Sveučilište Jurja Dobrile

Fakultet informatike u Puli

Matej Višnjić

Benzinska Postaja – MySQL

Seminarski rad

Pula, 30.07.2021

Sveučilište Jurja Dobrile

Fakultet informatike u Puli

Matej Višnjić

Benzinska Postaja – MySQL

Seminarski rad

JMBAG: 0303092423, redoviti student

Smjer: Informatika

Kolegij: Baze podataka 2

Mentor: doc. dr. sc. Goran Oreški

Pula, 30.07.2021

Sadržaj

1. UVOD

Baza podataka je organizirana skupina strukturiranih podataka ili podataka koji se pohranjuju u računalnom sustavu. Mogu pohraniti podatke o osobama, proizvodima, narudžbama i mnogo drugih stvari.

Podaci unutar baza podataka najčešće se modeliraju u retke i stupce u nizu tablica kako bi obrada i upit podataka bili učikovitiji. Postoje razno razni programski jezici za komuniciranje sa bazama podataka, kao na primjer SQL[1], MySQL[2], postgreSQL[3] i mnogi drugi. Svaka se razlikuje od drugih, MySQL je open-source sustav relacijske baze podataka.

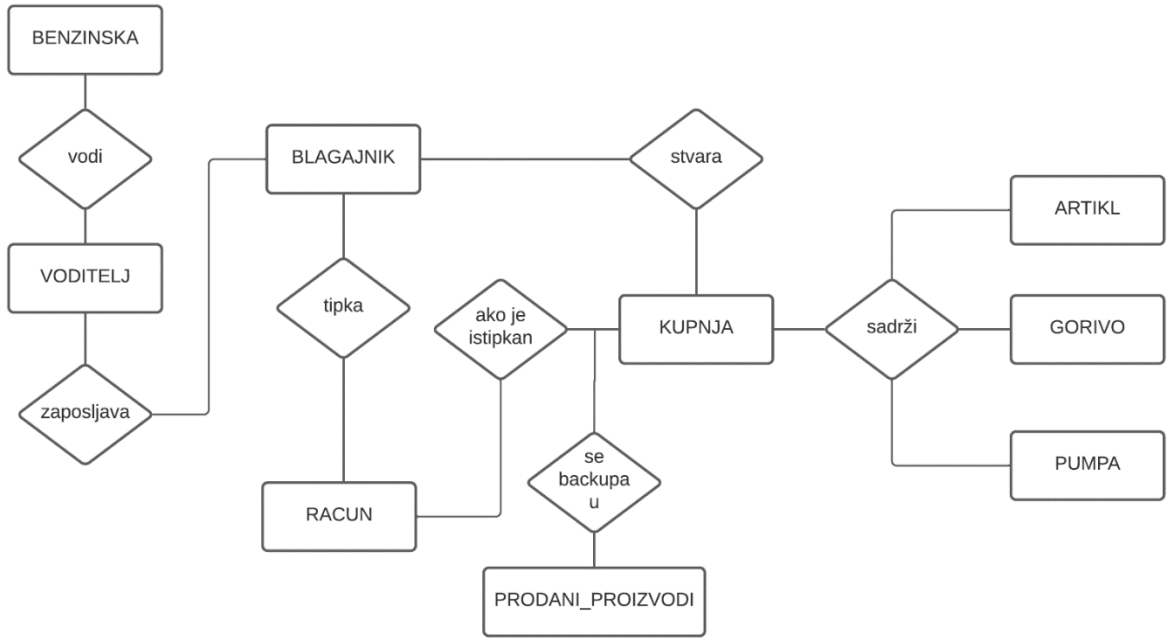
Tema projekta je baza podataka izmišljene „Benzinske postaje“. Projekt „Benzinska Postaja“ pisan je koristeći MySQL programski jezik i alat MySQL Workbench[4]. Projekt je zamišljen tako da prati benzinske postaje, voditelje, blagajnike, artikle, goriva, i pumpu. Isto tako sadržava opcije za kupnju artikala ili goriva, pratnju prodanih proizvoda i svaki istipkani račun.

[1] SQL – Strukturirani jezik upita, služi za komuniciranje sa bazom podataka
[2] https://www.mysql.com/
[3] https://www.postgresql.org/
[4] https://www.mysql.com/products/workbench/

2. IZRADA TABLICA

Baza podataka sadržava 9 tablica; benzinska, voditelj, blagajnik, artikl, gorivo, pumpa, kupnja, prodani_proizvodi, racun.

2.1 Jednostavni EER diagram



Slika 1. jednostavni EER diagram

Na slici 1. prikazan je jednostavan EER diagram bez atributa, koji prikazuje imena tablica i veze među njima. Koncept je osmišljen tako da svaka benzinska ima svoga voditelja, a voditelj zapošljava blagajnika. Blagajnik za svaku kupnju odabere stvari iz tablica artikla i goriva, te odabere određeni broj pumpe(pumpa broj 0 služi ako se kupuju stvari bez da je gorivo natankano). Nakon što je račun istipkan, stvari koji su bili u košarici se brišu iz košarice i dodavaju u prodani_proizvodi. Prodani proizvodi služe za analizu koliko je stvari prodano.

2.2 Opis tablica

```
CREATE TABLE benzinska (
    id INTEGER NOT NULL,
    ime VARCHAR(25),
    broj_telefona VARCHAR(30) NOT NULL,
    adresa VARCHAR(35),
    grad CHAR(25),
    poštanski_broj INTEGER NOT NULL,
    PRIMARY KEY(id)
);
```

Slika 2. tablica benzinska

Tablica 'Benzinska' sastoji se od 6 atributa; ID, ime, broj_telefona, adresa, grad, poštanski_broj. Služi za osnovne informacije kao što su adresa, broj telefona i slično. Id je tipa integer[1], not null[2] i primarni ključ tablice, ime varchar[3] od 25 znakova, broj_telefona varchar i not null. Atribut adresa je tip varchar od 35 znakova, a atribut grad je tipa char[4] od 25 znakova, poštanski broj je tipa integer i ne smije biti null. Svaka benzinska ima svog voditelja(o tablici 'voditelj' prikazano na slici 3.).

```
CREATE TABLE voditelj (
    id INTEGER AUTO_INCREMENT NOT NULL UNIQUE,
    sifra_v VARCHAR(11) NOT NULL UNIQUE,
    OIB VARCHAR(11) NOT NULL UNIQUE,
    ime CHAR(30),
    prezime CHAR(30),
    datum_zaposlenja DATETIME DEFAULT NOW(),
    placa INTEGER DEFAULT 6500,
    PRIMARY KEY(id),
    CONSTRAINT CHK_placa_voditelj CHECK (placa >= 6000)
);
```

Slika 3. tablica voditelj

Tablica 'voditelj' sastoji se od 7 atributa; ID, sifra_v, OIB, ime, prezime, datum_zaposlenja, placa. ID je tipa integer,auto_increment[5], not null, unique[6], i primarni ključ tablice. Sifra je tipa varchar od maksimalno 11 znakova, ne smije biti

[1] INTEGER – tip podataka koji ima 4 byta, i sprema vrijednosti od -2147483647 do 2147483647
[2] NOT NULL – ograničenje koje osigurava da vrijednosti pohranjene u stupcu nisu NULL.
[3] VARCHAR – tip podataka koji sprema vrijednosti od 0 do 65535 byta, te sprema slova i brojeve i posebne znakove
[4] CHAR – slično kao i varchar, samo sprema slova. Duljina može bit od 0 do 255
[5] AUTO_INCREMENT – koristi se za automatsko dodjeljivanje brojeva kada novi zapis dodajemo u tablicu
[6] UNIQUE – sve što je spremljeno u stupac je unikatno, ne može se ponavljati 2 iste vrijednosti.

null i unikatna je. OIB tip varchar od maksimalnih 11 znakova, not null i unique. Ime i prezime su tipa char od maksimalnih 30 znakova. Datum zaposlenja je tip datetime[1], a zadani datum je NOW()[2]. Sadrži još CHECK[3] ograničenje koje ograničava da plaća nije manja od 6000 kuna, ako prekršimo ograničenje javit će se greška CHK_placa_voditelj.

```
CREATE TABLE blagajnik (
    id INTEGER AUTO_INCREMENT NOT NULL UNIQUE,
    sifra_b VARCHAR(11) NOT NULL UNIQUE,
    OIB VARCHAR(11) NOT NULL UNIQUE,
    ime CHAR(30),
    prezime CHAR(30),
    datum_zaposlenja DATETIME DEFAULT NOW(),
    placa INTEGER DEFAULT 4500,
    PRIMARY KEY(id),
    CONSTRAINT CHK_placa_blagajnik CHECK (placa >= 4450)
);
```

Slika 4. tablica blagajnik

Tablica 'blagajnik' je identična kao i tablica 'voditelj' samo CHECK ograničenje nije postavljeno da je plaća manja od 6000 kuna, nego da nije manja od 4450 kuna.

```
CREATE TABLE artikl (
    id INTEGER NOT NULL UNIQUE,
    vrsta VARCHAR(20),
    naziv VARCHAR(50),
    cijena NUMERIC (10,2) NOT NULL,
    CONSTRAINT CHK_cijena_artikla CHECK (cijena > 0),
    CONSTRAINT CHK_vrsta_artikla CHECK (vrsta = 'GRICKALICE' OR vrsta = 'BEZALKOHOLNO' OR vrsta = 'ALKOHOL' OR vrsta = 'SLATKO' OR vrsta = 'OSTALO' OR vrsta = 'CIGARETE' OR vrsta = 'KAVA'),
    PRIMARY KEY (id)
);
```

Slika 5. tablica artikl

Tablica 'artikl' sastoji se od 4 atributa; ID koji je tip integer, not null, unikatan i primarni ključ tablice, ima svoju vrstu artikla koja je tip varchar koji prima 20 znakova, naziv varchar koji prima 50 znakova, i cijena koja je numeric[4] tip podataka koji sadrži 10 mjesta, i zakružuje na 2 decimale. Tablica sadrži i dva CHECK ograničenja, koji

[1] DATETIME – tip podataka koji služi za spremanje datuma i vremena
[2] NOW() – funkcija koja vraća trenutni datum i vrijeme.
[3] CHECK – ograničenje kad želimo nešto ograničiti. (npr. da plaća nije manja od 3800 kuna.)
[4] NUMERIC – tip podataka koji služi za brojeve sa decimalama. (lijevi broj u zagradi je ukupno brojeva, a desni znači na koliko decimala zaokružujemo)

ograničavaju da cijena ne može biti manja od nule, a drugi da se ne može unijeti vrsta proizvoda, osim onih koje su navedene u CHECK ograničenju.

```
CREATE TABLE gorivo (
    id INTEGER NOT NULL UNIQUE PRIMARY KEY,
    vrsta VARCHAR(20),
    ime VARCHAR(50),
    cijena NUMERIC (10,2) NOT NULL,
    CONSTRAINT CHK_cijena_goriva CHECK (cijena > 0),
    CONSTRAINT CHK_vrsta_goriva CHECK (vrsta = 'BENZIN' OR vrsta = 'DIZEL' OR vrsta = 'PLIN' OR vrsta = 'LOŽ ULJE')
);
```

Slika 6. tablica gorivo

Tablica 'gorivo' je identično napravljena kao i tablica 'artikl', kasnije su tablica 'artikl' i 'gorivo' povezani sa view[1](pogledi) što ćete više vidjeti u poglavlju 3.1 Pogled proizvodi

```
CREATE TABLE pumpa (
    id INTEGER NOT NULL UNIQUE PRIMARY KEY,
    CONSTRAINT CHK_id_pumpe CHECK (id>=0)

);
```

Slika 7. tablica pumpa

Tablica 'pumpa' sastoji se samo od jednog atributa ID koji je tip integer, not null, unikatan i primarni ključ tablice, ima jedno ograničenje koji provjerava da id pumpe nesmije biti manji ili jednak nuli.

[1] Pogled – *eng. view* - su virtualne tablice koje ne pohranjuju vlastite podatke već prikazuju podatke drugih tablica

```
CREATE TABLE kupnja (
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    id_pumpa INTEGER,
    id_proizvod INTEGER,
    kolicina NUMERIC (8,2) NOT NULL,
    CONSTRAINT CHK_kolicina CHECK (kolicina > 0),
    CONSTRAINT CHK_broj_pumpe CHECK (id_pumpa >= 0),
    FOREIGN KEY (id_pumpa) REFERENCES pumpa(id)
);
```

Slika 8. tablica kupnja

Tablica 'kupnja' sadrži 4 atributa; ID, id_pumpa, id_prizvod, kolicina. ID je tip integer, not null, auto_increment i ujedno i primarni ključ tablice. Id_pumpa je tip integer, te isto tako strani ključ[1] koji je povezan sa tablicom pumpa. Id_proizvod je tip integer. Količina je tip numeric koji ima 6 brojeva i 2 decimale. Također na slici vidimo i dva ograničenja, prvo ograničenje provjerava da količina nije manja od nule, a drugo ograničenje da broj pumpe nije manji ili jedan nuli.

```
CREATE TABLE prodani_proizvodi (
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    id_pumpa INTEGER,
    id_proizvod INTEGER,
    kolicina NUMERIC (8,2) NOT NULL,
    CONSTRAINT CHK_kolicina_prodanih CHECK (kolicina > 0),
    CONSTRAINT CHK_broj_pumpe_2 CHECK (id_pumpa >= 0),
    FOREIGN KEY (id_pumpa) REFERENCES pumpa(id)
);
```

Slika 9. tablica prodani_proizvodi

Tablica 'prodani_proizvodi' identični su kao i tablica 'kupnja', a služi samo za praćenje koliko je stvari prodano, kasnije pomoću procedura biti će omogućeno obaviti kupnju, dok će tablica 'kupnja' nakon svakog otipkanog računa biti će izbrisana, a proizvodi koji su istipkani prebačeni u prodani_proizvodi.

[1] Strani ključ – *eng. foreign key* vrsta ograničenja koja se mogu koristiti za održavanje itegriteta između tablica.

```
CREATE TABLE racun (
    id INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
    id_blagajnik INTEGER,
    datum_izdavanja DATETIME DEFAULT NOW(),
    osnovica NUMERIC (8,2),
    PDV NUMERIC(8,2),
    ukupno NUMERIC(8,2),
    FOREIGN KEY (id_blagajnik) REFERENCES blagajnik(id) ON DELETE SET NULL,
    CONSTRAINT CHK_osnovica CHECK (osnovica >= 0),
    CONSTRAINT CHK_PDV CHECK (PDV >= 0),
    CONSTRAINT CHK_ukupno CHECK (ukupno >= 0)
);
```

Slika 10. tablica racun

Tablica 'racun' sadrži 5 atributa; ID tipa integer, not null, unikatan i auto_increment, id_blagajnik tipa integer, datum_izdavanja tipa datetime sa funkcijom NOW(), osnovica koja je numeric ima šest brojeva i dvije decimale, PDV i ukupno koji su istog tipa kao i osnovica. Imamo u tablici strani ključ id_blagajnik koji je povezan sa tablicom 'blagajnik', te ako obrišemo blagajnika, postaviti će se null vrijednost u istipkanim računima(biti će prikazano kasnije). Također, sadrži tri check ograničenja koji 'paze' da osnovica, PDV i ukupno nije manje ili jednako nula.

2.3 Prikaz podataka

U ovome poglavnju prikazati ćemo neke od podataka koje sadržavaju tablice, i opisati ukratko za što služe.

2.3.1 Prikaz tablice benzinska

	id	ime	broj_telefona	adresa	grad	poštanski_broj
▶	1	CRObenz	052/000-111	Rovinjska 14	Pula	52220
*	NULL	NULL	NULL	NULL	NULL	NULL

Slika 11. prikaz podataka benzinska

Na slici 11. prikazan je podatak koji sadrži tablica 'benzinska'. Ova tablica sadrži samo jedan zapis i prikazuje nam informacije o benzinskoj postaji koja se trenutno koristi.

2.3.2 Prikaz tablice voditelj

id	sifra_v	OIB	ime	prezime	datum_zaposlenja	placa
1	12345	12345678910	Marko	Marić	2021-02-01 00:00:00	6700
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Slika 12. prikaz podataka voditelj

Svaka benzinska postaja ima svog voditelja, koji zapošljava ostale blagajnike. Voditelj ima svoj šifru kojom se prijavljuje u sustav, i za razliku od blagajnika može vidjeti više informacija, kao što su ukupni broj proizvoda koji su prodani, najprodavanije artikle i još mnoge druge stvari.

2.3.3 Prikaz tablice blagajnik

id	sifra_b	OIB	ime	prezime	datum_zaposlenja	placa
1	11	12345432110	Boris	Mileta	2021-02-02 00:00:00	4550
2	22	58490584322	Darko	Filipović	2021-02-02 00:00:00	4550
3	33	94384938938	Dario	Marković	2021-07-12 11:02:35	4500
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Slika 13. prikaz podataka blagajnik

Trenutno je zaposleno tri blagajnika, svaki se razlikuje po svom ID i šifri za prijavu u sustav. Kao blagajnik, nema previše mogućnosti, osim da može dodavati stavke za kupovinu, ispisivati račun i slično. Ostale stvari kao što su analiza prodaje, podizanje/spuštanje cijene goriva je blagajnicima onemogućena.

2.3.4 Prikaz tablice artikl

id	vrsta	naziv	cijena
1	KAVA	Veliki Machiatto/capucccino	10.00
2	KAVA	Mali Machiatto/capuccino	8.00
3	KAVA	Kakao	11.00
4	KAVA	Bijela Kava	14.00
5	KAVA	Espresso	7.00
6	GRICKALICE	Čips	17.99
7	GRICKALICE	Kroki	14.99
8	GRICKALICE	Štapići	12.99
9	GRICKALICE	Ribice	7.99
10	GRICKALICE	Indijski orah	24.99
20	BEZALKOHOLNO	Prirodna Voda 0.5	9.99

Slika 14. prikaz podataka artikala

Svaki artikl ima svoj jedinstveni ID, vrstu artikla, naziv i cijenu. U tablici 'artikl' prikazani su svi artikli. Također, svi artikli su prikazani u pogledima(*en.view*).

Imamo 7 različitih vrsta artikala, koji su razvrstani u pogledima. Tako možemo lakše pronaći proizvod koji želimo staviti u kupnju. **Na slici prikazano je prvih 20 artikala, sveukupno ih je 51*.*

2.3.5 Prikaz tablice gorivo

id	vrsta	ime	cijena
100	BENZIN	Eurosuper 90s	10.34
101	BENZIN	Eurosuper 95s	10.74
102	BENZIN	Eurosuper 100s	11.32
105	DIZEL	Eurodiesel	9.28
106	DIZEL	Eurodiesel +	9.88
107	DIZEL	Eurodiesel kamioni	8.28
108	DIZEL	Eurodiesel plavi	4.58
110	PLIN	Boca plina	102.00
111	PLIN	Autoplin LPG	4.70
120	LOŽ ULJE	Lož ulje	4.40
NULL	NULL	NULL	NULL

Slika 15. prikaz podataka gorivo

Kao što je prikazano na slici, tablica „gorivo“ je identična kao i tablica „artikl“. Ovdje su prikazana sva goriva koja nudi benzinska postaja. Artikli i gorivo su povezani u pogled „proizvodi“. Pogled „proizvodi“ prikazuje sve artikle i goriva skupa. Također, imamo procedure koje mogu promjeniti cijene goriva, koje radi voditelj poslovnice, to će biti prikazano u poglavlju 4.3.1 Procedura za uvećati/umanjiti određenu cijenu goriva i artikala.

2.3.6 Prikaz tablice pumpa

id
0
1
2
3
4
5
6
NULL

Slika 16. prikaz podataka pumpa

Benzinska postaja ima 6 pumpi goriva. Broj nula nam služi kada netko nije kupio gorivo, nego samo neke od artikala.

2.3.7 Prikaz tablice kupnja i prodani_proizvodi

Tablice „kupnja“ služi za unos artikala ili goriva koje blagajnik unosi kad neki kupac dođe na benzinsku. Kada ispisujemo račun tablica kupnja se briše, to jest truncate[1]. Više o tome prikazano je u poglavlju 4.3.6 Procedura za ispis računa. Tablica „Prodani proizvodi“ služi za pratiti koliko je artikala ili goriva prodano. Točnije, kada ispisujemo račun i tablica kupnja se obriše, stvari koji su bili u tablici kupnja, kopiraju se u tablicu „prodani proizvodi“.

2.3.8 Prikaz tablice račun

id	id_blagajnik	datum_izdavanja	osnovica	PDV	ukupno
1	1	2021-07-12 13:59:07	428.32	142.77	571.09
2	1	2021-07-12 13:59:59	72.00	24.00	96.00
NULL	NULL	NULL	NULL	NULL	NULL

Slika 17. prikaz podataka račun

Tablica „racun“ ima svoj ID koji predstavlja broj računa, ima ID blagajnika koji je istipkao račun, datum izdavanja, osnovicu, PDV i ukupni iznos računa. Također, postoji procedura koja prikazuje ime i prezime blagajnika koji je kucao račun, više o tome prikazano je u poglavlju 4.3.7 Procedure za pregled računa.

[1] TRUNCATE – briše sve podatke iz tablice, auto_increment brojčanik se resetira.

3. POGLEDI

Korištenje pogleda olakšavamo da spojimo dve ili više tablica u jednu virtualnu. U pogledima ne možemo dodavati nove stavke, niti brisati. Ja sam napravio 13 pogleda. Svi pogledi su napravljeni na jednostavan način i nisu prekompleksni.

3.1 Pogled proizvodi

Proizvodi su pogled gdje su spojene dvije tablice „artikl“ i „gorivo“. Sintaksa nije kompleksna, korišten je select[1] upit i union[2].

```
CREATE VIEW proizvodi AS
SELECT * from artikl
UNION
SELECT * FROM gorivo;
```

Slika 18. prikaz koda za pogled proizvodi

3.2 Pogled zaposlenici

```
CREATE VIEW zaposlenici AS
SELECT id,ime,prezime,datum_zaposlenja FROM voditelj v
UNION
SELECT id,ime,prezime,datum_zaposlenja FROM blagajnik;
```

Slika 19. prikaz koda za pogled zaposlenici

Pogled zaposlenici napravljen je slično kao i pogled „proizvodi“ samo nije select upitom odabrano sve, nego samo stupci koje smo naveli. Sadrži uniju kao i prethodni pogled. Slika 20. prikazuje podatke iz pogleda.

	id	ime	prezime	datum_zaposlenja
▶	1	Marko	Marić	2021-02-01 00:00:00
	1	Boris	Mileta	2021-02-02 00:00:00
	2	Darko	Filipović	2021-02-02 00:00:00
	3	Dario	Marković	2021-07-12 11:02:35

Slika 20. prikaz pogleda zaposlenici

[1] SELECT – koristi se za dohvaćanje redaka iz jedne ili više tablica
[2] UNION – operator koji kombinira rezultat iz dvaju ili više tablica. Kad koristimo union moramo imati isti broj stupaca. Isto tako stupci moraju imati slične tipove podataka.

3.3 Pogledi artikala i goriva

```
CREATE VIEW dizel AS
SELECT * FROM gorivo
WHERE vrsta = 'dizel';
```

```
CREATE VIEW kava AS
SELECT * FROM artikl
WHERE vrsta = 'kava';
```

Slika 21. kod pogleda dizel i kava

Pogledi za artikle i gorivo poredani su prema vrsti artikla/goriva. Ostali pogledi za određenu vrstu artikala i goriva napravljen su na identičan način. Na slici 22. prikazani su upiti za ova dva pogleda.

id	vrsta	ime	cijena
105	DIZEL	Eurodiesel	9.28
106	DIZEL	Eurodiesel +	9.88
107	DIZEL	Eurodiesel kamioni	8.28
108	DIZEL	Eurodiesel plavi	4.58

id	vrsta	naziv	cijena
1	KAVA	Veliki Machiatto/capucccino	10.00
2	KAVA	Mali Machiatto/capuccino	8.00
3	KAVA	Kakao	11.00
4	KAVA	Bijela Kava	14.00
5	KAVA	Espresso	7.00

Slika 22. prikaz pogleda „dizel“ i „kava“

4. OKIDAČI, FUNKCIJE I PROCEDURE

U ovome poglavlju proći ćemo kroz sve funkcije, procedure i okidače[1](*en.trigger*) koje su stvorene u bazi podataka za benzinsku postaju.

4.1 Okidači

Imamo samo dva okidača koji služe da ako unosimo novi artikl ili gorivo, da automatski prebacimo vrstu artikla/goriva u velika slova u slučaju da osoba koja je unosila nije pisala velikim slovima. Na slici 23. i 24. prikazana su ta dva okidača.

```
DELIMITER //
CREATE TRIGGER bi_artikl
BEFORE INSERT ON artikl
FOR EACH ROW
BEGIN
IF NEW.vrsta = LOWER(NEW.vrsta) THEN
SET NEW.vrsta = UPPER(NEW.vrsta);
END IF;
END//
DELIMITER ;
```

Slika 23. okidač za artikle

```
DELIMITER //
CREATE TRIGGER bi_gorivo
BEFORE INSERT ON gorivo
FOR EACH ROW
BEGIN
IF NEW.vrsta = LOWER(NEW.vrsta) THEN
SET NEW.vrsta = UPPER(NEW.vrsta);
END IF;
END//
DELIMITER ;
```

Slika 24. okidač za gorivo

4.2 Funkcije

Funkcije su skup SQL izraza koji izvršavaju neki zadatak ili operaciju i vraćaju jednu vrijednost. Funkcijski parametar prima samo ulazni parametar za razliku od procedura. U našoj bazi podataka ukupno imamo 6 različitih funkcija.

[1] OKIDAČ – *en.trigger* - pohranjeni program koji se automatski poziva kad se umetne, ažurira ili obriše zapis u tablici.

4.2.1 Funkcija za ispis ukupnog broja proizvoda

```
DELIMITER //
CREATE FUNCTION ukupan_broj_proizvoda() RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
DECLARE broj INTEGER;

SELECT COUNT(*) INTO broj
FROM proizvodi;

RETURN broj;
END//
DELIMITER ;
```

Slika 25. kod funkcije za ukupan broj proizvoda

Funkcija vraća ukupan broj proizvoda(artikli+gorivo), što iznosi 61.

4.2.2 Funkcija za prosječnu cijenu određene vrste goriva

```
DELIMITER //
CREATE FUNCTION prosjecna_cijena_goriva(p_vrsta VARCHAR(20)) RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
DECLARE prosjek NUMERIC(6,3);

SELECT AVG(cijena) INTO prosjek
FROM gorivo
WHERE vrsta = p_vrsta;

IF prosjek IS NULL THEN
RETURN CONCAT ('Ne postoji ta vrsta!');
ELSE
RETURN prosjek;
END IF;

END//
DELIMITER ;
```

Slika 26. kod funkcije za prosječnu cijenu goriva

Funkcija za izračun prosječne cijene goriva osmišljena je tako da kao parametar funkcije upisujemo vrstu goriva i onda funkcija vraća prosječnu cijenu određenog goriva.

4.2.3 Funkcija za količinu prodanog goriva

```
DELIMITER //
CREATE FUNCTION kolicina_prodanog_goriva (p_id_gorivo INTEGER) RETURNS VARCHAR(150)
DETERMINISTIC
BEGIN
DECLARE br NUMERIC(10,2);
DECLARE _ime VARCHAR(50);

SELECT SUM(kolicina) INTO br FROM prodani_proizvodi pp
INNER JOIN gorivo g ON pp.id_proizvod = g.id
WHERE g.id = p_id_gorivo;

SELECT g.ime INTO _ime FROM prodani_proizvodi pp
INNER JOIN gorivo g ON pp.id_proizvod = g.id
WHERE g.id = p_id_gorivo
ORDER BY g.id LIMIT 1;

IF p_id_gorivo NOT IN (SELECT id FROM gorivo)THEN
RETURN CONCAT('ID:', p_id_proizvod, ' ne postoji!');
ELSE
RETURN CONCAT('Proizvod: ',_ime,'. Prodana količina: ',br,' litara.');
END IF;

END//
DELIMITER ;
```

Slika 27. kod funkcije za količinu prodanog goriva

Funkcija vraća sveukupnu količinu prodanog goriva, ako gorivo nikad nije prodanog funkcija vraća null vrijednost. Kao parametar koristimo ID od određenog goriva. Primjer kada imamo neku prodanu količinu možete pogledati na slici 28.

	kolicina_prodanog_goriva (102)
▶	Proizvod: Eurosuper 100s. Prodana količina: 50.45 litara.

Slika 28. primjer funkcije za količinu prodanog goriva

4.2.4 Funkcija za količinu prodanog artikla

Funkcija za vraćanje količine određenog prodanog artikla napravljena je na sličan način kao i funkcija za količinu prodanog goriva. Ako niti jedan artikl nije prodan vraća vrijednost null, u suprotnom vraća broj prodanih artikala. Kao parametar isto koristimo ID određenog artikla. Kod funkcije možete pogledati na slici 28., a primjer funkcije na slici 29.

```
DELIMITER //
CREATE FUNCTION kolicina_prodanog_artikl (p_id_artikl INTEGER) RETURNS TINYTEXT
DETERMINISTIC
BEGIN
DECLARE br NUMERIC(10,2);
DECLARE _naziv VARCHAR(50);

SELECT SUM(kolicina) INTO br FROM prodani_proizvodi pp
INNER JOIN artikl a ON pp.id_proizvod = a.id
WHERE a.id = p_id_artikl;

SELECT naziv INTO _naziv FROM prodani_proizvodi pp
INNER JOIN artikl a ON pp.id_proizvod = a.id
WHERE a.id = p_id_artikl
ORDER BY a.id LIMIT 1;
IF p_id_artikl NOT IN (SELECT id FROM artikl)THEN
RETURN CONCAT('ID:', p_id_proizvod, ' ne postoji!');
ELSE
RETURN CONCAT('Proizvod: ',_naziv,'. Prodana količina: ',br,' komada.');
END IF;

END//
DELIMITER ;
```

Slika 29. kod funkcije za količinu prodanog artikla

	kolicina_prodanog_artikl(50)
▶	Proizvod: Marlboro. Prodana količina: 3.00 komada.

Slika 30. primjer funkcije za količinu prodanog artikla

4.2.5 Funkcija za ukupan iznos računa

Funkcija za ukupan iznos računa stvorena je isključivo za kasnije korištenje u proceduri za ispis račun, što možete vidjeti na 4.3.6 Procedura za ispis računa. Funkcija množi cijenu i količinu sa trenutne stavke računa, te vraća iznos računa. Kod možete pogledati na slici 30.

```
DELIMITER //
CREATE FUNCTION ukupan_iznos (p_id_pumpa INTEGER) RETURNS NUMERIC(8,2)
DETERMINISTIC
BEGIN
DECLARE ukupan_iznos NUMERIC (8,2);
SELECT SUM(cijena*kolicina) INTO ukupan_iznos FROM kupnja k
INNER JOIN proizvodi p ON k.id_proizvod = p.id
WHERE id_pumpa = p_id_pumpa;

RETURN ukupan_iznos;
END//
DELIMITER ;
```

Slika 31. kod funkcije za ukupan iznos računa

4.2.6 Funkcija za vraćanje ukupno istipkanih računa

Posljednja funkcija napravljena je da znamo podatak koliko je sveukupno računa istipkano od početka poslovanja. Funkcija ne prima niti jedan parametar, nije kompleksna, ima jedan jednostavni select upit koji sa funkcijom count[1] zbraja sve račune te vraća broj svih tih računa. Kod možete vidjeti na slici 31.

```
DELIMITER //
CREATE FUNCTION ukupno_racuna () RETURNS INTEGER
DETERMINISTIC
BEGIN
DECLARE br INTEGER;

SELECT COUNT(id) INTO br FROM racun;

RETURN br;

END//
DELIMITER ;
```

Slika 32. kod funkcije za ukupan broj izdanih računa

[1] COUNT – funkcija koja vraća broj redaka u tablici.

4.3 Procedure

Procedure ili pohranjena procedura je potprogram koji je pohranjen u bazi podataka. Procedura ima svoje ime, parametre koji mogu biti ulazni, izlazni i ulazno-izlazni i SQL naredbe. Poziv procedure ne vrši se sa select naredbom kao kod funkcija, nego s call naredbom. U našoj bazi imamo sveukupno 14 procedura. Sve procedure većinom su rađene u transakcijama[1].

4.3.1 Procedura za uvećati/umanjiti određenu cijenu goriva i artikala

Procedura je zamišljena tako da prima parametar uvećaj/umanji, vrstu goriva, i iznos za koji uvećajemo/umanjujemo cijenu. Ako krivo napišemo neke od parametara, procedura nam vraća grešku i cijena ne bude promjenjena sve dok se točno ne upisuju svi parametri.

```
DELIMITER //
CREATE PROCEDURE cijena_goriva(p_cijene VARCHAR(6), p_vrsta VARCHAR(20), p_iznos NUMERIC(5,2))
BEGIN

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

IF p_cijene != 'uvecaj' AND p_cijene != 'umanji'
THEN ROLLBACK;
SELECT CONCAT('Moraš koristiti ključnu riječ: uvecaj ili umanji!') AS GREŠKA;
ELSEIF p_vrsta NOT IN (SELECT vrsta FROM gorivo)
THEN ROLLBACK;
SELECT CONCAT('Nepoznata vrsta goriva!') AS GREŠKA;
ELSEIF p_iznos < 0
THEN ROLLBACK;
SELECT CONCAT('Iznos je manji od nule!') AS GREŠKA;

ELSEIF p_cijene = 'uvecaj' THEN
UPDATE gorivo
SET cijena = cijena + p_iznos
WHERE vrsta = p_vrsta;
SELECT CONCAT ('Vrsta: ', p_vrsta, ' je uvećana za ', p_iznos, ' kn') AS USPJEŠNO;
COMMIT;

ELSEIF p_cijene = 'umanji' THEN
UPDATE gorivo
SET cijena = cijena - p_iznos
WHERE vrsta = p_vrsta;
SELECT CONCAT ('Vrsta: ', p_vrsta, ' je umanjena za ', p_iznos, ' kn') AS USPJEŠNO;
COMMIT;
END IF;

END//
DELIMITER ;
```

Slika 33. kod procedure za uvećati/umanjiti cijenu goriva

Za cijenu artikala napravljena je još jedna potpuno identična procedura, samo se razlikuje u imenu procedure „cijena_artikala“, i u upitima ne koristimo tablicu

[1]TRANSAKCIJA – skupina izjava, upita, operacija koje se prije umetanja, ažuriranja ili brisanja mogu predati(commit) ili povratiti prijašnji podaci.

„gorivo“ nego tablicu „artikl“. Procedura javlja grešku ako unesemo nepoznatu vrstu goriva ili cijenu manju od nule ili ako koristimo krivu ključnu riječ. Primjer korištenja procedure artikla možete vidjeti na slici 33.

Slika 34. primjer procedure za uvećati/umanjiti cijenu artikala

4.3.2 Procedura za obaviti kupnju

```
DELIMITER //
CREATE PROCEDURE kupi (p_id_pumpa INTEGER, p_id_proizvod INTEGER, p_kolicina NUMERIC(6,2))
BEGIN
DECLARE utijeku INTEGER;
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
SELECT CONCAT('Došlo je do greške!') AS GREŠKA;
ROLLBACK;
END;

SELECT id_pumpa INTO utijeku FROM kupnja
WHERE id_pumpa!=p_id_pumpa
LIMIT 1;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

INSERT INTO kupnja VALUES (NULL, p_id_pumpa, p_id_proizvod, p_kolicina);

IF p_id_pumpa NOT IN (SELECT id FROM pumpa) THEN
ROLLBACK;
SELECT CONCAT('Krivi ID pumpe!') AS GREŠKA;
ELSEIF utijeku != p_id_pumpa THEN
ROLLBACK;
SELECT concat('Kupnja u tijeku! Isprintajte račun ili obrišite narudžbu!') AS GREŠKA;
ELSEIF p_kolicina < 0 THEN
ROLLBACK;
SELECT CONCAT('Količina ne smije biti manja od 0!') AS GREŠKA;
ELSEIF p_id_proizvod NOT IN (SELECT id FROM proizvodi) THEN
ROLLBACK;
SELECT CONCAT('Krivi ID proizvoda!') AS GREŠKA;
ELSE SELECT CONCAT('Uspješna narudžba!') AS USPJEŠNO;
COMMIT;
END IF;

END//
DELIMITER ;
```

Slika 35. kod procedure za kupnju

Procedura za obaviti kupnju osmišljena je na način da prima 3 parametra: ID pumpe, ID proizvoda, količinu proizvoda. Ako neki od parametra nije točan, na primjer krivi ID proizvoda, procedura javlja grešku i kupnja neće biti uspješna. U suprotnom dodavanje stavki u kupnju je uspješna. Ako želimo dodati neki proizvod na drugu pumpu javiti će nam grešku da, ili obrišemo cijelu narudžbu ili da prvo

isprintamo račun. Također, postoje opcije za micanje ili dodavanje količina proizvoda s trenutne kupnje ili za brisanje proizvoda s trenutne kupnje.

4.3.3 Procedura za pregled stavki na trenutnoj kupnji

```
DELIMITER //
CREATE PROCEDURE pregled_stavki_kupnje (p_id_pumpa INTEGER)
BEGIN

SELECT id_pumpa,p.id AS sifra_artikla,naziv,kolicina,cijena,SUM(cijena*kolicina) AS iznos FROM proizvodi p
INNER JOIN kupnja k ON k.id_proizvod=p.id
WHERE id_pumpa = p_id_pumpa
GROUP BY k.id;

END//
DELIMITER ;
```

Slika 36. kod procedure za pregled stavki trenutne kupnje

Procedura za pregled stavki jednostavna je i prima jedan parametar, ID pumpe i prikazuje nam artikle koje ćemo kupiti. Prikazuje 6 stupaca; ID pumpe, šifru artikla, naziv, količinu, cijenu, iznos. Primjer možete vidjeti na slici 36.

	id_pumpa	sifra_artikla	naziv	kolicina	cijena	iznos
▶	0	4	Bijela Kava	3.00	14.00	42.0000

Slika 37. primjer procedure za pregled stavki trenutne kupnje

4.3.4 Procedura za brisanje stavki trenutne kupnje

```
DELIMITER //
CREATE PROCEDURE brisanje_stavki_kupnje (p_id_proizvod INTEGER)
BEGIN
DECLARE art VARCHAR(50);

SELECT naziv INTO art FROM proizvodi
WHERE id=p_id_proizvod;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
IF p_id_proizvod NOT IN (SELECT id_proizvod FROM kupnja)
THEN ROLLBACK;
SELECT CONCAT('Ne postoji taj ID proizvoda!') AS GREŠKA;
ELSE
DELETE FROM kupnja
WHERE id_proizvod=p_id_proizvod;
COMMIT;
SELECT CONCAT('Artikl: ', art," je uklonjen sa kupnje!") AS USPJEŠNO;
END IF;

END//
DELIMITER ;
```

Slika 38. kod procedure za brisanje stavki trenutne kupnje

Procedura briše stavke iz košarice. Potrebno je kao parametar unijeti ID proizvoda (sifra_artikla) i ako se taj ID nalazi u košarici će se obrisati određeni artikl pod tom šifrom (IDem). U suprotnom vratit će „Ne postoji taj ID proizvoda“. Primjer možete vidjeti na slici broj 37.

USPJEŠNO
Artikl: Bijela Kava je uklonjen sa kupnje!

Slika 39. primjer procedure za brisanje stavki

4.3.5 Procedura za dodavanje/micanje količine stavki

```
DELIMITER //
CREATE PROCEDURE brisanje_kolicine_stavki (p_id_proizvod INTEGER, p_kol NUMERIC(8,2), p VARCHAR(6))
BEGIN
DECLARE naz VARCHAR(50);

SELECT naziv INTO naz FROM proizvodi
WHERE id = p_id_proizvod;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
IF p_id_proizvod NOT IN (SELECT id_proizvod FROM kupnja)
THEN ROLLBACK;
SELECT CONCAT('Ne postoji taj ID proizvoda!') AS GREŠKA;
ELSEIF p = 'dodaj' THEN
UPDATE kupnja
SET kolicina = kolicina+p_kol
WHERE id_proizvod=p_id_proizvod;
COMMIT;
SELECT CONCAT('Artikl: ', naz, ', kolicina je uvećana za ', p_kol,'.') AS USPJEŠNO;
ELSEIF p = 'oduzmi' THEN
UPDATE kupnja
SET kolicina = kolicina-p_kol
WHERE id_proizvod=p_id_proizvod;
COMMIT;
SELECT CONCAT('Artikl: ', naz, ', kolicina je umanjena za ', p_kol,'.') AS USPJEŠNO;
END IF;

END//
DELIMITER ;
```

Slika 40. kod procedure za dodavanje/micanje količine

Procedura prima 3 parametra: ID proizvod, količina, parametar p koji može biti samo „dodaj“ ili „oduzmi“, u suprotnom procedura javlja grešku. Također, javlja grešku i ako upišemo krivi ID proizvoda. Kada upišemo točan ID proizvoda koji se nalazi u košarici onda će se procedura izvršiti uspješno. Primjer uspješno uvećane količine možete pogledati na slici broj 39.

USPJEŠNO
Artikl: Bijela Kava, kolicina je uvećana za 2.00.

Slika 41. primjer procedure za dodavanje količine

4.3.6 Procedura za ispis računa

```
DELIMITER //
CREATE PROCEDURE ispis_racuna (p_id_blagajnik INTEGER, p_id_pumpa INTEGER)
BEGIN

DECLARE ukupno NUMERIC (8,2);
DECLARE pdv NUMERIC(8,2);
DECLARE osnovica NUMERIC(8,2);
DECLARE kol NUMERIC(8,2);
DECLARE p_id_proizvod INT;
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
SELECT CONCAT('Došlo je do greške,pokušajte ponovno!') AS GREŠKA;
ROLLBACK;
END;

SELECT ukupan_iznos(p_id_pumpa) INTO ukupno;

SELECT ukupno*0.25 INTO pdv;

SELECT ukupno-pdv INTO osnovica;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;

IF p_id_pumpa NOT IN (SELECT id_pumpa FROM kupnja) THEN
ROLLBACK;
SELECT CONCAT('Nema stavki!') AS GREŠKA;
ELSE
INSERT INTO prodani_proizvodi (id_pumpa,id_proizvod,kolicina)(SELECT id_pumpa,id_proizvod,kolicina INTO kol FROM kupnja);
INSERT INTO racun VALUES (NULL, p_id_blagajnik, NOW(), osnovica, pdv, ukupno);
TRUNCATE kupnja;
COMMIT;
SELECT CONCAT('Uspješno istipkan račun!') AS GREŠKA;
END IF;

END//
DELIMITER ;
```

Slika 42. kod procedure za ispis računa

Procedura za ispisivanje računa prima dva parametra: ID blagajnika i ID pumpe. Ako je neki parametar pogrešan procedura vraća grešku, isto tako ako u košarici nema stavki, vratiti će grešku. Nakon što je račun uspješno istipkan, podaci iz košarice prebacuju se u tablicu „prodani_proizvodi“, a tablica „kupnja“ se truncate(brisanje svih podataka iz tablice). Kada su točni svi parametri procedura vraća da je račun uspješno istipkan.

4.3.7 Procedure za pregled računa

Imamo dvije slične procedure za pregled svih istipkanih računa, i za pregled svih istipkanih računa određenog blagajnika. U proceduri za pregled svih računa iskorišten je select upit koji je left joinom[1] povezan sa tablicom blagajnik, procedura ne prima niti jedan parametar.

[1] LEFT JOIN – vraća podatke iz tablice 1 i tablice 2. NULL rezultat se javlja ako tablica 2 nema podudaranja sa tablicom 1.

Procedura određenih računa blagajnika napravljena je isto sa select upitom, ali je korišten inner join[1], i where[2] gdje je dan uvjet da je parametar jednak ID-u blagajnika.

```
DELIMITER //
CREATE PROCEDURE pregled_istipkanih_racuna ()
BEGIN

SELECT r.id AS racun_broj, r.id_blagajnik, b.ime,b.prezime,r.datum_izdavanja,r.osnovica,r.PDV,r.ukupno
FROM racun r
LEFT JOIN blagajnik b ON r.id_blagajnik = b.id;

END//
DELIMITER ;
```

Slika 43. kod procedure za pregled svih računa

```
DELIMITER //
CREATE PROCEDURE pregled_istipkanih_racuna_blagajnika (p_id_blagajnik INTEGER)
BEGIN

SELECT r.id AS racun_broj, r.id_blagajnik, b.ime,b.prezime,r.datum_izdavanja,r.osnovica,r.PDV,r.ukupno
FROM racun r
INNER JOIN blagajnik b ON r.id_blagajnik = b.id
WHERE b.id=p_id_blagajnik;

END//
DELIMITER ;
```

Slika 44. kod procedure za pregled računa određenog blagajnika

racun_broj	id_blagajnik	ime	prezime	datum_izdavanja	osnovica	PDV	ukupno
1	3	Dario	Marković	2021-07-22 13:53:01	360.40	120.13	480.53
2	1	Boris	Mileta	2021-07-22 13:53:33	513.05	171.02	684.07
3	2	Darko	Filipović	2021-07-22 13:54:11	853.16	284.39	1137.55

Slika 45. primjer procedure za pregled svih računa

racun_broj	id_blagajnik	ime	prezime	datum_izdavanja	osnovica	PDV	ukupno
2	1	Boris	Mileta	2021-07-22 13:53:33	513.05	171.02	684.07

Slika 46. primjer procedure za pregled računa određenog blagajnika

[1] INNER JOIN – vraća sve zapise koje imaju podudarne vrijednosti
[2] WHERE – omogućuje da odredimo uvjet pretraživanja tablica

4.3.8 Procedure za dodavanje ili brisanje blagajnika

```
DELIMITER //
CREATE PROCEDURE dodaj_blagajnika (p_sifra VARCHAR(11), p_OIB VARCHAR(11), p_ime CHAR(30), p_prezime CHAR(30), p_placa INTEGER)
BEGIN


SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
INSERT INTO blagajnik VALUES (NULL, p_sifra, p_OIB, p_ime, p_prezime, NOW(), p_placa);
SELECT CONCAT ('Uspjesno dodan zaposlenik!') AS USPJEŠNO;
COMMIT;

END//
DELIMITER ;
```

Slika 47. kod procedure za dodavanje blagajnika

Procedura za dodavanje blagajnika prima 5 parametara: sifra, OIB, ime, prezime i plaću. Procedura je napravljena jednostavno, samo insert u transakciji. Zamišljena je tako da ju samo voditelj smije koristiti za dodavanje novih blagajnika.

```
DELIMITER //
CREATE PROCEDURE brisanje_blagajnika (p_id INTEGER)
BEGIN
DECLARE v_ime CHAR(30);
DECLARE v_prezime CHAR(30);

SELECT ime INTO v_ime FROM blagajnik
WHERE id = p_id;
SELECT prezime INTO v_prezime FROM blagajnik
WHERE id = p_id;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
IF p_id NOT IN (SELECT id FROM blagajnik)
THEN ROLLBACK;
SELECT CONCAT('Ne postoji taj zaposlenik!') AS GREŠKA;
ELSE
SELECT CONCAT ('Zaposlenik ',v_ime, ' ',v_prezime,' ID: ', p_id, ' je uspješno obrisan') AS USPJEŠNO;
DELETE FROM blagajnik
WHERE id=p_id;
COMMIT;
END IF;

END//
DELIMITER ;
```

Slika 48. kod procedure za brisanje blagajnika

Procedura za brisanje blagajnika malo je kompleksnija od procedura za dodavanje blagajnika. Prvo provjerava da li postoji ID određenog zaposlenika kojeg želimo brisati, ako ne postoji onda vrati „Ne postoji taj zaposlenik!“, u suprotnom vrati nam da je „zaposlenik uspješno obrisan“. Ne preporučuje se nikad brisanje zaposlenika radi praćenja analize računa i slično.

4.3.9 Procedure za dodavanje ili brisanje artikla i goriva

```
DELIMITER //
CREATE PROCEDURE dodaj_artikl (p_id INTEGER, p_vrsta VARCHAR(20), p_naziv VARCHAR(50), p_cijena NUMERIC(10,2))
BEGIN

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
INSERT INTO artikl VALUES (p_id,p_vrsta, p_naziv, p_cijena);
SELECT CONCAT ('Uspjesno dodan artikl!') AS USPJEŠNO;
COMMIT;

END//
DELIMITER ;
```

Slika 49. kod procedure za dodavanje artikla

Procedura za dodavanje artikla ima vrlo jednostavan insert u transakciji. Prima 4 parametra: id, vrsta, naziv, cijena. Ako je artikl dodan uspješno vrati nam poruku „Uspješno dodan artikl“.

```
DELIMITER //
CREATE PROCEDURE dodaj_gorivo (p_id INTEGER, p_vrsta VARCHAR(20), p_naziv VARCHAR(50), p_cijena NUMERIC(10,2))
BEGIN

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
INSERT INTO gorivo VALUES (p_id,p_vrsta, p_naziv, p_cijena);
SELECT CONCAT ('Uspjesno dodan artikl!') AS USPJEŠNO;
COMMIT;

END//
DELIMITER ;
```

Slika 50. kod procedure za dodavanje goriva

Identična procedura kao i na slici 47. samo što se razlikuje što dodajemo u tablicu „gorivo“ a ne „artikl“ kao na proceduri iznad.

4.3.10. Procedura za brisanje proizvoda

```
#-PROCEDURA ZA BRISANJE ARTIKLA/GORIVA PREKO IMENA 14.
DELIMITER //
CREATE PROCEDURE brisanje_proizvoda (p_naziv VARCHAR(50))
BEGIN
DECLARE p_id INTEGER;

SELECT id INTO p_id FROM proizvodi
where naziv=p_naziv;

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
IF p_naziv NOT IN (SELECT naziv FROM proizvodi)
THEN ROLLBACK;
SELECT CONCAT('Ne postoji taj proizvod!') AS GREŠKA;
ELSEIF p_id IN (SELECT id FROM artikl) THEN
SELECT CONCAT ('Proizvod ID: ',p_id,' Naziv: ',p_naziv,' je uspješno obrisan') AS USPJEŠNO;
DELETE FROM artikl
WHERE id=p_id;
COMMIT;
ELSE
SELECT CONCAT ('Proizvod ID: ',p_id,' Naziv: ',p_naziv,' je uspješno obrisan') AS USPJEŠNO;
DELETE FROM gorivo
WHERE id=p_id;
COMMIT;
END IF;

END//
DELIMITER ;
```

Slika 51. kod procedure za brisanje proizvoda

Procedura za brisanje proizvoda prima jedan jedini parametar, naziv, koji upisujemo za brisanje proizvoda. Ako nije dobar naziv proizvoda procedura vraća „Ne postoji taj proizvod“. Ako je naziv točan onda se proizvod uspješno obriše. Ovakva procedura je moguća zato što imamo pogled proizvodi, gdje se nalaze svi proizvodi.

Ovo je ujedno i zadnja procedura koju ima naša benzinska postaja.

5. KORISNICI

Korisnici (*en. users*) u MySQL-u je zapis u USER tablici poslužitelja koji sadrži podatke za prijavu (korisničko ime i lozinka), privilegije računa i podatke o hostu za MySQL račun. Kada stvaramo korisnika imamo mu pravo davati privilegije što smije a što ne smije dirati. U našoj benzinskoj, voditelj će imati pristup svim procedurama, funkcijama i slično, dok blagajnici će biti ograničeni na samo nekoliko procedura koje su potrebne za obavljati kupnju, tipkati račun i slično.

5.1 Voditelj

```
CREATE USER benza_voditelj IDENTIFIED BY '000';
GRANT ALL PRIVILEGES ON benza.* TO benza_voditelj;
```

Slika 52. kod korisnika „voditelj“

Korisnik „voditelj“ lako je stvoren, doslovno dvije linije koda. Druga linija prikazuje da smo mu komandom „grant all privileges on“ dali pravo na korištenje svih procedura, funkcija, pogleda i tablica.

5.2 Blagajnik

```
CREATE USER benza_blagajnik IDENTIFIED BY '111';
GRANT ALL PRIVILEGES ON benza.kupnja TO benza_blagajnik;
GRANT SELECT ON benza.artikl TO benza_blagajnik;
GRANT SELECT ON benza.gorivo TO benza_blagajnik;
GRANT SELECT ON benza.proizvodi TO benza_blagajnik;
GRANT SELECT ON benza.benzinska TO benza_blagajnik;
GRANT SELECT ON benza.alkohol TO benza_blagajnik;
GRANT SELECT ON benza.benzin TO benza_blagajnik;
GRANT SELECT ON benza.bezalkoholno TO benza_blagajnik;
GRANT SELECT ON benza.cigarete TO benza_blagajnik;
GRANT SELECT ON benza.dizel TO benza_blagajnik;
GRANT SELECT ON benza.grickalice TO benza_blagajnik;
GRANT SELECT ON benza.kava TO benza_blagajnik;
GRANT SELECT ON benza.loz_ulje TO benza_blagajnik;
GRANT SELECT ON benza.ostalo TO benza_blagajnik;
GRANT SELECT ON benza.plin TO benza_blagajnik;
GRANT SELECT ON benza.slatko TO benza_blagajnik;
GRANT EXECUTE ON PROCEDURE kupi TO benza_blagajnik;
GRANT EXECUTE ON PROCEDURE brisanje_stavki_kupnje TO benza_blagajnik;
GRANT EXECUTE ON PROCEDURE brisanje_kolicine_stavki TO benza_blagajnik;
GRANT EXECUTE ON PROCEDURE ispis_racuna TO benza_blagajnik;
GRANT EXECUTE ON PROCEDURE pregled_stavki_kupnje TO benza_blagajnik;
GRANT EXECUTE ON PROCEDURE pregled_istipkanih_racuna TO benza_blagajnik;
GRANT EXECUTE ON FUNCTION ukupan_broj_proizvoda TO benza_blagajnik;
```

Slika 53. kod korisnika „blagajnik“

Korisnik „blagajnik“ razlikuje se od korisnika „voditelj“ jer nema pravo na sve kao voditelj, nego na određene poglede, funkcije, procedure. Kao blagajnik ne može dodavati ili brisati nove korisnike, artikle i goriva nego to radi samo voditelj. Od blagajnika mnogo procedura i funkcija je skriveno.

6. ZAKLJUČAK

U današnjem modernom dobu, to jest „doba interneta“, baze podataka susrećemo svakodnevno, na razno raznim portalima, aplikacijama i slično. To bi značilo da poznavanje rada u bazi podataka su vrlo bitne i vrlo lako ih je primjenjivati na razne sustave.

Projekt je osmišljen da što bolje obuhvaća sva pravila, osigurava sigurnost podataka, te jednostavnije korištenje baze podataka za našu osmišljenu benzinsku postaju. Ima razno raznih procedura, funkcija i tablica za obavljanje kupnje, tipkanje računa, dodavanje artikla, brisanje artikala i mnoge druge stvari. Ukupno ima 9 tablica, 14 procedura i 6 funkcija i 2 okidača. Postoje dvije vrste korisnika: voditelj i blagajnik. Sa grafičkim sučeljem poboljšali bi rad ovog projekta i bilo bi lakše koristiti sve te procedure.

7. LITERATURA

1. https://www.mysql.com/

2. https://hr.wikipedia.org/wiki/MySQL

3. https://www.w3schools.com/

4. www.youtube.com

5. www.google.com

6. https://www.postgresql.org/

7. https://www.mysql.com/products/workbench/

8. https://support.microsoft.com/hr-hr/office/osnove-baza-podataka-a849ac16-07c7-4a31-9948-3c8c94a7c204