MySQL - Transakcije

Evidencija račun

Potrebno je evidentirati račune izdane u trgovini. Za svaki račun potrebno je pratiti *broj_računa*, *datum_izdavanja* i *zaposlenika* koji je račun izdao, te kupca koji je platio račun. Za zaposlenika se prati: *ime*, *prezime*, *oib*, *datum_zaposlenja*, dok se za kupca prati *ime* i *prezime*. Na svakom računu se nalazi stavke koje u sebi sadrže *artikl* i *količinu*. Artikl se sastoji od *naziva* i *cijene*.

```
kupac(id, ime, prezime)
zaposlenik(id, ime, prezime, oib, datum_zaposlenja)
artikl(id, naziv, cijena)
racun(id, id_zaposlenik, id_kupac, broj, datum_izdavanja)
stavka_racun(id, id_racun, id_artikl, kolicina)
```

Baza podataka & tablice

```
CREATE DATABASE trgovina;
USE trgovina;

CREATE TABLE kupac (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE zaposlenik (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  oib CHAR(11) NOT NULL,
  datum_zaposlenja DATETIME NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE artikl (
  id INTEGER NOT NULL,
  naziv VARCHAR(20) NOT NULL,
  cijena NUMERIC(10,2) NOT NULL,
  PRIMARY KEY (id)
);
```

```sql
CREATE TABLE racun (
  id INTEGER NOT NULL,
  id_zaposlenik INTEGER NOT NULL,
  id_kupac INTEGER NOT NULL,
  broj VARCHAR(100) NOT NULL,
  datum_izdavanja DATETIME NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_zaposlenik) REFERENCES zaposlenik (id),
  FOREIGN KEY (id_kupac) REFERENCES kupac (id)
);

CREATE TABLE stavka_racun (
  id INTEGER NOT NULL,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  kolicina INTEGER NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_racun) REFERENCES racun (id) ON DELETE CASCADE,
  FOREIGN KEY (id_artikl) REFERENCES artikl (id),
  UNIQUE (id_racun, id_artikl)
);
```

Unos podataka

```sql
INSERT INTO kupac VALUES (1, 'Lea', 'Fabris'),
                         (2, 'David', 'Sirotić'),
                         (3, 'Tea', 'Bibić');

INSERT INTO zaposlenik VALUES
  (11, 'Marko', 'Marić', '123451', STR_TO_DATE('01.10.2020.', '%d.%m.%Y.')),
  (12, 'Toni', 'Milovan', '123452', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.')),
  (13, 'Tea', 'Marić', '123453', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.'));

INSERT INTO artikl VALUES (21, 'Puding', 5.99),
                          (22, 'Milka čokolada', 30.00),
                          (23, 'Čips', 9);

INSERT INTO racun VALUES
  (31, 11, 1, '00001', STR_TO_DATE('05.10.2020.', '%d.%m.%Y.')),
  (32, 12, 2, '00002', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.')),
  (33, 12, 1, '00003', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.'));

INSERT INTO stavka_racun VALUES (41, 31, 21, 2),
                                (42, 31, 22, 5),
                                (43, 32, 22, 1),
                                (44, 32, 23, 1);
```

Provjera i promjena zastavice *autocommit*:

```
# Ispisivanje vrijednosti
SELECT @@autocommit;

# Isključivanje autocommit-a
SET AUTOCOMMIT = OFF;

# Uključivanje autocommit-a
SET AUTOCOMMIT = ON;
```

Commit vs Rollback: (sa *autocommit = OFF*)

```
### Rollback ###
INSERT INTO artikl VALUES (24, 'Kruh', 7.00);

-- Ispis
SELECT * FROM artikl;

-- Odbacivanje promjena
ROLLBACK;

-- Ispis
SELECT * FROM artikl;


### Commit ###
INSERT INTO artikl VALUES (24, 'Kruh', 7.00);

SELECT * FROM artikl;

-- Spremanje promjena
COMMIT;

SELECT * FROM artikl;
```

Eksplicitne transakcije:

```
START TRANSACTION;

DELETE FROM artikl WHERE naziv = 'Kruh';

SELECT * FROM artikl;

COMMIT;
```

Greška u transakciji: (napomena: potreban je korak obrade greške - sljedeće vježbe)

```
START TRANSACTION;

INSERT INTO artikl VALUES (24, 'Kruh', 7.00);

INSERT INTO stavka_racun VALUES (45, 100, 24, 2);

ROLLBACK;
```

ISOLATION LEVELS

Kreirati ćemo novog korisnika za testiranje:

```
CREATE USER blagajnik IDENTIFIED BY 'blagajnik';
GRANT ALL PRIVILEGES ON trgovina.* TO blagajnik;
```

READ UNCOMMITTED

```
# Radimo sa root korisnikom
# 1.
SET SESSION TRANSACTION ISOLATION LEVEL READ
UNCOMMITTED;
START TRANSACTION;

# 2.
INSERT INTO artikl VALUES (24, 'Kruh', 7.00);









# 6.
COMMIT;


-- Čišćenje podataka
DELETE FROM artikl WHERE naziv = 'Kruh';
COMMIT;
```

```
# Radimo sa blagajnik korisnikom









# 3.
SET SESSION TRANSACTION ISOLATION LEVEL READ
UNCOMMITTED;
START TRANSACTION;

# 4.
SELECT * FROM artikl;

# 5.
INSERT INTO artikl VALUES (25, 'Mlijeko', 5.00);


# 7.
ROLLBACK;
```

READ COMMITTED

```
# Radimo sa root korisnikom
# 1.
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

# 2.
INSERT INTO artikl VALUES (24, 'Kruh', 7.00);









# 6.
COMMIT;




-- Čišćenje podataka
DELETE FROM artikl WHERE naziv = 'Kruh';
COMMIT;
```

```
# Radimo sa blagajnik korisnikom









# 3.
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

# 4.
SELECT * FROM artikl;

# 5.
INSERT INTO artikl VALUES (25, 'Mlijeko', 5.00);


# 7.
SELECT * FROM artikl;

# 8.
ROLLBACK;
```

REPEATABLE READ

```
# Radimo sa root korisnikom
# 1.
SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE
READ;
START TRANSACTION;

# 2.
INSERT INTO artikl VALUES (24, 'Kruh', 7.00);









# 6.
COMMIT;




-- Čišćenje podataka
DELETE FROM artikl WHERE naziv = 'Kruh';
COMMIT;
```

```
# Radimo sa blagajnik korisnikom









# 3.
SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE
READ;
START TRANSACTION;

# 4.
SELECT * FROM artikl;

# 5.
INSERT INTO artikl VALUES (25, 'Mlijeko', 5.00);


# 7.
SELECT * FROM artikl;

# 8.
ROLLBACK;
```

SERIALIZABLE

```
# Radimo sa root korisnikom
# 1.
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;

# 2.
INSERT INTO artikl VALUES (24, 'Kruh', 7.00);









# 6.
COMMIT;




-- Čišćenje podataka
DELETE FROM artikl WHERE naziv = 'Kruh';
COMMIT;
```

```
# Radimo sa blagajnik korisnikom









# 3.
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;

# 4.
SELECT * FROM artikl;

# 5.
INSERT INTO artikl VALUES (25, 'Mlijeko', 5.00);


# 7.
SELECT * FROM artikl;

# 8.
ROLLBACK;
```

Zanimljive varijable:

```
# Ispis
SELECT @@innodb_lock_wait_timeout;
SELECT @@transaction_isolation;

# Izmjena
SET @@innodb_lock_wait_timeout = 5; -- timeout 5 sekundi
```

Zadatak: Napiši proceduru i transakciju koja će povećati cijenu artikla sa nazivom *'Čips'* za 10% prosječne cijene svih artikala.

```
DROP PROCEDURE IF EXISTS povecaj_cijenu_cipsa;
DELIMITER //
CREATE PROCEDURE povecaj_cijenu_cipsa()
BEGIN
  DECLARE rez DECIMAL(10, 2);

  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
  START TRANSACTION;

  SELECT AVG(cijena) * 0.1 INTO rez
    FROM artikl;

  UPDATE artikl
    SET cijena = cijena + rez
    WHERE naziv = 'Čips';

  COMMIT;
END //
DELIMITER ;

-- Testiranje
SELECT * FROM artikl;

CALL povecaj_cijenu_cipsa();

SELECT * FROM artikl;
```

Zadatak: Napiši proceduru i transakciju koja će prema predanim argumentima (*p_id, p_ime, p_prezime, p_oib*) kreirati novog zaposlenika i novog kupca.

```
DROP PROCEDURE IF EXISTS spremi_osobu;
DELIMITER //
CREATE PROCEDURE spremi_osobu(p_id INTEGER, p_ime VARCHAR(10), p_prezime VARCHAR(15),
p_oib CHAR(11))
BEGIN

  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
  START TRANSACTION;

  INSERT INTO kupac VALUES (p_id, p_ime, p_prezime);

  INSERT INTO zaposlenik VALUES (p_id, p_ime, p_prezime, p_oib, NOW());

  COMMIT;
END //
DELIMITER ;

-- Testiranje
SELECT * FROM kupac;
SELECT * FROM zaposlenik;

CALL spremi_osobu(14, 'Spremi', 'Osobu', '12345678901');

SELECT * FROM kupac;
SELECT * FROM zaposlenik;
```

> ❗ **Međutim** ako u nekom koraku dođe do greške će transakcija biti prekinuta. Npr. ako postoji jedinstveni ključ na stupcu *oib*.

```
ALTER TABLE zaposlenik ADD CONSTRAINT zaposlenik_oib_uk UNIQUE (oib);

SELECT * FROM kupac;
SELECT * FROM zaposlenik;

CALL spremi_osobu(15, 'Test', 'sa UK', '12345678901', NOW());

SELECT * FROM kupac;
SELECT * FROM zaposlenik;

ROLLBACK; # ili COMMIT -> će završiti transakciju, nije ono što želimo
```

Zadatak: Napiši proceduru i transakciju koja će prema predanim argumentima (*p_id, p_ime, p_prezime, p_oib*) kreirati novog zaposlenika i novog kupca, s time da postoji jedinstveni ključ na stupcu *oib* tablice zaposlenik odnosno `ALTER TABLE zaposlenik ADD CONSTRAINT zaposlenik_oib_uk UNIQUE (oib);`:

```
DROP PROCEDURE IF EXISTS spremi_osobu;
DELIMITER //
CREATE PROCEDURE spremi_osobu(p_id INTEGER, p_ime VARCHAR(10), p_prezime VARCHAR(15),
p_oib CHAR(11))
BEGIN
  DECLARE EXIT HANDLER FOR 1062 -- kod greške za kršenje UK-a
    BEGIN
      ROLLBACK;
      SELECT CONCAT('Zaposlenik sa oibom "', p_oib, '" već postoji!');
    END;
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
      ROLLBACK;
      SELECT 'Došlo je do greške, procedura je obustavljena!';
    END;

  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
  START TRANSACTION;

  INSERT INTO kupac VALUES (p_id, p_ime, p_prezime);

  INSERT INTO zaposlenik VALUES (p_id, p_ime, p_prezime, p_oib, NOW());

  COMMIT;
END //
DELIMITER ;

-- Testiranje
SELECT * FROM kupac;
SELECT * FROM zaposlenik;

CALL spremi_osobu(15, 'Test', 'sa UK', '12345678901');

SELECT * FROM kupac;
SELECT * FROM zaposlenik;
```

SAVEPOINT

Savepoint & Rollback To Savepoint

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

SELECT * FROM artikl;

INSERT INTO artikl VALUES (24, 'Kruh', 7.00);

-- Kreiranje SAVEPOINT-a
SAVEPOINT my_savepoint;

INSERT INTO artikl VALUES (25, 'Mlijeko', 5.00);

SELECT * FROM artikl;

-- Vraćanje na SAVEPOINT
ROLLBACK TO SAVEPOINT my_savepoint;

SELECT * FROM artikl;

INSERT INTO artikl VALUES (26, 'Fanta', 10.00);

COMMIT;

SELECT * FROM artikl;


-- Čišćenje podataka
DELETE FROM artikl WHERE naziv IN ('Kruh', 'Fanta');
COMMIT;
```

Savepoint & Release Savepoint

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

SELECT * FROM artikl;

INSERT INTO artikl VALUES (24, 'Kruh', 7.00);

-- Kreiranje SAVEPOINT-a
SAVEPOINT my_savepoint;

INSERT INTO artikl VALUES (25, 'Mlijeko', 5.00);

SELECT * FROM artikl;

-- Otpuštanje SAVEPOINT-a (ne može se više koristiti u Rollback To Savepoint)
RELEASE SAVEPOINT my_savepoint;

SELECT * FROM artikl;

INSERT INTO artikl VALUES (26, 'Fanta', 10.00);

COMMIT;

SELECT * FROM artikl;


-- Čišćenje podataka
DELETE FROM artikl WHERE naziv IN ('Kruh', 'Mlijeko', 'Fanta');
COMMIT;
```

Zadatak: Napiši proceduru i transakciju koja će prema predanim argumentima (*p_id, p_ime, p_prezime, p_id_racun, p_broj*) kreirati novog kupca, novog zaposlenika i račun. U slučaju greške samo kupac treba biti spremljen, dok se ostale promjene moraju poništiti.

```
# Pretpostavimo da imamo
ALTER TABLE zaposlenik DROP CONSTRAINT zaposlenik_oib_uk;
ALTER TABLE racun ADD CONSTRAINT racun_broj_uk UNIQUE (broj);

DROP PROCEDURE IF EXISTS spremi_osobu_i_racun;
DELIMITER //
CREATE PROCEDURE spremi_osobu_i_racun(p_id INTEGER, p_ime VARCHAR(10), p_prezime
VARCHAR(15), p_oib CHAR(11), p_id_racun INTEGER, p_broj VARCHAR(100))
BEGIN
  DECLARE EXIT HANDLER FOR 1062 -- kod greške za kršenje UK-a
    BEGIN
      ROLLBACK TO SAVEPOINT kupac_spremljen;
      COMMIT;
      SELECT CONCAT('Račun sa brojem "', p_broj, '" već postoji, ali je kupac uspješno
spremljen.');
    END;
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
      ROLLBACK TO SAVEPOINT kupac_spremljen;
      COMMIT;
      SELECT 'Došlo je do greške, procedura je obustavljena!';
    END;

  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
  START TRANSACTION;

  INSERT INTO kupac VALUES (p_id, p_ime, p_prezime);

  SAVEPOINT kupac_spremljen;

  INSERT INTO zaposlenik VALUES (p_id, p_ime, p_prezime, p_oib, NOW());

  INSERT INTO racun VALUES (p_id_racun, p_id, p_id, p_broj, NOW());

  COMMIT;
END //
DELIMITER ;


-- Testiranje
SELECT * FROM kupac;
SELECT * FROM zaposlenik;
SELECT * FROM racun;
```

```
CALL spremi_osobu_i_racun(16, 'Spremi', 'osobu&račun (1)', 'AAAAAAAAAAA', 34, '00004');

SELECT * FROM kupac;
SELECT * FROM zaposlenik;
SELECT * FROM racun;

CALL spremi_osobu_i_racun(17, 'Spremi', 'osobu&račun (2)', 'CCCCCCCCCCC', 35, '00004');

SELECT * FROM kupac;
SELECT * FROM zaposlenik;
SELECT * FROM racun;
```

Zaključavanje tablica

✅ Zaključavanje tablica je vezano za **sesiju**, a **ne za** svaku **transakciju** zasebno.

Zaključavanje tablice za čitanje

```
LOCK TABLE artikl READ;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

SELECT * FROM artikl; -- možemo normalno čitati

INSERT INTO artikl VALUES (24, 'Kruh', 7.00); -- Greška: nitko ne može
upisivati/mijenjati podatke zaključane tablice

COMMIT;

UNLOCK TABLES;
```

> 💡 Ako se tablica zaključa sa `READ` onda sve transakcije mogu čitati podatke, ali ih nitko ne može mijenjati.

Zaključavanje tablice za mijenjanje

```
LOCK TABLE artikl WRITE;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

SELECT * FROM artikl;

INSERT INTO artikl VALUES (24, 'Kruh', 7.00);

COMMIT;

UNLOCK TABLES;


-- Čišćenje podataka
DELETE FROM artikl WHERE naziv = 'Kruh';
COMMIT;
```

> 💡 Ako se tablica zaključa sa `WRITE` onda samo transakcija koja je zaključala tablicu može čitati i izmjenjivati podatke, dok druge transakcije **ne mogu** čitati niti pisati u tablicu sve dok se tablica ne otključa.

Zadaci:

1. Napiši transakciju koja će kreirati novi račun sa nekoliko stavki računa, a pritom transakcija tijekom izvođenja treba moći čitati potvrđene promjene drugih transakcija.
2. Napiši transakciju koja će izmijeniti cijene artikala tako da ih poveća za 10%, a pritom transakcija tijekom izvođenja treba moći čitati potvrđene i nepotvrđene promjene drugih transakcija.

3. Pretpostavimo da imamo ograničenje: `ALTER TABLE artikl ADD CONSTRAINT artikl_cijena_ck CHECK (cijena > 0 AND cijena < 100);`
   Napiši proceduru i transakciju koja prihvaća sljedeće parametre: *p_id, p_naziv, p_cijena*. Procedura će kreirati tri artikla: jedan artikl sa predanim parametrima, jedan artikl sa parametrima *p_id+1, p_naziv + ' small', p_cijena * 0.5* i jedan artikl sa parametrima *p_id+2, p_naziv + ' big', p_cijena * 1.5*.
   U slučaju greške nastale prekršajem *CHECK* ograničenja je potrebno poništiti samo unos *'big'* i *'small'* artikala, dok je artikl kreiran sa predanim parametrima potrebno spremiti. Transakcija tijekom izvođenja treba moći čitati potvrđene i nepotvrđene promjene drugih transakcija.