UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Elektronske lekcije o osnovama sistema za upravljanje bazama podataka MySQL

MASTER RAD

Student
Duško Vešić

Mentor
prof. dr Miroslav Marić

Beograd
Jun, 2019

Sadržaj

Uvod

Svakodnevnica savremenog čoveka je nezamisliva bez interneta, koristi se radi zabave, učenja, poslovanja itd. Povećan pristup internetu i povećanje sadržaja koji su dostupni korisnicima kao i brz razvoj veb tehnologija dovode do povećane potražnje za stručnjacima informacionih nauka. Elektronske platforme za učenje veb tehnologija imaju veliku popularnost jer su pogodne za sticanje početnog znanja o raznim veb alatima. Jedna od malobrojnih elektronskih platformi na srpskom jeziku je eŠkola veba [1]. Interaktivne elektronske lekcije za učenje većine najkorišćenijih veb tehnologija su dostupne na platformi eŠkola veba. Platforma je nastala u okviru projekta radne grupe za obrazovni softver Matematičkog fakulteta Univerziteta u Beogradu i javno je dostupna na adresi `http://www.edusoft.matf.bg.ac.rs/eskola_veba`. U ovom radu će biti opisane elektronske lekcije o osnovama sistema za upravljanje bazama podataka MySQL.

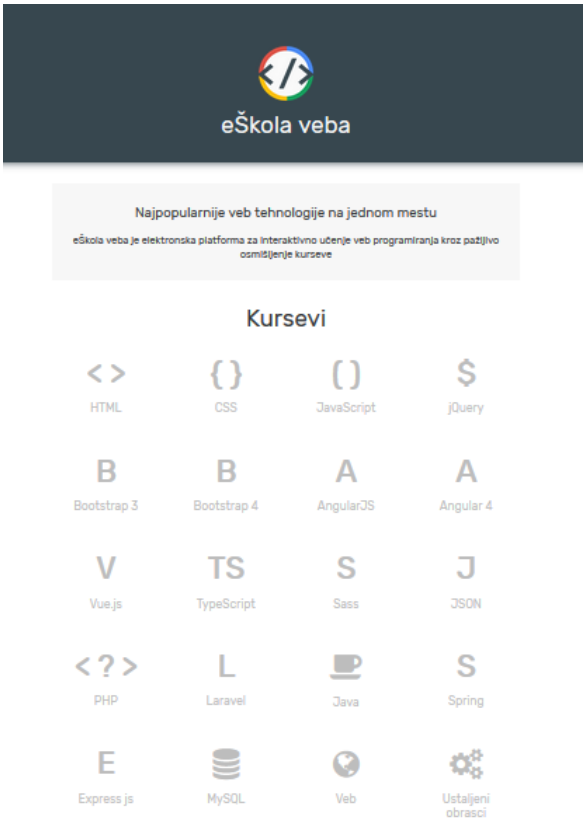
Efikasan način čuvanja obimnih podataka koji se često ažuriraju su relacione baze podataka. U njima su podaci smešteni u tabele koje su međusobno povezane. Za održavanje relacionih baza podataka potrebni su određeni alati. Sistem za upravljanje bazama podataka MySql je softverski paket otvorenog koda koji se koristi za kreiranje i održavanje relacionih baza podataka. Sistem radi kao server i omogućava interakciju korisnika ili neke aplikacije sa bazom podataka. Za pisanje upita koristi strukturirani jezik za upite SQL. Švedska kompanije MySQL AB je 23. maja 1995. godine prvi put objavila sistem za upravljanje bazama podataka MySQL. Osnivači kompanije i tvorci sistema su Dejvid Aksmark (engl. *David Axmark*), Alan Larson (engl. *Allan Larsson*) i Majkl Vidinijus (engl. *Michael Widenius*). Njihov proizvod i njegove unapređene verzije se masovno koriste zbog svoje pouzdanosti i brzine pristupa i ažuriranja podataka u bazi. Deo popularnosti duguje i tome što mnogi programski jezici (PHP, Java, Python itd.) sadrže biblioteke koje podržavaju njegov rad. Sastavni deo je WAMP i LAMP softverskih paketa namenjenih za razvoj veb aplikacija, koji na korisničkom računaru simuliraju rad servera. Sistem MySQL često je deo sadržaja u školovanju budućih veb programera.

Za potrebe rada kreirane su elektronske lekcije kako bi se što bolje predstavile mogućnosti koje pruža sistem MySQL. Lekcije su namenjene svima koji prvi put dolaze u kontakt sa ovim sistemom za upravljanje bazama podataka. U prvom poglavlju rada opisane su kreirane elektronske lekcije, a u naredna dva poglvalja obrađena je instalacija sistema u okviru WAMP softverskog paketa i povezivanje korisnika sa serverom na kojem se nalazi baza podataka pomoću programskog jezika PHP. Za ovaj deo se od korisnika očekuje da ima određena znanja programskog jezika PHP. Ostatak rada je posvećen pisanju raznih upita. Na početku su prikazani upiti za kreiranje baza i tabela, unos podataka i promenu strukture tabele. Zatim sledi poglavlje u kom je detaljno objašnjena pretraga tabele. Tada su se stekli uslovi da se u poglavlju 8 prikaže brisanje podataka iz baze i promena već unetih vrednosti. Naredna četiri poglavlja posvećena su raznim mogućnostima koje sistem MySQL pruža prilikom pretrage tabele. Od trinaestog poglavlja pa do kraja rada opisano je povezivanje tabela i njihova pretraga.

1 O elektronskim lekcijama

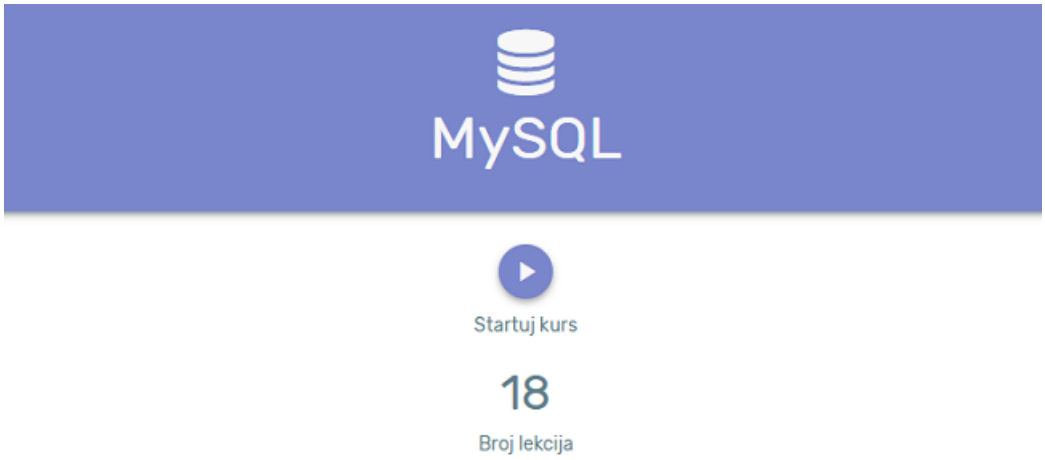
Cilj rada i elektronskih lekcija koje su kreirane za potrebe rada je da se početnici u svetu veb programiranja što pre osposobe da samostalno naprave funkcionalan sajt. Zbog toga, u nekoliko uvodnih lekcija, veća pažnja je posvećena pripremi za rad sa bazama podataka, nego samom sistemu MySQL.

Elektronske lekcije o osnovama sistema za upravljanje bazama podataka MySQL namenjene su svima a naročito onima koji nisu imali dodira sa relacionim bazama podataka. Prilagođene su početnicima, ali mogu biti korisne i za obnavljanje i unapređivanje prethodno stečenih znanja. Sve elektronske lekcije se nalaze na platformi eŠkola veba i dostupne su na veb adresi `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/sql`. Sve lekcije su na srpskom jeziku, javno dostupne i besplatne. Početni izgled platforme eŠkola veba prikazan je na slici 1.



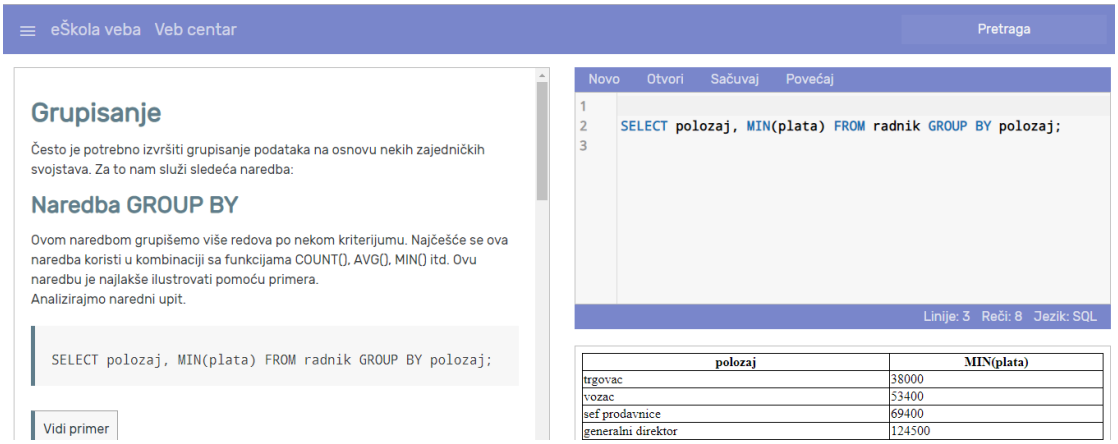
Slika 1: eŠkola veba

Kao što je prikazano na slici 1, na platformi se nalaze kursevi za učenje mnogih veb tehnologija. Među njima se nalazi i kurs posvećen sistemu MySQL, koji je napravljen za potrebe ovog rada. Izborom tog kursa otvara se njegova početka strana koja je prikazana na slici 2.



Slika 2: Naslovna strana kursa

Svaka elektronska lekcija se sastoji iz tri dela. Prvi deo je sadržaj te lekcije i nalazi se sa leve strane prozora. Drugi deo je prikazan u gornjem desnom uglu prozora gde se nalazi kôd primera koji se može menjati i izvršavati direktno na platformi. Treći deo je smešten u donjem desnom uglu prozora i u njemu se prikazuje rezultat izvršavanja primera. Izgled jedne lekcije je prikazan na slici 3.



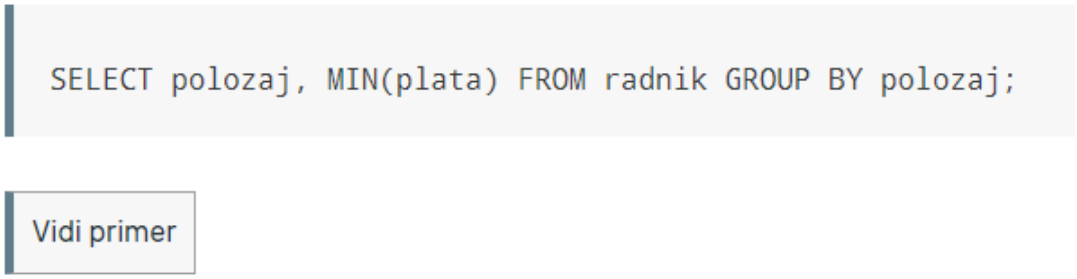
Slika 3: Izgled lekcije

U listi koja sledi navedene su elektronske lekcije koje su kreirane za potrebe ovog rada.

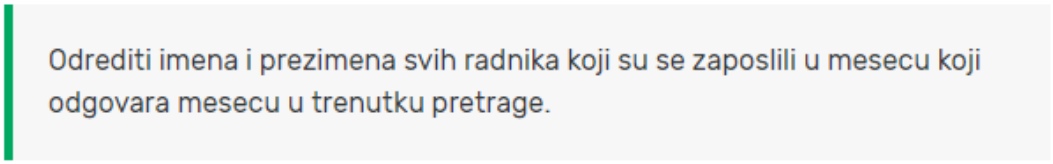
- Uvod i instalacija sistema MySQL;
- Komunikacija sa bazom podataka pomoću jezika PHP;
- Upiti za kreiranje baza i tabela;

- Upiti za izmenu strukture i brisanje tabela;
- Popunjavanje i pregled tabela;
- Pretraga tabele;
- Izmena i brisanje podataka u tabeli;
- Formatiranje izlaza;
- Ugrađene funkcije;
- Povezivanje tabela;
- Unije;
- Podupiti.

Za svaku lekciju navedene su definicije i opisi pojmova koji su potrebni za razumevanje konkretne teme. Teorijski pojmovi su ilustrovani i odgovarajućim primerima koji se izvršavaju direktno na platformi. Uz svaki primer nalazi se i tumačenje njegovog sadržaja. Pored definicija i primera, u svakoj lekciji se nalazi jedan ili više zadataka koje korisnik može samostalno da uradi kako bi proverio stečena znanja. Za svaki zadatak priloženo je i njegovo rešenje. Na slici 4 prikazan je izgled jednog primera, a izgled zadatka namenjenog korisnicima prikazan je na slici 5.



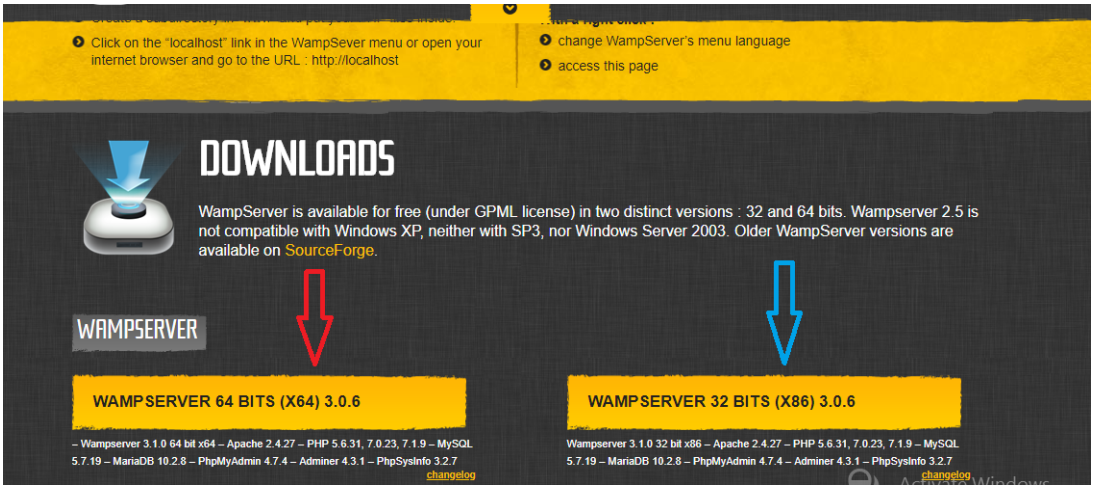
Slika 4: Prikaz primera kojim se ilustruje određeni upit



Slika 5: Zadatak namenjen korisnicima

2 Instalacija i upotreba sistema WAMP

Sistem MySQL ima linijski korisnički intefejs. Zbog toga se često koristi u okviru softverskih paketa WAMP i LAMP koji imaju grafički korisnički interfejs što korisniku olakšava upotrebu sistema MySQL. Softverski paketi WAMP i LAMP na korisničkom računaru simuliraju rad servera i namenjeni su za razvoj veb aplikacija. Pošto je *Windows* najrasprostranjeniji operativni sistem u Srbiji, a i šire, u radu i lekcijama je korišćen sistem MySQL u okviru softverskog paketa WAMP; softverski paket LAMP simulira rad servera na operativnim sistemima iz porodice UNIX. Instalacija se preuzima sa zvanične veb strane WAMP softverskog paketa `http://www.wampserver.com/en/`. Na navedenoj adresi treba posetiti odeljak za preuzimanje softvera *Downloads*, koji je prikazan na slici 6.



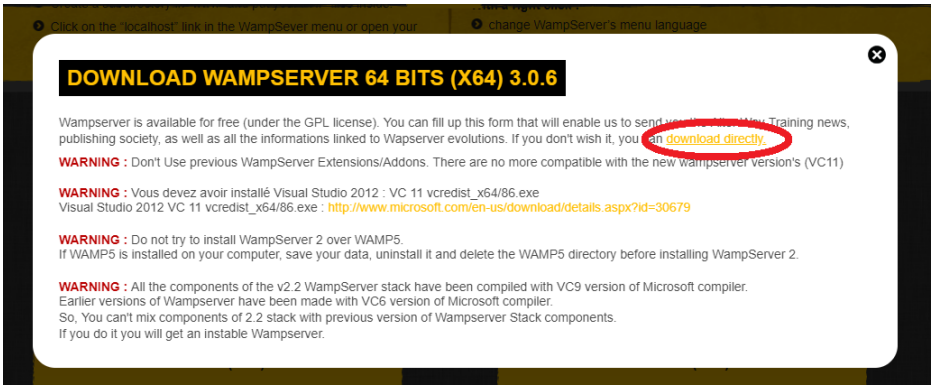
Slika 6: Izgled stranice za preuzimanje WAMP paketa

Potrebno je odabrati instalacioni fajl koji je najpogodniji računaru. Ukoliko je dužina procesorske reči računara 32 bita treba posetiti stranicu obeleženu plavom strelicom, a ako je dužina 64 bita treba kliknuti na karticu na koju upućuje crvena strelica.

Nakon odabira odgovarajuće verzije potrebno je početi preuzimanje instalacionog fajla. Kako najbrže početi proces preuzimanja prikazano je na slici 7.

Posle preuzimanja instalacionog fajla potrebno je pokrenuti proces instalacije koji nije komplikovan, samo treba ispratiti ponuđene korake. Po završetku instalacije, pokretanje WAMP paketa, a time i sistema MySQL, vrši se dvostrukim klikom na ikonicu na radnoj površini. Ako na radnoj površini nema ikonice WAMP paketa, pokretanje se može izvršiti praćenjem sledećih koraka *Start/All Programs/WampServer/Start WampServer*. Paket je pravilno pokrenut ako je WAMP ikonica, koja se nalazi na traci zadataka, u donjem desnom uglu ekrana, iz crvene promenila boju u zelenu.

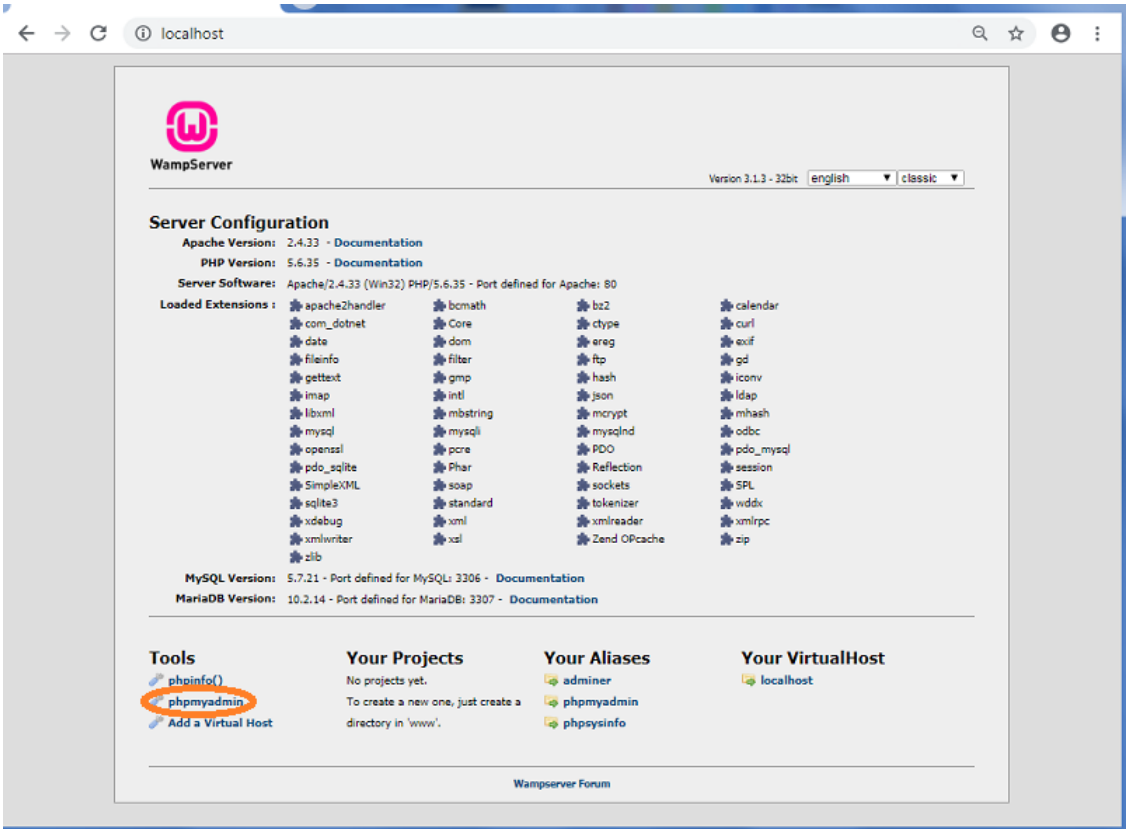
Sve datoteke i fajlove koji imaju veze sa sistemom MySQL i programskim jezikom PHP, čuvaju se u datoteci *www*. Ona se nalazi u datoteci *wamp* koja se automatski kreira prilikom instalacije WAMP paketa i nalazi se na putanji *C:\wamp*. Naravno, prilikom



Slika 7: Preuzimanje instalacionog fajla

procesa instalacije moguće je promeniti lokaciju pravljenja tih datoteka.

Šta je sve dostupno pokretanjem WAMP servera može se videti ako se iz bilo kog internet pregledača ode na adresu `http://localhost`. Posetom navedene stranice otvara se prozor koji je prikazan na slici 8.



Slika 8: Izgled stranice *localhost*

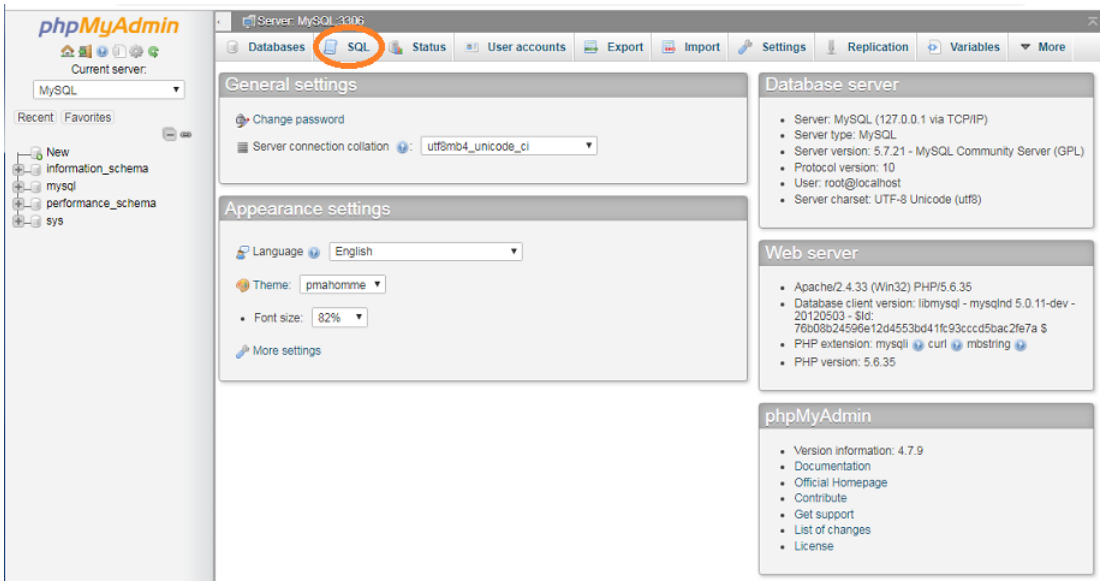
Ispod podnaslova *Your Projects* nabrajaju se nazivi svih projekata koji se nalaze u datoteci *www*. Svi programi koji koriste MySql baze podataka ili su pisani jezikom PHP izvršavaju se na serveru. Softver WAMP ima ulogu servera na personalnom računaru i sav sadržaj koji treba da bude na serveru se čuva u pomenutoj datoteci. Prilikom prve posete *localhost* strani neće biti nikakvih projekata i biće ispisana poruka kao na slici 8.

Najinteresantniji hipertekst na ovoj stranici je *phpmyadmin*. Klikom na njega, pregledač otvara aplikaciju *phpMyAdmin* namenjenu za upravljanje MySQL bazama podataka. Pomoću nje je moguće praviti nove baze podataka, kreirati njihove strukture, menjati sadržaj u njima i vršiti ostale relevantne akcije. Klikom na pomenuti hipertekst, ili unosom adrese `http://localhost/phpmyadmin` u polje za unos veb adrese u pregledaču, otvara se stranica kao na slici 9.

Slika 9: Pristup stranici *phpMyAdmin*

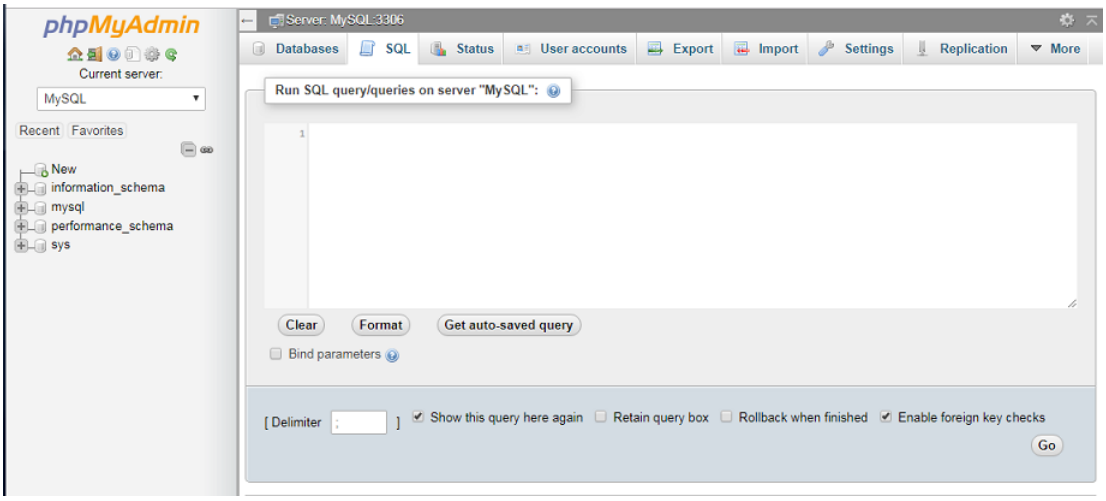
Kao što je prikazano na slici 9, potrebno je korisničko ime i lozinka za pristup aplikaciji *phpMyAdmin*. Ukoliko nije došlo do promene ustaljenih podešavanja prilikom instalacije WAMP paketa, korisničko ime za pristup aplikaciji je *root*, a lozinka je prazna niska, tj. polje za unos lozinke ostaje nepopunjeno. Nakon popunjavanja forme otvara se stranica aplikacije *phpMyAdmin*, koja je prikazana na slici 10.

U levom delu prozora nalazi se pregled kreiranih baza podataka. Četiri navedene baze se automatski kreiraju prilikom instalacije WAMP paketa i one nemaju veliki značaj za ovaj rad. Sada se dolazi do ključnog dela aplikacije *phpMyAdmin*. Izborom kartice *SQL* iz glavnog dela prozora, pojavljuje se opcija za pisanje upita. Upiti koji se pišu, odnose



Slika 10: Izgled stranice *phpMyAdmin*

se na pretragu baze podataka koja je selektovana u levom delu prozora. Izgled polja za pisanje upita prikazan je na slici 11.



Slika 11: Izgled kartice *SQL*

3 Komunikacija sa bazom podataka pomoću jezika PHP

Za svaku akciju koja se odnosi na bazu ili za podatke u njoj, potrebno je nekako proslediti upit serveru. Veza između korisnika i MySQL baze podataka ostvaruje se

posredovanjem nekog programskog jezika namenjenog za veb programiranje. Cilj ovih lekcija je da se početnici osposobe za izradu malo složenijih veb sajtova, a takvi sajtovi se ne mogu zamisliti bez neke baze podataka. U takvoj situaciji, najčešći izbor je da se komunikacija uspostavi pomoću programskog jezika PHP. U ovom poglavlju je dat opis funkcija *mysqli* biblioteke programskog jezika PHP. Ta biblioteka raspolaže svim funkcijama potrebnim za komunikaciju sa serverom na kom se nalazi MySQL baza podataka, tj. raspolaže funkcijama za povezivanje sa bazom podataka, za slanje upita, za čitanje upita itd. Sledi pregled najvažnijih funkcija iz *mysqli* biblioteke.

Funkcija *mysqli_connect*

Za uspostavljanje veze, odnostno konekcije sa serverom na kom se nalazi baza podataka, koristi se funkcija *mysqli_connect*.

Primer 1: Sintaksa funkcije *mysqli_connect*

```
<?php
    $konekcija = mysqli_connect($server, $username, $pass);
    ?>
```

Promenljiva *$konekcija* čuva podatke o uspostavljenoj vezi. Preko nje se ostvaruje sva dalja komunikacija sa serverom. Da bi se uspešno ostvarila veza sa serverom, funkciji *mysqli_connect* je potrebno proslediti tri argumenta. Prvi argument je niska koja predstavlja putanju do servera. Drugi argument je korisničko ime za pristup serveru, a treći argument je lozinka.

Za povezivanje sa WAMP serverom koji je instaliran na korisnikovom računaru, funkciji *mysqli_connect* se prosleđuju sledeći argumenti. Putanja do servera je *localhost*, korisničko ime je *root*, dok je lozinka prazna niska. Isti parametri kao kada se pristupa *phpMyAdmin* aplikaciji. Opisano povezivanje je prikazano primerom 2.

Primer 2: Povezivanje sa WAMP serverom

```
<?php
    $konekcija = mysqli_connect('localhost', 'root', '');
    ?>
```

Nakon svakog pokušaja povezivanja sa serverom potrebno je proveriti uspešnost veze. Moguće je da su funkciji za uspostavljanje veze prosleđeni pogrešni argumeniti ili da konekcija nije uspela iz nekog drugog razloga. Funkcija *mysqli_errno* vrši proveru konekcije. Prosleđuje joj se jedan argument, a to je vrednost koju je vratila funkcija *mysqli_connect*. U ovom slučaju to je vrednost koja se čuva u promenljivoj *$konekcija*. Kako se vrši provera uspešnosti veze sa serverom prikazano je primerom 3.

Primer 3: Provera konekcije

```
<?php
    if(mysqli_errno($konekcija))
      die('Neuspešna konekcija');
    ?>
```

Prikazano je kako se uspostavlja veza sa serverom, ali da bi se vršila pretraga ili izmene podataka u nekoj bazi, potrebno je ostvariti vezu sa tačno tom bazom. I to se može ostvariti funkcijom *mysqli_connect*. U odnosu na povezivanje sa serverom, jedina razlika je u tome što se funkciji prosleđuje i četvrti argument - niska koja predstavlja ime baze. Ako se na WAMP serveru nalazi baza podataka pod nazivom *moja_baza*, jedan od načina kako se vrši direktno povezivanje sa njom prikazan je primerom 4.

Primer 4: Veza sa bazom

```
<?php
    $konekcija = mysqli_connect('localhost', 'root', '', 'moja_baza');
    ?>
```

Kao što je izvršena provera uspostavljanja veze sa serverom, tako se mora proveriti i uspešnost povezivanja sa bazom podataka. Provera se izvodi identično u oba slučaja i ilustrovana je primerom 3.

Funkcija *mysqli_query*

Za prosleđivanje upita bazi podataka koristi se funkcija *mysqli_query*.

Primer 5: Sintaksa funkcije *mysqli_query*

```
<?php
    $rezultat = mysqli_query($konekcija, $upit);
    ?>
```

Funkciji se prosleđuju dva argumenta. Prvi argument predstavlja vezu sa bazom podataka, drugi argument je niska sa upitom koji je potrebno izvršiti. Povratna vrednost funkcije, smeštena u promenljivoj *$rezultat*, je objekat sa rezultatima upita ili *FALSE* ako upit nije uspešno izvršen.

Kako ne bi došlo do neželjenih grešaka, nakon slanja upita bazi, potrebno je proveriti uspešnost izvršavanja upita. Najjednostavnije je proveriti povratnu vrednost funkcije *mysqli_query*. Upit je uspešno izvršen ako je ona vratila vrednost različitu od *FALSE*. Primerom 6 je prikazano prosleđivanje upita bazi i provera o uspešnosti njegovog izvršavanja. Ako upit nije realizovan, prekida se izvršavanje programa i korisniku se ispisuje obaveštenje o tome.

Primer 6: Provera uspešnosti izvršavanja upita

```
<?php
    $rezultat = mysqli_query($konekcija, $upit);
    if(!$rezultat)
      die('Neuspešno izvršen upit');
    ?>
```

Funkcija *mysqli_num_rows*

Podaci o uspešno izvršenom upitu za pretragu baze se čuvaju u promenljivoj *$rezultat*, tj. povratna vrednost funkcije *mysqli_query* se čuva u toj promenljivoj. Pomoću

funkcije *mysqli_num_rows* se određuje koliko rezultata je dohvatio upit.

Primer 7: Sintaksa funkcije *mysqli_num_rows*

```
<?php
    $n = mysqli_num_rows($rezultat)
    ?>
```

Funkciji se prosleđuje jedan argument sa podacima o realizaciji upita. Povratna vrednost funkcije je ceo broj koji predstavlja broj redova dohvaćenih upitom, odnosno broj rezultata koji zadovoljavaju kriterijume navedene u upitu.

Funkcija *mysqli_num_rows* najčešće se koristi za proveru da li je ispravno izvršen upit dohvatio neki rezultat. Nije retka situacija da se upit uspešno izvrši, ali da ni jedan podatak u bazi ne ispunjava uslove navedene u upitu. Zbog toga se, nakon realizacije upita, a pre pokušaja ispisivanja rezultata upita, vrši provera da li je upit dohvatio neke podatke. Kako se vrši ta provera, može se videti u primeru 8.

Primer 8: Provera da li je upit dohvatio neki red

```
<?php
    $rezultat = mysqli_query($konekcija, $upit);
    if(mysqli_num_rows($rezultat))
      die('Nije pronađen traženi podatak');
    ?>
```

Funkcija *mysqli_fetch_assoc*

Ako je upit uspešno izvršen i ustavnoljeno je da je dohvaćen određen broj redova, treba ispisati rezultat izvršavanja upita. Jedan od najlakših načina ispisa dohvaćenih redova je pomoću funkcije *mysqli_fetch_assoc*.

Primer 9: Sintaksa funkcije *mysqli_fetch_assoc*

```
<?php
    $red = mysqli_fetch_assoc($rezultat);
    ?>
```

Funkciji se prosleđuje jedan argument sa podacima o izvršavanju upita. Ti podaci su povratna vrednost funkcije *mysqli_query*. Za prosleđeni argument, funkcija *mysqli_fetch_assoc* vraća jedan po jedan red iz baze koji zadovoljava uslove upita. Nakon poslednjeg dohvaćenog reda, funkcija vraća vrednost *FALSE*.

Prilikom ispisa podataka dohvaćenih upitom, obično se funkcija *mysqli_fetch_assoc* navodi kao uslov *while* petlje. Na taj način postiže se ispisivanje svih dohvaćenih redova. Malo detaljniji opis procesa ispisivanja rezultata upita sledi nakon primera 10.

Primer 10: Ispis redova dohvaćenih upitom

```
<?php
    while($red = mysqli_fetch_assoc($rezultat))
    {
      echo $red['ime_kolone_1'];
```

```
5       echo $red['ime_kolone_2'];
6         .
7         .
8         .
9       echo $red['ime_kolone_n'];
10    }
11    ?>
```

Jedan rezultat dohvaćen upitom je zapravo jedan red iz tabele. U tom redu ima onoliko podataka koliko ta tabela ima kolona. U promenljivoj *$red* nalazi se jedan dohvaćeni red, predstavljen asocijativnim nizom indeksiranim nazivima kolona tabele. Nakon ispisivanja željenih podataka iz jednog reda, u promenljivu *$red* se smešta sledeći dohvaćeni red i tako dalje dok ima redova koji odgovaraju uslovima upita. Tada se vrednost *FALSE* dodeljuje promenljivoj *$red* i izlazi se iz *while* petlje [5].

Funkcija *mysqli_close*

Ako je potrebno komunicirati sa nekom bazom podataka, treba uspostaviti vezu sa serverom na kom se ona nalazi. Pored toga, kada prestane potreba za vezom sa serverom, potrebno je i prekinuti uspostavljenu konekciju sa serverom ili bazom podataka. Prekid uspostavljene veze se ostvaruje funkcijom *mysqli_close*, koja prima jedan argument, a to je promenljiva koja čuva podatke o uspostavljanju veze, odnosno ona promenljiva u kojoj je smeštena povratna vrednost funkcije *mysqli_connect*.

Primer 11: Sintaksa funkcije *mysqli_close*

```
1 <?php
2     mysqli_close($konekcija);
3     ?>
```

U ovom poglavlju su nabrojane samo one funkcije koje se najčešće koriste, deteljan pregled *mysqli* biblioteke dostupan je na zvaničnom sajtu programskog jezika PHP [2].

4 Upiti za kreiranje baza i tabela

U prethodnom poglavlju je opisan način povezivanja korisnika sa serverom na kom se nalazi baza podataka. Na početku novog projekta treba kreirati bazu namenjenu tom projektu. Bazu podataka čini skup tabela koje su međusobno povezane, dakle, treba napraviti i te tabele. U ovom poglavlju su prikazani upiti za pravljenje i brisanje baza podataka i upiti za kreiranje i brisanje tabela unutar baze.

Upiti za kreiranje i brisanje baze podataka

Nova baza kreira se jednostavnim upitom koji je prikazan primerom 12.

Primer 12: Upit za kreiranje nove baze

```
1 CREATE DATABASE ime_baze;
```

Naredba za pravljenje nove baze podataka je *CREATE DATABASE*. Iza nje je potrebno navesti željeni naziv baze koja se kreira.

Upit za brisanje postojeće baze podataka je veoma sličan upitu za njeno kreiranje, jedina razlika je u tome što se naredba *CREATE DATABASE* zamenjuje naredbom *DROP DATABASE*. Naravno, iza ključnih reči naredbe se navodi ime baze koja se briše. Jedan upit za brisanje baze podataka se nalazi u primeru 13.

Primer 13: Upit za brisanje baze podataka

```
DROP DATABASE ime_baze;
```

Na ovom mestu je zgodno reći nekoliko reči o sintaksnim pravilima koja se moraju poštovati prilikom pisanja upita. Svaki upit se mora završiti simbolom tačka-zapeta (*;*). Ključne reči se mogu pisati i malim i velikim slovima, svejedno je da li piše *CREATE DATABASE* ili *create database*. Situacija se komplikuje kada je reč o imenima baza, tabela i kolona u tabelama. Ako je sistem MySQL instaliran na *Windows* operativnom sistemu, neće se praviti razlika između malih i velikih slova, ali to pravilo ne važi ako se MySQL nalazi na nekom operativnom sistemu iz porodice *Unix*. Da ne bi došlo da zabune i neželjenih problema, preporuka je da se svi nazivi baza, tabela itd. pišu kao da nema razlike između malih i velikih slova. Maksimalno izbegavati da se različitim tabelama daju imena poput *Radnik* i *radnik*. Nazivi baza podataka, tabela i kolona u tabelama mogu sadržati sve karaktere osim znakova navoda, tačke i kosih crta.

Upiti za kreiranje i brisanje tabela

Kada se napravi baza podataka, potrebno je kreirati tabele unutar nje. Upiti za pravljenje tabela su malo složeniji od upita za kreiranje baza podataka.

Primer 14: Sintaksa upita za kreiranje tabele

```
CREATE TABLE ime_tabele(
    kolona1_ime tip_podatka,
    kolona2_ime tip_podatka,
    .
    .
    .
    ime_koloneN tip_podatka
);
```

Naredba za pravljenje nove tabele je *CREATE TABLE*, iza nje se navodi ime tabele, pa se u zagradi nabraju imena kolona tabele i tipovi podataka koji se nalaze u kolonama. Na taj način se deklariše svaka kolona u tabeli. Na slici 12 mogu se videti tipovi podataka koji se najčešće koriste i koja su im značenja. Detaljan opis svih tipova podataka može se pronaći u [3].

Iza tipa podatka može se u zagradi navesti broj koji predstavlja maksimalnu veličinu podatka, npr. *VARCHAR(30)* označava da je podatak tekstualnog tipa i da ima najviše 30 karaktera.

Svaka tabela treba da ima primarni ključ. Jedna kolona, ili grupa kolona, koja jednoznačno identifikuje red u tabeli, naziva se primarni ključ tabele. Ako je primarni

Tip	Opis
INT	celi brojevi
DOUBLE	decimalni brojevi
DECIMAL	decimalni brojevi
CHAR	tekst fiksne dužine
VARCHAR	tekst promenljive dužine
TEXT	tekst
DATE	datum (YYYY-MM-DD)
TIME	vreme (HH:MM:SS)
DATETIME	datum i vreme (YYYY-MM-DD HH:MM:SS)
YEAR	godina (YYYY ili YY)

Slika 12: Tipovi podataka

ključ jedna kolona, podatak u toj koloni ne sme biti isti za dva ili više redova tabele. Ako je primarni ključ skup više kolona, onda podaci u tim kolonama moraju biti različiti za svaki red u tabeli. U upitu za kreiranje tabele treba definisati koje kolone su primarni ključ tako što se nakon nabrajanja svih imena kolona i tipova podataka u njima dodaje naredba *PRIMARY KEY()* i u zagradi se navode imena kolona koje su primarni ključ. Ovo nije jedini način za određivanje primarnog ključa, može se odrediti i tako što bi se prilikom definisanja kolona koje su primarni ključ, nakon imena i tipa podatka, dodala naredba *PRIMARY KEY*.

Upitom iz primera 15 kreira se tabela **osoba**.

Primer 15: Kreiranje tabele **osoba**

```
CREATE TABLE osoba(
    jmbg INT(13),
    ime VARCHAR(20),
    prezime TEXT(50),
    godine INT,
    PRIMARY KEY(jmbg)
);
```

Analizom upita se zaključuje da tabela **osoba** ima četiri kolone. Prva kolona čuva celobrojne podatke koji predstavljaju jedinstveni matični broj građana (JMBG) i ima maksimalnu dužinu od 13 karaktera. Ime i prezime osobe se skladište u drugoj i trećoj koloni koje su tekstualnog tipa, a u četvrtoj koloni se nalazi broj godina osobe. Kako je

JMBG jedinstven za svakog građanina, logično je izabrati tu kolonu za primarni ključ tabele.

Za svaku kolonu tabele se definiše tip podatka koji prihvata. Pored toga, moguće je još više opisati kolonu. Prilikom pisanja upita za kreiranje nove tabele, iza imena kolone i tipa podatka koji prihvata, mogu se navesti ključne reči *NOT NULL*, *DEFAULT* ili *AUTO_INCREMENT*.
Ako polja određene kolone ne smeju da ostanu nepopunjena, pri njenoj deklaraciji navode se rezervisane reči *NOT NULL*.
Često je potrebno upisati podrazumenavanu vrednost u polje neke kolone ako korisnik nije ništa uneo, tada se pri deklaraciji te kolone navodi rezervisana reč *DEFAULT* i odmah iza nje ta podrazumevana vrednost.
Ako pri prvom unosu vrednosti u tabelu, nekom polju određene kolone treba upisati broj 1, a pri svakom sledećem unosu upisivati broj za jedan veći od prethodnog, pri deklaraciji te kolone navode se rezervisane reči *AUTO_INCREMENT*.

Primer 16: Kreiranje tabele **clan_kluba**

```
CREATE TABLE clan_kluba(
    ime VARCHAR(20) NOT NULL,
    prezime TEXT(50) NOT NULL,
    br_karte INT AUTO_INCREMENT PRIMARY KEY,
    godine INT,
    grad VARCHAR(30) DEFAULT 'Beograd'
);
```

Upitom iz primera 16 kreira se tabela **clan_kluba**. Tabela sadrži kolone koje čuvaju ime, prezime, broj članske karte, godine i grad iz kog je član kluba.
Polje u koloni *ime* čuva tekstualni podatak koji ima najviše 20 karaktera i to polje ne sme ostati prazno.
Polje u koloni *prezime* čuva tekstualni podatak koji može imati najviše 50 karaktera i to polje mora biti popunjeno.
Polje u koloni *br_karte* čuva ceo broj, koji se svakim novim dodatim članom povećava za jedan. Kako svaki član kluba ima različit broj članske karte, kolona *br_karte* predstavlja primarni ključ.
Polje u koloni *godine* čuva celobrojni podatak.
Polje u koloni *grad* čuva tekstualni podatak koji može imati najviše 30 karaktera i ako se prilikom popunjavanja tabele ne navede grad novog člana kluba, u tabelu se upisuje podrazumevana vrednost „Beograd“.

5 Upiti za izmenu strukture i brisanje tabele

Neretko se dešava da postojeću tabelu treba modifikovati; dodati ili odbaciti neku kolonu, promeniti tip podatka koji kolona prihvata i sl. Da se ne bi kreirana tabela brisala, pa pravila nova skoro ista kao izbrisana, moguće je vršiti određene promene na već napravljenoj tabeli. Naredba za izmenu strukture tabele je *ALTER TABLE*.

U svim primerima u ovom poglavlju prikazuju se promene na tabeli **osoba**. Upit za njeno kreiranje je dat u prethodnom poglavlju u primeru 15.

Promena imena tabele

Primer 17: Sintaksa upita za promenu imena tabele

```
ALTER TABLE staro_ime_tabele RENAME novo_ime_tabele;
```

Upit kojim se ime tabele **osoba** menja u **gradjanin** je dat narednim primerom.

Primer 18: Upit za promenu imena tabele

```
ALTER TABLE osoba RENAME gradjanin;
```

Promena tipa podatka kolone

Primer 19: Sintaksa upita za promenu tipa podatka kolone

```
ALTER TABLE ime_tabele MODIFY COLUMN ime_kolone nov_tip_podatka;
```

Upitom iz primera 20 menja se tip podatka kolone *godine* tabele **osoba**. Iz tipa *INT* menja se u tip *YEAR*.

Primer 20: Promena tipa podatka kolone

```
ALTER TABLE osoba MODIFY COLUMN godine YEAR;
```

Promena imena kolone i tipa podatka u njoj

Primer 21: Sintaksa upita za promenu imena kolone i tipa podatka u njoj

```
ALTER TABLE ime_tabele CHANGE COLUMN staro_ime novo_ime nov_tip_podatka;
```

Moguće je samo promeniti ime kolone bez promene tipa podatka u njoj. To se realizuje tako što se u upitu iza navođenja starog i novog imena kolone, umesto novog tipa podatka zapisuje postojeći tip.

Upitom iz primera 22 se kolona *godine* tabele **osoba** samo preimenuje u *starost*, a upitom iz primera 23 se toj koloni menja ime u *godiste* i menja joj se tip u *YEAR*.

Primer 22: Promena imena kolone

```
ALTER TABLE osoba CHANGE COLUMN godine starost INT;
```

Primer 23: Promena imena kolone i njenog tipa podatka

```
ALTER TABLE osoba CHANGE COLUMN godine godiste YEAR;
```

Brisanje kolone iz tabele

Primer 24: Sintaksa upita za brisanje kolone iz tabele

```
ALTER TABLE ime_tabele DROP ime_kolone;
```

Upitom iz primera 25 briše se kolona *godine* iz tabele **osoba**.

Primer 25: Brisanje kolone

```
ALTER TABLE osoba DROP godine;
```

Dodavanje kolone u tabelu

Primer 26: Sintaksa upita za dodavanje nove kolone

```
ALTER TABLE ime_tabele ADD ime_nove_kolone tip_podatke_nove_kolone;
```

Primerom 27 je prikazan upit kojim se tabeli **osoba** dodaje kolona *nadimak*.

Primer 27: Dodavanje kolone

```
ALTER TABLE osoba ADD nadimak VARCHAR(20);
```

Brisanje tabele

Primer 28: Sintaksa upita za brisanje tabele

```
DROP TABLE ime_tabele;
```

Ako je prestala potreba za nekom tabelom iz baze podataka, treba je izbrisati da ne bi zauzimala memorijski prostor na serveru.

Upitom iz pirmera 29 briše se tabela **osoba**.

Primer 29: Brisanje tabele

```
DROP TABLE osoba;
```

6 Upiti za pregled tabele i unos podataka

Prednost platforme eŠkola veba je u tome što su elektronske lekcije na njoj interaktivne. Prilikom prolaska kroz lekcije o sistemu MySQL, korisnik može pisati upite i odmah videti rezultat njihovog izvršavanja. Na platformi je napravljeno nekoliko tabela u okviru jedne test baze podataka. Korisnik može vežbati pisanje upita za pretragu te baze. Treba napomenuti da se pri svakom učitavanju nove lekcije otklanjaju sve promene u bazi koje je korisnik napravio. Test baza je uvek ista, bez obzira na to da li ju je korisnik menjao u prethodnim lekcijama.

Jedna od tabela u test bazi, na kojoj se izvršavaju upiti, je tabela **radnik**. Struktura te tabele može se videti u primeru 30, gde je prikazan upit kojim je ona kreirana.

Primer 30: Upit za kreiranje tabele **radnik**

```
CREATE TABLE radnik (
    id_radnika INT NOT NULL AUTO_INCREMENT,
    ime VARCHAR(20) NOT NULL,
    prezime VARCHAR(50) NOT NULL,
    plata INT NOT NULL,
    polozaj VARCHAR(30) NOT NULL,
    radno_mesto VARCHAR(30) NOT NULL,
    datum_zaposlenja DATE NULL,
    telefon VARCHAR(15) NULL,
    PRIMARY KEY (id_radnika)
);
```

Upiti za pregled sadržaja tabele

Nakon upita za kreiranje baze i tabela, na red su došli upiti kojima se može videti sadržaj tabele i vršiti pretraga podataka u njima.

Kako videti ceo sadržaj tabele **radnik** prikazano je primerom 31. U primeru se nalazi upit za izlistavanje sadržaja tabele. Rezultat izvršavanja ovog upita prikazan je na slici 13.

Primer 31: Upit za izlistavanje sadržaja tabele **radnik**

```
SELECT * FROM radnik;
```

Naredba za pregled i pretragu sadržaja tabela je *SELECT*. Sintaksa pisanja upita sa naredbom *SELECT* prikazana je primerom 32.

Primer 32: Sintaksa naredbe *SELECT*

```
SELECT ime_kolone1, ime_kolone2, ..., ime_koloneN FROM ime_tabele;
```

Upit je prilično intuitivan. Iza naredbe *SELECT* navode se imena željenih kolona, pa se iza rezervisane reči *FROM* navodi ime tabele čiji sadržaj se pregleda. Ako je potrebno pregledati celu tabelu, pisanje imena svih njenih kolona se može zameniti stavljanjem zvezdice (*) kao što je urađeno u primeru 31.

Upit kojim se dobijaju samo imena i prezimena osoba iz tabele *radnik* dobija se upitom koji je naveden u primeru 33. Zbog veličine rezultata, na slici 14, prikazan je deo tabele koji se dobije izvršavanjem upita iz primera 33.

Primer 33: Upit za dobijanje imena i prezimena radnika

```
SELECT ime, prezime FROM radnik;
```

id_radnika	ime	prezime	plata	polozaj	radno_mesto	datum_zaposlenja	telefon
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914
2	Pavle	Spasic	41300	trgovac	Nis	2017-04-25	0187215302
3	Vojin	Markovic	53400	vozac	Novi Sad	2017-12-08	0213068124
4	Jovan	Petovic	41200	trgovac	Novi Sad	2018-04-06	0214580312
5	Marko	Pavic	71200	sef prodavnice	Beograd	2016-04-23	0117210034
6	Milos	Ilic	39200	trgovac	Novi Sad	2017-07-25	0216396452
7	Milica	Jovic	69400	sef prodavnice	Nis	2018-01-13	0184522019
8	Paja	Maric	38900	trgovac	Novi Sad	2018-04-11	02132008496
9	Vuk	Petrovic	59100	vozac	Beograd	2017-08-28	0113987589
10	Mirko	Vojinovic	124500	generalni direktor	Beograd	2008-11-23	0116369369
11	Marija	Jovovic	42100	trgovac	Nis	2018-01-03	01819674258
12	Jovana	Urosevic	39600	trgovac	Nis	2017-08-02	0189246381
13	Veselin	Markovic	41400	trgovac	Beograd	2018-03-29	0114215367
14	Stefan	Popovic	72600	sef prodavnice	Novi Sad	2017-11-27	0216342003
15	Lazar	Tosic	38800	trgovac	Beograd	*NULL*	0110314251
16	Filip	Tadic	67100	vozac	Nis	2017-07-08	0186311421
17	Ana	Spasic	42100	trgovac	Beograd	2018-06-06	0114258998
18	Uros	Pavic	38000	trgovac	Novi Sad	2018-02-20	*NULL*
19	Tomislav	Popadic	73400	vozac	Beograd	2016-12-13	0113103520
20	Svetlana	Savic	40300	trgovac	Beograd	2017-08-26	0118462153
21	Mira	Ostojic	38300	trgovac	Nis	2018-06-20	0184012403
22	Mina	Antic	38800	trgovac	Beograd	2017-08-03	0116942015

Slika 13: Sadržaj tabele **radnik**

ime	prezime
Ivona	Jankovic
Pavle	Spasic
Vojin	Markovic
Jovan	Petovic
Marko	Pavic
Milos	Ilic
Milica	Jovic
Paja	Maric

Slika 14: Deo rezultata dobijenih upitom iz primera 33

Upiti za dodavanje podataka u tabelu

Narednih nekoliko primera prikazuje upite za dodavanje podataka u tabelu neke baze. Dodavanje sadržaja u tabelu predstavlja dodavanje jednog novog reda. Naredba za dodavanje novog reda u tabelu je *INSERT INTO*.

Primer 34: Sintaksa naredbe *INSERT INTO*

```
INSERT INTO ime_tabele (ime_kolone1, ime_kolone2, ..., ime_koloneN)
VALUES (podatak1, podatak2, ..., podatakN);
```

Nakon naredbe za dodavnje novog reda navodi se ime tabele u koju se dodaju podaci, potom se u zagradi navode imena kolona u koje se smeštaju podaci. Zatim sledi rezervisana reč *VALUES*, pa se nakon nje, u zagradi, navode podaci koji se unose u tabelu. Podatak koji je prvi napisan se smešta u kolonu čije je ime navedeno prvo, drugi podatak u drugu navedenu kolonu itd. Ne mora se uneti podatak u svaku kolonu tabele, pa je zato potrebno navoditi imena kolona u koje će se upisati podaci.

Dodavanje nove osobe u tabelu **radnik** je prikazano primerom 35.

Primer 35: Dodavanje novog radnika

```
INSERT INTO radnik (ime, prezime, plata, polozaj, radno_mesto,
                                datum_zaposlenja, telefon)
VALUES ('Petar', 'Petrovic', 40000, 'trgovac', 'Beograd',
                                '2017-05-10', '0119876543');
```

Ovim upitom se u tabelu **radnik** dodaje Petar Petrovic, trgovac iz Beograda. U upitu nije navedena kolona *id_radnika*. U polje te kolone automatski se upisuje broj za jedan veći od broja iz prethodnog reda tabele (pogledati upit iz primera 30 kojim se kreira tabela **radnik**). Koloni koja čuva tekstualni tip podatka se vrednost za upisivanje mora proslediti pod znacima navoda, jer se tako zapisuju niske. Potrebno je obratiti pažnju i na zapis datuma. Koloni koja prihvata podatke tipa *DATE* se vrednost za upis mora proslediti u obliku YYYY-MM-DD. Prve četiri cifre predstavljaju godinu, sledeći par cifara je mesec, a poslednje dve cifre su dan; npr. 13. maj 2016. godine se u traženom formatu zapisuje na sledeći način: 2016-05-13.

Moguće je jednim upitom dodati više redova u tabelu. Takav upit je gotovo identičan upitu za dodavanje jednog reda. Samo je potrebno nabrojati podatke za svaki novi red tabele. Podaci za jedan red pišu se u zagradi, a zagrade se razdvajaju zapetama. Upit kojim se dodaju tri nova radnika u tabelu **radnik** može se videti u primeru 36.

Primer 36: Dodavanje novih radnika

```
INSERT INTO radnik (ime, prezime, plata, polozaj, radno_mesto,
                                datum_zaposlenja, telefon)
VALUES ('Marko', 'Markovic', 40000, 'trgovac', 'Beograd',
                                '2017-05-10', '0119876544'),
       ('Pera', 'Peric', 40000, 'trgovac', 'Beograd',
                                '2017-05-10', '0118876544'),
       ('Jova', 'Jovic', 40000, 'trgovac', 'Beograd',
                                '2017-05-10', '0119876554');
```

7 Upiti za uslovnu pretragu tabela

Retko kada je potrebno prikazati sadržaj cele tabele iz baze. Uglavnom je potrebno odrediti redove tabele koji ispunjavaju neki uslov. Za postavljanje uslova pretrage ili kriterijuma pretrage koristi se naredba *WHERE*.

Primer 37: Sintaksa uslovnog upita

```
SELECT ime_kolone1, ..., ime_koloneN
FROM ime_tabele
WHERE uslov;
```

Uslov iza naredbe *WHERE* je neki izraz u kom se koristi relacioni operator koji vraća logičku vrednost *TRUE* ili *FALSE*. Relacioni operatori koji se mogu koristiti prikazani su u tabeli na slici 15. Pored svakog operatora je navedeno i njegovo značenje.

Operator	Značenje
=	jednako
>	veće od
<	manje od
>=	veće ili jednako
<=	manje ili jednako
!= ili <>	različito
IS NULL i IS NOT NULL	da li je vrednost NULL ili nije

Slika 15: Relacioni operatori

Kako napisati upit kojim se pronalaze radnici koji imaju platu veću od 50000? Treba napisati dobar uslov pretrage, a to je da je vrednost u koloni *plata* veća od 50000. Taj upit se nalazi u primeru 38, a rezultat izvršavanja upita prikazan je na slici 16.

Primer 38: Uslovni upit

```
SELECT ime, prezime, plata
FROM radnik
WHERE plata > 50000;
```

Moguće je pisati uslove i za kolone koje čuvaju tekstualne podatke. Upit kojim se traže svi vozači je napisan u primeru 39. Rezultat izvršavanja i ovog upita prikazan je na slici 16.

Primer 39: Uslovni upit; vozači

```
SELECT ime, prezime, polozaj
FROM radnik
WHERE polozaj = 'vozac';
```

ime	prezime	plata
Vojin	Markovic	53400
Marko	Pavic	71200
Milica	Jovic	69400
Vuk	Petrovic	59100
Mirko	Vojinovic	124500
Stefan	Popovic	72600
Filip	Tadic	67100
Tomislav	Popadic	73400

ime	prezime	polozaj
Vojin	Markovic	vozac
Vuk	Petrovic	vozac
Filip	Tadic	vozac
Tomislav	Popadic	vozac

Slika 16: Rezultat upita iz primera 38 (levo) i rezultat upita iz primera 39 (desno)

Složeni uslovi

Dosadašnji upiti su pronalazili redove iz tabele koji ispunjavaju samo jedan uslov. Moguće je u jednom upitu kombinovati više uslova. Za povezivanje uslova koriste se logički operatori *AND* (konjukcija), *OR* (disjunkcija) i *NOT* (negacija). Prilikom izračunavanja najveći prioritet ima negacija, zatim sledi konjukcija, dok je disjunkcija najslabijeg prioriteta. Naravno, na prioritet izvršavanja operatora može se uticati postavljanjem zagrada.

Složeni uslovi se koriste kada je potrebno rešite probleme kao što je određivanje radnika koji imaju platu veću od 40000 i rade kao trgovci. Upit kojim se rešava ovakav problem može se videti u primeru 40.

Primer 40: Složeni uslovni upit; trgovac

```
SELECT ime, prezime, polozaj, plata
FROM radnik
WHERE polozaj = 'trgovac'
                AND plata > 40000;
```

Malo kompleksniji primer složenog upita je prikazan primerom 41. Kao i u prethodnom slučaju potrebno je odrediti sve trgovce sa platom većom od 40000, ali da im je radno mesto u Nišu ili Novom Sadu. Rezultat izvršavanja upita iz primera 40 prikazan je na slici 17, a rezultet izvršavanja upita iz primera 41 prikazan je na slici 18.

Primer 41: Složeni uslovni upit; trgovac iz Niša ili Novog Sada

```
SELECT ime, prezime, polozaj, plata, radno_mesto
FROM radnik
WHERE polozaj = 'trgovac'
                    AND plata > 40000
                    AND (radno_mesto = 'Nis' OR radno_mesto = 'Novi Sad');
```

U ovom upitu je potrebno promeniti prioritet izvršavanja operatora. Da nisu stavljene zagrade upit bi vratio sve trgovce sa platom većom od 40000 iz Niša i sve radnike iz Novog Sada.

ime	prezime	polozaj	plata
Pavle	Spasic	trgovac	41300
Jovan	Petovic	trgovac	41200
Marija	Jovovic	trgovac	42100
Veselin	Markovic	trgovac	41400
Ana	Spasic	trgovac	42100
Svetlana	Savic	trgovac	40300

Slika 17: Rezultat izvršavanja upita iz primera 40

ime	prezime	polozaj	plata	radno_mesto
Pavle	Spasic	trgovac	41300	Nis
Jovan	Petovic	trgovac	41200	Novi Sad
Marija	Jovovic	trgovac	42100	Nis

Slika 18: Rezultat izvršavanja upita iz primera 41

Operatori *IN* i *BETWEEN*

Prilikom pisanja upita sa složenim uslovima korisno je koristiti operatore *IN* i *BETWEEN* jer znatno skraćuju zapis uslova, a i upit napisan pomoću njih je dosta razumljiviji za čitanje. Operator *IN* označava da se vrednost u koloni, koja je navedena sa njegove leve strane, mora nalaziti u skupu napisanom sa njegove desne strane. Operator *BETWEEN* označava da se vrednost u koloni navedenoj sa njegove leve strane mora nalaziti između graničnih vrednosti (uključujući i granične vrednosti) koje su napisane sa njegove desne strane.

Pomoću ovih operatora se lako i kompaktno može napisati upit za određivanje svih radnika koji rade u Nišu ili Novom Sadu, a imaju platu između 40000 i 60000. Traženi upit se nalazi u primeru 42.

Primer 42: Složeni uslovi upit napisan pomoću operatora *IN* i *BETWEEN*

```
SELECT ime, prezime, plata, radno_mesto
FROM radnik
WHERE radno_mesto IN ('Nis', 'Novi Sad')
            AND plata BETWEEN 40000 AND 60000;
```

U primeru 43 predstavljen je upit kojim se rešava isti problem samo je napisan bez korišćenja operatora *IN* i *BETWEEN*.

Primer 43: Složeni uslovi upit bez upotrebe operatora *IN* i *BETWEEN*

```
SELECT ime, prezime, plata, radno_mesto
FROM radnik
WHERE (radno_mesto = 'Nis' OR radno_mesto = 'Novi Sad')
            AND plata >= 40000
            AND plata <= 60000
```

Upoređujući poslednja dva primera vidi se koliko korišćenje ova dva operatora olakšava pisanje i čitanje upita.

Primer 44 je odlična ilustracija upita za rešavanje problema sledećeg tipa. Potrebno je odrediti sve redove neke tabele tako da vrednost u nekoj koloni ne pripada određenom skupu. Tada je zgodno koristiti negaciju (*NOT*) i operator *IN*. Konkretno, potrebno je odrediti sve radnike koji nisu trgovci ni vozači.

Rezultat izvršavanja upita iz primera 42 i 44 prikazan je na slici 19.

Primer 44: Složeni uslovni upit sa upotrebom operatora *NOT*

```
SELECT ime, prezime, polozaj
FROM radnik
WHERE polozaj NOT IN ('trgovac', 'vozac');
```

ime	prezime	plata	radno_mesto
Pavle	Spasic	41300	Nis
Vojin	Markovic	53400	Novi Sad
Jovan	Petovic	41200	Novi Sad
Marija	Jovovic	42100	Nis

ime	prezime	polozaj
Marko	Pavic	sef prodavnice
Milica	Jovic	sef prodavnice
Mirko	Vojinovic	generalni direktor
Stefan	Popovic	sef prodavnice

Slika 19: Rezultat upita iz primera 42 (levo) i rezultat upita iz primera 44 (desno)

Operator *LIKE*

Ovaj operator se koristi za postavljanje uslova podacima tekstualnog tipa. Pomoću operatora *LIKE* mogu se pronaći sve niske koje počinju ili se završavaju traženim ka-

rakterom, niske koje imaju dve reči, niske koje imaju tačno traženi broj karaktera i sl.

Primer 45: Sintaksa operatora *LIKE*

```
SELECT ime_kolone1, ime_kolone2, ..., ime_koloneN
FROM ime_tabele
WHERE ime_kolone LIKE obrazac;
```

Obrazac se piše pod znacima navoda i opisuje uslov koji traženi tekstualni podatak treba da ispuni. Prilikom pisanja obrazaca koriste se dva specijalna karaktera: procenat (%) i donja crta (_). Procenat (%) označava da se na njegovom mestu može (a i ne mora) nalaziti neki karakter ili više njih. Donja crta (_) označava da se na njenom mestu mora nalaziti bilo koji, ali tačno jedan karakter. Primeri nekih obrazaca zajedno sa opisom njihovog značenja prikazani su na slici 20.

Primer	Značenje
'a%'	Označava sve reči koje počinju slovom "a"
'%a'	Označava sve reči koje se završavaju slovom "a"
'%na'	Označava sve reči koje se završavjau nizom slova "na"
'b_a'	Označava sve troslovne reči koje počinju slovm "b" a završavaju se slovm "a"
'%a_a'	Označava sve reči koje imaju bar tri karaktera, završavaju se na "a" i treći karakter od pozadi im je "a"

Slika 20: Primeri obrazaca

Treba napomenuti da se prilikom pisanja obrazaca ne pravi razlika između malih i velikih slova. Npr. sledeća dva obrasca imaju isto značenje: *'M%'* i *'m%'*.

Upitom iz primera 46 pronalaze se svi radnici čije ime počinje slovom „M“. U primeru broj 47 nalazi se upit za pretragu svih radnika kojima je slovo „o“ drugo slovo prezimena, dok je u primeru 48 napisan upit koji pronalazi sve radnike koji rade na položaju koji ima dve reči. Rezultati izvršavanja ova tri upita prikazani su na slici 21.

Primer 46: Upit koji koristi obrasce; prvo slovo imena

```
SELECT ime, prezime
FROM radnik
WHERE ime LIKE 'M%';
```

Primer 47: Upit koji koristi obrasce; drugo slovo prezimena

```
SELECT ime, prezime
FROM radnik
WHERE prezime LIKE '_o%';
```

Primer 48: Upit koji koristi obrasce; dve reči

```
SELECT ime, prezime, polozaj
FROM radnik
WHERE polozaj LIKE '% %';
```

ime	prezime
Marko	Pavic
Milos	Ilic
Milica	Jovic
Mirko	Vojinovic
Marija	Jovovic
Mira	Ostojic
Mina	Antic

ime	prezime
Milica	Jovic
Mirko	Vojinovic
Marija	Jovovic
Stefan	Popovic
Lazar	Tosic
Tomislav	Popadic

ime	prezime	polozaj
Marko	Pavic	sef prodavnice
Milica	Jovic	sef prodavnice
Mirko	Vojinovic	generalni direktor
Stefan	Popovic	sef prodavnice

Slika 21: Rezultati upita iz primera 46 (levo), primera 47 (u sredini) i primera 48 (desno)

8 Upiti za promenu i brisanje podataka iz tabele

Baza podataka je „živ organizam“, tabele se stalno ažuriraju, dodaju se novi podaci, postojeći se menjaju, za nekim podacima se izgubila potreba pa se oni brišu i sl. Ovo poglavlje je posvećeno ažuriranju postojećih podataka u tabeli i brisanju nepotrebnih podataka. Za promenu nekih podataka u tabeli koristi se naredba *UPDATE*, a za brisanje naredba *DELETE FROM*.

Promena podataka u tabeli

Primer 49: Sintaksa naredbe *UPDATE*

```
UPDATE ime_tabele
SET ime_kolone1 = nova_vrednost1, ... , ime_koloneN = nova_vrednostN
WHERE uslov;
```

Vrednost se menja tako što se koloni „dodeljuje“ nova vrednost. Kao što je prikazano primerom 49, gde je opisana sintaksa pisanja upita za ažuriranje podataka, moguće je jednim upitom promeniti podatke u više kolona. Uslov se navodi kako bi se ažurirale

samo određene kolone koje ispunjavaju navedeni uslov. Ako bi se izostavio uslov, svim redovima u tabeli bi se postavila ista vrednost u navedenim kolonama.

U primeru 50 prikazan je upit kojim se radnici Ivoni Jankovic postavlja nova plata u vrednosti od 41000. Tim upitom se ažurira samo jedno polje u tabeli **radnik**.

Primer 50: Ažuriranje jednog podatka

```
UPDATE radnik
SET plata = 41000
WHERE ime = 'Ivona' AND prezime = 'Jankovic';
```

Kako se menja vrednost u više redova nekoliko kolona jedne tabele prikazano je primerom 51, gde se nalazi upit kojim se svim šefovima prodavnica menja naziv položaja u rukovodilac prodavnice i svima se postavlja identična plata od 70000.

Primer 51: Ažuriranje više kolona

```
UPDATE radnik
SET polozaj = 'rukovodilac prodavnice', plata = 70000
WHERE polozaj = 'sef prodavnice';
```

U pirmeru 52 se ne navodi uslov, pa se svim radnicima plata povećava za 10%.

Primer 52: Ažuriranje više kolona; povećanje plate

```
UPDATE radnik
SET plata = plata * 1.1;
```

Pošto povećanje nije isto za svakog radnika, postavlja se pitanje kako ažurirati tabelu. U svako polje kolone *plata* se upisuje vrednost koja se dobije tako što se stara vrednost tog polja pomnoži brojem 1,1. Na taj način je izvršeno traženo povećanje plata od 10%.

Brisanje podataka iz tabele

Primer 53: Sintaksa naredbe *DELETE FROM*

```
DELETE FROM ime_tabele
WHERE uslov;
```

Naredbom *DELETE FROM* briše se jedan ili više redova iz tabele koji ispunjavaju navedeni uslov. Ako se izostavi uslov, biće izbrisani svi redovi u tabeli, tako da treba biti pažljiv prilikom pisanja takvih upita.

U primeru 54 nalazi se upit kojim se brišu svi vozači iz tabele **radnik**.

Primer 54: Brisanje redova

```
DELETE FROM radnik
WHERE polozaj = 'vozac';
```

9 Naredbe za formatiranje izlaza

Rezultat izvršavanja upita je neka tabela. U ovom poglavlju opisane su neke naredbe kojima se može upravljati izgledom rezultujuće tabele. Navedeno je kako se menja zaglavlje kolone, eliminišu isti redovi (duplikati) iz tabele, ograničava broj redova tabele i kako se sortiraju podaci u tabeli.

Naredba *DISTINCT*

Ovom naredbom se elimišu duplikati iz rezultujuće tabele, tj. onemogućava se pojava dva ili više redova sa istim podacima u rezultujućoj tabeli.

Spisak svih radnih mesta zaposlenih iz tabele **radnik** može se dobiti upitom iz primera 55.

Primer 55: Radna mesta

```
SELECT radno_mesto
FROM radnik;
```

Deo rezultata dobijenih izvršavanjem ovog upita prikazan je na slici 22.

radno_mesto
Beograd
Nis
Novi Sad
Novi Sad
Beograd
Novi Sad
Nis
Novi Sad

radno_mesto
Beograd
Nis
Novi Sad

Slika 22: Deo rezultata upita iz primera 55 (levo) i rezultati upita iz primera 56 (desno)

Iz prikazanog dela rezultata izvršavanja upita primećuje se da se gradovi ponavljaju više puta. Rezultujuća tabela bi bila daleko preglednija kada bi se eliminisali duplikati. Upit kojim se dobija takva tabela je napisan u primeru 56. Tom upitu je samo dodata naredba *DISTINCT* iza naredbe *SELECT*. Rezultat njegovog izvršavanja je takođe prikazan na slici 22.

Primer 56: Radna mesta bez ponavljanja

```
SELECT DISTINCT radno_mesto
FROM radnik;
```

Naredba *AS*

Prilikom ispisa podataka neke tabele, u zaglavlju svake kolone rezultujuće tabele piše ime te kolone. Pomoću naredbe *AS* korisnik može definisati šta će pisati u zaglavlju kolone. Kaže se da korisnik dodeljuje alijas ime toj koloni. Dodeljenim alijas imenom korisnik se može služiti i u samom upitu. Skraćivanje zapisa upita često se vrši tako što se kolonama sa dugačkim imenima dodele neki kratki alijasi i njima se korisnik služi pri pisanju tog upita. Dodeljeno alijas ime se gubi čim se upit izvrši.

Primer 57: Sintaksa naredbe *AS*

```
SELECT ime_kolone1 AS alijas1, ime_kolone2 AS alijas2, ...,
                                        ime_koloneN AS alijasN
FROM ime_tabele;
```

Pisanje rezervisane reči *AS* se može izostaviti, što je i prikazano primerom 58. Preglednosti i uniformnosti radi, u ovom radu nije izostavljana ključna reč *AS* prilikom davanja alijas imena.

Primer 58: Sintaksa naredbe *AS*

```
SELECT ime_kolone1 alijas1, ime_kolone2 alijas2, ..., ime_koloneN alijasN
FROM ime_tabele;
```

Narednim upitom u primeru 59 dodeljuje se alijas ime koloni *plata* tabele **radnik**. Deo rezultata izvršavanja upita prikazan je na slici 23.

Primer 59: Dodeljivanje alijas imena

```
SELECT ime, prezime, plata AS zarada
FROM radnik;
```

ime	prezime	zarada
Ivona	Jankovic	39500
Pavle	Spasic	41300
Vojin	Markovic	53400
Jovan	Petovic	41200
Marko	Pavic	71200
Milos	Ilic	39200

Slika 23: Deo rezultata upita iz primera 59

Naredba *ORDER BY*

Sortiranje redova tabele po određenim kolonama postiže se naredbom *ORDER BY*. Oznaka za rastuće soritiranje je *ASC*, a za opadajuće *DESC*.

Primer 60: Sintaksa naredbe *ORDER BY*

```
SELECT ime_kolone1, ime_kolone2, ..., ime_koloneN
FROM ime_tabele
WHERE uslov
ORDER BY kolona_po_kojoj_se_sortira1 ASC/DESC,
                        kolona_po_kojoj_se_sortira2 ASC/DESC... ;
```

Ukoliko se ne navede oznaka sortiranja, podrazumeva se rastuće sortiranje. Sortiranje se može vršiti po nekoliko nivoa. Ako u koloni koja je prva navedena iza naredbe *ORDER BY* ima više istih podataka, onda se njihovo sortirajne vrši po drugoj navedenoj koloni. Podaci tekstualnog tipa se sortiraju leksikografski.

Upit kojim se ispisuju imena i prezimena radnika sortiranih opadajuće po visini plate napisan je u primeru 61.

Primer 61: Upit za opadajuće sortiranje

```
SELECT ime, prezime, plata
FROM radnik
ORDER BY plata DESC;
```

U primeru 62 napisan je upit kojim se tabela **radik** rastuće sortira po prezimenu radnika.

Primer 62: Upit za rastuće sortiranje

```
SELECT *
FROM radnik
ORDER BY prezime;
```

Primer 63 sadrži upit kojim se dobijaju svi trgovci, sortirani po prezimenu i imenu.

Primer 63: Upit za sortiranje

```
SELECT prezime, ime
FROM radnik
WHERE polozaj = 'trgovac'
ORDER BY prezime ASC, ime ASC;
```

Naredba *LIMIT*

Za ograničavanje broja redova rezultujuće tabele, koristi se naredba *LIMIT*.

Primer 64: Sintaksa naredbe *LIMIT*

```
SELECT ime_kolone1, ime_kolone2, ..., ime_koloneN
FROM ime_tabele
WHERE uslov
LIMIT broj_prvih_redova_za_odbacivanje, broj_redova_za_ispisivanje;
```

Odmah treba napomenuti da se, ako iza naredbe *LIMIT* stoji samo jedan broj, podrazumeva ispisivanje toliko prvih dohvaćenih redova.

ime	prezime	plata ▼ 1
Mirko	Vojinovic	124500
Tomislav	Popadic	73400
Stefan	Popovic	72600
Marko	Pavic	71200
Milica	Jovic	69400
Filip	Tadic	67100
Vuk	Petrovic	59100
Vojin	Markovic	53400

prezime ▲ 1	ime ▲ 2
Antic	Mina
Ilic	Milos
Jankovic	Ivona
Jovovic	Marija
Maric	Paja
Markovic	Veselin
Ostojic	Mira
Pavic	Uros

Slika 24: Delovi rezultata upita iz primera 61 (levo) i primera 63 (desno)

Upitom iz primera 65 dobijaju se prva 4 dohvaćena radnika, a upitom iz primera 66 odbacuju se prva dva dohvaćena radnika i ispisuju se naredna 4 radnika. Rezultat izvršavanja ova dva upita prikazan je na slici 25.

Primer 65: Prva četiri dohvaćena reda

```
SELECT ime, prezime
FROM radnik
LIMIT 4;
```

Primer 66: Četiri dohvaćena reda, počevši od trećeg

```
SELECT ime, prezime
FROM radnik
LIMIT 2, 4;
```

ime	prezime
Ivona	Jankovic
Pavle	Spasic
Vojin	Markovic
Jovar	Petovic

ime	prezime
Vojin	Markovic
Jovan	Petovic
Marko	Pavic
Milos	Ilic

Slika 25: Rezultati upita iz primera 65 (levo) i primera 66 (desno)

Naredba *LIMIT* često se koristi u kombinaciji sa naredbom za sortiranje tabela *ORDER BY*. Njihova kombinacija je korisna jer je moguće dobiti prvih nekolika redova

tabele koji ispunjavaju određeni uslov. Npr. u primeru 67 je napisan upit kojim se dobija prvih pet radnika sa najvećom platom koji rade kao trgovci.

Primer 67: Uparivanje naredbi *LIMIT* i *ORDER BY*

```
SELECT ime, prezime, polozaj, plata
FROM radnik
WHERE polozaj = 'trgovac'
ORDER BY plata DESC
LIMIT 5;
```

ime	prezime	polozaj	plata 1
Ana	Spasic	trgovac	42100
Marija	Jovovic	trgovac	42100
Veselin	Markovic	trgovac	41400
Pavle	Spasic	trgovac	41300
Jovan	Petovic	trgovac	41200

Slika 26: Rezultat izvršavanja upita iz primera 67

10 Ugrađene funkcije

Prilikom pisanja upita za pretragu baze podataka, korisniku je na raspolaganju mnoštvo ugrađenih funkcija. Za svaki tip podatka koji se može naći u tabeli postoje odgovarajuće funkcije za njihovu obradu. Za rad sa vremenom i datumima ima mnoštvo korisnih ugrađenih funkcija. Pretraga i obrada tekstualnih podataka se mnogo pojednostavljuje korišćenjem funkcija. Podrazumeva se postojanje raznih matematičkih funkcija. Zanimljiva familija funkcija su agregatne ili grupne funkcije.

Agregatne funkcije

Agregatne funkcije primenjuju se na ceo skup rezltata upita i vraćaju jednu vrednost. Redove rezultujuće tabele je moguće podeliti u više grupa, pa na svaku grupu primeniti neku agregatnu funkciju. Zbog toga se one nazivaju i grupnim funkcijama.

- *MAX* - povratna vrednost ove funkcije je najveća vrednost u koloni koja joj je prosleđena kao argument.

  U primeru 68 napisan je upit kojim se dobija vrednost najveće plate.

Primer 68: Upotreba funkcije *MAX*

```
SELECT MAX(plata)
FROM radnik;
```

- *MIN* - povratna vrednost ove funkcije je najmanja vrednost u koloni koja joj je prosleđena kao argument.

  U primeru 69 napisan je upit kojim se dobija datum zaposlenja radnika koji je najduže u firmi.

Primer 69: Upotreba funkcije *MIN*

```
SELECT MIN(datum_zaposlenja)
FROM radnik;
```

- *SUM* - povratna vrednost ove funkcije je zbir vrednosti iz svih polja kolone koja joj je prosleđena kao argument.

  U primeru 70 napisan je upit kojim se dobija zbir plata svih radnika.

Primer 70: Upotreba funkcije *SUM*

```
SELECT SUM(plata)
FROM radnik;
```

- *AVG* - povratna vrednost ove funkcije je prosečna vrednosti svih polja kolone koja joj je prosleđena kao argument.

  U primeru 71 napisan je upit kojim se dobija prosečna plata radnika.

Primer 71: Upotreba funkcije *AVG*

```
SELECT AVG(plata)
FROM radnik;
```

- *COUNT* - povratna vrednost ove funkcije je ukupan broj upisanih vrednosti u koloni koja joj je prosleđena kao argument.

  U primeru 72 napisan je upit kojim se dobija broj trgovaca.

Primer 72: Upotreba funkcije *COUNT*

```
SELECT COUNT(ime)
FROM radnik
WHERE polozaj = 'trgovac';
```

Rezultati izvršavanja upita iz prethodnih pet primera prikazani su na slici 27.

MAX(plata)	MIN(datum_zaposlenja)	SUM(plata)	AVG(plata)	COUNT(ime)
124500	2008-11-23	1150200	52281.8182	14

Slika 27: Redom prikazani rezultati upita iz primera 68, 69, 70, 71 i 72

Funkcije za rad sa znakovnim vrednostima

Leksikografsko upoređivanje tekstualnih podataka se vrši pomoću standardnih relacionih operatora koji su prikazani na slici 12. Problem nastaje kada je potrebno nadovezati dve niske ili zameniti neki karakter u tekstu. U narednoj listi navedene su ugrađene funkcije koje se koriste za rešavanje takvih problema.

- *CONCAT(str1, str2, ..., strN)* - vraća nisku nastalu nadovezivanjem niski koje su joj prosleđene kao argumenti.
- *LENGTH(str)* - povratna vrednost ove funkcije je dužina niske *str*.
- *LOCATE(str1, str2, poz)* - vraća poziciju prvog pojvaljivanja niske *str1* u nisci *str2*, pretraga počinje od pozicije *poz* u nisci *str2*.
- *REPLACE(str1, str2, str3)* - vraća nisku *str1* u kojoj su sva pojavljivanja podniske *str2* zamenjena niskom *str3*.
- *LOWER(str)* - vraća nisku *str* u kojoj su sva velika slova zamenjena malim.
- *UPPER(str)* - vraća nisku *str* u kojoj su sva mala slova zamenjena velikim.

U primeru 73 prikazan je jedan malo složeniji upit, kojim se odgovara na sledeći zadatak. Ispisati prezimena i imena (prezime da bude ispisano velikim slovima) u jednoj koloni, položaj i platu svih radnika. Ako je radnik šef prodavnice, onda reč „sef“ zameniti rečju „rukovodilac“.

Primer 73: Funkcije namenjene niskama

```
SELECT CONCAT(UPPER(prezime),' ', ime) AS Prezime_i_ime,
                    REPLACE(polozaj, 'sef', 'rukovodilac') AS polozaj, plata
FROM radnik;
```

Potrebno je naglasitit da pri nadovezivanju niski koje predstavljaju ime i prezime, između njih treba dodati jednu belinu (blanko ili razmak) da bi se izbeglo njihovo „slepljivanje“. Deo rezultata izvršavanja upita iz primera 73 prikazan je na slici 28.

Prezime_i_ime	polozaj	plata
JANKOVIC Ivona	trgovac	39500
SPASIC Pavle	trgovac	41300
MARKOVIC Vojin	vozac	53400
PETOVIC Jovan	trgovac	41200
PAVIC Marko	rukovodilac prodavnice	71200

Slika 28: Deo rezultata upita iz primera 73

Funkcije za rad sa datumima

Datum i vreme se čuvaju kao niska zapisana u tačno definisanom formatu. Datumi se pišu u obliku YYYY-MM-DD. Prve četiri cifre predstavljaju godinu, sledeći par cifara je mesec, a poslednje dve cifre su dan. Vreme se piše u formatu HH:MM:SS. Prvi par je rezervisan za časove, drugi za minute, a treći za sekunde.

- *NOW* - vraća trenutni datum i vreme u formatu YYYY-MM-DD HH:MM:SS.
- *CURDATE* ili *CURRENT_DATE* - vraća trenutni datum u formatu YYYY-MM-DD.
- *CURTIME* ili *CURRENT_TIME* - vraća trenutno vreme u formatu HH:MM:SS.
- *EXTRACT(tip FROM datum)* - vraća vrednost određene komponente iz datuma koji je prosleđen argumentom *datum*. Komponenta može biti godina, mesec, dan, sat, minut ili sekund. Komponenta se prosleđuje parametrom *tip* i to, za godinu prosleđuje se *YEAR*, za mesec *MONTH*, za dan *DAY*, za sate *HOUR*, za minute *MINUTE* i za sekunde *SECOND*.

Upit kojim se određuju imena i prezimena svih radnika koji su se zaposlili u mesecu koji odgovara mesecu u trenutku pretrage prikazan je primerom 74.

Primer 74: Funkcije za rad sa datumima

```
SELECT ime, prezime, datum_zaposlenja
FROM radnik
WHERE EXTRACT(MONTH FROM datum_zaposlenja) = EXTRACT(MONTH FROM NOW());
```

Neke matematičke funkcije

Sledi pregled korisnih matematičkih funkcija.

- *ABS(n)* - vraća apsolutnu vrednost broja n.
- *POWER(n, m)* - vraća broj n pomnožen sam sa sobom m puta, odnosno vraća vrednost izraza n^m.
- *SQRT(n)* - vraća kvadratni koren broja n.
- n *DIV* m - vraća rezultat celobrojnog deljenja braja n brojem m.
- *MOD(n, m)* - vraća ostatak pri deljenju broja n brojem m.
- *RAND()* - vraća slučajan broj između nula i jedan.
- *FLOOR(n)* - zaokrugljuje broj n na najbliži manji (ili jednak) ceo broj.
- *CEILING(n)* - zaokrugljuje broj n na najbliži veći (ili jednak) ceo broj.

- *ROUND(n)* - zaokrugljuje broj n na najbliži ceo broj.
- *ROUND(n, d)* - zaokrugljuje broj n na najbliži broj sa d decimala.

Funkcija *MOD(n, m)* može se zapisati i ovako: n *MOD* m, ali pokušaj da se funkcija *DIV* zapiše u obliku *DIV(n, m)* će izazvati pojavu sintaksne greške.

Više o ovim funkcijama, kao i pregled svih funkcija, može se pronaći na zvaničnom sajtu sistema za upravljanje bazama podataka MySQL [4].

11 Grupisanje podataka

Naredba kojom se omogućava grupisanje podataka po nekom zajedničkom svojstvu je naredba *GROUP BY*. Najučinkovitija je kada se koristi u kombinaciji sa nekom od agregatnih funkcija.

Analizom upita iz primera 75 najbolje se može objasniti kako funkcioniše ova naredba.

Primer 75: Upit kojim se grupišu podaci

```
SELECT polozaj, MIN(plata)
FROM radnik
GROUP BY polozaj;
```

polozaj	MIN(plata)
generalni direktor	124500
sef prodavnice	69400
trgovac	38000
vozac	53400

Slika 29: Rezultat upita iz primera 75

Upit iz primera 75 izvršava se tako što se traži najmanja plata u svakoj grupi radnika. Radnici se grupišu po položaju na kom se nalaze. Jednu grupu radnika čine vozači, drugu grupu trgovci itd. Nakon izvršenog grupisanja, za svaku grupu radnika se posebno određuje najmanja plata. Na kraju, rezultujuća tabela ima dve kolone. U jednoj koloni su nazivi položaja, a u drugoj minimalna plata za svaki položaj.

Kako odrediti prosečnu platu radnika za svaki grad u kome rade prikazano je upitom iz primera 76. Rezultati su soritrani opadajuće po visini prosečne plate.

Primer 76: Prosečna plata u svakom gradu

```
SELECT radno_mesto, AVG(plata) AS prosecna
FROM radnik
GROUP BY radno_mesto
ORDER BY prosecna DESC;
```

Prilikom pretrage, za postavljanje uslova koje mora da ispuni jedan red tabele koristi se ključna reč *WHERE*. Moguće je postaviti uslov grupi podataka grupisanoj pomoću naredbe *GROUP BY*. Za postavljanje uslova grupi podataka koristi se rezervisana reč *HAVING*.

Kako bi uloga ključne reči *HAVING* bila što slikovitije opisana, prvo je dat jedan upit bez nje. U primeru 77 napisan je upit koji vraća broj radnika u svakom gradu.

Primer 77: Broj radnika u svakom gradu

```
SELECT COUNT(*) AS broj_radnika, radno_mesto
FROM radnik
GROUP BY radno_mesto;
```

Na slici 30 (levo) prikazan je rezultat izvršavanja upita iz primera 77. Zaključak je da u Beogradu radi 10 radnika, a u Nišu i Novom Sadu po 6.

U primeru 78 prikazan je upit kojim se ispisuju samo oni gradovi u kojima radi manje od 10 radnika. Za ovakav i slične probleme, koristi se rezervisana reč *HAVING*, jer se svakoj grupi radnika grupisanoj po mestu u kom rade postavlja uslov da ih ima manje od 10. Rezultat izvršavanja ovog upita prikazan je na slici 30 (desno).

Primer 78: Upotreba naredbe *HAVING*

```
SELECT COUNT(*) AS broj_radnika, radno_mesto
FROM radnik
GROUP BY radno_mesto
HAVING broj_radnika < 10;
```

broj_radnika	radno_mesto
10	Beograd
6	Nis
6	Novi Sad

broj_radnika	radno_mesto
6	Nis
6	Novi Sad

Slika 30: Rezultat upita iz primera 77 (levo) i rezultat upita iz primera 78 (desno)

12 Naredbe *IF* i *CASE*

Prilikom pisanja upita moguće je upravljati tokom izvršavanja komandi pomoću naredbi *IF* i *CASE*. One su veoma slične iskazima *if* i *switch* iz većine programskih jezika.

Sintaksa naredbe *IF* je sledeća:
IF(uslov, rez1, rez2) - ako je logička vrednost izraza *uslov* tačna, onda se izvršava izraz *rez1*, u suprotnom se izvršava izraz *rez2*.

U primeru 79 napisan je upit kojim se pored imena i prezimena svakog radnika u dodatnoj koloni dopisuje da li mu je plata ispod ili iznad proseka u zavisnosti od toga

da li ima primanja manja ili veća od 55000. Rezultat izvršavanja ovog upita prikazan je na slici 31.

Primer 79: Prosečna plata

```
SELECT ime, prezime,
                IF(plata < 55000, 'ispod proseka', 'iznad proseka') AS opis
FROM radnik;
```

Potrebno je ispisati imena i prezimena svih radnika, ali ako radnik ima platu manju od 55000, ime i prezime treba da mu piše velikim slovima. Upit kojim se rešava ovaj problem je napisan u pirmeru 80, a rezultat njegovog izvršavanja je takođe prikazan na slici 31.

Primer 80: Prosečna plata i funkcije za obradu teksta

```
SELECT IF(plata < 55000, UPPER(CONCAT(ime, ' ', prezime)),
                    CONCAT(ime, ' ', prezime)) AS Ime_i_prezime, plata
FROM radnik;
```

ime	prezime	opis
Ivona	Jankovic	ispod proseka
Pavle	Spasic	ispod proseka
Vojin	Markovic	ispod proseka
Jovan	Petovic	ispod proseka
Marko	Pavic	iznad proseka
Milos	Ilic	ispod proseka
Milica	Jovic	iznad proseka
Paja	Maric	ispod proseka

Ime_i_prezime	plata
IVONA JANKOVIC	39500
PAVLE SPASIC	41300
VOJIN MARKOVIC	53400
JOVAN PETOVIC	41200
Marko Pavic	71200
MILOS ILIC	39200
Milica Jovic	69400
PAJA MARIC	38900

Slika 31: Delovi rezultata upita iz primera 79 (levo) i upita iz primera 80 (desno)

Naredba *CASE* je slična naredbi *IF*. Razlika je u tome što kod nje ima više rezultujućih mogućnosti, dok su kod naredbe *IF* moguće samo dve.

Primer 81: Sintaksa naredbe *CASE*

```
CASE
WHEN uslov1 THEN rezultat1
WHEN uslov2 THEN rezultat2
  .
  .
  .
WHEN uslovN THEN rezultatN
ELSE rezultatN+1
END
```

Ako je logička vrednost izraza *uslov1* tačna onda se izvršava izraz *rezultat1*, ako je logička vrednost izraza *uslov2* tačna onda se izvršava izraz *rezultat2* itd. Ako ni u jednom slučaju logička vrednost uslova nije tačna, izvršava se izraz *rezultatN+1*.

U primeru 82 napisan je upit kojim se rešava sledeći problem. Ispisati imena i prezimena svih radnika i u dodatnoj koloni, ako rade u Novom Sadu ispisati „Vojvodina“, ako rade u Beogradu - „Glavni grad“ i ako rede u Nišu - „Jug Srbije“. Deo rezultata izvršavanja ovog upita prikazan je na slici 32.

Primer 82: Regioni

```
SELECT ime, prezime, radno_mesto,
        CASE
        WHEN radno_mesto = 'Novi Sad' THEN 'Vojvodina'
        WHEN radno_mesto = 'Beograd' THEN 'Glavni grad'
        ELSE 'Jug Srbije'
        END AS region
FROM radnik;
```

ime	prezime	radno_mesto	region
Ivona	Jankovic	Beograd	Glavni grad
Pavle	Spasic	Nis	Jug Srbije
Vojin	Markovic	Novi Sad	Vojvodina
Jovan	Petovic	Novi Sad	Vojvodina
Marko	Pavic	Beograd	Glavni grad
Milos	Ilic	Novi Sad	Vojvodina

Slika 32: Delovi rezultata upita iz primera 82

13 Povezivanje tabela

Jednu bazu podataka čini određeni broj tabela koje su međusobno povezane. Povezivanje tabela se vrši radi uštede memorijskog prostora i izbegavanje pisanja istih podataka na više mesta. Nakon sledećeg primera biće detaljnije objašnjeno kako se ostvaruje ušteda memorijskog prostora i zašto se izbegava pisanje istih podataka na više mesta. U elektronskim lekcijama na platformi eŠkola veba, kreirana je tabela **radnik**, za koju su pisani primeri upita iz ovog rada. Pored te tabele kreirana je i tabela **sluzbeni_put** u koju se evidentiraju podaci o službenim putovanjima radnika. Tabela **sluzbeni_put** je kreirana upitom iz primera 83.

Primer 83: Upit za kreiranje tabele **sluzbeni_put**

```
CREATE TABLE sluzbeni_put (
    id_puta INT NOT NULL AUTO_INCREMENT,
    putnik INT NOT NULL,
    datum DATE NULL,
    PRIMARY KEY (`id_puta`)
);
```

Za svaki sluzbeni put se beleži ko je i kada išao na putovanje. Prvih nekoliko redova tabele **sluzbeni_put** prikazani su na slici 33.

id_puta	putnik	datum
1	9	2018-05-06
2	3	2018-05-11
3	16	2018-05-21
4	19	2018-04-10
5	3	2018-04-15
6	16	2018-03-01
7	9	2017-10-10
8	3	2018-06-11

Slika 33: Deo sadržaja tabele **sluzbeni_put**

Primetno je da u koloni *putnik* tabele **sluzbeni_put** pišu brojevi, a ne imena radnika koji su išli na službeno putovanje, kako je to možda očekivano. Podaci o svim radnicima su dostupni u tabeli **radnik**. Prilikom popunjavanja kolone *putnik* tabele **sluzbeni_put** nema potrebe ponovo pisati ime radnika, dovoljno je zabeležiti jedinstvenu identifikaciju tog radnika, tj. upisati podatke iz kolone primarnog ključa tabele **radnik**. Na taj način se, iz tabele **sluzbeni_put**, korisnik preusmerava na tabelu **radnik** gde može da pročita sve podatke o radniku koji je putovao. Na ovaj način se štedi memorijski prostor jer se ne pamti niska sa imenom i prezimenom radnika već samo podaci iz kolone primarnog ključa. Još jedna pozitivna strana povezivanja tabela je lako ažuriranje podataka. Ako neko od radnika promeni prezime, dovoljno je samo u tabeli **radnik** promeniti podatak u koloni *prezime* pa će se dobiti novo prezime radnika prilikom pretrage tabela u kojima je upisana idetifikacija tog radnika.

Povezivanje više tabela u jednom upitu vrši se pomoću ključne reči *JOIN*.

Primer 84: Sintaksa povezivanja dve tabele

```
SELECT tabela1.kolona1, ..., tabela1.kolonaN,
                        tabela2.kolona1, ..., tabela2.kolonaN
FROM tabela1 JOIN tabela2 ON kriterijum_povezivanja;
```

Do sada su se iza naredbe *SELECT* pisala imena traženih kolona. U tim slučajevima je bilo jasno kojoj tabeli pripadaju te kolone jer se pretraga odnosila samo na jednu tabelu. Ako se pretražuju dve tabele, potrebno je prvo navesti ime tabele kojoj pripada željena kolona, tačku, pa zatim ime kolone. Ovakva notacija je potrebna jer dve tabele mogu imati kolone sa istim imenima. Odmah treba napomenuti da je moguće izostaviti ovakav zapis i pisati samo imena kolona, pod uslovom da tabele koje se pretražuju nemaju istoimene kolone. Zbog veće razumljivosti upita, u lekcijama koje su napravljene za potrebe ovog rada i u samom radu, u svim upitima je pored imena kolone pisano i ime tabele kojoj pripadaju. Ključna reč *JOIN* se piše između imena tabela koje se povezuju. Zatim sledi rezervisana reč *ON* pa se navodi kriterijum po kom se tabele povezuju.

Upit kojim se dolazi do imena i prezimena radnika koji su išli na službeno putovanje napisan je u primeru 85, a rezultat njegovog izvršavanja prikazan je na slici 34.

Primer 85: Imena i prezimena putnika

```
SELECT radnik.ime, radnik.prezime
FROM radnik JOIN sluzbeni_put ON radnik.id_radnika = sluzbeni_put.putnik;
```

Analizom upita može se zaključiti da je kriterijum za povezivanje ove dve tabele taj da podatak u koloni *id_radnika* tabele **radnik** mora da bude jednak podatku u koloni *putnik* tabele **sluzbeni_put**. Dakle, ispisuju se imena i prezimena samo onih radnika čija se jedinstvena identifikacija nalazi u koloni *putnik* tabele **sluzbeni_put**.

Pokazano je kako se pomoću naredbe *AS* zadaju alijas imena kolonama. Pomoću nje se može zadati alijas ime tabeli. Ako su nazivi tabela malo duži, često im se daju neka kratka alijas imena i time se postiže znatno skraćivanje zapisa upita.

U primeru 86 napisan je upit kojim se dobijaju imena svih vozača iz Beograda koji su išli na službeni put. Rezultat njegovog izvršavanja prikazan je na slici 34.

Primer 86: Vozači iz Beograda

```
SELECT r.ime, r.prezime, r.radno_mesto, r.polozaj, s.datum
FROM radnik AS r JOIN sluzbeni_put AS s ON r.id_radnika = s.putnik
WHERE r.radno_mesto = 'Beograd' AND r.polozaj = 'vozac'
ORDER BY s.datum;
```

ime	prezime
Vuk	Petrovic
Vojin	Markovic
Filip	Tadic
Tomislav	Popadic
Vojin	Markovic

ime	prezime	radno_mesto	polozaj	datum ▲ 1
Tomislav	Popadic	Beograd	vozac	2017-05-21
Vuk	Petrovic	Beograd	vozac	2017-10-10
Tomislav	Popadic	Beograd	vozac	2018-04-10
Vuk	Petrovic	Beograd	vozac	2018-05-06
Tomislav	Popadic	Beograd	vozac	2018-07-03

Slika 34: Deo rezultata upita iz primera 85 (levo) i rezultat upita iz primera 86 (desno)

Potrebni su podaci iz kolona *ime*, *prezime*, *radno_mesto* i *polozaj* iz tabele **radnik** i podatak iz kolone *datum* tabele **sluzbeni_put**. Radi kraćeg zapisa upita, tabeli **radnik** je dodeljen alijas **r**, a tabeli **sluzbeni_put** alijas **s**. Kriterijum za povezivanje tabela je isti kao i u primeru 85 - da se poklapaju podaci u kolonama *id_radnika* tabele **radnik** i *putnik* tabele **sluzbeni_put**. Pošto se traži pronalazak samo vozača iz Beograda, ti uslovi su dodati u upit iza ključne reči *WHERE*. Na kraju je pomoću naredbe *ORDER BY* obezbeđeno sortiranje podataka po koloni *datum* iz tabele **sluzbeni_put**.

Slede dva malo složenija upita. Prvi je napisan u primeru 87 i njime se određuje broj službenih putovanja radnika iz Niša i Novog Sada. U drugom upitu napisanom u primeru 88, određuje se broj službenih putovanja iz svakog grada. Rezultati oba upita prikazani su na slici 35.

Primer 87: Broj putovanja iz Niša i Novog Sada

```
SELECT COUNT(s.putnik)
FROM radnik AS r JOIN sluzbeni_put AS s ON r.id_radnika = s.putnik
WHERE r.radno_mesto IN ('Nis', 'Novi Sad');
```

Primer 88: Broj putovanja iz svakog grada

```
SELECT COUNT(s.putnik), r.radno_mesto
FROM radnik AS r JOIN sluzbeni_put AS s ON r.id_radnika = s.putnik
GROUP BY r.radno_mesto;
```

Upit iz primera 88 zahteva dodatni komentar. Kako se traži broj službenih putovanja radnika iz svakog grada, potrebno je grupisati radnike naredbom *GROUP BY* po kriterijumu grada u kom rade, što se čuva u koloni *radno_mesto*. Tek se, nakon izvršenog grupisanja pomoću agregatno funkcije *COUNT()*, vrši brojanje radnika u svakoj grupi.

COUNT(s.putnik)
11

COUNT(s.putnik)	radno_mesto
6	Beograd
6	Nis
5	Novi Sad

Slika 35: Rezultat upita iz primera 87 (levo) i rezultat upita iz primera 88 (desno)

14 Vrste spojeva tabela

U prethodnom poglavlju opisano je povezivanje tabela pomoću ključne reči *JOIN*. Takvo povezivanje naziva se unutrašnje povezivanje tabela (umesto ključne reči *JOIN* može se pisati i *INNER JOIN*) i ono se najčešće koristi. Kod unutrašnjeg povezivanja tabela prikazuju se samo redovi iz obe tabele koji su upareni po navedenom kriterijumu. Pored takve vrste spoja dva tabele postoji još nekoliko načina povezivanja.

Dekartov proizvod dve tabele

Dekartovim proizvodom dobijaju se sve moguće kombinacije redova iz dve tabele. Ključne reči za ovo povezivanje su *CROSS JOIN* i ne navodi se nikakav kriterijum za povezivanje.

Dekartov proizvod tabela **radnik** i **sluzbeni_put** napisan je u upitu iz primera 89.

Primer 89: Dekartov proizvod dve tabele

```
SELECT radnik.*, sluzbeni_put.*
FROM radnik CROSS JOIN sluzbeni_put;
```

Kod ove vrste povezivanja mogu se izostaviti ključne reči *CROSS JOIN*. U pirmeru 90 napisan je isti upit kao iz primera 89 samo je izostavljeno pisanje ključnih reči *CROSS JOIN*. Izvršavanjem oba upita dobija se isti rezultat. Deo tih rezultata prikazan je na slici 36.

Primer 90: Dekartov proizvod dve tabele bez ključnih reči *CROSS JOIN*

```
SELECT radnik.*, sluzbeni_put.*
FROM radnik, sluzbeni_put;
```

Pogledom na rezultate koji se dobiju izvršavanjem upita iz primera 89 i 90, jasno je da ovakvo uparivanje tabela nema mnogo smisla. Jednostavno je prvi dohvaćeni red iz tabele **radnik** uparen sa svim redovima tabele **sluzbeni_put**, zatim ja na isti način uparen i drugi dohvaćeni red iz tabele **radnik** itd.

Smislene rezultate moguće je dobiti i korišćenjem Dekartovog proizvoda dve tabele. Potrebno je navesti kriterijum za uparivanje iza ključne reči *WHERE*, kao što je napisano u upitu iz primera 91. Tim upitom se dobijaju podaci samo o radnicima koji su išli na službeno putovnaje. Rezultat izvršavanja tog upita prikazan je na slici 37.

Primer 91: Dekartov proizvod dve tabele i naredba *WHERE*

```
SELECT radnik.*, sluzbeni_put.*
FROM radnik, sluzbeni_put
WHERE radnik.id_radnika = sluzbeni_put.putnik;
```

Pogledom na rezultate prikazane na slici 37 jasno je da ti rezultati imaju smislia. Vidi se kada je koji radnik išao na put. Ovaj način možda deluje jednostavnije od unutrašnjeg povezivanja, ali ga treba izbegavati jer može dovesti do usporavanja izvršenja upita i grešaka ako se ne navede korektan kriterijum za uparivanje tabela. Pored toga, sam upit je razumljiviji i lakši za shvatanje ako se za ovakve tipove upita koristi rezervisana reč *JOIN*, jer su tada uslovi za selekciju podataka odvojeni od kriterijuma za povezivanje tabela.

id_radnika	ime	prezime	plata	polozaj	radno_mesto	datum_zaposlenja	telefon	id_puta	putnik	datum
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	1	9	2018-05-06
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	2	3	2018-05-11
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	3	16	2018-05-21
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	4	19	2018-04-10
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	5	3	2018-04-15
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	6	16	2018-03-01
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	7	9	2017-10-10
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	8	3	2018-06-11
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	9	16	2018-02-23
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	10	19	2017-05-21
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	11	19	2018-07-03
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	12	3	2018-03-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	13	16	2018-01-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	14	7	2018-01-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	15	5	2018-01-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	16	14	2018-01-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	17	30	2018-01-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	18	7	2018-01-26
2	Pavle	Spasic	41300	trgovac	Nis	2017-04-25	0187215302	1	9	2018-05-06
2	Pavle	Spasic	41300	trgovac	Nis	2017-04-25	0187215302	2	3	2018-05-11
2	Pavle	Spasic	41300	trgovac	Nis	2017-04-25	0187215302	3	16	2018-05-21

Slika 36: Deo rezultata izvršavanja upita iz primera 89 i 90

id_radnika	ime	prezime	plata	polozaj	radno_mesto	datum_zaposlenja	telefon	id_puta	putnik	datum
9	Vuk	Petrovic	59100	vozac	Beograd	2017-08-28	0113987589	1	9	2018-05-06
3	Vojin	Markovic	53400	vozac	Novi Sad	2017-12-08	0213068124	2	3	2018-05-11
16	Filip	Tadic	67100	vozac	Nis	2017-07-08	0186311421	3	16	2018-05-21
19	Tomislav	Popadic	73400	vozac	Beograd	2016-12-13	0113103520	4	19	2018-04-10
3	Vojin	Markovic	53400	vozac	Novi Sad	2017-12-08	0213068124	5	3	2018-04-15
16	Filip	Tadic	67100	vozac	Nis	2017-07-08	0186311421	6	16	2018-03-01
9	Vuk	Petrovic	59100	vozac	Beograd	2017-08-28	0113987589	7	9	2017-10-10
3	Vojin	Markovic	53400	vozac	Novi Sad	2017-12-08	0213068124	8	3	2018-06-11
16	Filip	Tadic	67100	vozac	Nis	2017-07-08	0186311421	9	16	2018-02-23
19	Tomislav	Popadic	73400	vozac	Beograd	2016-12-13	0113103520	10	19	2017-05-21
19	Tomislav	Popadic	73400	vozac	Beograd	2016-12-13	0113103520	11	19	2018-07-03
3	Vojin	Markovic	53400	vozac	Novi Sad	2017-12-08	0213068124	12	3	2018-03-26
16	Filip	Tadic	67100	vozac	Nis	2017-07-08	0186311421	13	16	2018-01-26
7	Milica	Jovic	69400	sef prodavnice	Nis	2018-01-13	0184522019	14	7	2018-01-26
5	Marko	Pavic	71200	sef prodavnice	Beograd	2016-04-23	0117210034	15	5	2018-01-26
14	Stefan	Popovic	72600	sef prodavnice	Novi Sad	2017-11-27	0216342003	16	14	2018-01-26
7	Milica	Jovic	69400	sef prodavnice	Nis	2018-01-13	0184522019	18	7	2018-01-26

Slika 37: Rezultat izvršavanja upita iz primera 91

Levo spoljašnje povezivanje

Reči rezervisane za levo spoljašnje povezivanje su *LEFT OUTERN JOIN* ili samo *LEFT JOIN*.

Primer 92: Sintaksa levog spoljašnjeg povezivanja

```
SELECT tabela1.kolona1, ..., tabela1.kolonaN,
                            tabela2.kolona1, ..., tabela2.kolonaN
FROM tabela1 LEFT JOIN tabela2 ON uslov;
```

Kod ovakvog povezivanja prikazuju se redovi iz odabranih kolona tabele1 i redovi iz odgovarajućih kolona tabele2 koji su upareni po navedenom kriterijumu (isto kao kod unutrašnjeg povezivanja). Pored tih redova ispisuju se i svi redovi iz odabranih kolona tabele1 (tabela1 je sa leve strane ključne reči JOIN) koji nisu upareni sa nekim redom iz tabele2, a u polje za ispis podataka iz tabele2 se upisuje *NULL* vrednost.

Nakon primera 93 u kom je napisan upit za levo spoljašnje povezivanje tabela **radnik** i **sluzbeni_put** sledi detaljno objašnjenje. Deo rezultata izvršavanja tog upita prikazan je na slici 38 (levo).

Primer 93: Levo spoljašnje povezivanje tabela **radnik** i **sluzbeni_put**

```
SELECT radnik.*, sluzbeni_put.*
FROM radnik LEFT JOIN sluzbeni_put ON radnik.id_radnika=sluzbeni_put.putnik;
```

Gledajući upit i deo rezultata koji vraća, može se malo bolje shvatiti kako funkcioniše levo spoljašnje povezivanje tabela. U prva tri reda rezultujuće tabele prikazani su putnici koji su išli na put, zajedno sa informacijama o putovanju. Taj deo je isti kao kod unutrašnjeg povezivanja. U druga tri reda, iste tabele, prikazani su radnici koji nisu išli na službeno putovanje. U tim redovima su polja namenjana za detalje o putu popunjena *NULL* vrednošću.

Spoljašnje povezivanje je veoma korisno za upite slične primeru 94. U njemu je napisan upit kojim se određuju radnici koji nisu išli na službeni put. Deo dobijenih rezultata izvršavanjem upita prikazan je na slici 38 (u sredini).

Primer 94: Radnici koji nisu išli na službeni put

```
SELECT radnik.*
FROM radnik LEFT JOIN sluzbeni_put ON radnik.id_radnika = sluzbeni_put.putnik
WHERE sluzbeni_put.putnik IS NULL;
```

Desno spoljašnje povezivanje

Desno spoljašnje povezivanjeje veoma je slično levom spoljašnjem povezivanju. Pored toga što se ispisuju svi upareni redovi iz obe tabele, jedina razlika je u tome što se u ovom slučaju ispisuju redovi iz tabele koja je desno od ključne reči JOIN koji nisu upareni sa nekim redom iz druge tabele. Ključne reči za ovo povezivanje su *RIGHT OUTERN JOIN* ili samo *RIGHT JOIN*.

Upit za desno spoljašnje povezivanje tebela **radnik** i **sluzbeni_put** napisan je u primeru 95. Deo rezultata izvršavanja tog upita prikazan je na slici 38 (desno).

id_radnika	ime	prezime	plata	polozaj	radno_mesto	datum_zaposlenja	telefon	id_puta	putnik	datum
5	Marko	Pavic	71200	sef prodavnice	Beograd	2016-04-23	0117210034	15	5	2018-01-26
14	Stefan	Popovic	72600	sef prodavnice	Novi Sad	2017-11-27	0216342003	16	14	2018-01-26
7	Milica	Jovic	69400	sef prodavnice	Nis	2018-01-13	0184522019	18	7	2018-01-26
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914	*NULL*	*NULL*	*NULL*
2	Pavle	Spasic	41300	trgovac	Nis	2017-04-25	0187215302	*NULL*	*NULL*	*NULL*
4	Jovan	Petovic	41200	trgovac	Novi Sad	2018-04-06	0214580312	*NULL*	*NULL*	*NULL*

id_radnika	ime	prezime	plata	polozaj	radno_mesto	datum_zaposlenja	telefon
1	Ivona	Jankovic	39500	trgovac	Beograd	2017-06-19	0113256914
2	Pavle	Spasic	41300	trgovac	Nis	2017-04-25	0187215302
4	Jovan	Petovic	41200	trgovac	Novi Sad	2018-04-06	0214580312
6	Milos	Ilic	39200	trgovac	Novi Sad	2017-07-25	0216396452
8	Paja	Maric	38900	trgovac	Novi Sad	2018-04-11	02132008496
10	Mirko	Vojinovic	124500	generalni direktor	Beograd	2008-11-23	0116369369

id_radnika	ime	prezime	plata	polozaj	radno_mesto	datum_zaposlenja	telefon	id_puta	putnik	datum
5	Marko	Pavic	71200	sef prodavnice	Beograd	2016-04-23	0117210034	15	5	2018-01-26
14	Stefan	Popovic	72600	sef prodavnice	Novi Sad	2017-11-27	0216342003	16	14	2018-01-26
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	17	30	2018-01-26
7	Milica	Jovic	69400	sef prodavnice	Nis	2018-01-13	0184522019	18	7	2018-01-26

Slika 38: Delovi rezultata nastalih izvršavanjem upita iz primera 93 (levo), 94 (u sredini) i 95 (desno)

Primer 95: Desno spoljašnje povezivanje tabela **radnik** i **sluzbeni_put**

```
SELECT radnik.*, sluzbeni_put.*
FROM radnik RIGHT JOIN sluzbeni_put ON radnik.id_radnika=sluzbeni_put.putnik;
```

Prikazani su svi upareni redovi obe tabele i jedan red iz tabele **sluzbeni_put** koja je napisano desno od reči *JOIN*. Polja tog reda namenjena za podatke iz tabele **radnik** su popunjena *NULL* vrednošću.

Samospoj

Samospoj je termin koji predstavlja povezivanje jedne tabele sa samom sobom. Realizuje se tako što se jednoj tabeli dodele dva različita alijas imena i onda se sa njima barata kao sa dve različite tabele.

Samospoj možda deluje malo nelogično i nepotrebno, ali kako bi se rešio problem pronalaska svih radnika koji rade na istom položaju kao radnik *Stefan Popovic*. Upit za rešavanje ovog problema napisan je u primeru 96, a rezultat njegovog izvršavanja prikazan je na slici 39.

Primer 96: Samospoj

```
SELECT r1.ime, r1.prezime, r1.polozaj
FROM radnik AS r1 JOIN radnik AS r2 ON r1.polozaj = r2.polozaj
WHERE r2.ime = 'Stefan' AND r2.prezime = 'Popovic';
```

ime	prezime	polozaj
Marko	Pavic	sef prodavnice
Milica	Jovic	sef prodavnice
Stefan	Popovic	sef prodavnice

Slika 39: Rezultat izvršavanja upita iz primera 96

U [6] možete pronaći veliki broj primera upita u kojima se kombinuje više vrsta povezivanja tabela.

15 Povezivanje više tabela

Kako se u jednoj bazi podataka nalazi više tabela, da bi se dobile tražene informacije nije uvek dovoljno povezati samo dve tabele. Povezivanje tri, četiri ili više tabela je slično povezivanju dve tabele.

Pored tabela **radnik** i **sluzbeni_put**, za potrebe lekcija i rada, kreirane su još tri tabele: **kupac**, **proizvod** i **narudzbina**. One su kreirane sledećim upitima.

Primer 97: Upit za kreiranje tabele **kupac**

```
CREATE TABLE kupac (
  id_kupca INT NOT NULL AUTO_INCREMENT,
  ime VARCHAR(20) NOT NULL,
  prezime VARCHAR(50) NOT NULL,
  grad VARCHAR(30) NOT NULL,
  ulica VARCHAR(50) NOT NULL,
  broj INT NOT NULL,
  PRIMARY KEY (`id_kupca`)
);
```

Primer 98: Upit za kreiranje tabele **proizvod**

```
CREATE TABLE proizvod (
  id_proizvoda INT NOT NULL AUTO_INCREMENT,
  naziv VARCHAR(40) NOT NULL,
  cena INT NOT NULL,
  PRIMARY KEY (`id_proizvoda`)
);
```

Primer 99: Upit za kreiranje tabele **narudzbina**

```
CREATE TABLE narudzbina (
  id_narudzbine INT NOT NULL AUTO_INCREMENT,
  kupac INT NOT NULL,
  prodavac INT NOT NULL,
  proizvod INT NOT NULL,
  kolicina INT NOT NULL,
  cena INT NOT NULL,
  datum DATE NULL,
  PRIMARY KEY (`id_narudzbine`)
);
```

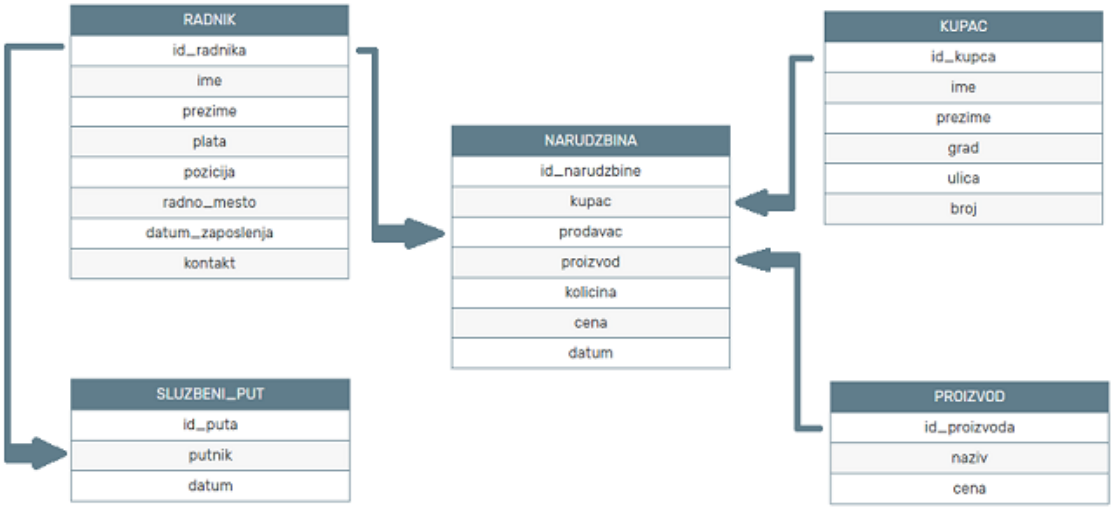
Ove tri tabele, zajedno sa prethodno korišćenim tabelama, čine jednu bazu podataka nekog preduzeća koje se bavi prodajom hrane za kućne ljubimce.

Grafički prikaz povezivanja tabela prikazan je na slici 40.

Bazu podataka dostupnu na platformi eŠkola veba namenjenu za učenje pisanja upita čini pet tabela: **radnik**, **sluzbeni_put**, **kupac**, **proizvod** i **narudzbina**. U tabeli **radnik** nalaze se podaci o svim zaposlenima u firmi. U tabeli **proizvod** su proizvodi koje firma prodaje. Jednostavnosti radi, na raspolaganju su samo hrana za pse i hrana za mačke. Sledeća je tabela **kupac**. U njoj se čuvaju podaci o kupcima koji kupuju hranu za svoje kućne ljubimce. U tabeli **sluzbeni_put** skladište se podaci koji su u vezi sa službenim putovanjima radnika firme. Peta tabela je tabela **narudzbina**. Ona sadrži podatke o tome kada i kod koga je neko kupio određeni proizvod.

Napisati opšti oblik povezivanja više tabela je tehnički zahtevno, a i pitanje je koliko bi bilo razumljivo. Zbog toga je detaljno analiziran jedan upit primenljiv na bazu koja je dostupna na platformi eŠkola veba.

Kako bi se odredili svi kupci koji su kupili hranu za pse, moraju se povezati tri tabele: **kupac**, **narudzbina** i **proizvod**. Zašto je to potrebno? Uslov pretrage je da



Slika 40: Grafički prikaz povezivanja tabela

je naziv proizvoda „Hrana za pse“. U tabeli **narudzbina**, u koloni *proizvod*, upisan je samo *id_proizvoda*, a ne njegov naziv. Da bi se pristupilo nazivu proizvoda, mora se povezati tabela **narudzbina** sa tabelom **proizvod**. Analogno, u tabeli **narudzbina** nije upisnao ime i prezime kupca već samo njegova identifikacija. Da bi se dobilo njegovo ime i prezime, mora se povezati tabela **narudzbina** sa tabelom **kupac**. Nakon zaključka koje tabele treba povezati treba ustanoviti po kom kriterijumu se one povezuju. Da bi se tabele **narudzbina** i **proizvod** korektno povezale potrebno je da podatak u koloni *proizvod* tabele **narudzbina** bude identičan podatku u koloni *id_proizvoda* tabele **proizvod**. Po istom principu se vrši povezivanje tabela **narudzbina** i **kupac**. Podaci u koloni *kupac* tabele **narudzbina** i u koloni *id_kupca* tabele **kupac** treba da budu isti. Broj kriterijuma za povezivanje tabela uvek je za jedan manji od broja tabela koje se povezuju.

Upit kojim se određuju svi kupci koji su kupili hranu za pse napisan je u primeru 100, a rezultat njegovog izvršavanja prikazan je na slici 41.

Primer 100: Kupci hrane za pse

```
SELECT DISTINCT kupac.ime, kupac.prezime
FROM narudzbina JOIN kupac ON narudzbina.kupac = kupac.id_kupca
           JOIN proizvod ON proizvod.id_proizvoda = narudzbina.proizvod
WHERE proizvod.naziv = 'Hrana za pse';
```

U ovom upitu upotrebljena je naredba *DISTINCT* kako bi se izbeglo višestruko pojavljivanje osoba koje su nekoliko puta kupovale hranu za pse.

Više tabela može se povezati i pomoću Dekartovog proizvoda. U tom slučaju se kriterijumi za povezivanje pišu iza ključne reči *WHERE*. Prednost ovakvog povezivanja je u tome što se tabele koje se povezuju pišu odvojeno od kriterijuma za povezivanje, a mana je što se kriterijumi za povezivanje pišu zajedno sa ostalim uslovima pretrage.

ime	prezime
Jovana	Pavlovic
Filip	Spasojevic
Svetozar	Puzic
Janko	Ivankovic
Veljko	Bojanic
Tanja	Pajic
Darko	Ivanovic
Pavle	Gajic
Ivan	Radivojevic
Tatjana	Velickovic
Marko	Obradovic

Slika 41: Rezultat izvršavanja upita iz primera 100

Analizirani problem određivanja svih kupaca hrane za pse moguće je rešiti korišćenjem Dekartovog proizvoda tabela, što je i prikazano primerom 101.

Primer 101: Kupci hrane za pse dobijeni Dekartovim proizvodom tabela

```
SELECT DISTINCT kupac.ime, kupac.prezime
FROM narudzbina, kupac, proizvod
WHERE narudzbina.kupac = kupac.id_kupca
                AND proizvod.id_proizvoda = narudzbina.proizvod
                AND proizvod.naziv = 'Hrana za pse';
```

Upiti kojima se određuju imena prodavaca koji su prodali hranu za pse u junu mesecu 2018. godine napisani su u sledeća dva primera. Upit sa unutrašnjim povezivanjem tabela napisan je u primeru 102, a u primeru 103 upit je napisan korišćenjem Dekartovog proizvoda tabela. Izvršavanjem oba upita dobija se isti rezultat koji je prikazan na slici 42.

Primer 102: Prodavci hrane za pse

```
SELECT DISTINCT radnik.ime, radnik.prezime
FROM narudzbina JOIN radnik ON narudzbina.prodavac = radnik.id_radnika
        JOIN proizvod ON narudzbina.proizvod = proizvod.id_proizvoda
WHERE proizvod.naziv = "Hrana za pse"
                AND EXTRACT(YEAR FROM narudzbina.datum) = 2018
                AND EXTRACT(MONTH FROM narudzbina.datum) = 6;
```

Primer 103: Prodavci hrane za pse dobijeni Dekartovim proizvodom tabela

```
SELECT radnik.ime, radnik.prezime
FROM radnik, narudzbina, proizvod
WHERE radnik.id_radnika = narudzbina.prodavac
            AND narudzbina.proizvod = proizvod.id_proizvoda
            AND proizvod.naziv = "Hrana za pse"
            AND EXTRACT(YEAR FROM narudzbina.datum) = 2018
            AND EXTRACT(MONTH FROM narudzbina.datum) = 6;
```

ime	prezime
Jovan	Petovic
Uros	Pavic

Slika 42: Rezultat izvršavanja upita iz primera 102 i 103

16 Unije

Za objedinjavanje rezultata dva ili više upita se koriste unije. Upiti se povezuju tako što se između upita piše ključna reč *UNION*.

Primer 104: Sintaksa upotrebe unija

```
upit1 UNION upit2 UNION ... UNION upitN
```

Prilikom pisanja ovakvih upita mora se voditi računa o tome da svi upiti koji se povezuju unijama moraju imati isti broj kolona. Pored toga, podaci u tim kolonama moraju biti istog tipa, tj. ako prvi upit vraća tabelu od tri kolone gde se u prvoj koloni čuvaju niske, u drugoj celobrojne vrednosti, a u trećoj datumi, onda i rezultujuće tabele ostalih upita moraju imati tri kolone koje čuvaju niske, cele brojeve i datume - tim redosledom.

U primeru 105 napisan je upit kojim se dobijaju imena i prezimena radnika i kupaca.

Primer 105: Imena radnika i kupaca

```
SELECT ime, prezime FROM radnik
UNION
SELECT ime, prezime FROM kupac;
```

Upitom iz primera 105 objedinjavaju se rezultati dva upita. Prvi upit je *SELECT ime, prezime FROM radnik*, a drugi je *SELECT ime, prezime FROM kupac*. Upit je korektno napisan jer oba upita koja su u uniji vraćaju rezultujuću tabelu od dve kolone. Dodatno se mora voditi računa da u prvim kolonama oba upita budu podaci istog tipa, to pravilo mora da važi i za druge kolone obe tabele.

Prilikom korišćenja unija, u rezultujućoj tabeli ispisuju se samo oni redovi koji nisu isti. Ako je potrebno ispisati sve redove, bez obzira na to da li su oni isti ili ne, koriste

se klučne reči *UNION ALL*. Razliku između *UNION* i *UNION ALL* najlakše je uočiti pomoću sledećeg primera.

Potrebno je pronaći sve gradove u kojima rade radnici firme i gradove u kojima žive kupci.

U primeru 106 ovaj problem je rešen korišćenjem rezervisane reči *UNION*, a u primeru 107 pomoću ključnih reči *UNION ALL*. Rezultati izvršavanja oba upita prikazani su na slici 43.

Primer 106: Gradovi dobijeni upotrebom ključne reči *UNION*

```
1 SELECT radno_mesto FROM radnik
2 UNION
3 SELECT grad FROM kupac;
```

Primer 107: Gradovi dobijeni upotrebom ključnih reči *UNION ALL*

```
1 SELECT radno_mesto FROM radnik
2 UNION ALL
3 SELECT grad FROM kupac;
```

radno_mesto
Beograd
Nis
Novi Sad

radno_mesto
Beograd
Nis
Novi Sad
Novi Sad
Beograd
Novi Sad

Slika 43: Rezultat upita iz primera 106 (levo) i deo rezultata upita iz primera 107 (desno)

U rezultujućoj tabeli upita iz primera 106 svaki grad je naveden samo jednom, a u rezultujućoj tabeli upita iz primera 107 se za svakog radnika i kupca ispisuje grad u kom rade ili žive, bez obzira na to da li se on već pojavio u tabeli ili ne.

Za naglašavanje razlike između redova rezultujuće tabele, koji su dobijeni prvim ili drugim upitom, dodaje se jedna nova kolona, kao što je urađeno u primeru 108.

Primer 108: Dodavanje nove kolone

```
1 SELECT ime, prezime, 'radnik' AS nova_kolona FROM radnik
2 UNION
3 SELECT ime, prezime, 'kupac' FROM kupac;
```

Dozvoljeno je povezivanje unijama uslovnih upita. U primeru 109 napisan je upit kojim se dobijaju svi kupci iz Beograda i svi radnici koji rade u Beogradu.

Primer 109: Kupci i radnici iz Beograda

```
SELECT ime, prezime, grad FROM kupac WHERE grad = 'Beograd'
UNION
SELECT ime, prezime, radno_mesto FROM radnik WHERE radno_mesto = 'Beograd';
```

ime	prezime	nova_kolona
Ivona	Jankovic	radnik
Pavle	Spasic	radnik
Vojin	Markovic	radnik
Jovan	Petovic	radnik
Marko	Pavic	radnik
Milos	Ilic	radnik

ime	prezime	grad
Janko	Ivankovic	Beograd
Jovana	Pavlovic	Beograd
Dejana	Avramovic	Beograd
Veljko	Bojanic	Beograd
Filip	Spasojevic	Beograd
Tanja	Pajic	Beograd

Slika 44: Deo rezultata dobijenih izvršavanjem upita iz primera 108 (levo) i 109 (desno)

17 Podupiti

Podupit ili ugnježden upit je upit unutar nekog upita. Svaki upit koji koristi podupite može se napisati i bez njih. Podupiti su dodati u kasnije verzije sistema MySQL jer se njihovom upotrebom izbegava pisanje složenih spojeva tabela, pa je samim tim i čitanje upita dosta jednostavnije.

Razlikuju se dva tipa podupita. Prvi tip su podupiti koji se pišu iza ključne reči *FROM* nekog upita i oni se nazivaju podupiti za izvedene tabele. Drugi tip su podupiti koji se pišu iza ključne reči *WHERE* nekog upita i oni se nazivaju podupiti za izraze.

Podupiti za izvedene tabele

Ovim tipom podupita dobija se izvedena tabela koja se dalje pretražuje u glavnom delu upita.

Nakon primera 110 sledi analiza upita koji je napisan u njemu. Tim upitom dobijaju se svi radnici koji rade na položaju vozača.

Primer 110: Vozači

```
SELECT vozac.ime, vozac.prezime
FROM (SELECT * FROM radnik WHERE polozaj = 'vozac') AS vozac;
```

Podupit ovog upita je *SELECT * FROM radnik WHERE polozaj = 'vozac'*. Izvršavanjem podupita dobije se izvedena tabela u kojoj se nalaze svi podaci o radnicima koji rade kao vozači. Toj izvedenoj tabeli dodeljen je alijas **vozac**. Nakon izvršravanja

podupita i formiranja izvedene tabele nastavlja se izvršavanje upita. Tada se pretražuje izvedena tabela **vozac** i vraćaju se imena i prezimena svih vozača.

U primeru 111 napisan je upit kojim se dobijaju svi radnici iz Niša koji su išli na službeni put.

Primer 111: Službena putovanja iz Niša

```
SELECT DISTINCT nislija.ime, nislija.prezime
FROM sluzbeni_put JOIN
                (SELECT radnik.id_radnika, radnik.ime, radnik.prezime
                FROM radnik
                WHERE radno_mesto = 'Nis') AS nislija
        ON nislija.id_radnika = sluzbeni_put.putnik;
```

Prvo se izvršava podupit i formira se izvedena tabela kojoj je dodeljeno alijas ime **nislija** koja ima tri kolone: *id_radnika*, *ime* i *prezime*. Ta tabela čuva podatke o radnicima koji rade u Nišu. Zatim se izvedena tabela **nislija** povezuje sa tabelom **sluzbeni_put** po kriterijumu da se podatak u koloni *id_radnika* tabele **nislija** poklapa sa podatkom u koloni *putnik* tabele **sluzbeni_put**. Nakon toga se ispisuju imena i prezimena svih radnika čija se identifikacija nalazi u tabeli **nislija**.

Rezultati izvršavanja ovog upita, kao i upita iz prmera 110, prikazani su na slici 45.

ime	prezime
Vojin	Markovic
Vuk	Petrovic
Filip	Tadic
Tomislav	Popadic

ime	prezime
Filip	Tadic
Milica	Jovic

Slika 45: Rezultat upita iz primera 110 (levo) i rezultat upita iz primera 111 (desno)

Podupiti za izraze

Podupiti za izraze ili preciznije podupiti za logičke izraze koriste se kao jedan od operanada relacionih operatora. Pored toga mogu se koristiti i sa operatorima *IN* i *BETWEEN*.

Kako pomoću podupita odrediti radnika sa najvećom platom prikazano je u primeru 112.

Primer 112: Radnik sa najvećom platom

```
SELECT ime, prezime
FROM radnik
WHERE plata = (SELECT MAX(plata) FROM radnik);
```

Podupit ovog upita je *SELECT MAX(plata) FROM radnik*. Njime se određuje vrednost najveće plate, pa se zatim ispisuje ime i prezime radnika koji ima tu platu.

Upotrebe podupita i operatora *IN* prikazana je primerom 113 gde je napisan upit kojim se određuju svi radnici koji nisu išli na službeni put.

Primer 113: Radnici koji nisu išli na službeni put

```
SELECT radnik.ime, radnik.prezime
FROM radnik
WHERE radnik.id_radnika NOT IN (SELECT sluzbeni_put.putnik FROM sluzbeni_put)
```

U podupitu ovog upita se određuju identifikacije svih radnika koji su išli na službeni put. Nakon toga se za svakog radnika iz tabele **radnik** proverava da li se njegova identifikacija nalazi u rezultatu koji je dobijen podupitom; ako se ne nalazi, ime i prezime tog radnika se vraća kao rezultat upita.

Jedan upit može u sebi imati dva ili više podupita. Lep primer takvog upita je prikazan primerom 114, gde je napisan upit kojim se dobijaju imena i prezimena svih radnika koji imaju platu veću od najveće plate trgovca, ali manju od najveće plate vozača.

Primer 114: Upit sa dva podupita

```
SELECT ime, prezime, polozaj
FROM radnik
WHERE plata BETWEEN (SELECT MAX(plata) FROM radnik WHERE polozaj = 'trgovac')
                    AND
                    (SELECT MAX(plata) FROM radnik WHERE polozaj = 'vozac');
```

Prvim podupitom dobija se najveća plata trgovca, a drugim najveća plata vozača. Te dve dobijene vrednosti predstavljaju granične vrednosti operatora *BETWEEN*. Zatim se ispisuju imena, prezimena i položaji svih radnika koji imaju platu u navedenom opsegu.

Rezultati izvršavanja sva tri prethodna upita prikazana su na slici 46.

Operatori *IN* i *BETWEEN* spominjani su i ranije, ali prilikom korišćenja podupita mogu se koristiti i neki novi operatori. To su operatori *EXISTS*, *ANY* i *ALL*.

Operator *EXISTS* se koristi kada se u uslovu podupita koriste podaci iz glavnog dela upita.

U primeru 115 napisan je upit kojim se dobijaju imena svih kupaca koji su kupovali hranu za ljubimce.

Primer 115: Svi kupci

```
SELECT kupac.ime, kupac.prezime
FROM kupac
WHERE EXISTS (SELECT narudzbina.kupac
                        FROM narudzbina
                        WHERE kupac.id_kupca = narudzbina.kupac);
```

U podupitu ovog upita koristi se podatak iz kolone *id_kupca* tabele **kupac** koja se nalazi u glavnom delu upita, pa je zbog toga potrebno koristiti operator *EXISTS*. U podupitu se dobija identifikacija onih kupaca koji su zaista nešto naručivali, zatim se za svaku osobu iz tabele **kupac** proverava da li se njena identifikacija nalazi u rezultatu podupita. Ako se nalazi, onda se ime i prezime te osobe ispisuje kao rezultat izvršavanja upita.

ime	prezime
Mirko	Vojinovic

ime	prezime
Ivona	Jankovic
Pavle	Spasic
Jovan	Petovic
Milos	Ilic
Paja	Maric
Mirko	Vojinovic
Marija	Jovovic
Jovana	Urosevic
Veselin	Markovic
Lazar	Tosic
Ana	Spasic
Uros	Pavic
Svetlana	Savic
Mira	Ostojic
Mina	Antic

ime	prezime	polozaj
Vojin	Markovic	vozac
Marko	Pavic	sef prodavnice
Milica	Jovic	sef prodavnice
Vuk	Petrovic	vozac
Marija	Jovovic	trgovac
Stefan	Popovic	sef prodavnice
Filip	Tadic	vozac
Ana	Spasic	trgovac
Tomislav	Popadic	vozac

Slika 46: Rezultat izvršavanja upita iz primera 112 (levo), 113 (sredina) i 114 (desno)

Kombinovanjem operatora *NOT* i *EXISTS* zamenjuje se upotreba spoljašnjeg povezivanja. Npr. upit za određivanje svih osoba iz tabele **kupac** koje nisu ništa naručivale moguće je dosta kraće i razumljivije napisati korišćenjem operatora *EXIST* nego pomoću spoljašnjeg povezivanja tabela. U primeru 116 napisan je taj upit.

Primer 116: Osobe koje nisu naručivale hranu

```
SELECT kupac.ime, kupac.prezime
FROM kupac
WHERE NOT EXISTS (SELECT narudzbina.kupac
                                FROM narudzbina
                                WHERE kupac.id_kupca = narudzbina.kupac);
```

Prvo se, izvršavanjem podupita, pronalaze sve osobe koje su naručivale hranu za kućne ljubimce, a onda se ispisuju samo one osobe iz tabele **kupac** koje se ne nalaze u rezultatu podupita.

Operator *ANY* koristi se prilikom pisanja podupita i vraća logičku vrednost *TRUE* ako bilo koja vrednost dobijena podupitom odgovara postavljenom uslovu.

Narednim upitom iz primera 117 određuju se svi radnici koji imaju platu veću od

najslabije plaćenog vozača.

Primer 117: Radnici sa platom većom od najmanje vozačeve

```
SELECT ime, prezime, polozaj, plata
FROM radnik
WHERE plata > ANY (SELECT plata
                   FROM radnik
                   WHERE polozaj = "vozac");
```

Upit se izvršava na sledeći način. U podupitu se dobiju plate svih vozača, zatim se za svaki red iz tabele **radnik** upoređuje plata radnika sa prvom platom dobijenom iz podupita i ispisuju se svi radnici koji imaju veću platu od te prve iz podupita. Nakon toga se sve plate iz tabele **radnik** upoređuju sa drugom platom dobijenom iz podupita i ako im je plata veća oni se ispisuju (ako je taj radnik već ispisan, ne vrši se dupliranje). Analogno se radi do kraja rezultata podupita. Na kraju se ispišu svi radnici koji imaju veću platu od najmanje plate koja je dobijena izvršavanjem podupita.

ime	prezime	polozaj	plata
Marko	Pavic	sef prodavnice	71200
Milica	Jovic	sef prodavnice	69400
Vuk	Petrovic	vozac	59100
Mirko	Vojinovic	generalni direktor	124500
Stefan	Popovic	sef prodavnice	72600
Filip	Tadic	vozac	67100
Tomislav	Popadic	vozac	73400

Slika 47: Rezultat izvršavanja upita iz primera 117

Operator *ALL* je veoma sličan operatoru *ANY*. Razlika je u tome što operator *ALL* vraća logičku vrednost *TRUE* ako sve vrednosti dobijene podupitom odgovaraju zadatom uslovu.

U primeru 118 napisan je upit kojim se dobijaju radnici koji imaju platu veću od najplaćenijeg vozača.

Primer 118: Radnici sa platom većom od najveće vozačeve

```
SELECT ime, prezime, polozaj, plata
FROM radnik
WHERE plata > ALL (SELECT plata
                   FROM radnik
                   WHERE polozaj = "vozac");
```

Upit se izvršava sledećim redom. Prvo se u podupitu dobiju plate svih vozača. Nakon toga se plata svakog radnika iz tabele **radnik** upoređuje sa svim rezultatima podupita i

ispisuju se samo oni radnici koji imaju platu veću od svih plata dobijenih izvršavanjem podupita.

ime	prezime	polozaj	plata
Mirko	Vojinovic	generalni direktor	124500

Slika 48: Rezultat izvršavanja upita iz primera 118

Na sajtu platforme za učenje veb tehnologija *w3schools* [7] možete pronaći više primera upita u kojima se koriste operatori *EXISTS*, *ANY* i *ALL*.

18 Razni upiti

U ovom poglavlju predstavljeni su raznovrsni upiti u kojima se povezuje više tabela, koriste funkcije za rad sa tekstualnim podacima, kombinuje unutrašnje i spoljašnje povezivanje itd.

Prvi upit u ovom poglavlju napisan je u primeru 119, gde je prikazano kombinovanje podupita i spoljašnjeg povezivanja tabela. Izvršavanjem upita dobijaju se imena kupaca pored imena radnika kod kojih su kupovali, ali se prikazuju i kupci koji nikada nisu naručivali hranu za kućne ljubimce.

Primer 119: Levo spoljašnje povezivanje i podupit

```
SELECT kupac.ime, kupac.prezime, radnik2.ime, radnik2.prezime
FROM kupac LEFT JOIN
        (SELECT radnik.ime, radnik.prezime, narudzbina.kupac
        FROM radnik JOIN
            narudzbina ON radnik.id_radnika = narudzbina.prodavac) AS radnik2
    ON radnik2.kupac = kupac.id_kupca;
```

Podupitom se dobija tabela od tri kolone: *ime* i *prezime* iz tabele **radnik** i kolona *kupac* iz tabele **narudzbina**. Njoj je dodeljeno alijas ime **radnik2**. Izvedena tabela dobija se unutrašnjim povezivanjem tabela **radnik** i **narudzbina** po kriterijumu da su podaci u kolonama *id_radnika* i *prodavac* isti. Tabela **kupac** se levim spoljašnjim povezivanjem spaja sa izvedenom tabelom **radnik2** po kriterijumu da je podatak u koloni *id_kupca* tabele **kupac** isti kao podatak u koloni *kupac* izvedene tabele **radnik2**. Kako se traži određivanje i onih kupaca koji nisu ništa naručivali, tabela **kupac** se nalazi sa leve strane rezervisanih reči *LEFT JOIN*.

Uz minimalne izmene upita iz primera 119, može se dobiti rešenje istog problema upotrebom desnog spoljašnjeg povezivanja, što je i prikazano u primeru 120.

Primer 120: Desno spoljašnje povezivanje i podupit

```
SELECT CONCAT(kupac.ime, " ", kupac.prezime) AS ime_kupca,
                CONCAT(radnik2.ime, " ", radnik2.prezime) AS ime_prodavca
FROM (SELECT radnik.ime, radnik.prezime, narudzbina.kupac
        FROM radnik JOIN
            narudzbina ON radnik.id_radnika = narudzbina.prodavac) AS radnik2
    RIGHT JOIN kupac ON radnik2.kupac = kupac.id_kupca;
```

Jedina razlika u odnosu na upit iz primera 119 je u tome što podupit mora biti napisan sa leve, a tabela **kupac** sa desne strane ključnih reči *RIGHT JOIN*. Radi bolje preglednosti krajnje tabele, po dve rezultujuće kolone su spojene u jednu i dodeljena su im alijas imena.

Delovi rezultata izvršavanja upita iz primera 119 i 120 prikazani su na slici 49. Krajnji podaci su isti u oba slučaja, ali se uočava jasna razlika u formatu i preglednosti tabela.

ime	prezime	ime	prezime
Janko	Ivankovic	Mina	Antic
Janko	Ivankovic	Ivona	Jankovic
Jovana	Pavlovic	Lazar	Tosic
Jovana	Pavlovic	Lazar	Tosic
Jovana	Pavlovic	Ana	Spasic
Dejana	Avramovic	*NULL*	*NULL*
Veljko	Bojanic	Svetlana	Savic
Veljko	Bojanic	Svetlana	Savic

ime_kupca	ime_prodavca
Janko Ivankovic	Mina Antic
Janko Ivankovic	Ivona Jankovic
Jovana Pavlovic	Lazar Tosic
Jovana Pavlovic	Lazar Tosic
Jovana Pavlovic	Ana Spasic
Dejana Avramovic	*NULL*
Veljko Bojanic	Svetlana Savic
Veljko Bojanic	Svetlana Savic

Slika 49: Delovi rezultata upita iz primera 119 (levo) i iz primera 120 (desno)

U primeru 121 napisan je upit kojim se dobijaju imena prodavaca i novčanu vrednost hrane koju su prodali.

Primer 121: Vrednost prodate hrane

```
SELECT CONCAT(radnik.ime, ' ', radnik.prezime) AS Prodavac,
                        SUM(narudzbina.cena) AS Ukupno
FROM radnik JOIN narudzbina ON narudzbina.prodavac = radnik.id_radnika
GROUP BY Prodavac ORDER BY Ukupno DESC;
```

Tabele **radnik** i **narudzbina** povezuju se po kriterijumu istih podataka u kolonama *id_radnika* i *prodavac*. Nakon toga se, za svakog radnika, sabiraju vrednosti iz kolone *cena* tabele **narudzbina** i vrši se ispisivanje rezultata. Vrednosti iz kolona *ime* i *prezime* objedinjavaju se u jednu kolonu. Ispis redova rezultujuće tabele se opadajuće sortira po vrednosti prodate hrane.

U prethodnom upitu prikazano je povezivanje dve tabele i upotreba agregatne funkcije, dok se u sledećem upitu ilustruje upotreba agregate funkcije pri povezivanju tri tabele.

Upit kojim se dobija tačan broj naručivanja hrane za pse svakog kupca napisan je u primeru 122.

Primer 122: Vrednost prodate hrane

```
SELECT CONCAT(kupac.ime, " ", kupac.prezime) AS Kupac, COUNT(kupac) AS n
FROM kupac JOIN narudzbina ON narudzbina.kupac = kupac.id_kupca
           JOIN proizvod ON narudzbina.proizvod = proizvod.id_proizvoda
WHERE proizvod.naziv = "Hrana za pse"
GROUP BY Kupac ORDER BY n DESC;
```

Prodavac	Ukupno 1
Ivona Jankovic	230900
Veselin Markovic	114400
Mina Antic	111300
Jovan Petovic	110000
Pavle Spasic	81000
Mira Ostojic	67200
Milos Ilic	21900
Paja Maric	15200
Ana Spasic	13500
Lazar Tosic	13200
Svetlana Savic	11600
Uros Pavic	7600
Marija Jovovic	3900
Jovana Urosevic	2600

Kupac	n 1
Jovana Pavlovic	3
Ivan Radivojevic	2
Janko Ivankovic	2
Svetozar Puzic	1
Tanja Pajic	1
Darko Ivanovic	1
Tatjana Velickovic	1
Filip Spasojevic	1
Veljko Bojanic	1
Pavle Gajic	1
Marko Obradovic	1

Slika 50: Rezultat upita iz primera 121 (levo) i iz primera 122 (desno)

U primeru 123 napisan je upit kojim se dobijaju informacije o najvrednijoj narudžbini - ko je kupac, ko prodavac, šta je kupljeno i vrednost narudžbine.

Primer 123: Najvrednija narudžbina

```
SELECT narudzbina.cena AS Cena,
        CONCAT(kupac.ime, " ", kupac.prezime) AS Kupac,
        CONCAT(radnik.ime, " ", radnik.prezime) AS Prodavac,
        proizvod.naziv AS Proizvod
FROM radnik, kupac, narudzbina, proizvod
WHERE narudzbina.kupac = kupac.id_kupca
        AND narudzbina.prodavac = radnik.id_radnika
        AND narudzbina.proizvod = proizvod.id_proizvoda
        AND narudzbina.cena = (SELECT MAX(narudzbina.cena) FROM narudzbina);
```

U ovom upitu povezuju se četiri tabele i piše se podupit. Povezivanje tabela je izvršeno Dekartovim proizvodom. U podupitu se dobija najveća novčana vrednost narudžbine, pa se zatim ispisuju traženi podaci za tu narudžbinu.

Cena	Kupac	Prodavac	Proizvod
220000	Janko Ivankovic	Ivona Jankovic	Hrana za pse

Slika 51: Rezultat izvršavanja upita iz primera 123

Više o podupitima može se pronaći na sajtu platforme za učenje veb tehnologija [8].

Zaključak

U elektronskim lekcijama koje su kreirane za potrebe ovog rada obrađeni su i sistematizovani osnovni koncepti sistema MySQL. Relacionim bazama podataka pridaje se veliki značaj u svim sferama poslovanja. Među mnogim sistemima za upravljanje tim bazama, sistem MySQL se izdvaja zbog svoje besplatne licence, jednostavnosti sintakse, velike kompatibilnosti sa mnogim programskim jezicima i lake upotrebljivosti na vebu. Elektronske lekcije imaju za cilj da na brz i efikasan način obuče korisnika osnovama sistema MySQL kako bi na tržištu rada što bolje iskoristili prednosti njegovog poznavanja. Zbog toga su lekcije metodički prilagođene korisnicima koji nemaju ranija iskustva u ovoj oblasti i predstavljaju dobru osnovu za dalje usavršavanje i proširivanje znanja. Savladavanjem osnova sistema MySQL, početnicima u svetu veb programiranja otvaraju se velike mogućnosti za dalje napredovanje. Elektronske lekcije kreirane za potrebe ovog rada predstavljaju veoma dobar izbor za usvajanje početnih znanja o relacionim bazama podataka i o sistemu MySQL jer su pisane jednostavnim jezikom kako bi korisnici što lakše, brže i kvalitetnije usvojili przentovanu teoriju. Teme obrađene elektronskim lekcijama pažljivo su birane kako bi se njihovim savladavanjem omogućilo dalje napredovanje u učenju i usvajanje naprednih mogućnosti sistema MySQL uz samostalni ili mentorski rad.

Literatura za učenje ovog sistema je obimna i lako dostupna na internetu, ali zahteva dobro razumevanje strane literature. Zbog toga elektronske lekcije kreirane za potrebe ovog rada imaju veliku prednost u odnosu na drugu literaturu jer su dostupne na srpskom jeziku. Još jedna velika prednost elektronskih lekcija je direktno pokretanje koda što omogućava korisniku lekcija da odmah vidi rezultate izvšavanja upita i daje mu slobodu da sam piše nove upite. Sve elektronske lekcije dostupne su na platformi eŠkola veba na adresi `http://www.edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/sql`.

Literatura

[1] Jurić Nemanja, Marić Miroslav, *eŠkola Veba, VII simpozijum Matematika i primene*, Matematički fakutet, Beograd, 5. novembar 2016.

[2] Zvanični sajt programskog jezika PHP, http://php.net/

[3] Luke Welling, Laura Thompson - *Priručnik za MySQL*, Mikro knjiga, 2005.

[4] Zvanični sajt sistema za upravljanje bazama podataka MySQL, https://www.mysql.com/

[5] Aleksandar Veljković, *Predavanje iz predmeta Veb programiranje* http://poincare.matf.bg.ac.rs/ aleksandar/files/web/skripta.pdf

[6] Sajt platforme namenjene učenju sistema za upravljanje bazama podataka MySQL, http://www.mysqltutorial.org/

[7] Sajt platforme za interaktivno učenje veb tehnologija https://www.w3schools.com/

[8] Sajt platforme za interaktivno učenje veb tehnologija https://www.w3resource.com/