



UNIVERZITET SINGIDUNUM
FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Baza podataka agencije za promet nekretnina

- Diplomski rad -

Mentor:

Mladen Veinović

Student:

Oliver Božinovski

Br. indeksa: 24/2006

Beograd, 2011.



FAKULTET ZA INFORMATIKU I RAČUNARSTVO

UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Beograd, Zorana Đinđića 44

Kandidat: Oliver Božinovski

Broj indeksa: 24/2006

Smer: programiranje i projektovanje

Tema: Baza podataka agencije za promet nekretnina

Zadatak: Realizovati MySQL bazu podataka i prateću aplikaciju za rad agencije za promet nekretnina

Datum odobrenja rada:
Beograd, _____.____._____.

MENTOR

Prof. dr Mladen Veinović

DEKAN

Prof. dr Mladen Veinović

Baza podataka agencije za promet nekretnina

Sažetak: U ovom diplomskom radu projektovana je i realizovana MySQL baze podataka za potrebe agencije za promet nekretninama. Svi primeri koji su obrađeni odnose se na poslovanje agencija sličnog tipa. Tabele su prilagođene i namenski napravljene za korisnike koji nisu eksperti u informacionim tehnologijama već tipični korisnici računara, koji su zaposleni kao službenici agencija za promet nekretnina. Zato su unos, obrada zahteva i ispis rezultata pretrage napravljeni kao servis i podrška zaposlenima. Korišćen softver je MySQL i PHP razvojno okruženje kao podrška pri zadavanju upita i unosu podataka u tabele. Spomenute su neki od standardnih postupaka i načina rešavanja problema sa kojima se susreću programeri prilikom izrade ovakvih i sličnih baza podataka.

Ključne reči: baza, MySQL, PHP, nekretnine, unos, ispis, ažuriranje, izveštaj...

Database for real estate trade

Abstract: In this paper is designed MySQL database for real estate trade company. All examples are applicable for real estate trade company, and adapted for their employees. Database is made for insert, update and read necesery information about owners or buildings or contracts witch are realized with clients. Software used in this paper ilustration are MySQL RDBMS and PHP web support.

Keywords: MySQL, PHP, database, real estate, insert, update, create tables, user...

SADRŽAJ

1	Uvod	- 3 -
2	O MySQL-u	- 4 -
2.1	Performanse.....	- 4 -
2.2	Stabilnost.....	- 4 -
2.3	Lakoća upotrebe.....	- 5 -
2.4	Cena	- 5 -
2.5	Mogućnosti.....	- 6 -
3	Model baze podataka agencije za promet nekretnina.....	- 7 -
3.1	Podaci koje čuvaju agencije za promet nekretnina.....	- 7 -
3.2	Pristup izradi modela	- 8 -
3.3	Principi kod projektovanja baza podataka	- 9 -
3.3.1	Redundantnost podataka	- 10 -
3.3.2	Anomalije podataka	- 10 -
3.3.3	Vrednost Null	- 10 -
4	Kreiranje baze podataka agencije za promet nekretnina	- 11 -
4.1	Kreiranje tabela.....	- 11 -
4.2	Forme za unos podataka.....	- 18 -
4.3	Kreiranje PHP stranice.....	- 20 -
4.3.1	Unos fotografija.....	- 22 -
4.3.2	Napomene prilikom unosa podataka.....	- 26 -
5	Realizacija upita	- 27 -
5.1	Jednostavni upiti	- 27 -
5.2	Složeni upiti.....	- 30 -
5.2.1	Levi i desni spoj	- 30 -
5.3	Pristup bazi podataka pomoću PHP-a.....	- 31 -

5.4	Ažuriranje zapisa	- 32 -
6	Zaključak	- 33 -
	Literatura	- 34 -

1 Uvod

U diplomskom radu biće obrađen postupak pravljenja MySQL baze podataka za poslovnu primenu, u ovom slučaju agencije za promet nekretninama. Na početku će biti reči o samom načinu rada agencije za promet nekretnina. Samim tim biće uočene specifičnosti kod poslovanja ovih tržišnih organizacija, i ustanoviće se za kojim vrstama podataka je potrebno praviti evidenciju u bazi koju kreiramo. Zatim će biti više reči o MySQL sistemu za rad sa bazama podataka. Tu ćemo predstaviti prednosti ovog sistema. Zašto se opredeliti za ovaj sistem za rad sa bazama podataka a ne neki drugi sistem kao što su npr. Oracle, Microsoft Access, Filemaker, DB2.

Biće obrazložen konkretan primer kreiranja baze podataka agencije za promet nekretnina, i predstaviti model za pravljenje tabela i njihove međusobne povezanosti, kao i samu realizaciju projekta. Koje tabele su glavne, koje pomoćne, kakvi podaci će se čuvati u svakoj od njih i u kom tipu odnosno formatu, biće prezentovano kroz tekst i primere. Postupak unosa, izmene, i brisanja podataka biće detaljno obrađen.

Izvršavanje upita u različitim varijantama i pravljenje izveštaja, takođe će biti deo ovog projekta. Postupci u kojima se prikazuje manipulacija podacima od strane krajnjih korisnika sistema biće obrađena u delu teksta gde je prisutna PHP podrška ovoj bazi podataka.

Na kraju će biti izveden zaključak, kao sumirani rezultat rada na ovom projektu. Biće ocenjena i njegova realizacija u praksi, kao i uslovi koji su neophodni da bi se ovakav projekat ostvario.

2 O MySQL-u

MySQL predstavlja višenitni SQL sistem za rad sa bazama podataka. Sistem radi kao server, obezbeđujući istovremeno pristup bazi podataka za više korisnika. Proizvod je kompanije MySQL AB, a u vlasništvu je poznate firme Sun Microsystems. Predstavlja softver otvorenog koda, tj. otvoren je za proučavanje, testiranje, nadogradnju i poboljšanje od strane bilo kog programera. Na raspolaganju je kao besplatan i kao komercijalni softver. Može se nabaviti pod uslovima opšte GNU licence, a ukoliko je potrebna komercijalna licenca, može se kupiti.

MySQL je sistem za upravljanje relacijskim bazama podataka. Relacijska baza podataka je baza podataka koja podatke radije sprema u odvojene tabele nego sve u jednu tabelu, zbog izbegavanja redundantnosti podataka. Ovo daje brzinu i fleksibilnost bazama podataka. Tablice su povezane definisanim relacijama što omogućuje kombinovanje podataka iz nekoliko tablica u slučaju da postoji zahtev za to. SQL znači eng. "Structured Query Language" (strukturni jezik za pretraživanje) – najčešći standardizovan jezik za pristupanje bazama podataka.

Zašto koristiti MySQL? Vrlo je brz, pouzdan, i lagan za korišćenje. Takođe ima vrlo praktične dodatne opcije razvijene u bliskoj saradnji sa korisnicima. MySQL je originalno razvijan za manipulaciju vrlo velikih baza podataka, mnogo brže od postojećih rešenja i uspešno se koristi u visoko zahtevnim produktivnim okruženjima već godinama. Pristupačnost, brzina i sigurnost čine MySQL vrlo pogodnim za pristupanje bazama podataka na Internetu.

2.1 Performanse

Na MySQL-ovoj zvaničnij web lokaciji mogu se pogledati rezultati poređenja MySQL-a i drugih baza podataka. Ti podaci pokazuju da uglavnom primetno nadmašuje brzinom svoje rivale. U načelu, rezultate bilo kog testiranja treba prihvatiti sa izvesnom rezervom, naročito zbog toga što ih je izvršio proizvođač baze podataka. Međutim, i nezavisni testovi pokazuju da je MySQL jedan od najbržih proizvoda na raspolaganju. Brzina je oduvek bila ključna odlika o kojoj projektanti MySQL-a vode računa. U MySQL se ugrađuju nove mogućnosti samo ako nisu na štetu performansi. Zbog toga se ponekad nove mogućnosti dodaju sporije nego što to korisnici žele, ali se time obezbeđuje da MySQL uvek bude brz.

2.2 Stabilnost

Projektanti su uvek smatrali da je stabilnost sistema od prvorazrednog značaja. Sve verzije MySQL-a izdate u binarnom obliku, čak i alfa izdanja, moraju proći kroz skup testova stabilnosti. Postupak obuhvata testiranje funkcija i drugih osobina, kao i ispitivanje rezultata operacija pri kojima su u prošlosti otkrivene i ispravljene greške, čime se obezbeđuje da se ponovo ne uvede već ispravljena greška. Ispravljanju grešaka projektanti moraju da daju viši prioritet od drugih poslova. U suštini, projektant prekida tekući posao sve dok se ne otkloni greška koja spada u njegovu oblast stručnosti. Pravilo nalaže da svako izdanje MySQL-a mora biti očišćeno od dotad poznatih grešaka koje se mogu reprodukovati u poznatim uslovima. Razume se, određeni problemi ne mogu se rešiti a da pritom ne prouzrokuju druge probleme na drugom mestu. To naročito važi za produkcione

verzije u koje ne bi trebalo ugrađivati suštinske izmene koje bi mogle da utiču na stabilnost verzije. U takvim slučajevima, problem se dokumentuje i otklanja u jednoj od narednih verzija. I najzad, kvalitet softvera obezbeđuju MySQL-ovi klijenti i zajednica korisnika. Preko četiri miliona korisnika širom sveta rade u veoma različitim okruženjima, što pruža jedinstvenu priliku za otkrivanje grešaka već u početnim fazama razvoja. Budući da je MySQL-ov sistem za otkrivanje i dokumentovanje grešaka javan, svaki korisnik ima uvid u sve što su drugi korisnici otkrili i može da dodaje svoje napomene.

2.3 Lakoća upotrebe

Treća ključna odlika MySQL-a jeste lakoća upotrebe. Nije potreban složen postupak podešavanja da biste počeli da radite. MySQL server radi kako treba čim ga raspakujete. Standardne vrednosti parametara su takve da minimiziraju utrošak prostora diska i memorije. Razume se, da bi se postigle optimalne performanse određenih funkcija koje su važne u produkcionom okruženju, kao što je npr. prijavljivanje korisnika za rad u bazu podataka, potrebno je finije doterivanje tih standardnih vrednosti radnih parametara MySQL-a. Pri tome vam mogu pomoći primeri konfiguracionih datoteka koje dobijate uz MySQL.

2.4 Cena

Cena je stavka koja se možda najlakše poredi sa konkurentima. Za mnoge uslove upotrebe, MySQL je besplatna aplikacija. Uslovi licence tipa GPL omogućavaju da neograničeno koristite softver, da menjate njegov izvorni kôd i da MySQL distribuirate drugim ljudima takođe pod GPL uslovima. U određenim situacijama, kada želite da distribuirate MySQL kao sastavni deo komercijalnog proizvoda, moraćete da kupite komercijalnu licencu. Drugim rečima, za MySQL postoje dva sistema licenciranja, gde je besplatna upotreba određena GPL uslovima, a način komercijalne upotrebe određen je standardnim sporazumima EULA(End User Licence Agreement- sporazum o licenci sa krajnjim korisnikom) ili OEM(Original Equipment Manufacturer – izvorni proizvođač opreme) uslovima. Pravilo kompanije je sledeće: “Ako je vaš proizvod besplatan, i naš je; ako se vaš proizvod plaća – naš takođe”. Najvažniji konkurenti nude isključivo komercijalne licence, uz složene sisteme cena, koje zavise od nameravanog načina upotrebe, broja procesora po serveru i broja istovremenih korisnika koji će uspostavljati veze sa bazom podataka. Troškovi za Oracle server, Microsoft SQL Server, IBM-ov DB2 i Sybase-ov server, mogu dostići desetine hiljada dolara u okruženju sa umerenim opterećenjem, pa čak i stotine hiljada dolara za server sa više procesora na koji se povezuje veći broj klijenata. MySQL se ponekad poredi sa drugim bazama podataka otvorenog koda, kao što su PostgreSQL i Firebird. Od svih pomenutih baza podataka otvorenog koda, jedino je MySQL proizvod iza kojeg stoji jedna kompanija, koja je vlasnik svih prava intelektualne svojine i koja nudi celovite komercijalne licence, koje obuhvataju i odgovornost i odštetu u slučaju neispravnosti, kao što zahtevaju velike organizacije. Još jedna kategorija softvera sa kojom se MySQL poredi jesu jeftine baze podataka koje nisu namenjene radu u klijent/server okruženju, a ciljna tržišna grupa su im kućni korisnici ili male firme. U tu kategoriju spadaju Access i Filemaker Pro. Mada se grafički korisnički interfejs programa iz ove kategorije često jednostavno upotrebljava, njima uglavnom nedostaje važna funkcionalnost i nisu dovoljno stabilni, skalabilni i brzi da bi bili upotrebljivi za aplikacije od ključne važnosti za firmu.

2.5 Mogućnosti

Poređenje mogućnosti softvera u mnogome zavisi od toga koje mogućnosti smatrate važnim za sistem ili aplikaciju za koju vam je potreban. MySQL pruža određene mogućnosti, kao što su:

- Tekstualno pretraživanje
- Replikovanje
- Podrška za veoma obimne tabele

kojih u drugim jeftinim proizvodima uopšte nema ili ne rade najbolje. Međutim MySQL-u nedostaju neke mogućnosti kao što su uskladištene procedure i pogledi, koje su standardne u najskupljim proizvodima, a postoje čak i u nekim jeftinim proizvodima. Neke od tih nedostajućih mogućnosti (kao što su uskladištene procedure) planirane su za naredne verzije, dok će za druge biti potrebno više vremena. Neke MySQL-ove mogućnosti kao npr. zaključavanje na nivou reda, ne postoje čak ni u najskupljim sistemima. Na web lokaciji MySQL-a nalazi se link <http://www.mysql.com/information/features.html>, na kojoj se može naći uporedni pregled mogućnosti MySQL-a i dvadesetak konkurenata. Kratak spisak mogućnosti koje ne podržavaju svi MySQL-ovi konkurenti:

- Transakcije u skladu sa grupom pravila ACID
- Podrška za različite platforme
- Replikovanje
- Podrška za obimne tabele i baze podataka
- Tekstualno pretraživanje
- Podupiti
- Podrška za većinu sintakse po standardu SQL 92

3 Model baze podataka agencije za promet nekretnina

3.1 Podaci koje čuvaju agencije za promet nekretnina

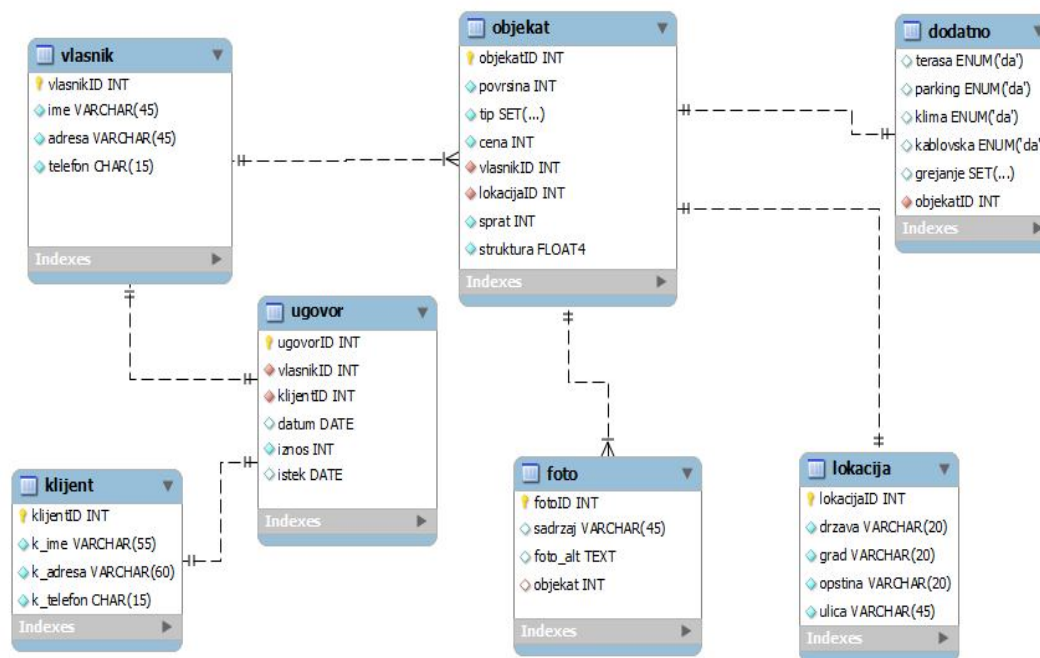
U našoj državi agencije za promet nekretnina su u privrednom registru registrovane za delatnost obavljanja prometa nepokretnostima.

Prema Zakonu o nepokretnostima u nepokretnosti spadaju: zemljište (poljoprivredno, građevinsko, šume i šumsko zemljište), zgrade, posebni delovi zgrada stanovi, poslovne prostorije, garaže i garažna mesta) i drugi građevinski objekti.

Pošto obavljaju poslove kupoprodaje i izdavanja nepokretnosti, agencije za promet nekretnina imaju potrebu da vode evidenciju o nepokretnostima, njihovim vlasnicima (ukoliko agencija nije vlasnik nekretnine, što uglavnom nije slučaj) i klijentima (odnosno kupcima ili klijentima koji su iznajmili nekretninu) i podacima vezanim za te poslove.

Agencije za promet nekretnina uglavnom vrše delatnost kao posrednici prilikom kupoprodaje ili izdavanja nekretnine.

Naknada za posredničke poslove definiše se ugovorom između vlasnika nepokretnosti i same agencije. To može biti procenat od vrednosti nepokretnosti ili na drugi načina definisana naknada.



Slika 1: Relacioni model baze podataka urađen u MySQL Workbench programu

Slika 1 grafički prikazuje šemu baze podataka agencije za promet nekretnina. Šema

je urađena u MySQL-ovom programu Workbench koji predstavlja grafički interfejs za realizaciju tabela i baza. Šema se može prikazati i tekstualno na sledeći način:

Vlasnik (**vlasnikID**, ime, adresa, telefon)

Lokacija (**lokacijaID**, država, grad, opština, ulica)

Objekat (**objekatID**, tip, površina, cena, sprat, struktura, vlasnikID, lokacijaID)

Dodatno (terasa, parking, klima, grejanje, kablovska, **objekatID**)

Foto (**fotoID**, sadrzaj, foto_alt, objekatID)

Klijent (**klijentID**, k_ime, k_adresa, k_telefon)

Ugovor (**ugovorID**, datum, iznos, istek, vlasnikID, klijentID)

Šema date baze podataka agencije za promet nekretnina ima sedam tabela. Kolone koje su ispisane podebljanim slovima predstavljaju primarne ključeve datih tabela, dok kolone koje su podvučene predstavljaju spoljne ključeve.

3.2 Pristup izradi modela

Prva stvar koju treba uraditi jeste neka vrsta istraživanja o samoj delatnosti firme ili sistema za koji projektujemo model baze podataka. U stvarnom životu ne postoje dva potpuno identična sistema, stoga svakom modelovanju treba pristupiti pojedinačno. Znači, treba uočiti i zapisati sve osnovne elemente, procese i odnose. Ti osnovni elementi koje ćemo da modelujemo su, u stvari, entiteti i relacije. Entiteti predstavljaju stvari iz stvarnog sistema o kojima čuvamo podatke u bazi podataka. U našem primeru, entitet je objekat, vlasnik objekta ili klijent o kojima čuvamo podatke u bazi. Svaki je entitet za sebe: vlasnik objekta je jedan entitet, objekat je drugi entitet, a klijent bi predstavljao treći entitet. Relacije su, u stvari veze između entiteta. One mogu biti različitih vrsta, pa tako imamo relacije tipa "jedan prema jedan", „jedan objekat naprema više“, i „više objekata prema više“. Relacija tipa „jedan prema jedan“ povezuje samo dva entiteta. Primer je da jedna nekretnina može biti samo na jednoj lokaciji. Primer za relaciju „jedan prema više“ jeste jedan vlasnik koji može posedovati više nekretnina ili jedna nekretnina može imati više fotografija u bazi.

MySQL je sistem za upravljanje relacionim bazama podataka, što znači da podržava baze u kojima postoje relacije između podataka koji se čuvaju u tabelama.

Kolone (atributi), predstavljaju kolone tabele koje opisuje određeni podatak koji se nalazi u svakom redu tabele. Reč klona i atribut koriste se kao sinonimi, ali tačnije je reći da je kolona deo tabele, a atribut se odnosi na entitet iz stvarnog sveta koji tabela modeluje.

Redovi tabele sadrže podatke iz kolona tabele. Redovi u tabeli se još nazivaju i zapisi ili n-torke. U našem primeru na slici 2 vidi se da svaki red u tabeli vlasnik predstavlja zapis o jednom vlasniku nekretnine.

```
mysql> use nekretnine;
Database changed
mysql> select * from vlasnik;
```

vlasnikID	ime	adresa	telefon
1	Mitar Mitrovic	Bajmoka 33	0113445223
2	Nebojsa Markovic	Vasina 23	011456322
3	Danijel Sretenovic	Rige od fere 2	011444321

```
3 rows in set (0.06 sec)
```

Slika 2: Prikaz tabele vlasnik

Ključevi su kolone koje identifikuju red tabele. Primarni ključ je kolona ili grupa kolona pomoću koje se može jednoznačno identifikovati pojedini red u tabeli. Mi hoćemo da u našem primeru odredimo kolonu po kojoj ćemo moći da identifikujemo pojedinačno svakog vlasnika nekretnine. U ovom slučaju to je prva kolona vlasnikID koja je numeričkog tipa i predstavlja unikatni redni broj zapisa. To je bolji ključ od kolone Ime jer se može dogoditi da dve osobe imaju isto ime i prezime. Kod izbora kolone za primarni ključ, najbolje je izabrati neku koja može imati jedinstvene vrednosti za svaki zapis.

Spoljni ključevi predstavljaju veze između tabela. To je uglavnom primarni ključ iz neke druge tabele kojim se napravi veza sa tom tabelom i svim podacima iz te tabele.

```
mysql> select * from ugovor;
```

ugovorID	datum	iznos	istek	vlasnikID	klijentID
1	2011-02-11	500	2011-03-11	1	3
2	2010-10-10	1000	2011-01-12	2	1

```
2 rows in set (0.00 sec)
```

Slika 3: Prikaz tabele ugovor

Redovi vlasnikID i klijentID u tabeli ugovor na slici 3 predstavljaju spoljne ključeve a odnose se na primarne ključeve u tabelama vlasnik i klijent.

3.3 Principi kod projektovanja baza podataka

Kod projektovanja baze podataka moramo sebi postaviti dva važna pitanja:

- Koje sve podatke treba čuvati, odnosno o kojim stvarima ili entitetima moramo da čuvamo podatke?
- Koje ćemo upite postavljati bazi podataka?

Na osnovu odgovora na ova pitanja možemo da napravimo bazu podataka čija će struktura biti takva da isključuje strukturne probleme kao što su redundantnost i anomalije podataka.

3.3.1 Redudantnost podataka

Kada projektujemo bazu podataka, to moramo uraditi tako da redudantnost (dupliranje) podataka bude minimalna, a da pri tome ne izgubimo nijedan podatak koji nam je neophodan. Redudantnim podatkom se smatra onaj koji se ponavlja u različitim redovima iste tabele ili u različitim tabelama jedne baze podataka. U našem primeru to bi moglo da se zamisli kad bi recimo u tabeli ugovor (slika 1) unosili podatke o klijentu i vlasniku (ime i prezime, adresu), umesto što sada imamo samo njihove ID vrednosti, pri čemu bi došlo do dupliranja podataka u bazi.

3.3.2 Anomalije podataka

Anomalije predstavljaju nešto složeniji problem i pojavljuju se kod loše osmišljene strukture baze podataka. Mogu se javiti kod dodavanja, brisanja i ažuriranja podataka. Recimo, u tabelu vlasnik uneli smo određenu osobu koja poseduje neku od nekretnina iz tabele objekat. Tabele vlasnik i objekat povezane su kolonom vlasnikID koja identifikuje jedinstvenog vlasnika. Ako želimo da promenimo vlasnika, kod ažuriranja može doći do anomalije ako nismo uveli pravilo referencijalnog integriteta da se pri ažuriranju jedne tabele ažurira automatski i druga tabela u kojoj se nalazi spoljni ključ. Isti problem može nastati kod brisanja zapisa iz tabele.

3.3.3 Vrednost Null

Još jedno pravilo dobrog projektovanja baza podataka nalaže da treba izbegavati šeme koje dozvoljavaju veliki broj nepopunjenih atributa. To nam govori da ti podaci nisu mnogo značajni za bazu (ako od 100 zapisa u jednoj koloni 95 imaju vrednost null), ili da su veoma malo zastupljeni, pa se umesto toga može napraviti nova tabela koja bi čuvala neki strani ključ i podatke iz kolone koji nisu null.

4 Kreiranje baze podataka agencije za promet nekretnina

4.1 Kreiranje tabela

Krenućemo sa kreiranjem tabela za našu bazu agencije za promet nekretninama. Pošto smo napravili šemu baze u kojoj postoji sedam tabela sada ćemo ih kreirati ručno u MySQL monitoru koji je uključen u sve instalacije MySQL-a, a komande se zadaju preko komandne linije. MySQL monitor nije jedini korisnički interfejs. Za pristup se mogu koristiti i MySQL Control Centar koji je grafički korisnički interfejs i zvanični je proizvod za rad sa MySQL-om, ali ne nalazi se na svim instalacijama. Možete ga preuzeti sa adrese: www.mysql.com/downloads/mysqlcc.html. Treći najzastupljeniji korisnički interfejs jeste phpMyAdmin, koji je web korisnički interfejs za upotrebu MySQL-a. Veoma je popularan kod hosting kompanija koje svojim korisnicima stavljaju na raspolaganje MySQL radi razvijanja web aplikacija. Za phpMyAdmin potrebno je da imate funkcionalan web server i instaliran PHP. No, da se vratimo MySQL monitoru. Pri startovanju potrebno je uneti lozinku koju smo zadali prilikom instalacije. Posle unosa lozinke možemo početi sa kreiranjem baze podataka.

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32 to server version: 5.0.16-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database nekretnine;
Query OK, 1 row affected (0.00 sec)

mysql> use nekretnine;
Database changed
mysql>
mysql> create table vlasnik(
    -> vlasnikID int not null auto_increment primary key,
    -> ime varchar(50),
    -> adresa varchar(45),
    -> telefon char(15)) type=InnoDB;
Query OK, 0 rows affected, 1 warning (0.17 sec)

mysql> create table lokacija(
    -> lokacijaID int not null auto_increment primary key,
    -> drzava varchar(20),
    -> grad varchar(20),
    -> opstina varchar(20),
    -> ulica varchar(45)) type=InnoDB;
Query OK, 0 rows affected, 1 warning (0.12 sec)

mysql> _
```

Slika 4: Kreiranje baze i tabela u MySQL monitoru

Kreiranje baze vrši se komandom:

CREATE DATABASE nekretnine;

pod uslovom da baza sa takvim imenom ne postoji u spisku baza kod MySQL-a. Ako niste sigurni da li neka baza sa tim imenom postoji onda se pre comande create database može zadati komanda: drop database if exists nekretnine; što znači da će se takva ako eventualno postoji obrisati.

Komanda : USE nekretnine; koristi se da bi selektovala bazu sa kojom želimo da radimo.

Izraz CREATE TABLE vlasnik (kolone tabele); kreira tabelu vlasnik sa zadatim kolonama, gde svaka kolona mora imati definisan tip podataka.

Tabela 1: Osnovni tipovi podataka koji se koriste u tabelama

Numerički tip	Znakovni tip	Vremenski tip
INT	CHAR	DATE
DECIMAL	VARCHAR	TIME
FLOAT	TEXT	DATETIME
DOUBLE	BLOB	TIMESTAMP
	ENUM	YEAR
	SET	

Kao što si i vidi u tabeli 1, postoji mnogo različitih tipova za smeštanje podataka.

Osnovni tipovi koji se koriste su:

- Numerički tipovi: tinyint, smallint, int, mediumint, bigint, numeric, decimal, float i double
- Znakovni tipovi: char, varchar, text i blob
- Za rad sa datumima i vremenima: date, time, datetime, timestamp i year
- Ostali tipovi podataka

Odabirom pravih tipova, tabele se optimizuju. Racionalnije se koristi prostor za memoriju podataka, a samim tim i realizovanje upita, odnosno sam odziv rezultata pretrage može se značajno ubrzati. Pravilo je upotrebiti najmanji tip podataka koji je dovoljan za unos podataka sa kojima će se raditi. Na primer, ako se u koloni tabele čuvaju brojevi od

jedan do deset, nije neophodno zadavati tip **int** , već tip **tinyint** . Što su kraći redovi i manje tabele, to će se one brže pretraživati.

Isto važi i za podatke tipa **varchar**, kod kojih se mora deklarirati maksimalna dužina podatka koji se može uneti. Može se zadati i tip **char**, koji predstavlja znakovnu vrednost fiksne dužine. Bez obzira da li je zadato **varchar** ili **char**, način rada sa tim podacima se neće razlikovati. Jedina razlika je oblik u kojem se podaci smeštaju u memoriju. Podatak tipa **varchar(15)**, zauzima onoliko prostora koliko je potrebno za smeštanje stvarnog broja karaktera, dok podatak tipa **char(15)**, uvek zauzima petnaest znakova u memoriji, bez obzira na to da li je u njega uneta bilo kakva vrednost.

Prva kolona koja je kreirana u tabeli vlasnik je kolona vlasnikID (slika 4). Ona je tipa **int** (eng. integer), a posle te deklaracije tipa slede ostale deklaracije koje određuju tu kolonu. Zadali smo da je kolona vlasnikID **not null** – drugim rečima, u svakom redu tabele ta kolona mora sadržati neku vrednost. Zatim smo zadali **auto_increment**, čime smo omogućili da svaki naredni red koji unesemo u kolonu dobija jedinstvenu vrednost po redosledu niza. **Primary key** znači da će ta kolona biti primarni ključ te tabele. Ako tabela ima samo jedan primarni ključ, onda se može zadati ovako. U slučaju da tabela ima dva ili više primarnih ključeva, onda se sintaksa razlikuje u tome što se posebno navede **primary key()**, a u zagradi se navedu kolone koje čine primarni ključ.

Zatim smo deklarirali tipove za kolone ime, adresa i telefon, i time smo definisali tabelu vlasnik. Tabela vlasnik trebala bi da sadrži osnovne i kontakt podatke o vlasnicima nekretnina.

Tip tabele koji smo deklarirali je InnoDB tabela. Standardni tip tabele u MySQL-u je MyISAM tabela, i za nju nije neophodno dodati naredbu type. MyISAM tabele su često brže od tabele tipa InnoDB. To je njihova prednost, dok im je mana to što ne podržavaju spoljne ključeve i transakcije.

```
mysql>
mysql> create table objekat(
  -> objekatID int not null auto_increment primary key,
  -> tip set('kuca','stan','hala/garaza','poslovni prostor'),
  -> površina int not null,
  -> cena int not null,
  -> sprat int not null,
  -> struktura float,
  -> vlasnikID int references vlasnik(vlasnikID) on delete cascade on update cascade,
  -> lokacijaID int references lokacija(lokacijaID) on delete cascade on update cascade)
  -> type=InnoDB;
Query OK, 0 rows affected, 1 warning (0.19 sec)

mysql> create table dodatno(
  -> terasa enum('da','ne'),
  -> parking enum('da','ne'),
  -> klima enum('da','ne'),
  -> kablovska enum('da','ne'),
  -> grejanje set('etazno','struja','centralno'),
  -> objekatID int references objekat(objekatID) on delete cascade on update cascade,
  -> primary key(objekatID)) type=InnoDB;
Query OK, 0 rows affected, 1 warning (0.17 sec)

mysql>
```

Slika 5: Prikaz kreiranja tabela objekat i dodatno

Na slici 5 prikazano je kreiranje tabele objekat koja je napravljena za čuvanje osnovnih vrednosti objekta nekretnine i tabele dodatno za smeštanje nekih dodatnih karakteristika nekretnine. I jedna i druga tabela imaju spoljne ključeve koji referenciraju na kolone u drugim tabelama. Tako je tabela objekat povezana sa tabelama Vlasnik i Lokacija, dok je tabela dodatno povezana sa tabelom Objekat.

Tip podataka ENUM, u zagradi definiše vrednosti koje se mogu nalaziti u redovima te kolone, ali može i sadržati vrednost null, ukoliko nije uneta nijedna od navedenih.

Tip podataka SET, takođe u zagradi definiše vrednosti koje se mogu nalaziti u redovima te kolone, s tim što može sadržati više opcionih vrednosti od tipa podataka ENUM.

Kolone koje su spoljni ključ u ovom primeru definisane su sa službenom reči references, što znači da pokazuju na koju tabelu a zatim u zagradi, na koju tačnu kolonu u toj tabeli se odnose. Tako, u našem primeru, kolona vlasnikID u tabeli objekat odnosi se na istoimenu kolonu u tabeli Vlasnik. Ime ove kolone koja predstavlja spoljni ključ moglo je biti i različito od kolone na koju se odnosi, ali meni je više odgovaralo da bude isto, što ću pokazati u narednim primerima kada se budu kreirali upiti za bazu.

Da objasnimo, reči ON UPDATE i ON DELETE govore nam o tome šta će se desiti sa podacima u koloni ako se ona kolona na koju oni referenciraju ažurira ili briše. Pa se tako može setovati da to bude cascade, restrict ili set null. Ako setujemo sa cascade, onda će se promena u glavnoj tabeli, izvršiti i u koloni sa spoljnim ključem. Znači promena u glavnoj dovodi do promene u zavisnoj povezanoj tabeli. Komanda restrict znači da će biti odbijeno da se izvrši promena u zavisnoj tabeli, dok komanda set null označava da će u slučaju ažuriranja ili brisanja zapisa u glavnoj tabeli dovesti do toga da vrednosti zapisa u

zavisnoj tabeli uzmu vrednost null. Ovim navedenim komandama se definišu pravila referencijalnog integriteta za bazu podataka.

Ostalo nam je da kreiramo tabele foto, za eventualno smeštanje fotografija nekretnina, zatim klijent i ugovor.

Tabelom klijent agencija za promet nekretnina čuvaće podatke o klijentima koji su kupili ili iznajmili neku nekretninu koja je u vlasništvu agencije ili vlasnika nekretnine.

Tabela ugovor bi trebalo da čuva podatke o stranama koje su potpisale ugovor, zatim iznosu, datumu zaključivanja ugovora kao i o isteku ugovora ako se radi o zakupu.

```
mysql> create table foto(
  -> fotoID int not null auto_increment primary key,
  -> sadrzaj varchar(45),
  -> foto_alt text,
  -> objekatID int references objekat(objekatID)) type=InnoDB;
Query OK, 0 rows affected, 1 warning (0.11 sec)

mysql> create table klijent(
  -> klijentID int not null auto_increment primary key,
  -> k_ime varchar(50),
  -> k_adresa varchar(60),
  -> k_telefon char(15)) type=InnoDB;
Query OK, 0 rows affected, 1 warning (0.12 sec)

mysql> create table ugovor(
  -> ugovorID int not null auto_increment primary key,
  -> datum date,
  -> istek date,
  -> iznos int,
  -> vlasnikID int references vlasnik(vlasnikID),
  -> objekatID int references objekat(objekatID ))
  -> type= InnoDB;
Query OK, 0 rows affected, 1 warning (0.14 sec)

mysql>
```

Slika 6: Prikaz kreiranja tabela foto, klijent i ugovor

Pošto smo kreirali sve tabele iz naše pretpostavljene šeme sa slike 1 sada možemo pogledati i njihovu strukturu u MySQL-u. Za prikaz strukture tabela u bazi koristićemo komandu **desc**.

```
mysql> desc vlasnik;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| vlasnikID | int(11) | NO | PRI | NULL | auto_increment |
| ime | varchar(50) | YES | | NULL | |
| adresa | varchar(45) | YES | | NULL | |
| telefon | char(15) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.10 sec)
```

Slika 7: Prikaz strukture tabele vlasnik

```
mysql>
mysql> desc lokacija;
```

Field	Type	Null	Key	Default	Extra
lokacijaID	int(11)	NO	PRI	NULL	auto_increment
drzava	varchar(20)	YES		NULL	
grad	varchar(20)	YES		NULL	
opstina	varchar(20)	YES		NULL	
ulica	varchar(45)	YES		NULL	

5 rows in set (0.06 sec)

Slika 8: Prikaz strukture tabele lokacija

```
mysql> desc objekat;
```

Field	Type	Null	Key	Default	Extra
objekatID	int(11)	NO	PRI	NULL	auto_increment
tip	set('kuca','stan','hala/garaza','poslovni prostor')	YES		NULL	
povrsina	int(11)	NO			
cena	int(11)	NO			
sprat	int(11)	NO			
struktura	float	YES		NULL	
vlasnikID	int(11)	YES		NULL	
lokacijaID	int(11)	YES		NULL	

8 rows in set (0.00 sec)

Slika 9: Prikaz strukture tabele objekat

```
mysql> desc dodatno;
```

Field	Type	Null	Key	Default	Extra
terasa	enum('da','ne')	YES		NULL	
parking	enum('da','ne')	YES		NULL	
klima	enum('da','ne')	YES		NULL	
kablovska	enum('da','ne')	YES		NULL	
grejanje	set('etazno','struja','centralno')	YES		NULL	
objekatID	int(11)	NO	PRI	0	

6 rows in set (0.00 sec)

```
mysql>
```

Slika 10: Prikaz strukture tabele dodatno

```
mysql>
mysql> desc foto;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| fotoID | int(11) | NO   | PRI | NULL    | auto_increment |
| sadrzaj | varchar(45) | YES |     | NULL    |               |
| foto_alt | text   | YES |     | NULL    |               |
| objekatID | int(11) | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

Slika 11: Prikaz strukture tabele foto

```
mysql>
mysql> desc klijent;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| klijentID | int(11) | NO   | PRI | NULL    | auto_increment |
| k_ime | varchar(50) | YES |     | NULL    |               |
| k_adresa | varchar(60) | YES |     | NULL    |               |
| k_telefon | char(15) | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

Slika 12: Prikaz strukture tabele klijent

```
mysql>
mysql> desc ugovor;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ugovorID | int(11) | NO   | PRI | NULL    | auto_increment |
| datum | date   | YES |     | NULL    |               |
| istek | date   | YES |     | NULL    |               |
| iznos | int(11) | YES |     | NULL    |               |
| vlasnikID | int(11) | YES |     | NULL    |               |
| objekatID | int(11) | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Slika 13: Prikaz strukture tabele ugovor

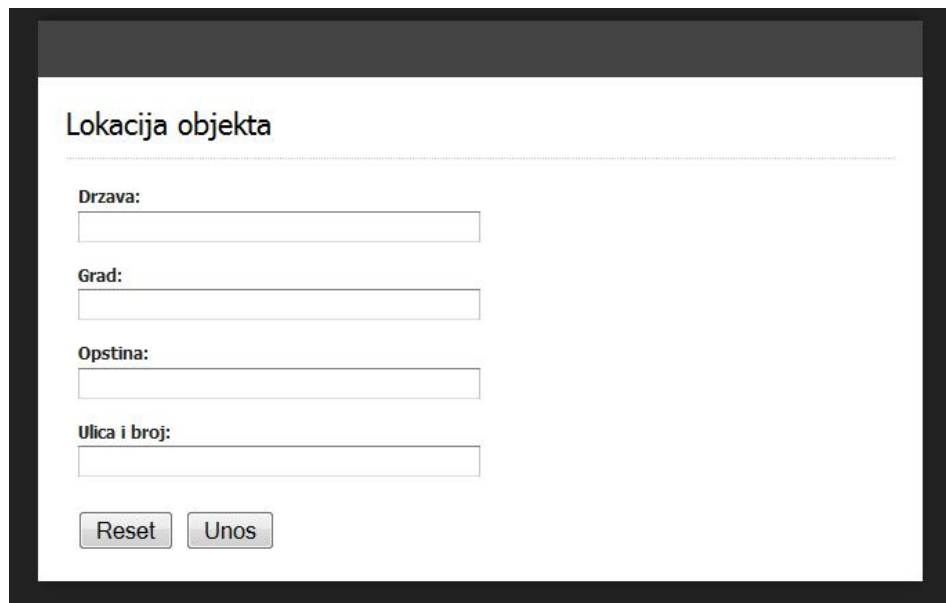
Nakon što smo ilustrovali izgled strukture tabela koje smo kreirali u bazi agencije za promet nekretnina, sada preostaje da popunimo tabele podacima.

Uzećemo pretpostavku da prosečan službenik agencije za promet nekretnina nema znanje i iskustvo u korišćenju SQL programa, pa ćemo morati da napravimo neku aplikaciju, ili neku formu za unos podataka, kako bi korisnicima zaposlenim u agenciji olakšali korišćenje same baze podataka.

4.2 Forme za unos podataka

Forme za unos podataka u bazu mogu biti jedna ili više html stranica, koje bi trebalo da sadrže polja koja postoje kao nazivi kolona u tabelama. Forme, znači treba napraviti po uzoru na strukturu tabele koju želimo da popunjavamo podacima.

Forma predstavlja i dobru kontrolu kod unosa podataka, pošto se na klijent strani prilikom unosa može ograničiti kakvi sve podaci mogu da se unesu u bazu. To se obično radi sa padajućim meni listama, zatim označavanjem checkbox ili radio dugmića, a može i javascript-om da se ograniči unos slova u polje rezervisanog za brojeve ili recimo definiše se uzorak (patern) za polje u koje se unosi adresa elektronske pošte, pa ako niste upisali znak @ ili imate zarez umesto tačke, unos neće biti dozvoljen.



Slika 14. izgled forme za unos podataka u tabelu Lokacija

Forma za unos podataka na slici 14 predstavlja rešenje za unos podataka u tabelu Lokacija i sadrži četiri polja za unos koja su tekstualnog tipa (za unos string varijabli).

Forma ima i dva dugmeta. Prvo dugme Reset je za poništavanje nepravilno unetih podataka u polje. Pritiskom na ovo dugme brišu se uneti stringovi u poljima i omogućava ponovni unos.

Drugo dugme Unos služi za slanje podataka na server (Apache server), gde ćemo preko PHP koda poslati podatke dalje u MySQL bazu podataka. Ovo dugme Unos iskoristićemo i kao link za prelaz na sledeću formu za unos, pošto imamo još tabela koje treba popuniti podacima.

```
<form method="post" action="unos1.php">  
  <div>
```

```
                <h2>Lokacija objekta</h2>
                <p></p>
            </div>
            <ul >
<li >
<label >Drzava: </label>
    <div><input name="drzava" type="text" maxlength="255" value=""/> </div>
    </li>
<li ><label >Grad: </label>
    <div> <input name="grad" type="text" maxlength="255"          value=""/> </div>
</li>
<li ><label >Opstina: </label>
    <div> <input name="opstina" type="text" maxlength="255"          value=""/>
</div>
</li>
<li > <label >Ulica i broj: </label>
    <div><input name="ulica" type="text" maxlength="255" value=""/> </div>
</li>
<li>
<input type="reset" name="reset" value="Obrisi" />
<input type="submit" name="submit" value="Unos" />
</li>
            </ul>
        </form>
```

Ovako bi trebalo da izgleda kôd forme koju smo videli na slici 14. u okviru html stranice, između tagova <body> i </body>.

Treba uočiti da je u tagu <form> metod setovan na „post“, a za action je zadata stranica sa imenom unos1 koja ima php ekstenziju. Metod je mogao biti setovan i kao „get“, ali bi se u tom sličaju promenljive prikazivale u adres baru prilikom učitavanja, a to smo ovim putem izbegli.

Php stranica unos1 poslužiće nam kao serverska izvršna stranica u koju ćemo ubaciti kôd za povezivanje sa našom MySQL bazom podataka i smeštanje podataka iz ove forme u tabelu Lokacija.

4.3 Kreiranje PHP stranice

Stranica za unos podataka u tabelu Lokacija, mogla bi da izgleda ovako:

```
<?php
$konekcija = mysql_connect("localhost","root","abc123");
if (!$con)
{
    die('Povezivanje sa bazom podataka nije moguće, došlo je do greške : ' . mysql_error());
}

mysql_select_db("nekretnine", $konekcija);

$sql="INSERT INTO Lokacija (Drzava, Grad, Opstina, Ulica)
VALUES
('$_POST[drzava]',$_POST[grad'],$_POST[opstina]', $_POST[ulica] )";

if (!mysql_query($sql,$konekcija))
{
    die('Greška: ' . mysql_error());
}
echo "1 zapis je dodan u tabelu Lokacija!";

mysql_close($konekcija)

header("Location: http://127.0.0.1/forma2.html");

?>
```

Pošto se radi o PHP fajlu, sav kôd se nalazi između znakova <?php i ?>, koji predstavljaju početak i završetak PHP stranice.

Pre nego što počnemo, potrebno je znati naziv servera na kome se nalazi baza podataka (ukoliko radite u lokalnu, naziv će biti "local", kao što je u ovom primeru), validno korisničko ime i lozinka za taj server. Zatim, startujemo sa PHP kôdom, kreiranjem konekcione varijable:

```
$konekcija = mysql_connect
("naziv_servera","korisničko_ime","lozinka")

or die("Povezivanje nije moguće.");
```

Funkcija die() koristi se za prekidanje skripta i štampanje poruke o grešci ukoliko prethodna funkcija propadne (u ovom slučaju, ako konekcija nije uspostavljena).

Kada uspostavite vezu, tj. konekciju, sledeći korak predstavlja izbor baze podataka sa kojom želimo da radimo, tj. u koju hoćemo da smestimo podatke iz forme. U našem primeru baza podataka na MySQL serveru naziva se Nekretnine, pa ćemo tako i uneti u funkciju za odabir baze:

```
$db = mysql_select_db("nekretnine", $konekcija)  
or die("Baza nije dostupna!");
```

MySQL ne pravi razliku između malih i velikih slova, tako da se ime baze može ispisati i malim slovima.

Funkcije koje smo koristili su:

- `mysql_connect` – funkcija kojom ostvarujemo vezu sa MySQL serverom; zahteva naziv hosta, korisničko ime i lozinku.
- `mysql_select_db` – funkcija za izbor baze podataka od mnogobrojnih koje se nalaze na MySQL serveru.
- `mysql_query` – funkcija za postavljanje SQL instrukcija.
- `mysql_close` – funkcija za prekid veze ka bazi podataka.

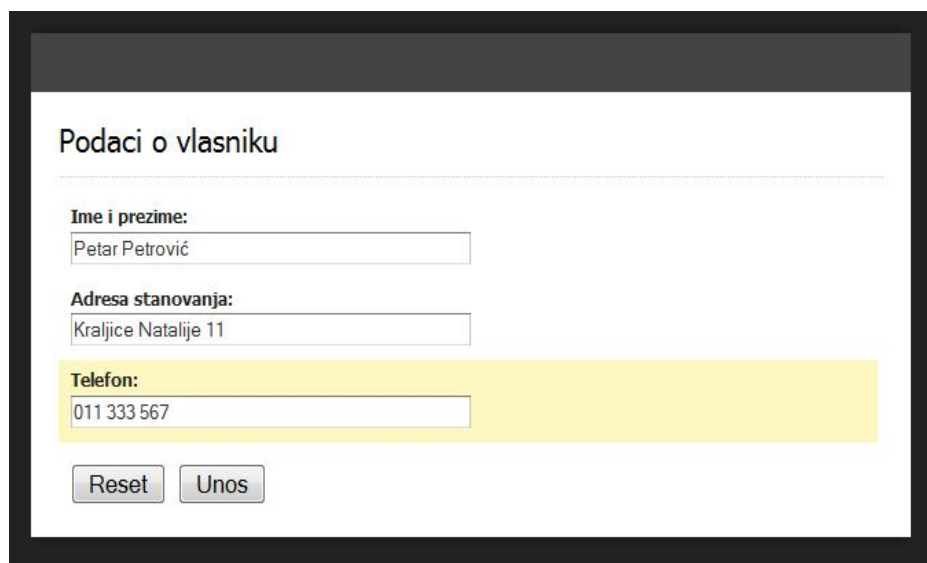
`Insert into` (imena kolona) `values` (vrednosti kolona) je standardna SQL komanda za unos podataka u tabelu. U našem primeru, preko `post` metoda pokupljeni su podaci uneti u formu i sa ovom komandom su uneti u tabelu.

Na kraju imamo redirekciju, preusmeravanje, na novu formu za unos podataka u drugu tabelu u našoj bazi za promet nekretnina.

```
header("Location: http://127.0.0.1/forma2.html");
```

Ovde se vidi da su naše html i php stranice smeštene na lokalnom apache serveru (bazično u folderu `htdocs`), mada one mogu biti smeštene na bilo kom serveru (i na internet mreži) koji ima podršku za MySQL baze podataka. Tako bismo dobili pristup bazi sa bilo kog kompjutera, sa bilo kog mesta.

Sada ćemo napraviti formu za unos podataka u tabelu `Vlasnik`. Njen izgled videćemo na slici 15.



Slika 15: Izgled forme za unos podataka u tabelu Vlasnik

Forma koju smo napravili služiće nam za unos podataka u tabelu Vlasnik, gde ćemo čuvati podatke o vlasniku nekretnine. Forma je jednostavna i sadrži tri polja za unos sa tastature. Ime i prezime, adresa i broj telefona osobe su polja koja imamo u tabeli Vlasnik u našoj MySQL bazi. Tu su i dva dugmeta za submit i reset podataka. Treba reći da u tabeli Vlasnik postoji i polje VlasnikID, ali ga nećemo staviti u formu. To je polje definisano kao auto_increment, i zato će to polje automatski dobiti vrednost pri unosu podataka u tabelu Vlasnik. Ovo polje će dobiti vrednost uvećanu za jedan, u odnosu na poslednji uneti zapis. Tako npr. ako je poslednja vrednost VlasnikID bila 101, sledeći zapis će dobiti vrednost 102 i tako redom.

Podatke u ostalim tabelama unećemo na isti način, kao što smo uneli i u ovom primeru kod tabele Lokacija. Može se malo razlikovati postupak unošenja prilikom unosa iz forme koja sadrži radio dugmad ili padajuću listu, ili neki drugi tip u formi, mada se povezivanje sa bazom i unos preko funkcije obavlja na isti način.

4.3.1 Unos fotografija

Kod osmišljavanja izgleda baze podataka za agenciju koja se bavi prometom nekretnina, prepostavili smo da bi mogli čuvati i fotografije objekta. Zato smo i napravili jednu tabelu Foto, koja će poslužiti za tu svrhu.

MySQL ima mogućnost za smeštanje fotografija u baze podataka. Tip podataka u koji se smeštaju fotografije je BLOB. Ovaj tip podataka koristi se za veće podatke kao što su: zvuk, fotografije, video zapis i slično.

Prilikom unosa fotografija u bazu preko BLOB tipa, vrši se njihova konverzija u binarni oblik, dok se kod eksportovanja opet vrši konverzija iz binarnog oblika. Pošto su podaci koji se unose memorijski veliki, mora se prilikom unosa smisliti način da se smanje ili kompresuju. To se može uraditi na više načina, na primer rezolucija i veličina fotografija se mogu smanjiti prilikom unosa php skriptom, a može se i ograničiti maksimalna veličina fajla koji se unosi.

Fotografije možemo na dva načina povezati sa bazom a to su:

- Možemo fotografije direktno uneti u bazu podataka u binarnom obliku – blob
- Možemo fotografije smestiti u neki folder na serveru, a u bazu upisati samo putanju do fotografije

Predpostavimo da mi u našem primeru ne koristimo samo MySQL, već imamo i apache web server i php podršku, kao što smo dosad unosili podatke preko web formi a ne preko MySQL Monitor interfejsa. U tom slučaju ne moramo deponovati fotografije u samu MySQL bazu, već ih možemo kopirati u folder na serveru, a u bazi možemo čuvati samo putanju do njih u vidu stringa. Ovakvo rešenje se pokazalo kao mnogo kvalitetnije u velikom broju slučajeva kod raznih web servisa koji imaju podršku baza podataka. Kao što u programiranju ne morate na svakom mestu da pravite identičan objekat, dovoljno je da napravite pokazivač na taj objekat, tako i mi u našem primeru u tabeli Foto čuvaćemo samo putanju do fotografije koju smo uneli u folder na serveru.

Naravno, ovaj unos (kopiranje) fotografija u folder na našem serveru, izvršićemo tako što ćemo kreirati jednu php stranicu koju ćemo nazvati upload.php i smestiti je u folderhtdocs na našem serveru. Folder za smeštanje fotografija nazvaćemo fotos.

The screenshot shows a web form titled "Unos fotografija" with a subtitle "Moguće je uneti maksimalno 6 fotografija, koje moraju biti u jpg ili gif formatu". At the top, there is a dropdown menu labeled "Uneti ID objekta". Below this, there are six identical groups of input fields. Each group consists of a text input labeled "Slika 1:" through "Slika 6:", a text input labeled "Naziv:", and a "Browse..." button next to the "Slika" input. At the bottom of the form, there is a button labeled "Unesi fotografije".

Slika 16: Izgled forme za unos podataka u tabelu Foto

Forma za unos fotografija u tabelu Foto, ima šest polja za unos čime smo ograničili broj fotografija koje korisnik može da unese.

Data je napomena za format fotografije u kojem moraju biti unete, u našem slučaju jpg i gif kao dva najzastupljenija formata fotografija u elektronskom obliku.

Kôd za stranicu upload.php koja se poziva pritiskom da dugme Unos mogao bi da izgleda ovako:

```
<?php
    $mysql_link = mysql_connect("127.0.0.1", "root", "test123");
    mysql_select_db("nekretnine") or die("Could not select database");

    $images_dir = "fotos";
    // inicijalizacija
    $konacni_rezultat= "";
    $brojac = 0;
    // tipovi fotografija za proveru
    $tipovi_fotografija = array( 'image/pjpeg' => 'jpg',
                                'image/jpeg' => 'jpg',
                                'image/gif' => 'gif');

    // GD Function List
    $gd_function_suffix = array( 'image/pjpeg' => 'JPEG',
                                'image/jpeg' => 'JPEG',
                                'image/gif' => 'GIF');

    // niz fotografija unetih preko forme
    $unete_foto = $_FILES['photo_filename'];
    // niz naziva koji su uneti u formi
    $naziv = $_POST['naziv'];
    while( $brojac <= count($unete_foto) )
        {if($unete_foto['size'][$brojac] > 0)
            {if(!array_key_exists($unete_foto['type'][$brojac], $tipovi_fotografija))
                {$konacni_rezultat.= "Fajl ".($brojac+1). " nije fotografija!<br />";}
            else
                {
                    mysql_query( "INSERT INTO foto(`photo_filename`, `naziv`, `photo_category`) VALUES('0',
                    '".addslashes($naziv[$brojac])."', '".addslashes($_POST['category'])."' )" );
                    $new_id = mysql_insert_id();
                    $filetype = $unete_foto['type'][$brojac];
                    $sextention = $tipovi_fotografija[$filetype];
                    $filename = $new_id." ".$sextention;
                    mysql_query( "UPDATE foto SET photo_filename='".addslashes($filename)."' WHERE
                    photo_id='".addslashes($new_id)."' );
                    // originalni fajl
                    copy($unete_foto['tmp_name'][$brojac], $images_dir."/".$filename);

                    // velicina thumbnail slikice
                    $size = GetImageSize( $images_dir."/".$filename );
                    if($size[0] > $size[1])
                        {
                            $thumbnail_width = 100;
                            $thumbnail_height = (int)(100 * $size[1] / $size[0]);
                        }
                    else
                        {
                            $thumbnail_width = (int)(100 * $size[0] / $size[1]);
                            $thumbnail_height = 100;
                        }

                    $function_suffix = $gd_function_suffix[$filetype];
```

```
$function_to_read = "ImageCreateFrom".$function_suffix;
$function_to_write = "Image".$function_suffix;

// očitava fajl
$source_handle = $function_to_read ( $images_dir."/". $filename );

if($source_handle)
{
    // prazan sablon za thumbnail
    $destination_handle = ImageCreate ( $thumbnail_width, $thumbnail_height );

    // Sada je skratimo(kreiramo thumbnail)
    ImageCopyResized( $destination_handle, $source_handle, 0, 0, 0, 0, $thumbnail_width,
    $thumbnail_height, $size[0], $size[1] );
}

// sacuvamo thumbnail
$function_to_write( $destination_handle, $images_dir."/tb_". $filename );
ImageDestroy($destination_handle );
//

$konacni_rezultat.= "<img src='".$images_dir. "/tb_". $filename.'" /> Fajl ".$brojac+1)." unet!<br
/>";
}
}
$brojac++;
}
echo <<<__HTML_END
<html>
<head>
<title>Unos fotografija</title>
</head>
<body>
$result_final
</body>
</html>
__HTML_END;
?>
```

Objašnjenje funkcija ovog php fajla je ovakva. Prvo je napravljena veza sa MySQL serverom, a zatim je izabrana baza sa kojom se radi. Zatim je dat naziv za folder na serveru gde će se smeštati fotografije.

Potom su inicijalizovani tipovi fotografija preko kojih će se proveravati da li su uneti fajlovi stvarno fotografije ili ne, preko provere ekstenzija.

Onda su u nizovima uneti fajlovi i njihovi nazivi iz polja koja se nalaze u formi za unos.

Fotografije koje smo uneli smanjili smo na male (100x100 piksela) i napravili paralelno i thumbnail-ove koje ćemo kasnije koristiti prilikom pregleda. Klikom na ove male, dobićemo kasnije kad budemo radili upite, fotografije u originalnoj veličini.

4.3.2 Napomene prilikom unosa podataka

Videli smo kako možemo uneti podatke preko formi koje smo napravili za naše tabele u bazi. Jednostavno, svako polje u formi za unos predstavlja podatak koji se preko php funkcija prosledi MySQL bazi podataka.

Unos primarnog ključa smo u nekim tabelama rešili tako što smo tu kolonu u tabeli definisali kao `auto_increment` polje tako da prilikom unosa podataka, to polje može i automatski da se popuni.

Kolona koja je u jednoj tabeli primarni ključ, može u drugoj tabeli da bude spoljni ključ. Da bi se te dve tabele povezale, odnosno da bi referencirale jedna na drugu, potrebno je imati istu vrednost u oba slučaja. Kako izvesti da automatski unesen primarni ključ, bude isti u drugoj tabeli kao spoljni ključ?

Verovatno postoji više načina na koji se to radi. U ovom slučaju iskoristili smo funkciju **`mysql_insert_id()`**; koja ima mogućnost da prosledi poslednji uneti ID u predhodnoj formi, ili automatski generisan ID iz kolone koja je `auto_increment`. Tako se jednostavno može pozvati ovakva funkcija prilikom unosa spoljnog ključa.

Drugi, često korišćen način za ovo rešenje jeste i korišćenje funkcije koja vraća realno vreme unosa, pa se npr. podaci uneti u nekom vremenskom intervalu stave pod jedinstveni ključ koji je generisan iz datuma ili vremena unosa.

5 Realizacija upita

Da bismo mogli da očitamo podatke koji su sačuvani u tabelama baze podataka, moramo postaviti upite nad tabelama. Službena reč koja se koristi kod upita je SELECT.

Njegov oblik izgleda ovako:

```
SELECT ime kolone  
FROM ime tabele  
[ WHERE uslovi ]  
[ GROUP BY grupisanje ]  
[ HAVING uslov_grupe ]  
[ ORDER BY redosled ]  
[ LIMIT ograničenja ]  
[ PROCEDURE ime_procedure(argumenti) ];
```

5.1 Jednostavni upiti

Upiti koji se postavljaju nad tabelama mogu biti:

- Jednostavni
- Složeni upiti

Jednostavni upit je onaj koji se postavlja nad jednom tabelom i nije previše opterećen uslovima. Primer takvog upita može biti:

```
SELECT * FROM Vlasnik;
```

Jedan od korisnih elemenata jeste operator * kojim se biraju sve kolone iz navedene tabele ili tabela.

vlasnikID	ime	adresa	telefon
1	Petar Petrovic	Bulevar Oslobođenja 33	011 334 881
2	Nemanja Draskovic	Mutapova 12	021 775 883
3	Milena Matic	Cvijiceva 47	011 943 755
4	Mitar Trifunovic	Golubacka 1	013 137 225
5	Gorica Vasic	Pozeska 103	011 553 834
6	Dejan Petkovic	Nemanjina 30	011 765 889
7	Slavisa Stojanovic	Kursulina 19	011 235 996
8	Jovana Eric Popovic	Bajmocka 12	025 238 771

Slika 17: Prikaz rezultata upita

Naravno, suština relacione baze nije u tome da vam daje sve podatke koje ste u nju uneli, već da vam omogući da pronađete određene podatke.

Da biste izdvojili određeni podskup redova iz tabele, potrebno je da zadate uslov za njihov izbor. Tome služi odredba **WHERE**. Na primer:

```
SELECT ime, telefon FROM Vlasnik WHERE telefon LIKE '011%';
```

ime	telefon
Petar Petrovic	011 334 881
Milena Matic	011 943 755
Gorica Vasic	011 553 834
Dejan Petkovic	011 765 889
Slavisa Stojanovic	011 235 996
Ivan Despotovic	011 666 321

Slika 18: Prikaz rezultata upita sa uslovom

Na slici 18 možemo da vidimo da su izdvojeni podaci, imena iz tabele Vlasnik čiji brojevi telefona počinju sa 011. Koristili smo rezervisanu reč LIKE i operator % koji nam je poslužio kao džoker znak za pravljenje uzorka na osnovu kojeg je filtrirana kolona sa brojevima telefona.

Odredba ORDER BY sortira redove po redosledu vrednosti iz jedne ili više kolona navedenih u iskazu. Ako nije navedeno, podrazumeva se rastući redosled, a ako želimo da podaci budi sortirani po opadajućem redosledu, koristi se reč DESC iza navedenog imena kolone.

Odredbom LIMIT može se ograničiti broj redova rezultata koje daje upit. Ova odredba može sadržati i dva parametra, pri čemu je prvi broj reda od kog počinje učitavanje a drugi je broj koji predstavlja maksimalan broj redova rezultata. Naročito je korisna kod web aplikacija koje koriste MySQL jer omogućava jednostavan mehanizam podele rezultata na stranice.

Uklanjanje dupliranih vrednosti u rezultatima pretrage može se postići pomoću odredbe DISTINCT.

Osim grupisanja podataka i upotrebe grupnih funkcija, rezultat grupne funkcije možete porediti sa zadatom vrednošću pomoću odredbe HAVING. Ta odredba piše se neposredno iza odredbe GROUP BY i slična je odredbi WHERE, ali se odnosi samo na grupe podataka i rezultate grupnih funkcija.

Tabela 2: Ugrađene funkcije za upotrebu u naredbi GROUP BY

Funkcija	Namena
Avg (kolona)	Vraća prosek vrednosti sadržanih u datoj koloni
Count(kolona)	Vraća ukupan broj vrednosti u datoj koloni
Min(kolona)	Vraća najmanju vrednost u datoj koloni
Max(kolona)	Vraća najveću vrednost u datoj koloni
Std(kolona)	Vraća standardnu devijaciju za vrednosti sadržane u datoj koloni
Sum(kolona)	Vraća zbir vrednosti sadržanih u datoj koloni

Tabela 3: Operatori koji mogu da se koriste sa naredbom WHERE

Operator	Primer	Opis
=	VlasnikID = 3	Ispituje da li su vrednosti jednake
>	cena > 50.00	Ispituje da li je prva vrednost veća od druge
<	cena < 50.00	Ispituje da li je prva vrednost manja od druge
>=	cena >= 50.00	Ispituje da li je prva vrednost veća od druge ili su jednake
<=	cena <= 50.00	Ispituje da li je prva vrednost manja od druge ili su jednake
!= ili <>	sprat != 0	Ispituje da li su dve vrednosti različite
IS NOT NULL	adresa is not null	Ispituje da li polje sadrži bilo kakvu vrednost
IS NULL	adresa is null	Ispituje da li je polje prazno
BETWEEN	iznos between 0 and 50.000	Ispituje da li je vrednost u zadatom opsegu
IN	grad in ("Srbija")	Ispituje da li je vrednost u skupu navedenih vrednosti
NOT IN	grad not in ("Srbija")	Ispituje da li je vrednost izvan skupa navedenih vrednosti

LIKE	ime like ("Marko %")	Ispituje da li postoji SQL podudarnost sa zadatim uzorkom
NOT LIKE	ime not like ("Marko %")	Proverava da li se vrednost ne podudara sa zadatim uzorkom
REGEXP	ime regexp	Ispituje da li postoji podudarnost sa zadatim regularnim izrazom

Poslednja tri reda u tabeli odnose se na operatore LIKE i REGEXP koji omogućavaju prepoznavanje uzoraka.

Operator LIKE omogućava SQL prepoznavanje jednostavnih uzoraka. Uzorci mogu da se sastoje od običnog teksta i znaka za procenat (%), koji predstavlja bilo koji broj znakova, odnosno podvlake (_), koja predstavlja samo jedan znak.

REGEXP omogućava prepoznavanje uzoraka pomoću regularnih izraza. Umesto rezervisane reči REGEXP možete upotrebiti njen sinonim RLIKE. Regularni izrazi u formatu POSIX takođe su podržani u PHP-u.

5.2 Složeni upiti

Složeni upiti izvršavaju se nad više tabela istovremeno, pa je potrebno tabele spojiti radi manipulacije nad podacima. Tabele koje povezujemo navodimo u odredbi FROM, zajedno sa vrstom spoja. Takođe treba zadati i uslov koji opisuje način na koji bi tabele trebalo da budu spojene. Primer:

```
SELECT ime.Vlasnik, telefon.Vlasnik, tip.Objekat, cena.Objekat FROM
```

```
Vlasnik, Objekat WHERE vlasnikID.Vlasnik = vlasnikID.Objekat;
```

U ovom primeru je zadat upit nad tabelama Vlasnik i Objekat, gde je uslov za spajanje tabela jednakost između primarnog ključa u tabeli Vlasnik i spoljnog ključa u tabeli Objekat. U odredbi FROM lista tabela je zadata zarezom između tabela. Umesto zareza može se koristiti rezervisana reč JOIN koja je sinonim.

Kada se zada ovakva vrsta spoja, MySQL pretražuje sve tabele koje su zadate i pokušava da pronađe najefikasniji način spajanja, pri čemu ne mora da spoji tabele redosledom na koji ste naveli. Ako hoćemo da to bude tačno po redosledu koji je naveden onda se koristi rezervisana reč STRAIGHT JOIN.

5.2.1 Levi i desni spoj

Spojevi sa JOIN rade u slučaju da su spojevi koje smo zadali jednakovredni. Šta se dešava kada želimo da pronađemo sve redove u jednoj tabeli za koju ne postoje odgovarajući redovi u drugoj tabeli?

Tada se koriste levi ili desni spoj. Levi spoj radi tako što za sve redove tabele iz prve koja je zadata (leva tabela), traži odgovarajuće redove u tabeli koja je pridružena. Pronađeni redovi se postavljaju pored leve tabele, a za svaki red koji nema parnjaka u desnoj tabeli, operator LEFT JOIN dodaje vrednost NULL.

5.3 Pristup bazi podataka pomoću PHP-a

U našem pristupu bazi nekretnina pomoću PHP-a odradićemo sledeće korake:

- Proveriti i filtrirati podatke koji stižu od korisnika
- Uspostaviti vezu sa odgovarajućom bazom podataka
- Zadati upit toj bazi
- Učitati rezultate
- Prikazati rezultate korisniku.

Neki od ovih koraka radili smo i prilikom unosa podataka u table baze i oni se neće previše razlikovati ni prilikom uspostavljanja upita preko neke vrste forme. To su uglavnom odnosi na uspostavljanje veze sa serverom i odabirom baze podataka sa kojom će se raditi. Provera podataka unetih od strane korisnika isto tako se u suštini ne razlikuje previše, pa ćemo obratiti pažnju na ono što dosad nije obrađeno.

Napravićemo jednostavan primer za ispis svih podataka iz tabele Vlasnik.

```
<?php
$con = mysql_connect("localhost","root","test123");
if (!$con)
{
    die('Nije uspela konekcija: ' . mysql_error());
}

mysql_select_db("nekretnine", $con);

$resultat = mysql_query("SELECT * FROM Vlasnik
LIMIT 5 ");

while($red = mysql_fetch_array($resultat))
{
    echo $red['vlasnikID'] . " " . $red['ime'] . " " . $red['adresa'] . " " . $red['telefon'];
    echo "<br />";
}
?>
```

Treba primetiti da je funkcija **mysql_query()** obradila sql upit nad tabelom Vlasnik i prosledila niz u promenjivu rezultat. Kasnije smo tu promenjivu rezultat iskoristili kao parametar funkcije **mysql_fetch_array()** koja je niz podataka pridružila varijabli red, koja predstavlja svaki zapis u tabeli Vlasnik.

Da varijablu red nismo stavili u petlju, dobili bismo samo prvi zapis iz tabele Vlasnik. Pošto postoji petlja, možemo da ispišemo svih pet zapisa iz tabele Vlasnik.

5.4 Ažuriranje zapisa

SQL komanda UPDATE omogućava ažuriranje postojećih redova u tabelama baze podataka.

U MySQL-ovoj dokumentaciji stoji da komanda UPDATE ima sledeći oblik:

```
UPDATE [LOW_PRIORITY] [IGNORE] ime_tabele
```

```
SET ime_kolone1 = izraz1, [ime_kolone2 = izraz2,...]
```

```
[WHERE uslov] [ORDER BY...] [LIMIT broj_redova]
```

Opcija WHERE može se dodati da bi se ažurirali samo određeni redovi u zadatoj tabeli. U suprotnom, ako se izostavi, ažuriraće se svi redovi tabele.

Komanda UPDATE omogućava ažuriranje i više tabela istovremeno. Ažuriraju se samo one kolone koje su navedene u odredbi SET.

Sada ćemo napraviti primer koji odgovara našem modelu baze podataka. Izabraćemo npr. tabelu Vlasnik i ažurirati broj telefona vlasnika, ukoliko je on u međuvremenu promenio broj, pa imamo potrebu da to promenimo i u našoj bazi.

```
UPDATE Vlasnik SET telefon='011 342 111' WHERE vlasnikID= 4;
```

U datom primeru ažuriraćemo broj telefona vlasnika koji je upisan u tabelu pod brojem 4 i promenićemo njegov telefonski broj brojem koji je naveden u odredbi SET. Ovo je primer prostog ažuriranja jednog zapisa koji obuhvata jednu kolonu u jednoj tabeli. Kako bismo to mogli da uradimo preko PHP-a?

To bi moglo da se uradi ovako. Prilikom ispisa rezultata zadatog upita, mogli bismo u tabelu u koju smeštamo rezultate, da na kraju dodamo još jednu kolonu sa linkom na PHP stranicu na serveru koja izvršava ažuriranje. U zavisnosti od toga u kojem redu kliknemo na link za ažuriranje, otvoriće nam se novi prozor za unos podataka za dati zapis. Unete podatke prihvaćemo preko metode POST i ažurirati preko stranice na serveru podatke u našoj MySQL bazi.

6 Zaključak

Kao što smo mogli da vidimo u dosadašnjem radu MySQL predstavlja dobro rešenje za realizaciju jednog ovakvog projekta. Relativno je lak za upotrebu, i praktično može se koristiti za ovakve male baze u poslovnoj primeni, mada ima primera da ga koriste i veliki sistemi i organizacije.

Uz podršku PHP-a na serverskoj strani, znatno je olakšan rad krajnjim korisnicima za manipulaciju podacima koji se čuvaju u bazi.

Sve važnije funkcionalnosti u radu sa bazama podataka su obrađene i uspešno izvedene u navedenim primerima.

Sa još nekim dodacima ako što su dodela privilegija (kod unosa, čitanja i ažuriranja) i ograničavanje pristupa bazi neovlašćenim korisnicima može se dodatno poboljšati kontrola i čuvanje podataka. Isto se može postići korišćenjem neke vrste validacije u okviru sesije koja se može odraditi uz pomoć PHP-a. No, taj deo već spada u zaštitu i svaka kompanija koristi različite načine i ima svoju definisanu politiku vezanu za zaštitu podataka.

Pravljenje rezervne kopije baze (eng. backup), takođe bi bilo poželjno realizovati, u slučaju nekog oštećenja servera ili bilo kog drugog načina gubitka podataka.

Ovaj projekat baze podataka agencije za promet nekretnina predstavlja trenutni model. Pošto se u informatici stvari brzo menjaju, sve je podložno promeni i usavršavanju, tako da bi za par godina izrada mogla da izgleda sasvim drugačije. Jedino što ostaje isto jeste glavna ideja i svrha korišćenja.

Literatura

- [1] Luke Welling, Laura Thomson, **PHP and MySQL web development**, četvrto izdanje, Micro Kniga, Beograd, 2008.
- [2] Luke Welling, Laura Thomson, **Priručnik za MySQL**, Micro Kniga, Beograd, 2006.
- [3] Veinović Mladen, Goran Šimić, **Uvod u baze podataka**, Univerzitet Singidunum, Beograd, 2008

.