučaju zaključavanja za ažuriranje samo jedna transakcija može da ga koristi u jednom trenutku. Ukoliko transakcija ažurira resurs onda to zaključavanje prelazi u ekskluzivno.

Zaključavanja s namerom koriste se za zaštitu zajedničkog i ekskluzivnog zaključavanja nad manjim resursima u hijerarhiji zaključavanja.

Hijerarhija sadrži resurse koji mogu biti redovi, indeks ključevi, stranice, grupe stranica, stabla ili hrpe, tabele, fajlovi i sama aplikacija.

Transakcije pre zaključavanja prvo provere zaključavanja s namerom na višem nivou kao što je nivo tabele da bi znale da li da zaključaju redove u tabeli čime se pojačavaju performanse sistema.

SQL Server koristi *dinamičko zaključavanje* - tehniku kojom se na osnovu samog upita i šeme dinamički odredjuje na kojem nivou će se zaključavanje odraditi.

Potpune zastoje rešava monitor zaključavanja - nit koja povremeno proverava sve zadatke na nivou servera. Osnovni interval je pet sekundi. Medjutim ukoliko se potpuni zastoj pronadje onda se aktivno prate sva sledeća zaključavanja da bi se pronašli ciklusi. Provere se rade povremeno zato što postoji mala verovatnoća da sistem dodje do potpunog zastoja a koristi se posebna nit da bi se smanjili troškovi provera.

Osim osnovnih nivoa izolacija koje SQL Server podržava - čitanje nepotvrdjenog, čitanje potvrdjenog, ponovljivo čitanje i serijalizabilno on podržava i još dva nivoa koja koriste verzioniranje redova:

- čitanje potvrdjene slike (eng. *read commited snapshot*) - kada je opcija baze READ_COMMITED_SNAPSHOT uključena tada izolacioni nivo čitanje potvrdjenog koristi verzioniranje redova za održavanje konzistencije baze prilikom čitanja podataka. Za

izmenu podataka koristi se ekskluzivno zaključavanjeKada ova opcija nije upaljena koristi se zajedničko zaključavanje pre čitanja.

- slika (eng. *snapshot*) - ovaj nivo izolacije koristi verzioniranje redova za čitanje isto kao i prethodni s tim što koristi *optimističku kontrolu konkurentnosti* (pri ažuriranju se proverava da li je neka druga transakcija već izmenila te podatke i ukoliko je došlo do tako nečeg transakcija se obustavlja i javlja greška)

7. Primeri korišćenja transakcija kod MS SQL Server-a

Za potrebe testiranja indeksa kreiran je SQL Server 2019 pomoću docker slike *mcr.microsoft.com/mssql/server:2019-latest.*

Korišćenjem skripte *CreateAndSeed.sql* kreirana je baza podataka sa tabelom *Automobiles* u koju je dodato 10 redova.

```
USE master
IF EXISTS(SELECT * FROM sys.databases WHERE NAME='transactionsdemo')
BEGIN
    ALTER DATABASE transactionsdemo SET SINGLE_USER WITH ROLLBACK IMMEDIATE
    DROP DATABASE transactionsdemo
END

CREATE DATABASE transactionsdemo

GO

USE transactionsdemo

CREATE TABLE Automobiles (
    ID int NOT NULL IDENTITY PRIMARY KEY,
    Model varchar(255),
    Price int,
);

GO

INSERT INTO Automobiles (Model, Price)
VALUES
    ('Audi', 5000), ('Audi', 6000),  ('Audi', 7000),  ('Audi', 8000),  ('Audi', 11000),
    ('BMW', 5000), ('BMW', 5500), ('BMW', 6000), ('BMW', 7000), ('BMW', 12000)

GO
```

Slika 7.1 *Kreiranje baze, tabele i dodavanje redova za potrebe testiranja*

7.1 Testiranje dirty read-a kod čitanje nepotvrdjenog nivoa izolacije

Za potrebe ovog testa prva transakcija će smanjiti cenu najjeftinijeg automobila marke „Audi" sa 5000 na 4500, zatim će čekati deset sekundi za obustavljanje. Za to vreme transakcija sa nivoom izolacije čitanje nepotvrdjeno će pokušati da nadje cenu najjeftinijeg automobila marke „Audi" i dobiti 4500, nevalidnu informaciju.

```
BEGIN TRANSACTION;
UPDATE Automobiles SET Price = 4500
WHERE Id = (
    SELECT TOP 1 Id
    FROM Automobiles
    WHERE Model = 'Audi'
    ORDER BY Price ASC
);

WAITFOR DELAY '00:00:10.000';
ROLLBACK TRANSACTION;
```

Slika 7.2 *Prva transakcija koja smanjuje cenu automobila i onda se poništava posle 10 sekundi*

```
1  SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
2
3  SELECT TOP 1 Price
4  FROM Automobiles
5  WHERE Model = 'Audi'
6  ORDER BY Price ASC;
```

Results | Messages

	Price
1	4500

Slika 7.3 *Druga transakcija koja čita nevalidnu cenu najjeftinijeg automobila marke „Audi"*

Podizanjem nivoa izolacije na *čitanje potvrdjenog* druga transakcija čeka koja je pokrenuta nakon prve čeka da se prva završi zato što je zaključala podatke i nakon toga čita validnu vrednost najjeftinijeg automobila.

```
1   SET TRANSACTION ISOLATION LEVEL READ COMMITTED
2
3   SELECT TOP 1 Price
4   FROM Automobiles
5   WHERE Model = 'Audi'
6   ORDER BY Price ASC;
```

Results Messages

	Price
1	5000

Slika 7.4 *Druga transakcija sa nivoom izolacije čitanje potvrdjenog čita validnu vrednost najjeftinijeg automobila*

Naredba na slici 7.5 omogućava korišćenje verzioniranja redova za nivo izolacija čitanje potvrdjenog.

```
ALTER DATABASE Automobiles
SET READ_COMMITTED_SNAPSHOT ON
WITH ROLLBACK IMMEDIATE;
GO
```

Slika 7.5 *Aktiviranje verzioniranja redova*

7.2 Testiranje neponovljivog čitanja kod nivoa izolacije čitanje potvrdjenog

U ovom testu prva transakcija će pročitati vrednosti top tri najskupljih automobila marke „BMW", stati na deset sekundi, zatim će se izvršiti transakcija broj dva koja će smanjiti cenu jednog od njih. Prvi transakcija će prilikom drugog čitanja dobiti druge rezultate za top tri automobila koji se ne poklapaju sa prvim. Ovakav problem može se rešiti podizanjem nivoa izolacije transakcije na ponovljivo čitanje. Tada će drugi upit morati da sačeka završetak prvog upita da bi se izvršio.

```
UPDATE Automobiles SET Price = 5000 WHERE Id = 8
```

Slika 7.5 *Druga transakcija - upitom koji menja cenu jednog od automobila*

```sql
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

BEGIN TRANSACTION;

-- Query 1 - first run
SELECT TOP 3 *
FROM Automobiles
WHERE Model = 'BMW'
ORDER BY Price DESC;

WAITFOR DELAY '00:00:10.000';

-- Query 1 - second run
SELECT TOP 3 *
FROM Automobiles
WHERE Model = 'BMW'
ORDER BY Price DESC;

COMMIT TRANSACTION;
```

Results Messages

	ID	Model	Price
1	10	BMW	12000
2	9	BMW	7000
3	8	BMW	6000

	ID	Model	Price
1	10	BMW	12000
2	9	BMW	7000
3	7	BMW	5500

Slika 7.6 *Prva transakcija koja dobija različite rezultate prilikom izvršavanja istog upita*

```sql
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

BEGIN TRANSACTION;

-- Query 1 - first run
SELECT TOP 3 *
FROM Automobiles
WHERE Model = 'BMW'
ORDER BY Price DESC;

WAITFOR DELAY '00:00:10.000';

-- Query 1 - second run
SELECT TOP 3 *
FROM Automobiles
WHERE Model = 'BMW'
ORDER BY Price DESC;

COMMIT TRANSACTION;
```

Results Messages

	ID	Model	Price
1	10	BMW	12000
2	9	BMW	7000
3	8	BMW	6000

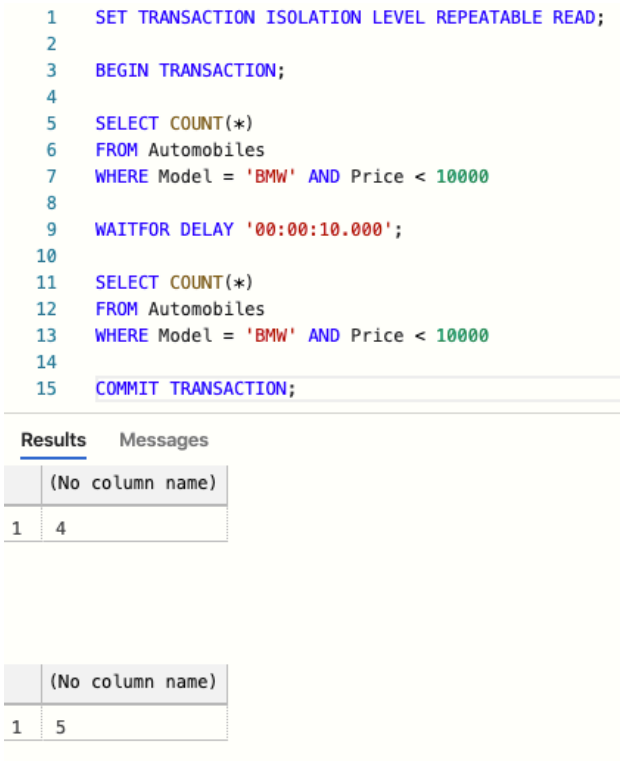
	ID	Model	Price
1	10	BMW	12000
2	9	BMW	7000
3	8	BMW	6000

Slika 7.7 *Postavljanjem izolacionog nivoa na ponovljivo čitanje dobijaju se istovetni rezultati*

7.3 Testiranje fantomskog problema kod ponovljivog čitanja

Treći test sastoji se od prve transakcije koja čita broj automobila marke „BMW" čija je cena ispod 10.000, zatim čeka deset sekundi za vreme kojih druga transakcija doda nov automobil sa cenom 5000 koja će povećati broj automobila sa 4 na 5 u drugom čitanju transakcije broj jedan.

Serijalizabilan nivo izolacije rešava ovaj problem i omogućava transakciji broj jedan da pročita vrednost 4 oba puta.



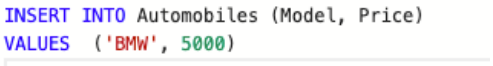
```
1   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
2
3   BEGIN TRANSACTION;
4
5   SELECT COUNT(*)
6   FROM Automobiles
7   WHERE Model = 'BMW' AND Price < 10000
8
9   WAITFOR DELAY '00:00:10.000';
10
11  SELECT COUNT(*)
12  FROM Automobiles
13  WHERE Model = 'BMW' AND Price < 10000
14
15  COMMIT TRANSACTION;
```

Results Messages

	(No column name)
1	4

	(No column name)
1	5

Slika 7.8 *Prva transakcija sa nivoom izolacije ponovljivo čitanje kod koje se javlja fantomski problem*



```
INSERT INTO Automobiles (Model, Price)
VALUES  ('BMW', 5000)
```

Slika 7.9 *Druga transakcija koja dodaje nov automobil*

```
1   SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
2
3   BEGIN TRANSACTION;
4
5   SELECT COUNT(*)
6   FROM Automobiles
7   WHERE Model = 'BMW' AND Price < 10000
8
9   WAITFOR DELAY '00:00:10.000';
10
11  SELECT COUNT(*)
12  FROM Automobiles
13  WHERE Model = 'BMW' AND Price < 10000
14
15  COMMIT TRANSACTION;
```

Results Messages

	(No column name)
1	4

	(No column name)
1	4

Slika 7.10 *Prva transakcija sa serijalizabilnim nivoom izolacije i rešenim fantomskim problemom*

8. Zaključak

Transakcije su izuzetno bitan mehanizam u konkurentnom izvršavanju poslova u radu sistema za upravljanje bazama podatka. Da bi transakcije ispunjavale svoju funkciju moraju imati četiri osobine poznatije pod akronimom ACID - atomičnost, konzistentnost, integritet i održivost.

Da bi sistem bio efikasan transakcije se mogu preplitati u izvršavanju. Planovi izvršavanja sastoje se od akcija više transakcija koje se prepliću i moraju poštovati njihov integritet.

Pri preplatanju akcija može doći do pojave anomalija koje se mogu razrešiti tehnikama zaključavanja ili verzioniranja redova kod MS SQL Server-a konkretno i podešavanjem nivoa izolacije.

U predzadnjem poglavlju obradjene su specifičnosti SQL Servera i njegove implementacije navedenih tehnika, dok je u zadnjem na primerima pokazano pojavljivanje anomalija i njihovo razrešavanje povećanjem nivoa izolacije.

LITERATURA

[1] Kurs *Sistemi za upravljanje bazama podataka* na Elektronskom fakultetu u Nišu - https://cs.elfak.ni.ac.rs/nastava/course/view.php?id=57

[2] „Database Management Systems“ - Raghu Ramakrishnan & Johannes Gehrke

[3] „Transaction locking and row versioning guide - SQL Server | Microsoft Docs“ - https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=aps-pdw-2016-au7

[4] „Deadlock in DBMS - GeeksForGeeks" - https://www.geeksforgeeks.org/deadlock-in-dbms/