ПРОГРАМИРАЊЕ ВЕБ АПЛИКАЦИЈА

Nastavana jedinica: 8

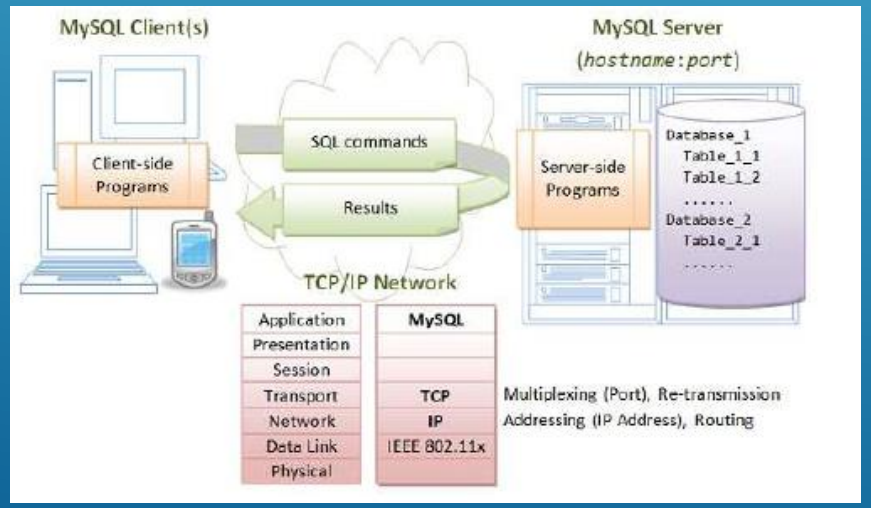
Rad sa bazama podataka

Pojam baze podataka

BAZA PODATAKA?

Osnovni pojmovi baze podataka

- Baza podataka predstavlja kolekciju podataka organizovanih tako da se podacima lako može manipulisati. Ukoliko uopšteno posmatramo, baze podataka bi se mogle podeliti prema tipu podataka koje sadrže pa bi tako postojale tekstualne, numeričke ili baze nekih drugih tipova podataka.
- Relacioni model je model po kome su podaci u bazi podataka organizovani u formi rednih lista (ordered list), a grupisani relacijama. Redne liste su u stvari tabele sa redovima i kolonama, a povezanosti među tabelama nazivaju se relacije.
- Sistem za upravljanje bazom podataka je specijalno dizajnirana softverska aplikacija koja omogućava interakciju korisnika, drugih aplikacija sa jedne strane i same baze podataka sa druge strane. Drugim rečima DBMS omogućava definisanje, kreiranje, pretragu, ažuriranje i administraciju jedne baze podataka.

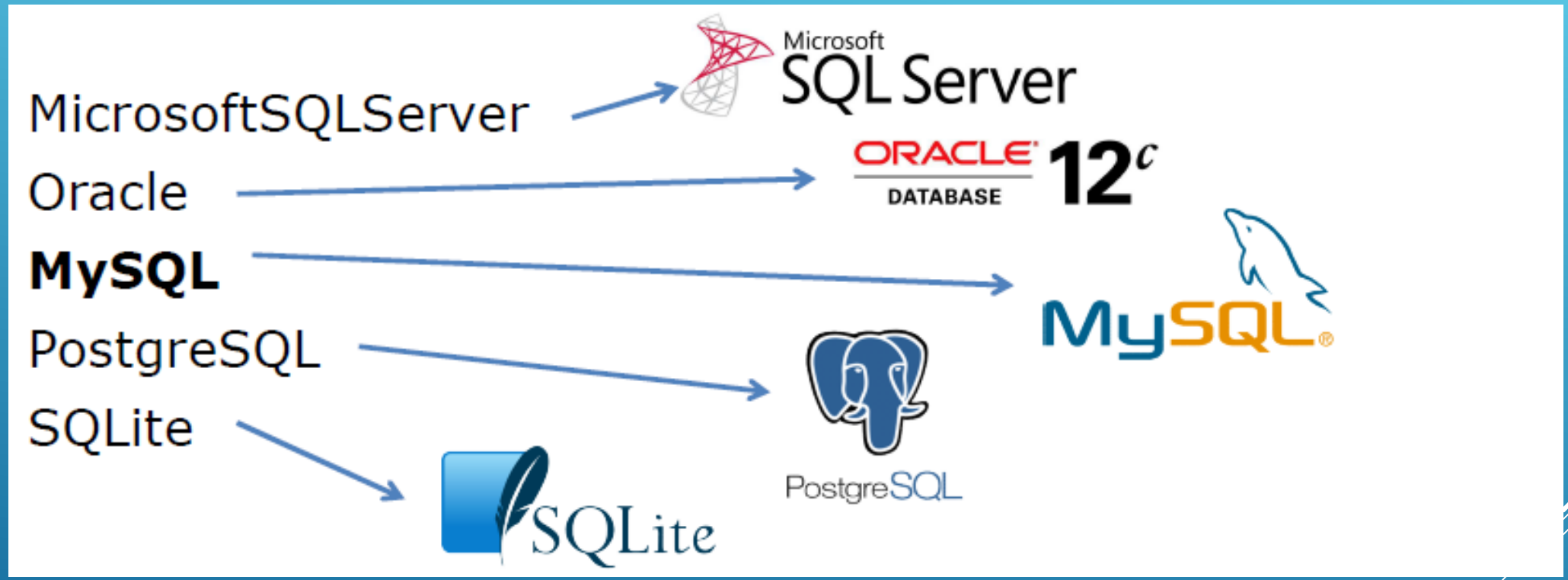


Najpoznatiji Database Menadžment sistemi

- Baza podataka predstavlja kolekciju podataka organizovanih tako da se podacima lako može manipulisati. Ukoliko uopšteno posmatramo, baze podataka bi se mogle podeliti prema tipu podataka koje sadrže pa bi tako postojale tekstualne, numeričke ili baze nekih drugih tipova podataka.
- Relacioni model je model po kome su podaci u bazi podataka organizovani u formi rednih lista (ordered list), a grupisani relacijama. Redne liste su u stvari tabele sa redovima i kolonama, a povezanosti među tabelama nazivaju se relacije.
- Sistem za upravljanje bazom podataka je specijalno dizajnirana softverska aplikacija koja omogućava interakciju korisnika, drugih aplikacija sa jedne strane i same baze podataka sa druge strane. Drugim rečima DBMS omogućava definisanje, kreiranje, pretragu, ažuriranje i administraciju jedne baze podataka.

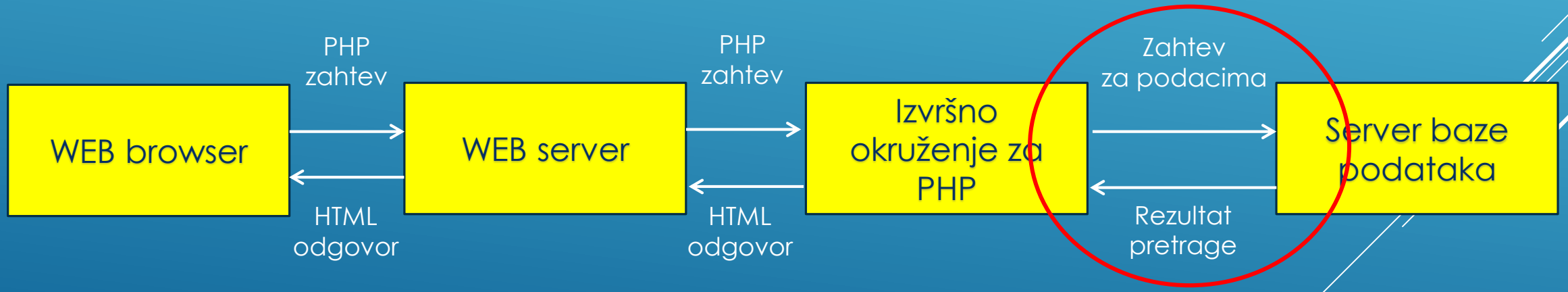
Najpoznatiji Database Menadžment sistemi

- Najpoznatiji sistemi za upravljanje relacionim bazama podataka su:



Komunikacija sa bazom podataka

- Upotreba baza podataka omogućava lakši, bezbedniji, brži i efikasniji način skladištenja, obrade i pristupa podacima.
- Postupak komunikacije web stranice i baze podataka može da se realizuje preko PHP koda.



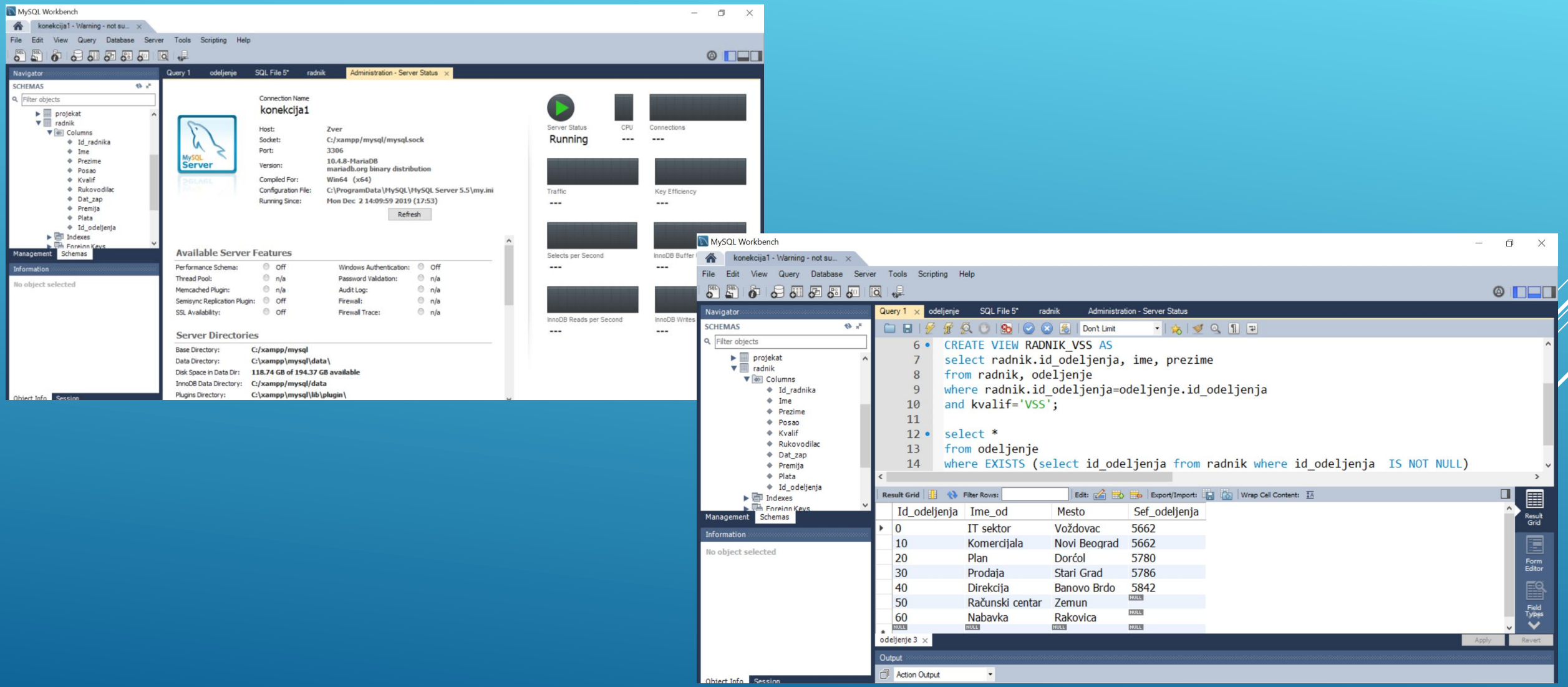
PHP i baze podataka

- Jedna od primarnih uloga serverskog jezika je komunikacija sa bazom podataka.
- PHP može da uspostavi konekciju sa velikim brojem baza podataka odnosno sistema za upravljanje bazama podataka.
- Jedan od najkorišćenijih DBMS sistem MySQL.
- MySQL server za rad sa bazama podataka, omogućava rad sa bazama proizvoljnih veličina, koristi standardni SQL, besplatan je i ima veliki broj korisnika.

MySQL

- MySQL je sistem za upravljanje bazama podataka.
- MySQL je najkorišćeniji sistem ove vrste.
- MySQL je projekat otvorenog koda, kreiran od strane švedske kompanije MySQL AB, u vlasništvu Oracle-a
- MySQL se isporučuje bez ikakvog alata sa grafičkim korisničkim okruženjem za manipulaciju podacima. Korisnici mogu da koriste integrisanu konzolu, odnosno komandni interfejs (CLI) ili da koriste neki od alata sa grafičkim korisničkim okruženjem koji dolaze odvojeno od samog MySQL sistema.
- MySQL kao sistem može raditi na mnogo različitih operativnih sistema , a najčešće se koristi na Apache web serverima u kombinaciji sa PHP-om.

MySQL

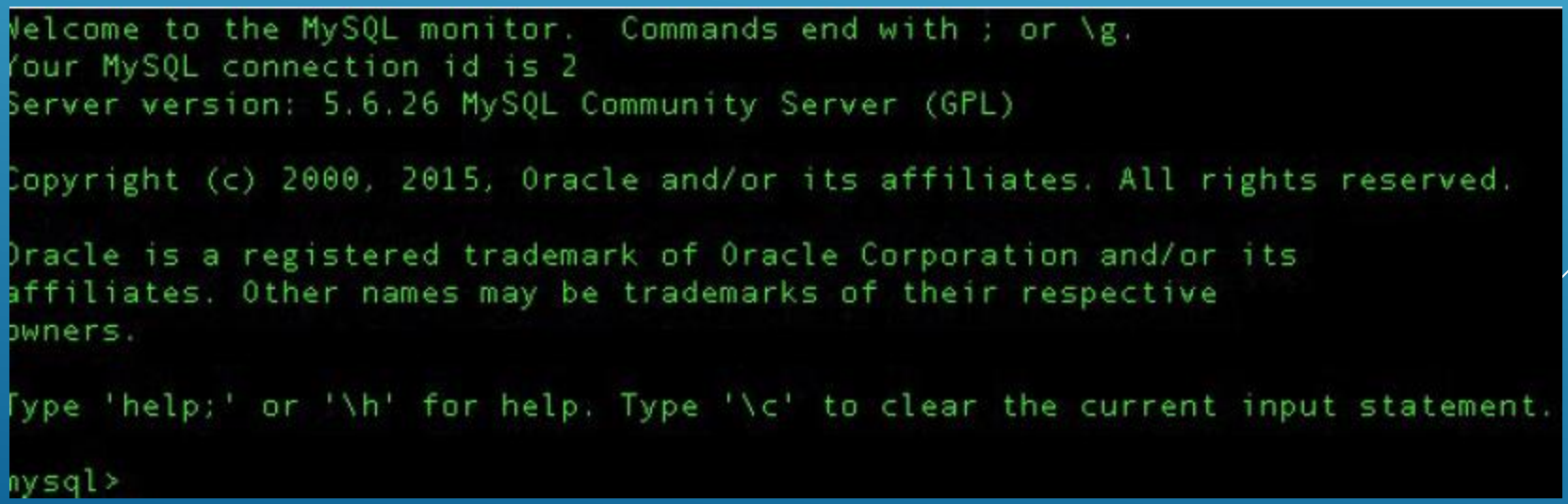

MySQL Workbench
konekcija1 - Warning - not su...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
projekat
radnik
Columns
Id_radnika
Ime
Prezime
Posao
Kvalif
Rukovodilac
Dat_zap
Premija
Plata
Id_odeljenja
Indexes
Management Schemas
Information
No object selected
Query 1 odeljenje SQL File 5* radnik Administration - Server Status
Connection Name
konekcija1
Host: Zver
Socket: C:/xampp/mysql/mysql.sock
Port: 3306
Version: 10.4.8-MariaDB mariadb.org binary distribution
Compiled For: Win64 (x64)
Configuration File: C:\ProgramData\MySQL\MySQL Server 5.5\my.ini
Running Since: Mon Dec 2 14:09:59 2019 (17:53)
Refresh
Server Status
Running
CPU
Connections
Traffic
Key Efficiency
Selects per Second
InnoDB Buffer
InnoDB Reads per Second
InnoDB Writes
Available Server Features
Performance Schema: Off
Thread Pool: n/a
Memcached Plugin: n/a
Semisync Replication Plugin: Off
SSL Availability: Off
Windows Authentication: Off
Password Validation: n/a
Audit Log: n/a
Firewall: n/a
Firewall Trace: n/a
Server Directories
Base Directory: C:/xampp/mysql
Data Directory: C:\xampp\mysql\data\
Disk Space in Data Dir: 118.74 GB of 194.37 GB available
InnoDB Data Directory: C:/xampp/mysql/data
Plugins Directory: C:\xampp\mysql\lib\plugin\
MySQL Workbench
konekcija1 - Warning - not su...
File Edit View Query Database Server Tools Scripting Help
Query 1 odeljenje SQL File 5* radnik Administration - Server Status
Don't Limit
6 CREATE VIEW RADNIK_VSS AS
7 select radnik.id_odeljenja, ime, prezime
8 from radnik, odeljenje
9 where radnik.id_odeljenja=odeljenje.id_odeljenja
10 and kvalif='VSS';
11
12 select *
13 from odeljenje
14 where EXISTS (select id_odeljenja from radnik where id_odeljenja IS NOT NULL)
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
Id_odeljenja Ime_od Mesto Sef_odeljenja
0 IT sektor Voždovac 5662
10 Komercijala Novi Beograd 5662
20 Plan Dorćol 5780
30 Prodaja Stari Grad 5786
40 Direkcija Banovo Brdo 5842
50 Računski centar Zemun NULL
60 Nabavka Rakovica NULL
odeljenje 3
Apply Revert
Output
Action Output

Šta je SQL?

- SQL (Structured Query Language) jeste jezik za upravljanje podacima posredstvom sistema za upravljanje bazom podataka
- SQL je standardizovan 1986. godine, ali i pored toga nije u potpunosti portabilan između različitih sistema za upravljanje bazama podataka
- SQL poseduje kompletnu sopstvenu sintaksu koju je potrebno savladati kako bi se na adekvatan način moglo rukovati podacima u bazi

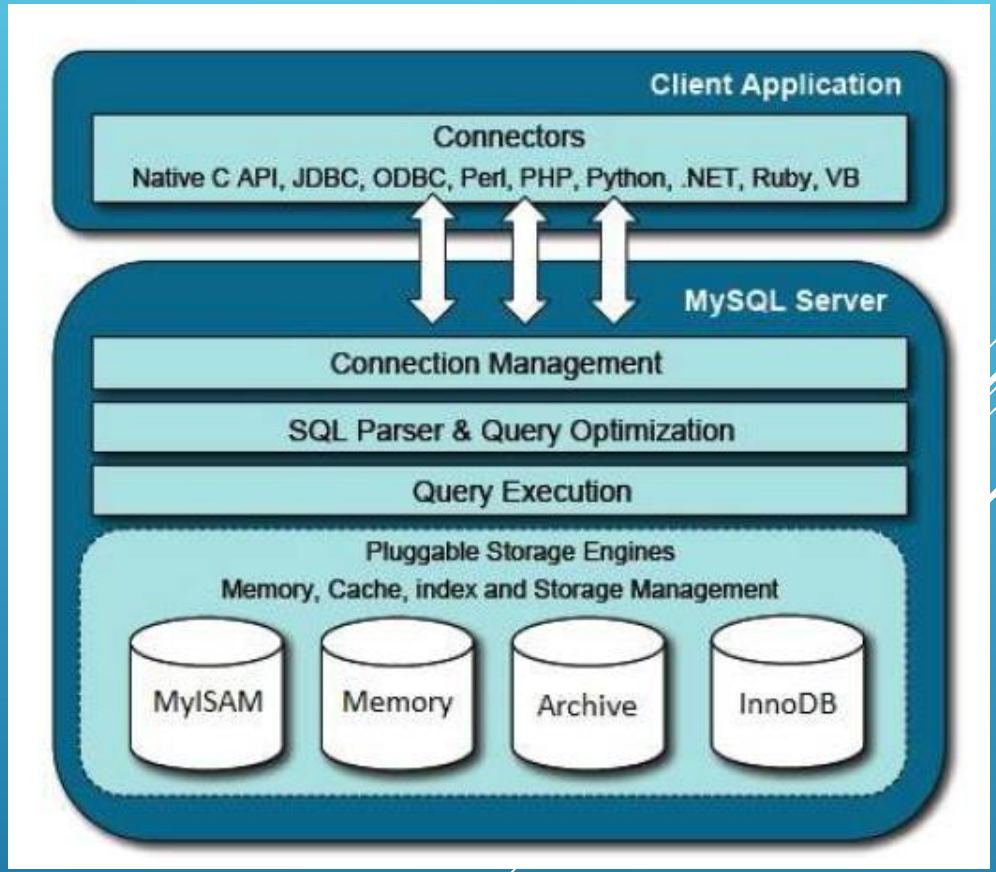
Elementi MySql okruženja

- Okruženje MySQL-a, sastoji se iz dva dela:
  - servera koji rukuje podacima
  - klijenta koji od server zahteva vršenje određenih radnji
- Jedini klijentski program koji stiže sa MySQL-om, jeste konzolni program MySQL Monitor.



Elementi MySql okruženja

- Kada se baza podataka koristi u nekoj aplikaciji, bila ona web, desktop ili mobilna, tada sama aplikacija postaje klijent baze podataka.
- Da bi aplikacija koju samostalno razvijamo bila u stanju da se sporazume sa serverom MySQL-a, tj. da bi poznavala njegov jezik koristi se konektor.
- Konektori se razlikuju u zavisnosti od platforme i programskog jezika u kome će biti upotrebljavani.
- To su zapravo biblioteke napisane na izvornom jeziku za koji su namenjene, koje sadrže gotove metode za komunikaciju.



Sistemske baze podataka

Podrazumevano, nakon instalacije, mysql server sadrži tri baze podataka

information_schema

- Omogućava pristup informacijama o podacima, tj. podacima o podacima. Podaci o podacima se drugačije nazivaju meta podaci (metadata)

mysql

- sadrži sve serverske informacije. Korisnici, relacije, privilegije i slično

performance_schema

- Podešavanjai statistike praćenja događaja niskog nivoa

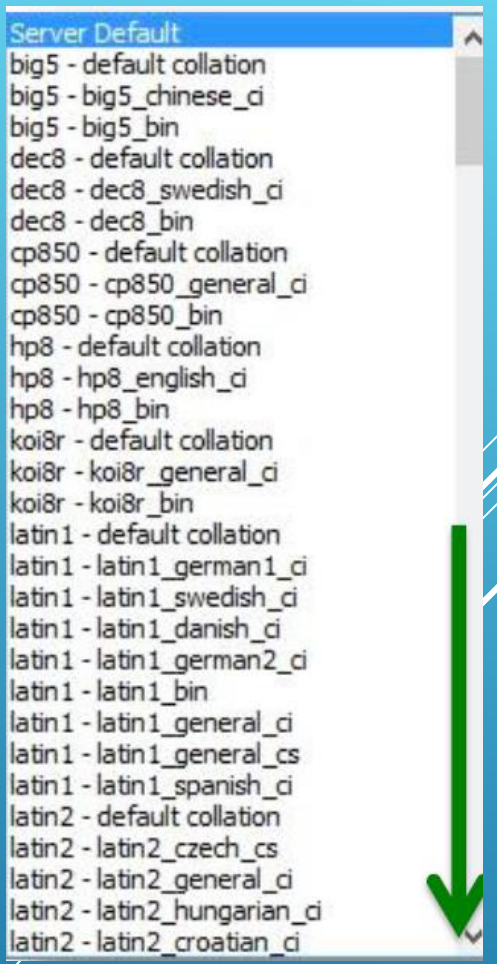
Kreiranje baze podataka

- Nakon prijave na MySQL server potrebno je kreirati bazu podataka.
- Komanda za kreiranje baze podataka.

```
CREATE DATABASE preduzece
CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

CharacterSets and Collations

- Ako treba da uporedimo karaktere A i B, najlakši način da se to uradi je poređenjem njihovih brojnih vrednosti, odnosno enkodinga. Collation upravo predstavlja pravila poređenje enkodinga.
- Character setovi i kolacije moraju odgovarati jedni drugima. Stoga je u padajućem meniju, koji se može otvoriti u prozoru za dodavanje baze, moguće odabrati kombinaciju karakter seta i kolacije.
- Na krajevima kolacija možemo da primetimo određene sufikse. Sufiksi i njihovo značenje su sledeći:
  - ci - Case Insensitive
  - cs - Case Sensitive
  - bin - Binary
- Ostavljanjem opcije Server Default zapravo se za karakter set i kolaciju podešava onaj karakter set i kolacija koji su podešeni na nivou servera.



SQL naredba za odabir baze podataka

Pre početka pravljenja tabela, indeksa, relacija ili upravljanja podacima u bazi podataka, neophodno je izdati MySQL serveru naredbu za odabir baze podataka na koju se odnose naredbe koje slede.

```
USE preduzece;
```

SQL naredba za pravljenje tabele - CREATE TABLE

Kreiranje tabele i postavljanje ograničenja definisanjem složenog primarnog i stranog ključa
SQL CREATE TABLE iskaz
SQL CONSTRAINT i REFERENCES ključne reči

```
CREATE TABLE ucesce
      (
       radnik_id          int              NOT NULL,
       projekat_id        int              NOT NULL,
       sredstva           float            NOT NULL,
       CONSTRAINT     ucesce_PK    PRIMARY KEY(radnik_id, projekat_id),
       CONSTRAINT     radnik_FK    FOREIGN KEY(radnik_id)    REFERENCES radnik(radnik_id),
       CONSTRAINT     projekat_FK  FOREIGN KEY(projekat_id)  REFERENCES projekat(projekat_id)
      ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

SQL naredba za pravljenje tabele - CREATE TABLE

U komandi CREATE TABLE deklarišemo unutar zagrada kolone koje nam trebaju, njihove tipove podataka i druge informacije koje se tiču strukture tabele.

Iza zatvarajuće zagrade možete zadati određene opcije koje se tiču tabele.

Ako ne zadate tip tabele, podrazumeva se MyiSAM. Ukratko, tip tabele može biti jedan od sledećih:

- My ISAM, koji se podrazumeva, veoma je brz i podržava indekse za tekstualno pretraživanje; zamenjuje nekadašnji podrazumevani tip ISAM.
- InnoDB je mašina baze podataka usklađena s grupom pravila ACID koja podržava transakcije, spoljne ključeve i zaključavanje podataka na nivou pojedinačnog reda tabele.

TIPOVI PODATAKA

- Osnovni tipovi podataka koje se koriste u MySQL implementaciji SQL jezika su:
  - Tekstualni: kodirani i nekodirani
  - Numerički: označeni i neoznačeni
  - Vremenski

- **Kodirani tekstualni tipovi** podataka se koriste za čuvanje tekstualnih vrednosti, gde tekst nije kodiran ASCII kodnom mapom, već Unicode kodnom mapom promenljive širine. Najčešće se dužina vrednosti ovih tipova podataka izražava u broju karaktera.
- **Nekodirani ili binarni tipovi** podataka se koriste za čuvanje binarnih zapisa, kao što su sadržaji datoteka ili binarnih serijalizacija objekata iz programa. Najčešće se dužina vrednosti ovih tipova podataka izražava u broju bajtova.

Tekstualni tipovi podataka

Specifični kodirani tekstualni tipovi podataka su:	
CHAR	Dužina je ograničena na 255 karaktera
VARCHAR	Dužina je ograničena na 65.535 karaktera
TEXT	Dužina je ograničena na 65.535 karaktera
TINYTEXT	Dužina je ograničena na 256 karaktera
MEDIUMTEXT	Dužina je ograničena na 16.777.215 karaktera.
LONGTEXT	Dužina je ograničena na 4.294.967.295 karaktera.
Specifični binarni tipovi podataka su:	
BINARY	Dužina zapisa je ograničena na 255 bajtova.
VARBINARY	Dužina zapisa je ograničena na 65.535 bajtova.
BLOB	Dužina je ograničena na 65.535 bajtova.
MEDIUMBLOB	Dužina je ograničena na 16.777.215 bajtova.
LONGBLOB	Dužina zapisa je ograničena na 4.294.967.295 bajtova.

Tekstualni tipovi podataka

ENUM i SET tipovi podataka omogućavaju da se unapred definisane tekstualne vrednosti čuvaju u poljima tih tipova uz manju potrošnju prostora.

ENUM	Najviše 65.535 jedinstvenih vrednosti. Polje može da sadrži samo jednu od vrednosti
SET	Najviše 64 jedinstvene vrednosti. Polje može da sadrži najviše 64 jedinstvene vrednosti.

Numerički tipovi podataka

Označeni celobrojni numerički tipovi podataka su:	
INT	Opseg od -2147483648 do 2147483647
TINYINT	Opseg od -128 do 127.
SMALLINT	Opseg od -32768 do 32767
MEDIUMINT	Opseg od -8388608 do 8388607
BIGINT	Od -9223372036854775808 do 9223372036854775807.

Neoznačeni celobrojni numerički tipovi podataka su:	
INT	Opseg od 0 do 4.294.967.295
TINYINT	Opseg od 0 do 255
SMALLINT	Opseg od 0 do 65.535
MEDIUMINT	Opseg od 0 do 16.777.215
BIGINT	Opseg od 0 do 18.446.744.073.709.551.615.

Numerički tipovi podataka

Označeni i neoznačeni realni numerički tipovi sa fiksnom decimalnom tačkom su
DECIMAL(C, D)
NUMERIC

Označeni i neoznačeni realni numerički tipovi sa promenjivom decimalnom tačkom su
FLOAT
DOUBLE

Vremenski tipovi podataka

YEAR Opseg od 1901 do 2155. godine. Može da bude i 0000
DATETIME Opseg od 1. 1. 1000. u 00.00.01 do 19. 1. 2038. u 03.14.08
DATE Opseg datuma od 1. 1. 1000. do 31. 12. 9999. godine.
TIME Opseg vremena od 00.00.00 do 23.59.59
TIMESTAMP Opseg od 1. 1. 1000. u 00.00.01 do 19. 1. 2038. u 03.14.08.

Tipovi podataka DATETIME i TIMESTAMP podržavaju zapis do milionitog delova sekunde, od 0,000001 do 0,999999 sekundi

Specijalni tipovi podataka

- Prostorni tipovi, koji nalaze primenu u određenim oblastima kao što su geografski informacioni sistemi i drugi sistemi koji rade sa prostornim podacima.
- MySQL podržava JSON tipove podataka. JSON tip je specifičan, jer u određenoj meri narušava principe relacionih baza podataka, jer obezbeđuje mehanizam za čuvanje složenih podataka proizvoljne strukture u poljima tabele.
- JSON podaci se u MySQL bazama podataka beleže kao tekstualni, ali posebne jezičke konstrukcije omogućavaju korišćenje i referenciranje struktura podataka smeštenih unutar polja koje sadrži JSON vrednost.

Odabir tipova podataka

Odabir adekvatnog tipa podatka je važan, jer u slučaju prvobitnog pogrešnog odabira tipa, u slučaju kasnije potrebe za promenom tipa, neophodno je izvršiti proces migracije.

- Ime osobe može da bude VARCHAR tip podatka ograničene dužine, ali ne može da bude DECIMAL(10,2) ili INT;
- Cena artikla može da bude DECIMAL(10,4) ako je potrebno evidentirati cene koje su realne vrednosti, ali ako su cene uvek celobrojne, dovoljno je koristiti tip podatka INT;
- ID broj kartice može da bude INT UNSIGNED ako se obim vrednosti broja ID kartice uklapa u okvire najvećeg podržanog broja neoznačene celobrojne numeričke vrednosti tipa INT. Međutim, ako ID broj kartice sadrži specijalne karaktere ili, eventualno, slovne karaktere, onda je bolje koristiti CHAR ili VARCHAR potrebne dužine;
- Kada je potrebno evidentirati status paketa ili pošiljke, moguće je koristiti tip podatka TINYINT UNSIGNED dužine 1 u kojem je pomoću vrednosti 1 ili 0 moguće ukazati na to da li je pošiljka poslata ili ne. Međutim, bolja opcija je definisati ENUM tip sa podržanim vrednostima koje mogu da ukažu na više koraka u postupku dostave paketa ili pošiljke, npr. pending, sent, lost, returned, delivered itd.

Naredbe za rukovanje sa podacima

NAREDBE ZA RUKOVANJE PODACIMA omogućavaju izveštavanje iz baze podataka (izdvajanje postojećih i izračunavanje novih informacija) i ažuriranje baze podataka u širem smislu.

Izveštavanje iz baze podataka:
SELECT – pristup podacima, prikaz pronađenih ili izračunatih sadržaja iz baze podataka.

Ažuriranje baze podataka:
INSERT – unošenje podataka, dodavanje novih redova u tabelu,
DELETE – brisanje podataka, izbacivanje redova iz tabele,
UPDATE – ažuriranje, izmena vrednosti podataka u koloni.

Čitanje podataka iz baze

Čitanje sadržaja kolona ime i grad iz tabele kupci

Primer:

```
select ime, grad
from kupci ;
```

Čitanje svih kolona iz tabele kupci

Primer:

```
select * from kupci
```

Čitanje podataka iz baze

Za izdvajanje dela podataka sa uslovom treba koristiti where pa uslov pretrage

Primer:

```
select *
from kupci
where grad = 'Vranje' ;
```

Kombinacija uslova u where klauzuli

```
select * from kupci
where grad = 'Vranje'
or|and ime='Pera Peric';
```

Operatori za poređenje su: =,<,>,>=,<=, !=, is not null, is null, between, in, not in, like,

Čitanje podataka iz više tabela istovremeno

Ukoliko se želi pročitati više podataka iz različitih tabela, a koji su na bilo koji način povezani, potrebno je izvršiti spajanje tabela.

- Izlistati sve porudžbine Pere Perića

Tabela	Kolone			
Kupci	ID kupca	Ime	Adresa	Grad
Narudzbine	ID narudzibe	ID kupca	Iznos	Datum

Primer:

```
select narudzbine.iznos, narudzbine.datum
from kupci, narudzbine
where kupci.ime = 'Pera Peric' and kupci.IDkupca = narudzbine.IDkupca;
```

Uslov spajanja je kupci.IDkupca = narudzbine.ID kupca

Spajanje više od dve tabele

Koji kupci su naručuli knjigu Baze (ISBN = 222333)

Tabela	Kolone			
Kupci	ID kupca	Ime	Adresa	Grad
Narudzbine	ID narudzibene	ID kupca	Iznos	Datum
Proizvodi	ID narudzibene	ISBN broj	Količina	
Knjige	ISBN broj	Autor	Naslov	Cena

Primer:

select kupci.ime from kupci, narudzbine, proizvodi, knjige
where kupci. ID kupca = narudzbine. ID kupca and narudzbine. ID narudzbine= proizvodi.IDnarudzbine
and proizvodi.ISBN broj = knjige.ISBN broj and knjige.naslov like ' %Baze% ’ ;

Def: Broj uslova za spajanje tabela je za jedan manji od broja tabela.
Za spajanje dve tabele potreban jedan uslov.

Spajanje tabele (JOIN)

Join je ključna reč u SQL-u, koja označava spoj između dve tabele.

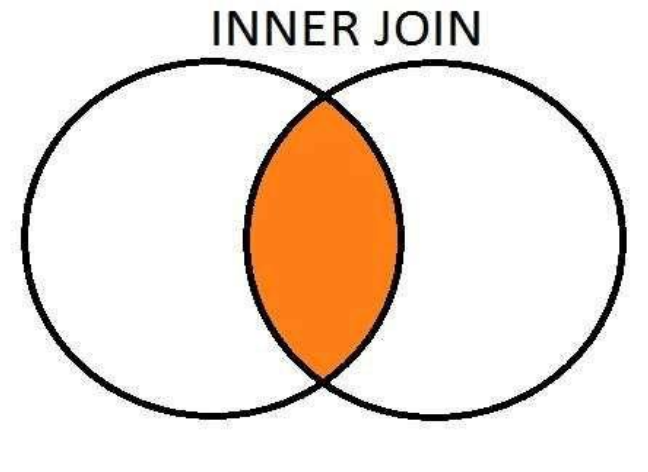
Dolazi u kompletu sa ključnom rečju ON (ali često i nekim drugim ključnim rečima).

Kada se u rezultatu spoljnjeg upita kombinuju podaci iz više tabela mora se izvršiti spajanje (JOIN) dveju ili više tabela.

Spajanje tabela vrši se korišćenjem zajedničkih atributa, tj. atributa koji su definisani nad istim domenima.

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

INNER JOIN



- Spaja zapise na osnovu jednog ili više zajedničkih polja. Vraća zapise u kojima su jednake vrednosti polja preko kojeg (kojih) se vrši spajanje.
- Spajanje se vrši preko polja primarnog ključa u jednoj tabeli i polja spoljnjeg ključa u drugoj tabeli (u relaciji jedan prema više).
- Ako za neku vrednost primarnog ključa jedne tabele u drugoj tabeli ne postoji ni jedan odgovarajući zapis, taj zapis se ne pojavljuje u rezultatu upita.

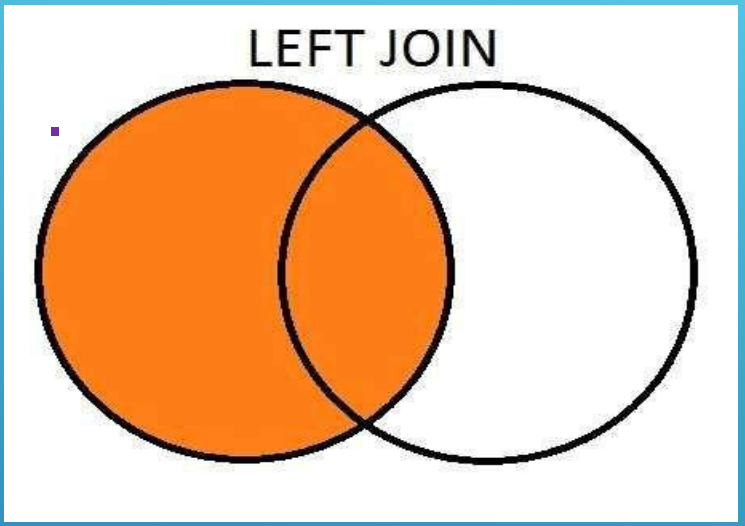
RADNIK <**#id_radnika**, ime, prezime, posao, kvalif, **rukovodilac$**, dat_zap, premija, plata,**Id_odeljenja$**>

ODELJENJE <**#id_odeljenja**, ime_od, mesto, **sef_odeljenja$**>

```
SELECT ime, prezime
FROM radnik JOIN odeljenje
ON odeljenje.id_odeljenja=radnik.id_odeljenja
WHERE mesto='Banovo brdo’
```

Prikazati spisak imena i prezimena zaposlenih koji rade na Banovom brdu.

LEFT JOIN



- LEFT JOIN vraća sve zapise iz tabele koju u spoju proglasimo kao “LEVU”, odnosno koja je prva navedena u izrazu za spajanje

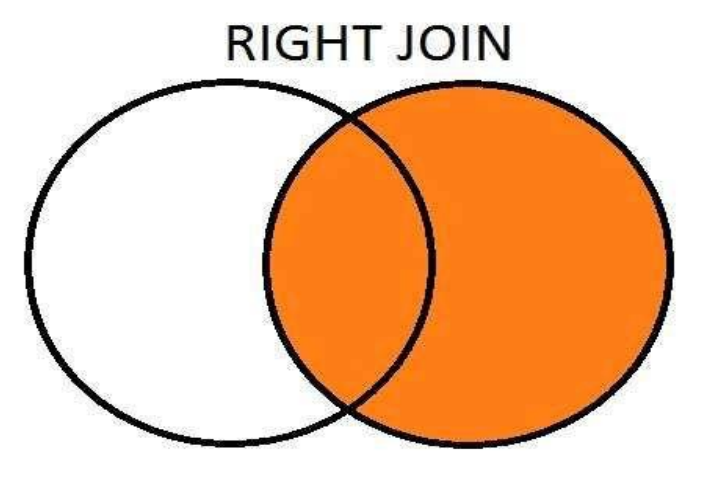
RADNIK <**#id_radnika**, ime, prezime, posao, kvalif, **rukovodilac$**, dat_zap, premija, plata,**Id_odeljenja$**>

ODELJENJE <**#id_odeljenja**, ime_od, mesto, **sef_odeljenja$**>

```
SELECT ime_od, ime, posao from
odeljenje left join radnik
on odeljenje.id_odeljenja=radnik.id_odeljenja
```

Prikazati nazive odeljenja, ime i posao svakog radnika koji u njima rade, uključujući i odeljenja u kojima nema raspoređenih radnika

RIGHT JOIN



- RIGHT JOIN vraća sve zapise iz tabele koju u spoju proglasimo ka“DESNU”, bez obzira na to da li se odgovarajući zapisi nalaze u “levoj” tabeli.

RADNIK <**#id_radnika**, ime, prezime, posao, kvalif, **rukovodilac$**, dat_zap, premija, plata,**Id_odeljenja$**>

ODELJENJE <**#id_odeljenja**, ime_od, mesto, **sef_odeljenja$**>

```
SELECT ime_od, ime, posao
from odeljenje right join radnik
on odeljenje.id_odeljenja=radnik.id_odeljenja
```

Prikazati nazive odeljenja, ime i posao svakog radnika koji u njima rade, uključujući i radnike koji nisu raspoređeni ni u jednom odeljenju.

Upotreba ALIJASA

- Alijasi su druga imena koja se dodeljuju privremeno ili trajno postojećim kolonama ili tabelama.

```
select k.ime,
from kupci as k, narudzbine as n, proizvodi as p, knjige as knj
where k.ID kupca = n.ID kupca and n.ID narudzbine= p.ID narudzbine and
n.ISBN = knj.ISBN and knj.naslov like ' %Baze% ";
```

Učitavanje po redosledu

- Ako se redovi koje treba prikazati kao rezultat upita žele sortirati koristi se naredba ORDER BY.
- ASC i DESC
- ASC podrazumevano sortiranje za MySQL

```
select ime, adresa, grad
from kupci
order by ime;
```

```
select ime, adresa, grad
from kupci
order by 3 desc;
```

```
select ime, adresa, grad
from kupci
order by 3 desc, ime;
```

Grupne funkcije

- Grupne funkcije se primenjuju na grupisane podatke (više kolona ili redova u kolonama)
- Neke grupne funkcije su:

Naziv	Opis
AVG (kolona)	Prosečna vrednost kolone
COUNT (stavke)	Ako je navedena kolona daje broj elemenata u koloni bez NULL, a ako se ispred naredbe doda DISTINCT dobije se broj jedinstvenih vrednosti u koloni.
MIN (kolona)	Minimalne vrednost kolone
MAX (kolona)	Maksimalna vrednost kolone
SUM (kolona)	Zbir vrednosti u koloni

Primeri grupnih funkcija

Izračunavanje srednje vrednosti porudžbina

```
select avg(iznos)
from narudzbine;
```

Izračunavanje srednje vrednosti porudžbina po određenim grupama dobija se grupisanjem po određenoj kategoriji

```
select IDkupca, avg(iznos)
from narudzbine
group by IDkupca
```

Traženje porudžbina koje premašuju vrednost 4500

```
select IDkupca, avg(iznos)
from narudzbine;
group by IDkupca
having avg(iznos) > 4500;
```

Unošenje podataka u bazu podataka

- Unošenje podataka vrši se naredbom INSERT
  - INSERT [INTO] tabela [(kolona1, kolona2, kolona3, ...)]
    VALUES(vrednost1, vrednost2, vrednost3, ...);

- Vrednosti kolona pisati pod znacima ' ili '', sem ako su brojevi ili datumi.

- Ako se upisuje ceo red (sve kolone) sintaksa je:
  INSERT [INTO] tabela VALUES(vrednost1, vrednost2, vrednost3, ...);

```
insert into kupci
values ('Pera Peric', 'Beograd', 'Svet oko nas', 2000, 1, 'knjiga za decu', 2007-03-01);
```

Unošenje podataka u bazu podataka

- Upis se može vršiti i za pojedine kolone proizvoljnim redosledom

```
insert into kupci (naslov, kolicina, grad) values ('Mali princ', 2, 'Vranje',);
```

- Ako je kolona primarni ključ definisana kao auto_increment, tada je kod upisa,dovoljno samo prvi put, definisati njenu vrednost.

Unošenje podataka u bazu podataka

- U tabelu se može uneti i više redova odjednom

```
insert into kupci values
(2, 'Pera Peric', 'Beograd', 'Svet oko nas', 2000, 1, 'knjiga za decu', '2007-03-01') ,
(3, 'Mile Vasic', 'Sabac', 'Geografija', 2500, 2, 'knjiga za omladinu', '2007-03-05') ,
(4, 'Ana Savic', 'Valjevo', 'Bukvar', 1500, 1, 'knjiga za decu', '2007-03-15') ;
```

Ažuriranje zapisa u bazi

- Često postoji potreba za promenom podataka u bazi
- (cena artikla, pdv, adresa...)
- Iskaz UPDATE i njegova sintaksa su:

Primer povećanja cene knjiga za 10%:

```
update knjige
set cena=cena*1.1;
```

Primer promene adrese kupca:

```
update kupci
set adresa = ' Bulevar mira 12'
where IDkupca = 3;
```

```
UPDATE imetabele
Set kolona1=izraz1, kolona2=izraz2, ...
[Where uslovi]
```

Naknadne izmene strukture tabela

Tabela 10.5 Izmene koje omogućava iskaz ALTER TABLE

Sintaksa	Opis
`ADD [COLUMN] opis_kolone [FIRST | AFTER kolona]`	Dodaje novu kolonu na navedenom mestu. Ako mesto umetanja nije navedeno, kolona se dodaje na kraj. Parametar *opis_kolone* definiše ime kolone i tip podataka, kao u iskazu CREATE.
`ADD [COLUMN] (opis_kolone, opis_kolone...)`	Dodaje jednu ili više kolona na kraj tabele.
`ADD INDEX [indeks] (kolona,...)`	Dodaje indeks po navedenoj koloni ili kolonama.

Primer izmene strukture tabela

Promena tipa podatka u koloni tako da prihvata imena max dužine 90 karaktera

```
alter table kupci
modify ime char(90) not null;
```

Dodavanje nove kolone (pdv) iza kolone iznos u tabeli porudžbine

```
alter table porudzbine
add pdv float (5,2) after iznos
```

Brisanje kolone pdv

```
alter table porudzbine
drop pdv;
```

Brisanje

Brisanje zapisa (redova) - Delete

```
DELETE [low_priority] [quit] [ignore]
from imetabele
[WHERE uslov]
```

```
DELETE from kupci
WHERE ime='Pera';
```

Brisanje tabela

```
DROP TABELE (naziv tabele koja se briše);
```

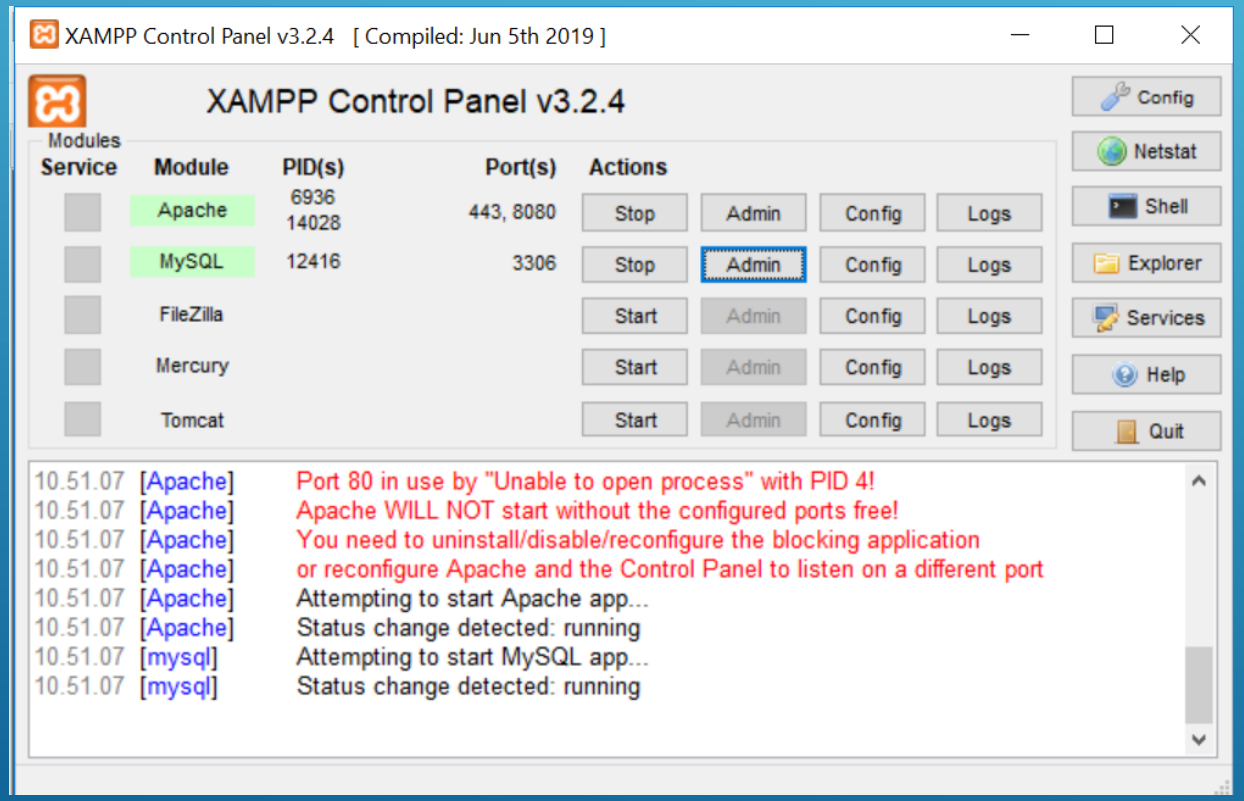
```
TRUNCATE TABELE (naziv tabele koja se briše);
```

Brisanje cele baze

```
DROP DATABASE (naziv baze podataka koja se briše);
```

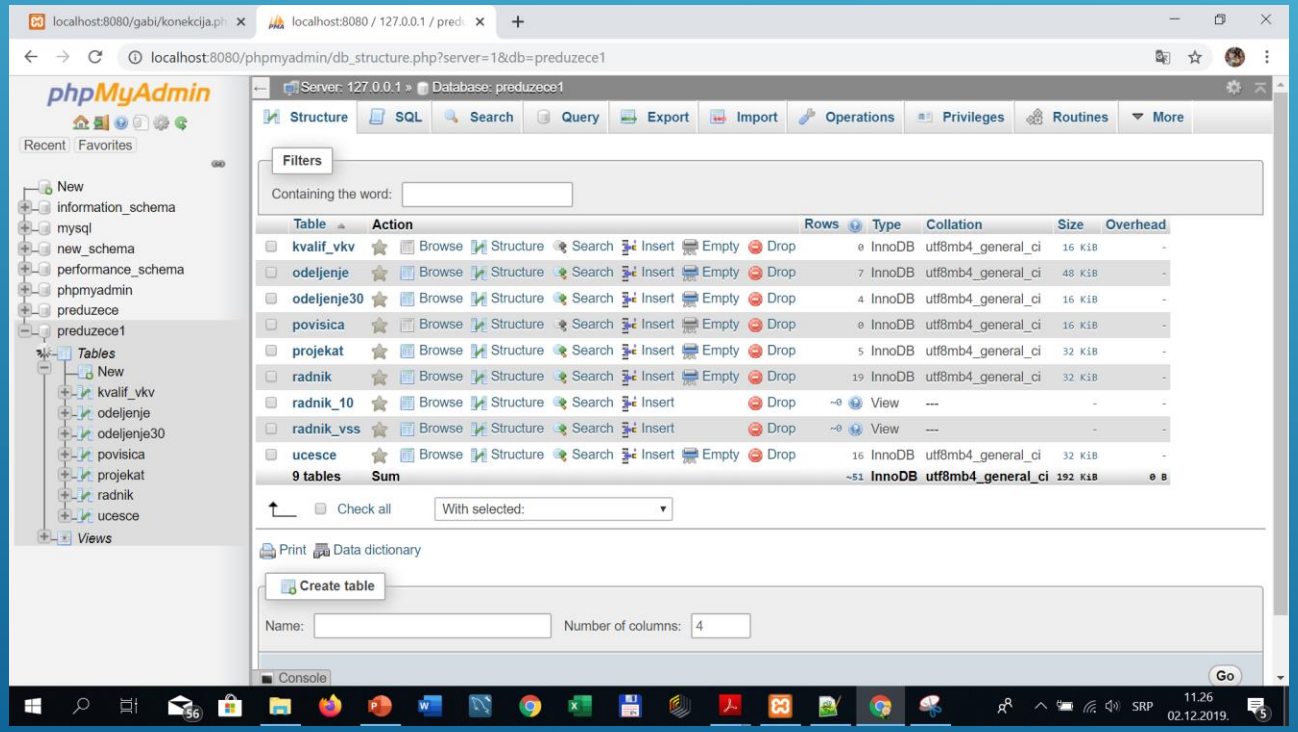
Aktivacija baze podataka

- Prvi korak je kreiranje baze podataka
- Ovo se realizuje ili kodom ili direktno u bazi podataka
- Za rad sa bazom podataka potrebno je da se pristupi bazi podataka
- Ako se radi u lokalu i koristi XAMPP potrebno je aktivirati opciju MySQL

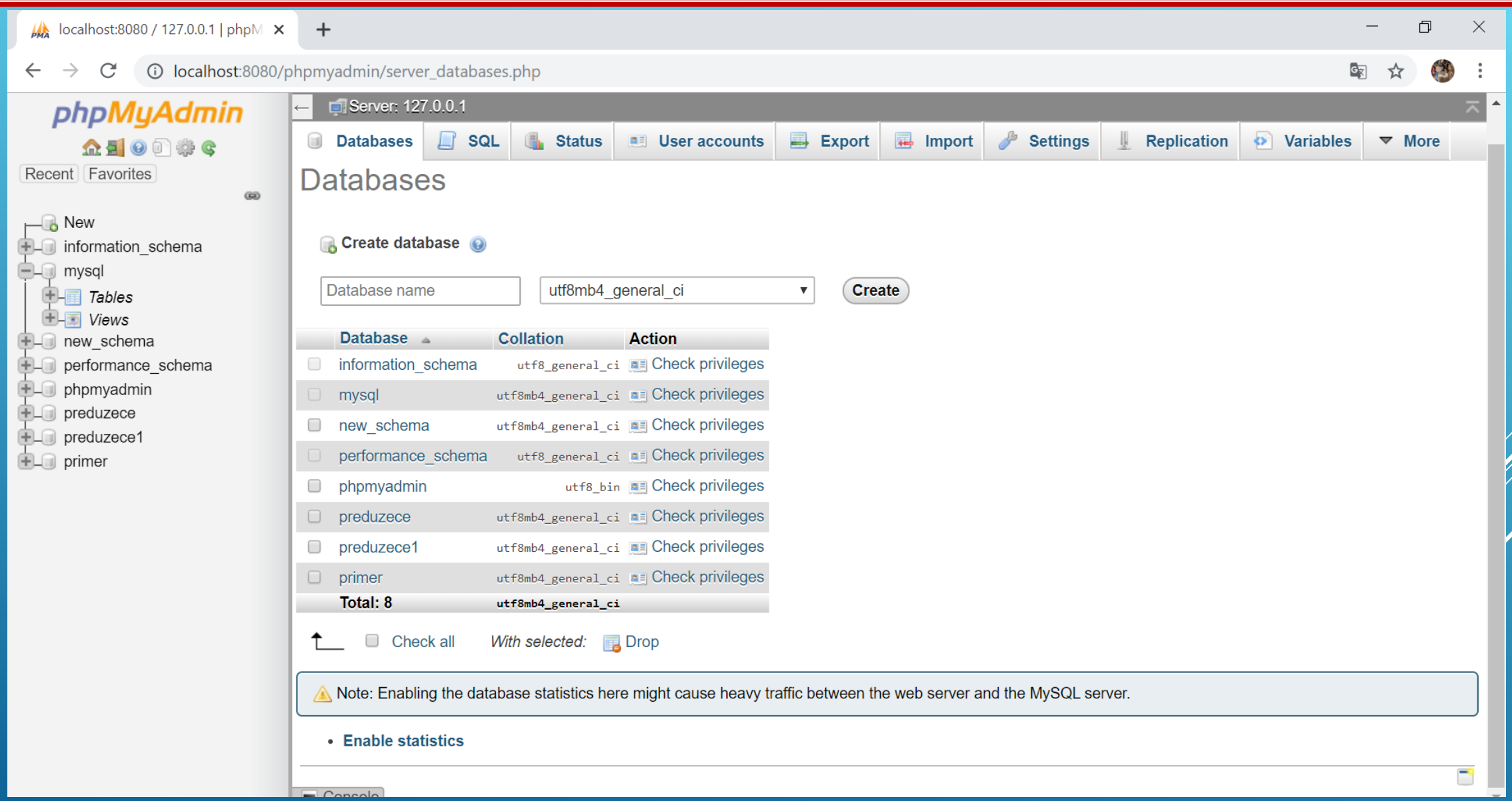


Grafički pristup bazi podataka

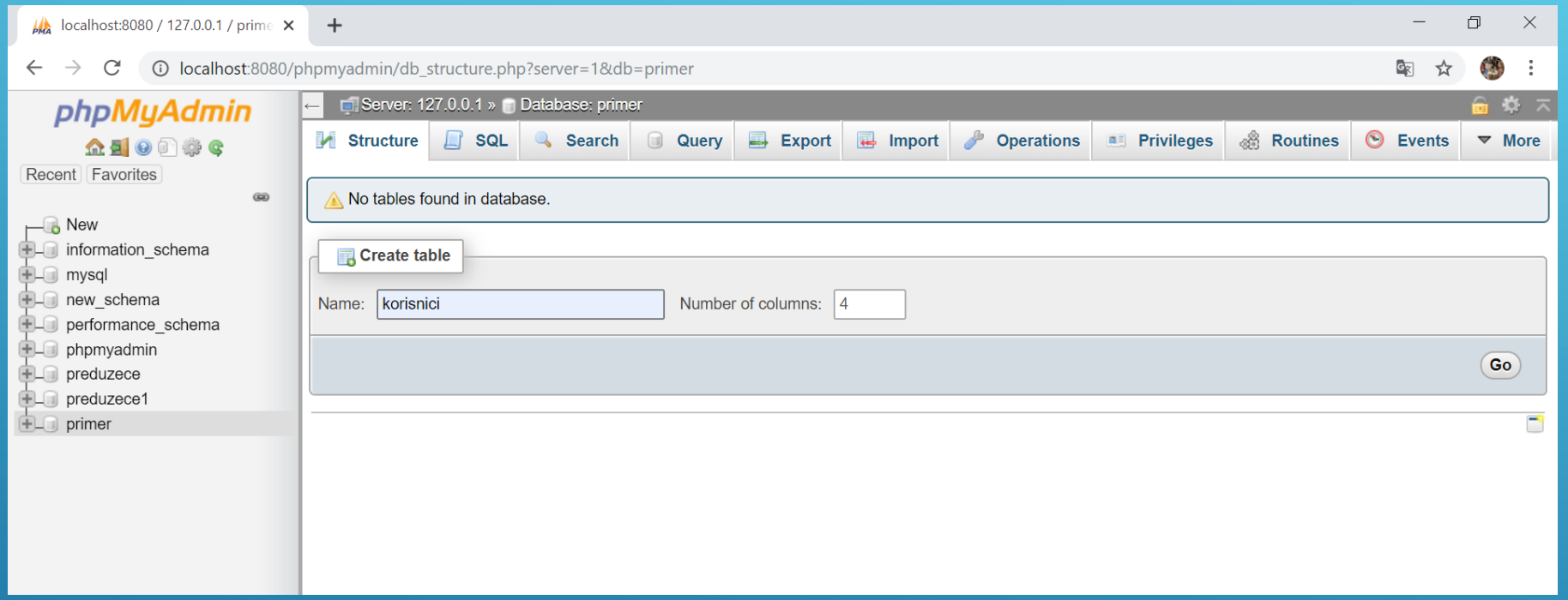
- Nakon pokretanja XAMPP-a bazi podataka MySQL se može pristupiti u grafičkom režimu preko aplikacije phpMyAdmin
- Potrebno je unet i adresu lokalnog servera http://localhost:8080/phpmyadmin
- Nakon toga, treba kreirati željenu bazu podataka i tabele.



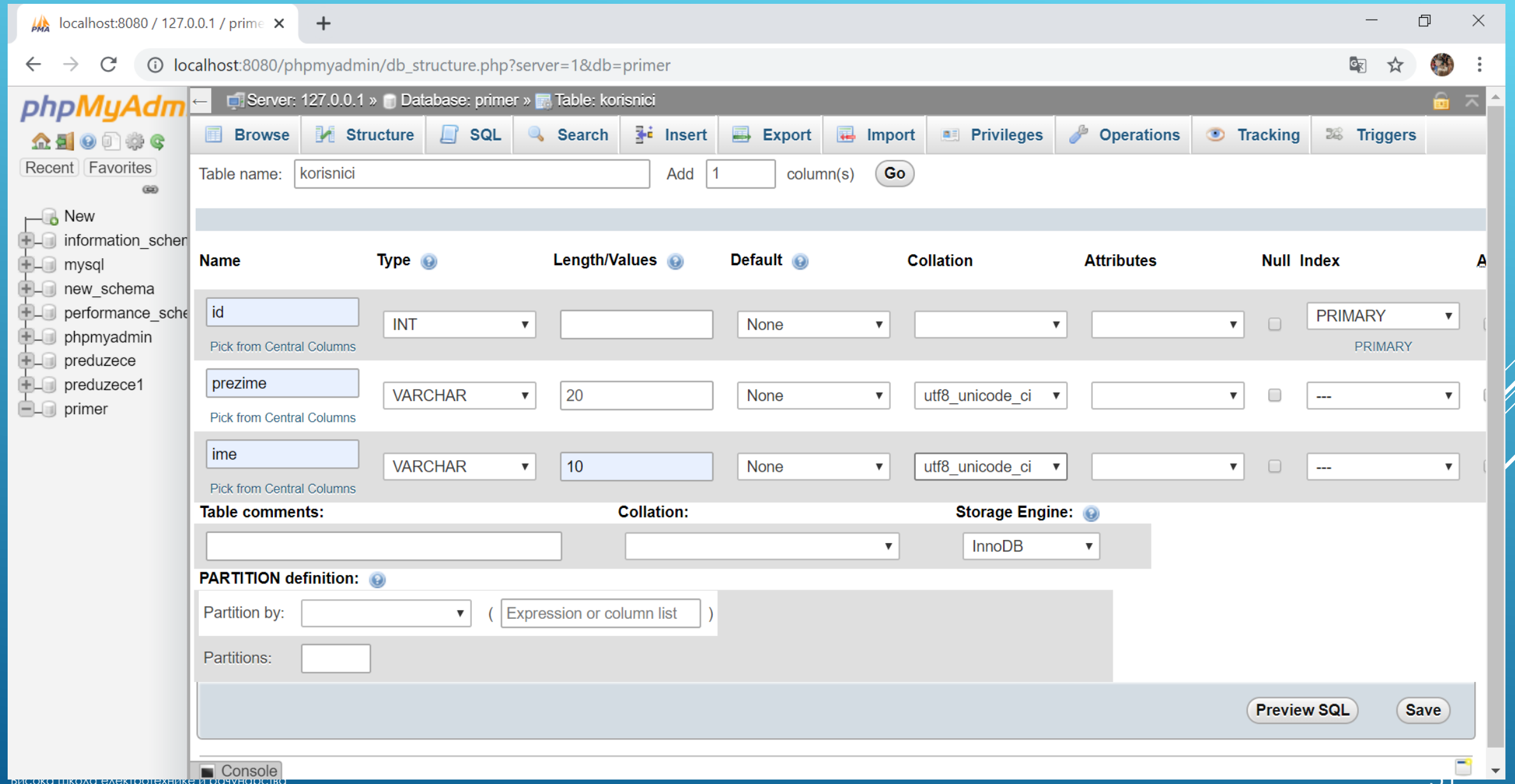
Grafički pristup bazi podataka



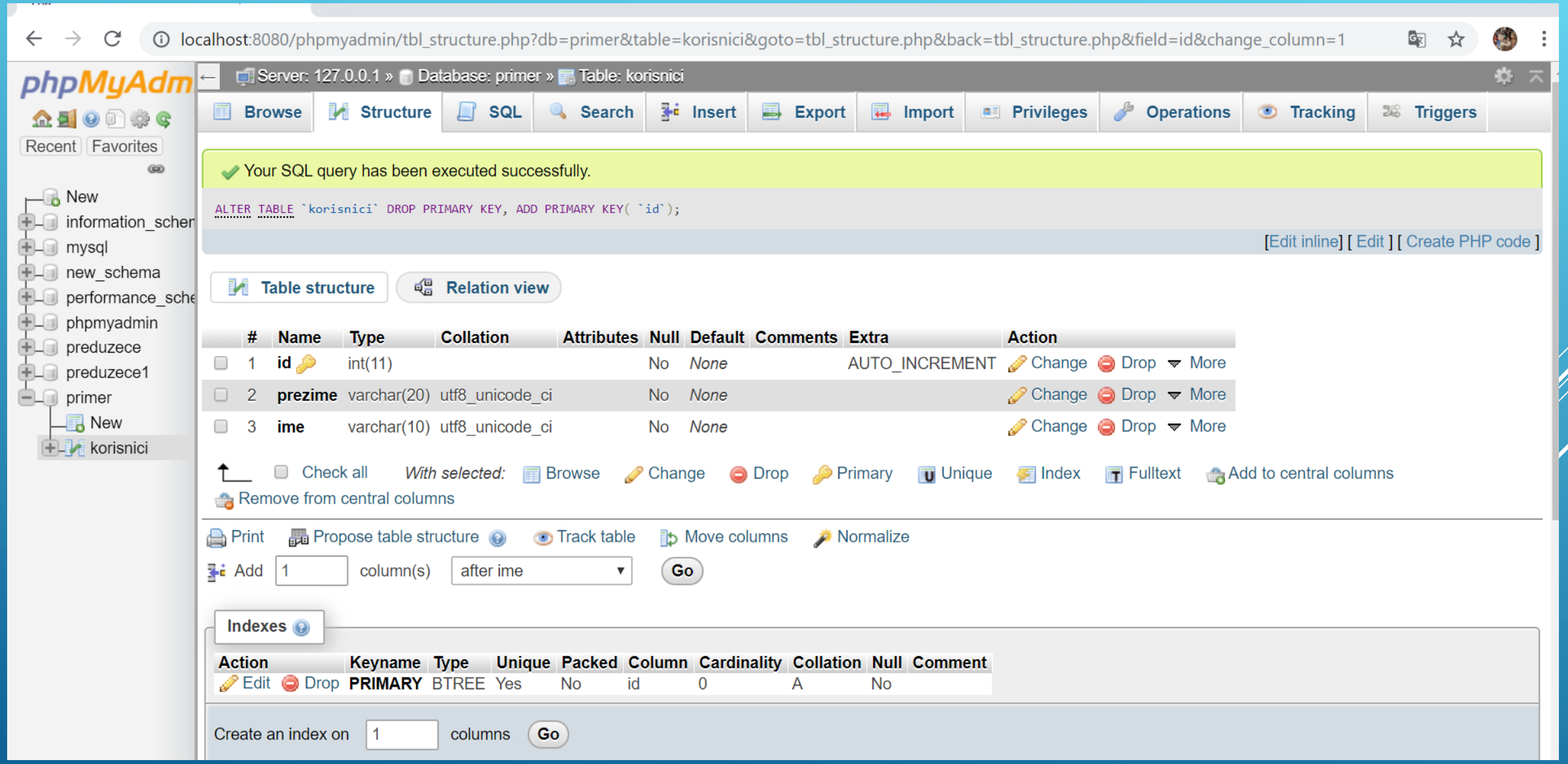
Grafički pristup bazi podataka



Grafički pristup bazi podataka



Grafički pristup bazi podataka



Otvaranje konekcije ka MySQL serveru

- Pre nego što se pristupi podacima u bazi podataka, potrebno je otvoriti konekciju ka MySQL serveru
- U PHPu, to se postiže mysqli_connect() funkcijom:

```
mysqli_connect(host,username,password,dbname);
```

Funkcija mysqli_connect()

Parametar	Opis
host	Ime hosta ili IP adresa
username	MySQL korisničko ime
password	MySQL lozinka za logovanje
dbname	Podrazumevana baza koja se koristi za opsluživanje upita

Primer konekcije

- U sledećem primeru, u cilju kasnije upotrebe u skriptu, konekcija se skladišti u promenljivoj ($con):
- Postoji mogućnost neuspele konekcije. Za proveru konekcije koriste nam funkcije **mysqli_connect_error()** i **mysqli_conect_errno()**
- Ukoliko je konekcija ostvarena, kao rezultat se vraća resurs baze podataka, ukoliko je konekcija neuspešna vraća se **FALSE**

```
<?php
//Pokušaj konekcije
$con=mysqli_connect("http://www.mysqldatabase.com", "pera", "12345", "mojaBaza");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
?>
```

Zatvaranje konekcije

- Po završetku skripta, konekcija će se automatski zatvoriti. Za zatvaranje konekcije pre toga, koristi se funkcija **mysqli_close():**

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("http://www.mysqldatabase.com", "pera", "12345", "mojaBaza");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}

/*
Rad sa bazom podataka
*/

//Zatvaranje konekcije
mysqli_close($con);
?>
```

Funkcija mysqli_query()

- Funkcija mysqli_query() obezbeđuje slanje upita bazi podataka:

```
mysqli_query (connection,query, resultmode);
```

Parametar	Opis
connection	Zahtevano. Specificira MySQL konekciju koja će se koristiti
query	Zahtevano. Specificira string koji predstavlja upit
resultmode	Opciono. Konstanta. Dve mogućnosti: •MYSQLI_USE_RESULT (Ovo se koristi za slanje velike količine podataka) •MYSQLI_STORE_RESULT (Ovo je podrazumevano)

Upit za encoding

- Važna stvar u radu sa bazama podataka jeste kodiranje konekcije
- Podrazumevano kodiranje konekcije je ANSI
- Ako u bazi imamo podatke kodirane sa UTF-8, moramo pripremiti bazu za rad sa UTF-8 encoding-om. To se radi upitom.
- Ukoliko ne izvršimo ovaj upit, na stranici se neće prikazati ćirilična slova i latinična slova đ,š,ž,ć,č

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()),
if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA:
".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");

/*
Rad sa bazom podataka
*/

//Zatvaranje konekcije
mysqli_close($con);
?>
```

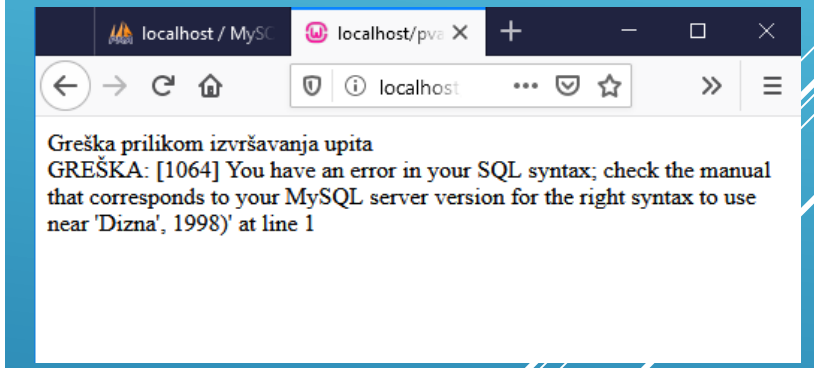
PRIMER

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");

//Izvršavanje INSERT upita
mysqli_query($con, "INSERT INTO korisnici (ime, prezime, godina) VALUES ('Mile',
'Dizna', 1998);");

//Zatvaranje konekcije
mysqli_close($con);
?>
```

Detektovanje greške prilikom izvršenja upita

- Upit ne mora uvek biti uspešan.
- Detektovanje greške prilikom izvršenja upita je obavezno. Ako se ne proveri stanje izvršenje upita, korisnik NEĆE ZNATI da upit nije uspešno izvršen
- Funkcije **mysqli_error($con)** i **mysqli_errno($con)** služe za proveru da li je upit uspešno izvršen

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje INSERT upita
mysqli_query($con, "INSERT INTO korisnici (ime, prezime, godina) VALUES ('Mile, 'Dizna', 1998)");

//Provera da li je upit uspešno izvršen
if(mysqli_error($con))
{
    echo "Greška prilikom izvršavanja upita<br>GREŠKA: [".mysqli_errno($con)."]
".mysqli_error($con);
}

//Zatvaranje konekcije
mysqli_close($con);
?>
```



Rezultat upita

- **INSERT**, **UPDATE** i **DELETE** upiti su **upiti bez resultset-a**, što znači da kao odgovor nećemo dobiti podatke već samo da li je upit izvršen uspešno (TRUE) ili neuspešno (FALSE)
- Nakon ovih upita možemo samo proveriti da li je upit uspešno izvršen i, eventualno, broj izmenjenih redova.
- **SELECT upiti** su **upiti sa resultset-om**, odnosno kao odgovor dobijamo podatke iz baze ako je upit dobar ili FALSE ako upit nije uspešno izvršen.
- Nakon ovog upita možemo proveriti da li je upit uspešno izvršen i ako jeste možemo obraditi odgovor iz baze.

Upiti bez resultset-a

- Nakon INSERT, UPDATE i DELETE upita se može proveriti koliko redova je izmenjeno u bazi podataka korišćenjem **mysqli_affected_rows()** funkcije
- Ukoliko nije došlo ni do jedne izmene funkcija vraća 0, ukoliko jeste vraća broj izmenjenih redova

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");
//INSERT upit
mysqli_query($con, "INSERT INTO korisnici (ime, prezime, godina) VALUES ('Joca', 'Karburator', 2000)");

//Provera koliko je dodato redova
if(mysqli_affected_rows($con)>0)
    echo „Dodato redova: ".mysqli_affected_rows($con);
else
    echo "Nijedan red nije dodat";

//Zatvaranje konekcije
mysqli_close($con);
?>
```

Mysqli_insert_id()

- Nakon INSERT upita se može saznati koji je id poslednjeg dodatog zapisa (pod pretpostavkom da je id primarni ključ u tabeli i da je autoincrement) korišćenjem funkcije **mysqli_insert_id()**

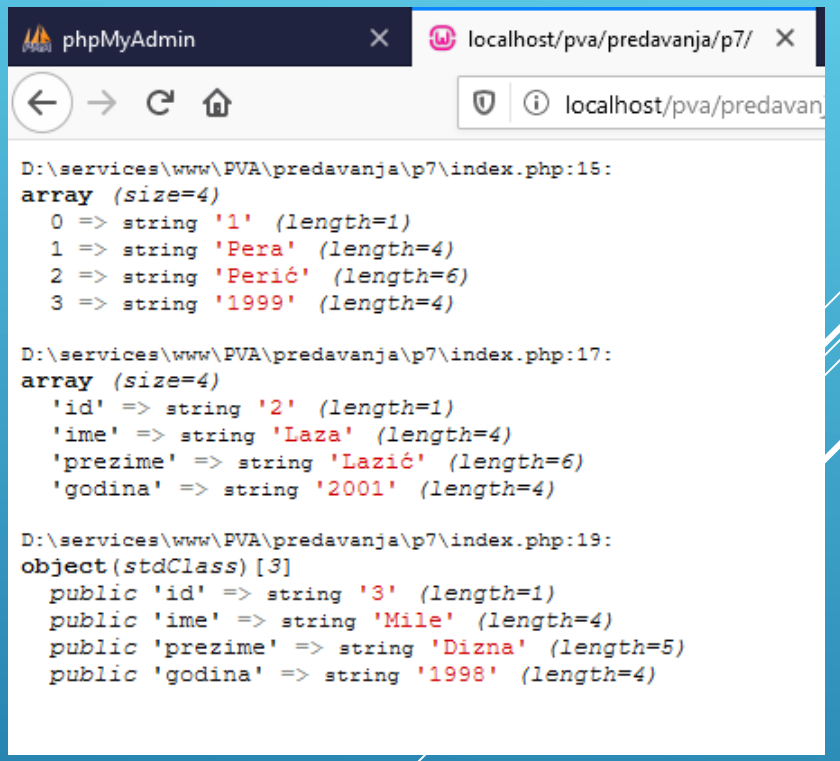
```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");
//INSERT upit
mysqli_query($con, "INSERT INTO korisnici (ime, prezime, godina) VALUES ('Joca', 'Karburator', 2000)");

//poslednji id
echo "Poslednji id: ".mysqli_insert_id($con);

//Zatvaranje konekcije
mysqli_close($con);
?>
```

Obrada rezultata upita

- Ukoliko je select upit uspešno izvršen, iz baze se vraća resultset.
- Postoji nekoliko načina za obradu rezultset-a
  - **mysqli_fetch_array($result)** – čitanje jednog reda i smeštanje i u numerički i u asocijativni niz
  - **mysqli_fetch_row($result)** – čitanje jednog reda rezultata i smeštanje u numerički niz
  - **mysqli_fetch_assoc($result)** – čitanje jednog reda rezultata i smeštanje u asocijativni niz
  - **mysqli_fetch_object($result)** – čitanje jednog reda rezultata i smeštanje u objekat
  - **mysqli_fetch_all($result, $tip)** – čitanje celokupnog odgovora i smeštanje u niz. Da li će to biti numerički ili asocijativni niz zavisi od drugog parametra $tip (**MYSQLI_NUM** ili **MYSQLI_ASSOC**)

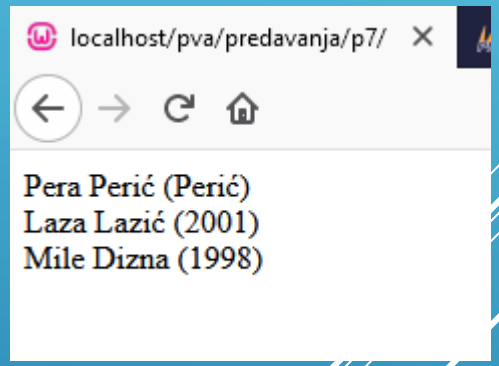
Obrada rezultata upita

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");
//SELECT upit
$result=mysqli_query($con, "SELECT * FROM korisnici");
$red=mysqli_fetch_row($result);
var_dump($red);
$red=mysqli_fetch_assoc($result);
var_dump($red);
$red=mysqli_fetch_object($result);
var_dump($red);
//Zatvaranje konekcije
mysqli_close($con);
?>
```



Prikaz rezultata upita

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");
//SELECT upit
$result=mysqli_query($con, "SELECT * FROM korisnici");
$red=mysqli_fetch_row($result);
echo $red[1]." ".$red[2]," (".$red[2].")<br>";
$red=mysqli_fetch_assoc($result);
echo $red['ime']." ".$red['prezime']," (".$red['godina'].")<br>";
$red=mysqli_fetch_object($result);
echo $red->ime." ".$red->prezime," (".$red->godina.")<br>";
//Zatvaranje konekcije
mysqli_close($con);
?>
```



Prolazak kroz ceo resultset

- Za prolazak kroz sve odgovore iz baze podataka možemo koristiti petlje FOR i WHILE
- Za broj odgovora možemo koristiti **mysqli_num_rows()** funkciju, ukoliko nam je potrebna za FOR petlju.
- Za podešavanje pokazivača u samom odgovoru možemo koristiti funkciju **mysqli_data_seek($result, $position)**
  - **$result –** rezultat koji obrađujemo
  - **$position** – pozicija na koju želimo da se postavimo (0 za početak)

Prolazak kroz ceo resultset

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");
//SELECT upit
$result=mysqli_query($con, "SELECT * FROM korisnici");
echo "<h4>FOR petlja</h4>";
for($i=0;$i<mysqli_num_rows($result);$i++){
    $red=mysqli_fetch_object($result);
    echo $red->ime." ".$red->prezime." (".$red->godina.")<br>";
}
mysqli_data_seek($result, 0);
echo "<h4>WHILE petlja</h4>";
while($red=mysqli_fetch_object($result))
    echo $red->ime." ".$red->prezime." (".$red->godina.")<br>";
//Zatvaranje konekcije
mysqli_close($con);
?>
```

Transakcije

- Transakcije omogućavaju da se jedna ili više SQL komandi enkapsulira u jedinstvenu operaciju, odnosno celinu, te na taj način one predstavljaju važnu komponentu kada se govori o integritetu podataka u bazi.
- Odgovor na pitanje zašto koristiti transakcije sadrži se u jednoj reči - **ACID**.
- **Atomicity** (atomičnost ili nedeljivost) – ovaj pojam označava da je transakcija nedeljiva, tj. da se može izvršiti ili ne izvršiti; polovična izvršavanja nisu moguća.
- **Consistency** (konzistentnost)– ovaj pojam označava da će nakon izvršavanja transakcije baza biti u konzistentnom stanju; ukoliko transakcija izazove neko nedozvoljeno stanje ili neadekvatno ažuriranje podataka, sve će biti vraćeno na početno stanje.
- **Isolation** (izolacija) – ovaj pojam označava da se više transakcija može izvršavati simultano, tj. u isto vreme bez ikakvih loših posledica.
- **Durability** (izdržljivost) – pojam koji se odnosi na osobinu transakcija da budu sačuvane čak iako se odmah nakon njihovog završetka dogodi otkaz sistema i prekid rada baze.

Transakcije

- Transakcija kao skup SQL upita se mogu izvršiti u celosti ili poništiti nakon realizacije, tj. vratiti unazad
- Kada se transakcija izvrši, sve promene u bazi, koje su posledica delovanja skupa SQL upita se izvrše samo ako se cela transakcija izvrši, tj. sve promene se ponište ako se transakcija poništi ili ne realizuje.
- Nema delimičnog izvršavanja jednog po jednog upita, kao u klasičnim upotrebama SQL upita.
- Transakcija se završava kada se ili izvrši - **COMMIT** ili poništi - **ROLLBACK**.

Transakcije

- Od novije verzije, MySQL podržava transakcije, ali mora tip tabele da bude InnoDB
  (Storage Engine= InnoDB)
- Podrazumevano, MySQL server se nalazi u takozvanom **autocommit modu**. Ovo znači da se svaka SQL komanda izvršava u okviru jedne male transakcije. Zato se za naredbe može reći da se odigravaju transakciono, u smislu toga da će ili biti urađene, ili neće biti urađene, ali svakako, neće biti urađene napola.
- Ovo znači da ukoliko brišemo neki zapis iz baze podataka komandom DELETE mi u stvari izvršavamo jednu transakciju, (naravno pod uslovom da baratamo tabelom koja je tipa InnoDB).

Transakcije

- Za transakcije su nam bitne 3 funkcije
  - **mysqli_begin_transaction($con)** – oznaka za početak transakcije
  - **mysqli_commit($con)** – oznaka da se svi upiti smeste u bazu
  - **mysqli_rollback($con)** – oznaka da je nešto pošlo po zlu i da se baza vrati u oblik pre transakcije

Transakcije

- U ovom primeru će se izvršiti 100 INSERT upita, ali zbog mysqli_rollback() nijedan od njih neće biti zapamćen u bazi (**NAPOMENA**: id – jevi korišćeni za nove zapise će biti iskorišćeni)

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");

mysqli_begin_transaction($con);

for($i=0;$i<100;$i++){
    mysqli_query($con, "INSERT INTO korisnici (ime, prezime, godina) VALUES ('Pera', 'Perić', 2000)");
}
mysqli_rollback($con);//mysqli_commit($con);
//Zatvaranje konekcije
mysqli_close($con);
?>
```

Transakcije

```php
<?php
require_once "konekcija.php";

if($konekcija){
    $upit1 = "insert into tabela1(ime, prezime) values('Laza', 'Lazic') ";
    $upit2 = "insert into tabela1(ime, prezime) values('Bane', 'Banic') ";
    $upit3 = "insert into tabela1(ime, prezime) values('Milica', 'Jovic') ";

    // pocetak transakcije
    $konekcija->beginTransaction();

    $konekcija->query($upit1);
    $konekcija->query($upit2);
    $konekcija->query($upit3);

    // realizacija transakcije
    $konekcija->commit();
}
else{
    echo "Nema konekcije sa bazom";
}
?>
```

Transakcije try-catch

```php
<?php
//Pokušaj konekcije
$con=mysqli_connect("localhost", "root", "", "pva");
//Provera konekcije
if(mysqli_connect_error())//if(mysqli_connect_errno()), if(!$con)
{
    echo "Neuspela konekcija na bazu!!!!<br>GREŠKA: ".mysqli_connect_error();
    exit();
}
//Izvršavanje upita za encoding
mysqli_query($con, "SET NAMES utf8");
//Početak transakcije
mysqli_begin_transaction($con);
try
{
    //Izvršavanje 100 INSERT upita
    for($i=0;$i<100;$i++)
    {
        mysqli_query($con, "INSERT INTO korisnici (ime, prezime, godina) VALUES ('Pera', 'Perić', 2000)");
        //mysqli_query ne baca exception, ako želimo da koristimo try...catch blok, moramo ručno da bacimo Exception
        if(mysqli_error($con))throw new Exception("Greška prilikom izvršavanja upita");
    }
    //Ako je sve OK radi se commit
    mysqli_commit($con);
}
catch(Exception $e)
{
    //Ako nije u redu hvata se exception
    echo $e->getMessage();
    //Radi se rollback
    mysqli_rollback($con);
}
//Zatvaranje konekcije
mysqli_close($con);
?>
```

PDO – PHP data object

- PHP ima podršku za rad sa svim bazama podataka
- Obzirom da različite baze podataka imaju različite mogućnosti, PHP je, od verzije 5, uveo PHP Data Object
- To je klasa za rad sa svim bazama podataka.
- Sve metode se identično koriste, samo se razlikuje string za konekciju

```php
<?php
$db1=new PDO("mysql:host=localhost;dbname=pva", "root", "");
$db2=new PDO("sqlite:baza.sqlite");
?>
```

PDO – PHP data object

- Konekcija sa MySQL serverom se realizuje preko objekta PDO tj. pomoću **new PDO()**
- Ključnu ulogu imaju parametri koji se definišu za putanju do servera baze, ime baze, username i password

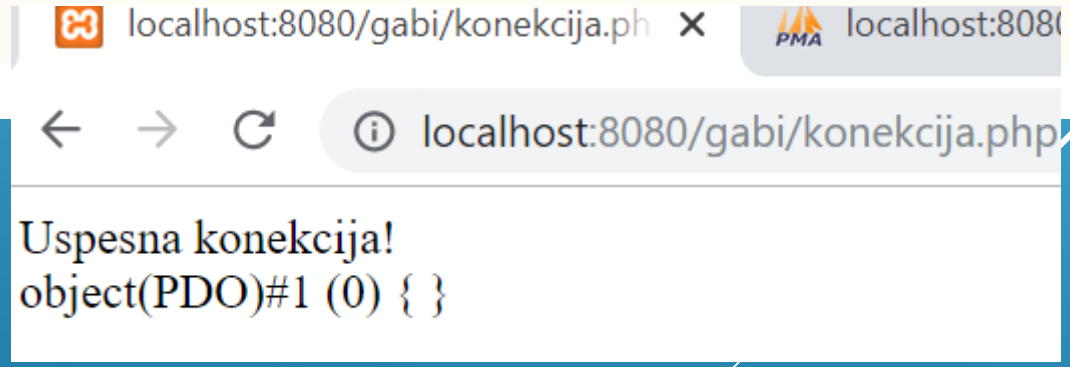
```
$konekcija=new PDO("mysql:host=localhost;dbname = test",$user,$pass);
```

- Ako se koristi lokalni web server tada je putanja **localhost** , ime baze ono koje smo napravili, username je **root** , a password je **prazan string.**
- Ako se koristi komercijalni server, ovo su parametri koji se dobiju od provajdera ili se kreiraju na serveru baze podataka.

PDO – PHP data object

- Parametri konekcije se mogu izdvojiti u posebne promenljive ili konstante

```php
<?php
$serverBaze = "localhost";
$username = "root";
$password = "";
$bazaPodataka = "preduzece1";
$konekcija = new PDO("mysql:host=$serverBaze;dbname=$bazaPodataka",$username,$password);
echo "Uspesna konekcija!<br>";
var_dump($konekcija);
?>
```

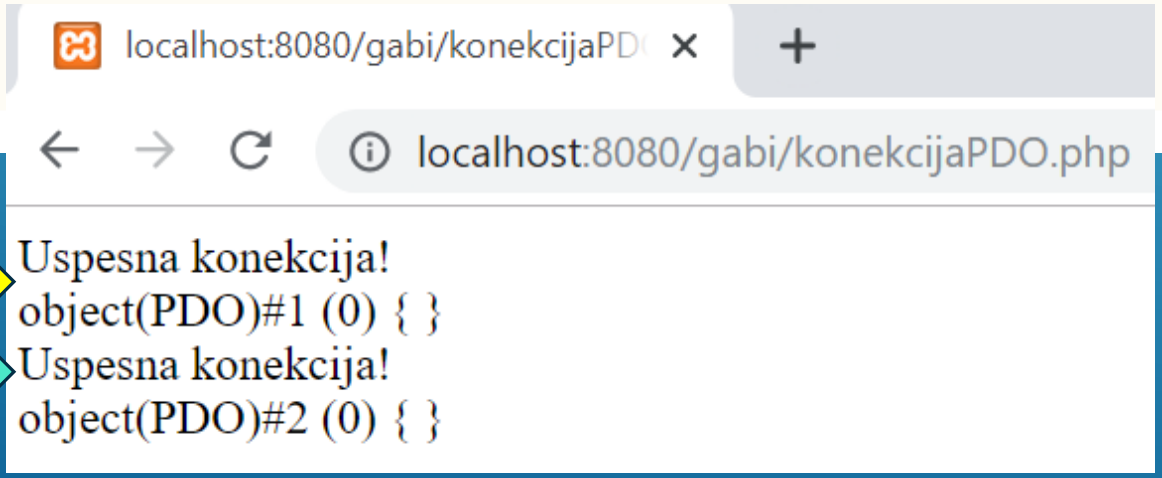


PDO – PHP data object

- Objekat PDO, kojim se definiše jedna konekcija, se smešta u neku promenljivu
- Ta promenljiva je identifikator konekcije i u sebi sadrži kreirani PDO objekat
- U ovoj promenljivoj se pamte svi podaci o tome gde se nalazi baza, koji su pristupni parametri i da li je uspostavljena konekcija i može se pozvati svaki put kada je potrebno bilo šta da se radi sa bazom podataka
- U istom kodu može se napraviti veći broj PDO objekata, sa različitim bazama i koristiti se istovremeno različitim konekcijama.

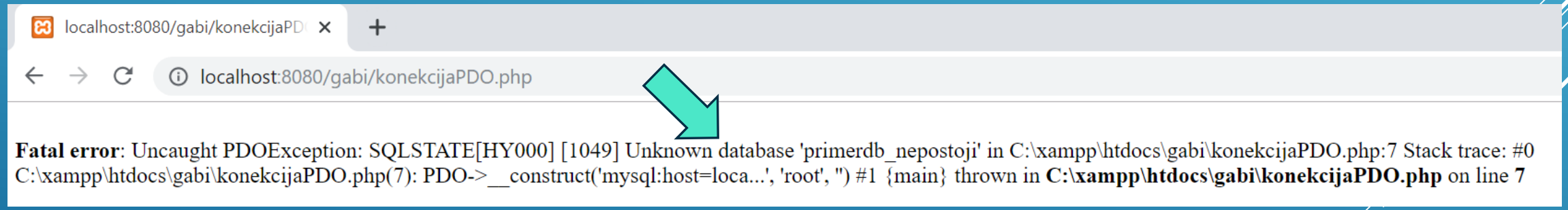
Kreiranje dva PDO objekta

```php
<?php
$serverBaze = "localhost";
$username = "root";
$password = "";
$bazaPodataka = "preduzece1";
$bazaPodataka1 = "preduzece";

$konekcija = new PDO("mysql:host=$serverBaze;dbname=$bazaPodataka",$username,$password);
echo "Uspesna konekcija!<br>";
var_dump($konekcija);

$konekcija1 = new PDO("mysql:host=$serverBaze;dbname=$bazaPodataka1",$username,$password);
echo "<br>";
echo "Uspesna konekcija!<br>";
var_dump($konekcija1);
?>
```



Konekcija na bazu koja ne postoji

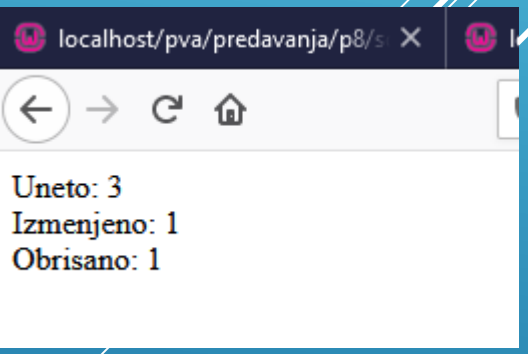
```php
<?php
$serverBaze = "localhost";
$username = "root";
$password = "";
$bazaPodataka = "primerDB_nepostoji";

$konekcija = new PDO("mysql:host=$serverBaze;dbname=$bazaPodataka",$username,$password);
echo "Uspesna konekcija!<br>";
var_dump($konekcija);
```



PDO i MySQL – upiti bez resultset-a

- Za razliku od mysqli klase, koja ima isti način izvršavanja upita, PDO ima posebne metode za upite sa resulset-om i bez resulset-a
- Metoda **exec()** se koristi za upite bez resultset-a
- Odgovor metode je broj izmenjenih/obrisanih redova

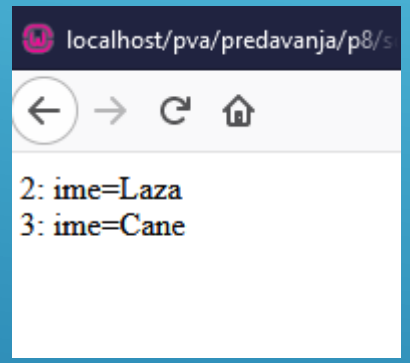
```php
<?php
$db=new PDO("mysql:host=localhost;dbname=pva", "root", "");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu";
    exit();
}
$brojUnetihRedova=$db->exec("INSERT INTO korisnici (ime, prezime, godina) VALUES ('Pera',
'Peric', 1976), ('Laza', 'Lazic', 1990), ('Cane', 'Kurbla', 1980)");
echo "Uneto: $brojUnetihRedova<br>";
$brojIzmenjenihRedova=$db->exec("UPDATE korisnici SET ime='Jovan' WHERE ime='Pera'");
echo "Izmenjeno: $brojIzmenjenihRedova<br>";
$brojObrisanihRedova=$db->exec("DELETE FROM korisnici WHERE ime='Jovan'");
echo "Obrisano: $brojObrisanihRedova<br>";
?>
```



PDO i MySQL – upiti sa resulset-om

- Metoda **query()** se koristi za upite sa resultset-om
- Metoda vraća i asocijativni i numerički niz

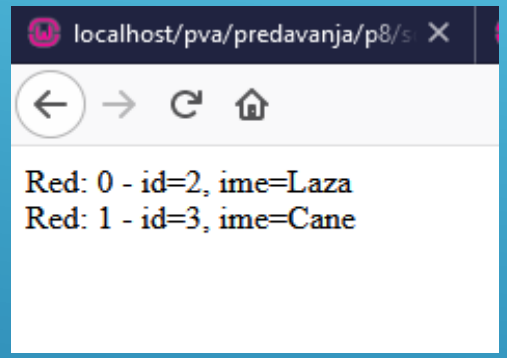
```php
<?php
$db=new PDO("mysql:host=localhost;dbname=pva", "root", "");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu";
    exit();
}
$rez=$db->query("SELECT * FROM korisnici");
foreach($rez as $red)
    echo $red[0].": ime=".$red['ime']."<br>";

?>
```



PDO i MySQL – upiti sa resulset-om

- Korišćenjem parametara metode query() može se dobiti asocijativni niz

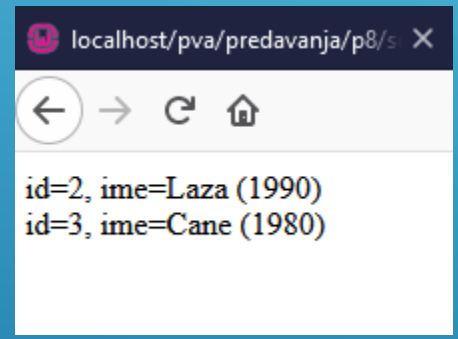
```php
<?php
$db=new PDO("mysql:host=localhost;dbname=pva", "root", "");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu";
    exit();
}
$rez=$db->query("SELECT * FROM korisnici");
$pomRez=$rez->fetchAll(PDO::FETCH_ASSOC);
foreach($pomRez as $k=>$v)
    echo "Red: ".$k." - id=".$v['id'].", ime=".$v['ime']."<br>";
?>
```



PDO i MySQL – upiti sa resulset-om

- Korišćenjem parametara metode query() rezultat možemo odmah staviti u objekat klase

```php
<?php
class Korisnici{
    public $id;
    public $ime;
    public $prezime;
    Public $godina;
}
$db=new PDO("mysql:host=localhost;dbname=pva", "root", "");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu";
    exit();
}
$rez=$db->query("SELECT * FROM korisnici");
$pomRez=$rez->fetchAll(PDO::FETCH_CLASS, "Korisnici");
foreach($pomRez as $k)
    echo "id=$k->id, ime=$k->ime ($k->godina)<br>";
?>
```



PDO i MySQL – parametarski upiti

- PDO nam omogućava i parametarske upite

```php
<?php
$db=new PDO("mysql:host=localhost;dbname=pva", "root", "");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu";
    exit();
}
$sql=$db->prepare("INSERT INTO korisnici (ime, prezime, godina) VALUES (:ime, :prezime, :godina)");
$ime="Joca";
$prezime="Karburator";
$godina=1999;
$sql->bindParam(":ime", $ime);
$sql->bindParam(":prezime", $prezime);
$sql->bindParam(":godina", $godina);
$sql->execute();
?>
```

PDO i MySQL – transakcije

- Slično kao i kod MySQL, PDO podržava transakcije.
- Razlika je što kod MySQLi klase sami moramo da bacimo Exception, dok kod PDO klase to se događa automatski

```php
<?php
$db=new PDO("mysql:host=localhost;dbname=pva", "root", "");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu";
    exit();
}
$db->beginTransaction();
try
{
    //Izvršavanje 100 INSERT upita
   for($i=0;$i<100;$i++)
   {
       $db->exec("INSERT INTO korisnici (ime, prezime, godina) VALUES ('Pera', 'Perić', 2000)");
   }
    //Ako je sve OK radi se commit
    $db->commit();
}
catch(Exception $e)
{
    //Ako nije u redu hvata se exception
    echo $e->getMessage();
    //Radi se rollback
    $db->rollBack();
}
?>
```

Raskid veze sa bazom podataka

- Do sada opisani način je definisao način uspostavljanja veze sa bazom, kreiranje PDO objekta tj. identifikatora veza sa bazom
- Zatim treba da slede operacije koje su predviđene za rad sa bazom
- Nakon tih operacija, vezu sa bazom treba raskinuti
- Raskidanje veze sa bazom, za identifikator $konekcija je setovanje na vrednost null
- **$konekcija = null;**

SQLite

- SQLite je mala, portabilna, efikasna i relativno brza baza podataka.
- Koristimo je kad želimo da rukujemo sa umerenom količinom podataka i kada nam nisu neophodne jake performanse.
- Ne zahteva nikakav sistem za skladištenje, već se smešta u običnu datoteku.
- Potrebno je da se uključi biblioteka za rad sa SQLite.

SQLite - konekcija

- Ukoliko baza (datoteka) ne postoji, biće kreirana.
- Ukoliko postoji vratiće se referenca na postojeću datoteku (neće biti izbrisana).

```php
<?php
$db=new SQLite3("baza.sqlite");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu!!!!";
    exit();
}
else
    echo "Uspešna konekcija!!!!";
?>
```

SQLite – provera konekcije

- U prethodnom primeru postoji provera konekcije na SQLite bazu.
- Provera konekcije u slučaju rada sa ovom bazom je nepotrebna jer konekcija nikad neće biti neuspešna.
- Baza se pravi na lokalnom računaru, tako de ne postoji mogućnost da je baza nedostupna.
- U najgorem slučaju možemo da pogrešimo ime baze, ali ni tada nećemo dobiti grešku, već će se otvoriti nova baza sa navedenim imenom.

SQLite – kreiranje baze

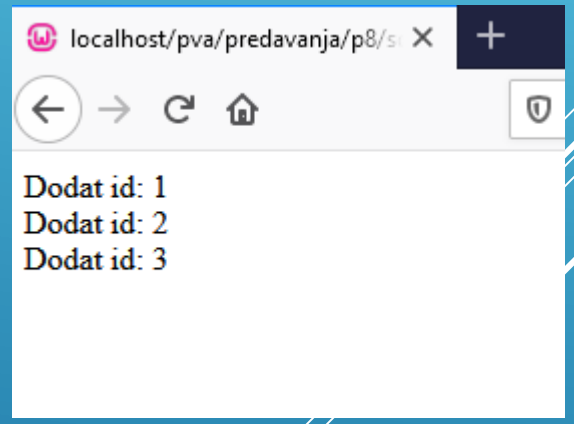
- Postoje razvojna okruženja za rad sa SQLite bazama, ali najčešći (i najjednostavniji) način rada je sa SQL upitima
- Za izvršavanje upita koristi se **exec()** metoda ako je upit bez resulset-a, a **query()** ako je rezultat upita resulset
- Parametar metode je SQL upit koji treba da se izvrši

```php
<?php
$db=new SQLite3("baza.sqlite");
if(!$db)
{
    echo "Greška prilikom konekcije na bazu!!!!";
    exit();
}
else
    echo "Uspešna konekcija!!!!";
$db->exec("CREATE TABLE korisnici id integer primary key, ime varchar(255)");
?>
```

SQLite

- Pored exec() metode, postoji još metoda i funkcija koje možemo koristiti za rad sa SQLite bazama podataka
- Metoda **lastInsertRowId()** vraća poslednji dodat id u bazi

```php
<?php
    $db=new SQLite3("baza.sqlite");
    $db->exec("CREATE TABLE korisnici id integer primary key, ime varchar(255)");
    $db->exec("INSERT INTO korisnici VALUES (null, 'Petar')");
    echo "Dodat id: ".$db->lastInsertRowID()."<br>";
    $db->exec("INSERT INTO korisnici VALUES (null, 'Laza')");
    echo "Dodat id: ".$db->lastInsertRowID()."<br>";
    $db->exec("INSERT INTO korisnici VALUES (null, 'Mile')");
    echo "Dodat id: ".$db->lastInsertRowID()."<br>";
?>
```



SQLite upiti

- Metoda **fetchArray()** služi za dohvatanje rezultat iz SQLite baze
- Postoje parametri kojima možemo dohvatiti asocijativni i numerički niz, samo asocijativni i samo numerički

```php
<?php
    $db=new SQLite3("baza.sqlite");
    $rez=$db->query("SELECT * FROM korisnici");
    //Asocijativni i numerički
    while($red=$rez->fetchArray())
        echo "ID ".$red[0].", ime=".$red['ime']."<br>";
    echo "<hr>";
    //Asocijativni
    while($red=$rez->fetchArray(SQLITE3_ASSOC))
        echo "ID ".$red['id'].", ime=".$red['ime']."<br>";
    echo "<hr>";
    //Numerički
    while($red=$rez->fetchArray(SQLITE3_NUM))
        echo "ID ".$red[0].", ime=".$red[1]."<br>";
?>
```

SQLite – Uništavanje konekcije

- Uništavanje konekcije se vrši metodom **unset()**
- Kao parametar ide konekcija na bazu

```
unset($db);
```