



Baza podataka služi da skladišti željene podatke, da te podatke organizuje po tabelama, i da ih zatim jednostavno i brzo dostavi 'klijentu' koji traži te podatke...

Verovatno se pitate kako sve to funkcioniše...

Postoji na jednoj strani server [gde se čuvaju podaci] - i postoji na drugoj strani klijent [koji traži podatke od servera].

Komunikacija između klijenta i servera se odvija SQL jezikom!

SQL sintaksa je veoma jednostavna, i razumljiva.

Dakle, klijent pošalje 'upit' serveru - i server pošalje klijentu odgovor - tj. željene podatke iz baze.

Evo jednog najprostijeg primera SQL upita [to se također zove i SQL query, ili samo query] :

Kod:

```
SELECT ime FROM tabela_korisnici WHERE prezime='Damnjanovic';
```

Server će vratiti podatak klijentu:

Kod:

```
Predrag
```

Tako sve to funkcioniše...

Klijent traži, ili šalje, podatke... server obradi zahtev, i pošalje odgovor klijentu...

SQL se uglavnom primenjuje u programskim jezicima.

Dakle, program traži/salje podatke serveru, server obradi zahtev, i vrati rezultat programu.

To je i svrha baza podataka, da brzo i efikasno skladišti podatke i radi sa njima.

U svakom programskom jeziku postoje funkcije/klase/komponente za povezivanje sa SQL bazom.

Ako vam treba pomoć oko povezivanja - napisite ovde u kom programskom jeziku programirate, i koju SQL bazu koristite, pa ćemo pokušati da rešimo vaš problem.

Ja ću da radim po MySQL sintaksi, posto to jedino i poznajem..

SQL sintakse se u principu ne razlikuju mnogo, tako da će 99% toga raditi na svim SQL bazama.

Ako na svom računaru nemate ni jednu bazu - instalirajte MySQL [linux korisnici to imaju] -

<http://dev.mysql.com/downloads/mysql/4.1.html>

Uputstvo za instalaciju - <http://www.mycity.co.yu/phpbb/viewtopic.php?t=57>

Bilo je dosta reči o MySQL instalaciji - tako da možete pogledati ovaj forum [i eventualno PHP forum].

Ako imate problema sa instalacijom, prijavite ih odmah... mada nije neophodno da imate instaliranu bazu...

Možete i samo ovako teorijski da učite, ali je ipak preporučljivo da sve to imate, i sami eksperimentirate, i sami radite sa bazom podataka [kada dovoljno naučite]...

Pa da posle krenemo sa prvom lekcijom...

1) Organizacija podataka

Baza podataka se sastoji iz [vise] tabela...

Dakle, podaci su u bazi organizovani po tabelama [a jedna baza moze imati mnogo tabela]

Same tabele sadrze podatke... u pitanju su obicne dvodimenzionalne tabele...

Na vrhu su polja [kolone]... (na primer: Ime, Prezime, Godiste...) a na dole se redjaju podaci...

Primer:

Kod:

```
tabela1 :
```

Ime	Prezime	Godiste
Predrag	Damnjanovic	83
Petar	Petrovcic	84
Jasmina	Nikolic	80

2) Tipovi polja

Polja [kolone] imaju svoj 'tip', tako da se tacno zna koji tip podataka moze da ide u koje polje.

Na primer... u polju 'Godiste' moze da se upise samo broj... ne moze tekst... jer je polje predvidjeno samo za brojcanе vrednosti.

Najcesce korisceni tipovi su (MySQL):

INT - za brojeve

REAL - za brojeve sa decimalama

CHAR - za samo jedan karakter

VARCHAR - za vise karaktera (do 255), s tim sto se navede broj karaktera za polje, recimo - VARCHAR(30)

TINYTEXT - za tekst do 256 karaktera

TEXT - za tekst do 65.000 karaktera

BLOB - za 'binarni' sadrzaj [do 65k bajta]...

MEDIUMTEXT, MEDIUMBLOB, BIGTEXT, BIGBLOB - sve isto, samo su polja veca...

DATE - za datume

TIME - za vreme...

Ako bismo hteli da kreiramo onu gore tabelu, iz primera... polja bi bila ovako 'organizovana' :

Ime - bilo bi tipa TINYTEXT, ili eventualno VARCHAR(20) ako bi smo hteli da uštedimo prostor, posto ne postoje imena duza od 20 slova

Prezime - isto kao i za Ime, TINYTEXT, ili VARCHAR(20)

Godiste - tu cemo staviti INT, posto se upisuju samo brojcanе vrednosti

Za Access korisnike: umesto INT staviti NUMBER, i umesto TINYTEXT staviti STRING

3) Kreiranje tabele

Kao sto smo rekli u prethodnoj lekciji... sve operacije sa bazom se obavljaju SQL komandama...

Komanda za kreiranje tabele je CREATE TABLE.

SQL query, koji trebamo da posaljemo serveru, da bi kreirali bazu, je :

Kod:

```
CREATE TABLE tabela1 (  
ime VARCHAR(20),  
prezime VARCHAR(20),  
godiste INT  
);
```

Server ce naravno poslati odgovor da je tabela "tabela1" uspesno kreirana...

Za one koji zele jos neke sitnice da saznaju (koje vama za sada nisu bitne) :

- 1) http://www.w3schools.com/sql/sql_create.asp
- 2) http://www.mysql.com/doc/en/CREATE_TABLE.html
- 3) <http://www.google.com/search?hl=en&lr=&q=create+table>

4) Za one kojima je dosadno (i koji trce ispred vremena)

Ovo ispod cemo sve uciti posebno, u sledecim lekcijama...

Tabela se puni na sledeci nacin:

Kod:

```
INSERT INTO tabela1 (ime, prezime, godiste)
VALUES ('Mika', 'Kostic', 80);
```

Opsirnije: http://www.w3schools.com/sql/sql_insert.asp

Tabela se cita na sledeci nacin... primeri:

Kod:

```
SELECT * FROM tabela1;
SELECT ime FROM tabela1;
SELECT ime, godiste FROM tabela1;
SELECT ime, prezime FROM tabela1;
SELECT ime, prezime FROM tabela1 WHERE godiste>80;
```

Opsirnije: http://www.w3schools.com/sql/sql_select.asp

Napominjem ponovo, ovo su samo primeri, da vam nije dosadno, za one koji zele da pozure sa lekcijama. Sve cemo ovo posebno objasniti, u sledecim lekcijama.

INSERT je SQL naredba/upit za dodavanje novih zapisa u bazu podataka. sintaxa INSERT naredbe je sledeca:

```
INSERT INTO ime_tabele [(spisak_atributa)] VALUES (spisak_vrednosti_atributa);
```

atribut je ime kolone (u ovoj tabeli, atributi su: ime, prezime i godiste).

INSERT naredba dodaje spisak_vrednosti_atributa u ime_tabele. spisak_atributa je neophodan ako zelite da dodate samo neka polja. ako zelim da dodam samo ime i prezime, ali ne i godiste (pod uslovom da se za polje godiste dozvoljava da bude ne uneseno).

SQL upit:

Kod:

```
INSERT INTO tabela1 (ime, prezime) VALUES ('Slobodan', 'Kasic');
```

ovaj sql upit ce generisati ovakvu tabelu

Kod:

```
tabela1 :
```

Ime	Prezime	Godiste
Predrag	Damnjanovic	83
Petar	Petrovcic	84
Jasmina	Nikolic	80
Slobodan	Kasic	

SQL nije casesensitive i nije bitno da li cete napisati InSeRt/INSERT/INSeRt/insert, potpuno je svejedno, ali ja pisem velikim slovima radi preglednosti, nista vise.
za attribute nikada ne stavljajte nesto kao text, string ili neki drugi tip polja, jer moze doci do greske pri izvrsavanju upita. malo je teze pronaci takvu gresku jer se ne nadas da je to problem 😊.ako je tip polja/atributa string/text, u INSERT upitu (a i ostalih upita), morate da stavite ' i ', ako je u pitanju broj, onda je bez '.

Peca je u prosloj lekciji dao linkove gde mozete dodatno da ucite SQL tako da ja necu stavljati ponovo... nadam se da nisam nista zaboravio 😊

Select upit sluzi kako bi iz neke tabele 'izvadili' selektovali neki zapis ili vise zapisa po nekom odredjenom kriterijumu.
sintaxta select upita je sledeca:

Kod:

```
SELECT * FROM ime_tabele [WHERE id=13];
```

znak * znaci da cemo iz tabele ime_tabele da selektujemo sve attribute, a ime_tabele je zapravo ime tabele u kojoj se

nalaze ti selektovani atributi.

ako zelimo da iz tabele selektujemo sve korisnike koji su rodjeni 83 godine upit bi izgledao ovako:

Kod:

```
SELECT * FROM tabelal WHERE Godiste=83;
```

kao rezultat ovog upita, dobicemo ovakvu tabelu

Kod:

Ime	Prezime	Godiste
Predrag	Damnjanovic	83

ova tabela je virtuelna tabela i nalazi se u memoriji racunara.

bazu cemo azurirati (update jos nismo radili, za dva casa)

Kod:

```
UPDATE tabelal SET Godiste=83 WHERE Prezime='Kasic';
```

ovaj sql upit ce u vec postojeci zapis, dodati ono prazno poje koje je bilo (Godiste) cela tabela izgleda ovako:

Kod:

Ime	Prezime	Godiste
Predrag	Damnjanovic	83
Petar	Petrovcic	84
Jasmina	Nikolic	80
Slobodan	Kasic	83

kada se sada izvrsi isti onaj SELECT upit, virtuelna tabela ce biti ovakva:

Kod:

Ime	Prezime	Godiste
Predrag	Damnjanovic	83
Slobodan	Kasic	83

sa ovakvim rezultatom upita, mozete da radite sta vam je volja. da ga sortirate ili ispisujete negde.

ako zelite da izracunate (izbrojite) koliko zapisa ima u tabeli sledeci sql upit resava to:

Kod:

```
SELECT count(*) FROM tabelal;
```

rezultat upita ce biti broj 4.

postoje i malo komplikovaniji delovi SELECT upita (ugnjezdeni upiti)

hipoteticka situacija:

imate dve tabele. u jednoj spisak CD/DVD, a u drugoj grupa u kojoj isti pripadaju. tabele ce ovako nekako izgledati:

Kod:

```
tabela: cddvd
```

Broj	Ime	Grupa
1	Worms	1
2	Flash	2
3	Linux	3
4	VS.NET	4

Kod:

tabela: grupa

Broj	Ime
1	Igre
2	Programi
3	OSystems
4	Programiranje

ovako realizovana tabela sprečava neke od anomalija (anomalije pri brisanju, dodavanj i azuriranju)

možli smo to uraditi i sa jednom tabelom stin sto bi onda u atributu Grupa zapravo bilo ime grupe, ali to nije dobro. tako se ne radi zbog pomenutih anomalija (o njima kasnije)

ove dve tabele su spojene sa atributima Grupa (iz prve tabele) i Broj (iz druge tabele)

Kod:

```
select cddvd.broj,cddvd.ime,grupa.ime from cddvd,grupa where cddvd.broj=grupa.broj;
```

ovaj sql upit ce vam dati ono sto zelite.
tabela je ovakva

Kod:

tabela: virtuelana

Broj	Ime	Grupa
1	Worms	Igre
2	Flash	Programi
3	Linux	OSystems
4	VS.NET	Programiranje

sve sto vam je potrebno je tu, a jos plus ste izbegli anomalije. malo je sql upit komplikovaniji, ali nije problem.

znaci, ono sto zelite da se prikaze u tabeli, morate navesti posle SELECT-a. kada radite sa dve ili vise tabele,

najbolje je da atribut pisete kao ime_tabele.ime_atributa da ne bi bilo zabune, a i da izbegnete greske, posebno ako

u dve tabele imate isti atribut (kao u mom slucaju Broj).

ovo je nesto sto ja najcesce koristim. ima tu jos ugnjezdenih f-ja. neke ne znam ni sam kako se koriste 😊

npr.

Kod:

```
SELECT max(Broj) FROM grupa;
```

vraca broj 4 posto je on najveći broj u atributu Broj

slučaj sa min(broj), vratice 1, kao najmanji.

umesto **SELECT *** mozete navoditi i polja koja zelite da dobijete, recimo **SELECT preizime, godiste...**

na primer:

SELECT ime FROM tabela1;

i dobicete

Kod:

Ime
Predrag
Petar
Jasmina
Slobodan

Posle malo duze pauze, nastavljam sa zapocetim poslom (za koji, BTW, jos nisam placen 🤖😊).
Potrudicu se da objasnim sto je moguće bolje.

WHERE je klauzula uslova. Sa WHERE određuje koje zapise cete brisati, selektovati, azurirati...
Opciona je, ali ako postoji, mora da ide posle FROM some_table naredbe.

Primer 1.0

Kod:

```
SELECT * FROM Table1 WHERE id=1;
```

Ovo prevedeno na obican jezik glasi: Izaberi sva polja iz tabele Table1 gde je id jednako 1.
WHERE mozete da kombinujete sa logickim izrazima tipa AND, OR, NOT.

Primer 1.1 (Selektuje sve zapise iz tabele Table1 gde je id 1 i 4)

Kod:

```
SELECT * FROM Table1 WHERE id=1 AND id=4;
```

Primer 1.1 (Selektuje sve zapise iz tabele Table1 gde je id ili 1 ili 4)

Kod:

```
SELECT * FROM Table1 WHERE id=1 OR id=4;
```

WHERE klauzula moze da sadrzi GROUP BY i ORDER BY klauzule.
GROUP BY ime_atributa za grupisanje po tom atributu

ORDER BY ime_atributa za prikazivanje u opadajucem (DESC) ili rastucem (ASC) redosledu.

WHERE klauzula takodje moze da sadrzi i drugu SELECT klauzulu (ne u MySQLu).
Ono sto je bitno kod ovakvog nacina pisanja SQL upita jeste da u jednom delu SQL upita izjednacite polje sa poljem iz drugog SQL upita (istog tipa) po kojem zelite da povežete te dve tabele.

Primer 1.2

Kod:

```
SELECT * FROM Tabela1 WHERE id1=(SELECT id2 FROM Tabela2 WHERE UserName='NekiUser');
```

Ovaj SQL upit ne radi nista pametno, ali sam samo hteo da pokazem syntaxu kako se pise.

Isti ovaj rezultat mozete postici na drugi nacin (ako neke baze podataka ne podrzavaju ugnjezdene upite kao mySQL npr.)

Primer 1.3

Kod:

```
SELECT * FROM Table1,Table2 WHERE Table1.id1=Table2.id2 AND UserName='NekiUser';
```

U ovom slucaju morate navesti ime_tabele.ime_atributa kako bi 'rekli' iz koje tabele da uzima podatke. Ovo isto moze i vama pomoci, posebno ako upit ne radi kako treba.

Ako ne zelite da pisete ime_tabele.ime_atributa, onda morate da proverite da ime svakog polja iz obe tabele bude unikatno kako ne bi doslo do gresaka. (Ovo se ne preporucuje)

Rezultat oba ova upita je isti.

Neke baze (MSSQL) kada poredite dva stringa u WHERE klauzuli, 'traze' da umesto = bude klauzula LIKE.

Bilo u slucaju koriscenja ugnjezdenog upita ili SQL upita iz Primera 1.3 kada SQL upit treba da vrati vise zapisa onda se umesto = koristi klauzula IN.

U klauzuli WHERE mozete da kombinujete logicke operacije po ovoj semi:

Argument1 OperatorUporedjivanja Argument2 i tako u nedogled 😊.

WHERE u DELETE

Sama DELETE klauzula ne bi bila ni malo upotrebljiva ako ne bi imali WHERE klauzulu osim u slucaju da zelimo da obrisemo sve zapise iz tabele.

Kada zelite neki odredjeni zapis (ili logicku kombinaciju zapisa) da obrisete iz tabele, u DELETE klauzulu morate da 'ukljucite' WHERE klauzulu.

Primer 1.4

Kod:

```
DELETE FROM Table1 WHERE UserName='NekiTamoUser';
```

Ovaj SQL upit ce obrisati citav jedan zapis iz tabele Table1 gde je atribut UserName = NekiTamoUser.

WHERE u UPDATE

Kao i DELETE, ni UPDATE ne bi bio upotrebljiv bez WHERE, osim ako ne zelimo da jednu promenu preslikamo na sve u tabeli.

Ako zelite da azurirate samo jedan zapis ili logicku kombinaciju vise zapisa koristite WHERE unutar UPDATE klauzule.

Primer 1.5 (Azuriranje zapisa gde je id=14)

Kod:

```
UPDATE SET ime_atributa=vrednost_atributa WHERE id=14;
```

Primer 1.6 (Azuriranje zapisa gde je id=14 i id=10 ukupno dve promene ime_atributa sa vrednost_atributa)

Kod:

```
UPDATE SET ime_atributa=vrednost_atributa WHERE id=14 AND id=10;
```

O UPDATEu nema sta puno da se kaze. (bar koliko ja znam 😊).
UPDATE klauzula služi za menjanje vrednosti jednog ili vise zapisa po nekom odredjenom kriterijumu.

Sadrzi WHERE uslov pomocu kojeg se menjaju neki atributi zapisa.

Kod:

```
UPDATE tabela SET ime_atribut1=vrednost_atribut1, ime_atribut2=vrednost_atribut2,  
ime_atributN=vrednost_atributN;
```

Posle spiska atributa cija vrednost treba da se promeni/izmeni, treba da stoji WHERE koji odredjuje koji zapis ce biti UPDATE-ovan.

Ako izostavimo WHERE uslov, svi zapisi u tabeli ce biti isti.

Kod:

```
UPDATE tabela SET grupa='industrija koze' WHERE id=14;
```

Ovaj sql upit ce promeniti vrednost atributa grupa u industrija koze svim zapisima kojima je ID=14.

Kao i za UPDATE, ni o DELETE nema mnogo stosta da se kaze.

DELETE služi za brisanje svih (time se ne brise definicija tabele, nego samo zapisi u njoj) zapisa ili samo selektovanih zapisa pomocu uslova.

Kod:

```
DELETE FROM tabela;
```

Ovaj sql upit brise sve zapise iz tabele.

Da bi ipak obrisali samo zeljeni zapis, treba se dodati uslov WHERE.

Kod:

```
DELETE FROM tabela WHERE id=14;
```

Ovaj sql upit brise sve zapise ciji je ID=14.

A. Advanced SQL-ing 😊

Ovo je Appendix 😊 kao sto imate po knjigama. Zelim da vam pokazem neke naprednije upite koji sam ja koristio i koju su veoma korisni.

Zamislite sledecu situaciju.

Zelite da napravite nesto, da kazemo katalog knjiga (strucna literatura 😊).

Izmedju ostalog, ta aplikacija treba da sadrzi i ime knjige i grupu u kojoj pripada radi lakseg pronalazenja iste. Vidjao sam da su neki sve te informacije (ime knjige, grupa, korisnik koji je uzeo knjigu...) dodavali u jednu tabelu. To nije dobro resenje jer se tu javljaju tri osnovne (i jedine) anomalije.

1. Pri dodavanju zapisa
2. Azuriranju zapisa (Update)
3. Brisanju zapisa

Stim sto jos pravite redudantne podatke (bespotrebno ponavljanje zapisa/podataka).

Da obajsnim svaku od anomalija.

1. Anomalija pri dodavanju zapisa:

Uzmimo primer ove nase aplikacije za knjige.

Ako imamo tabelu u kojoj su smesteni korisnici te biblioteke i tabelu u kojoj su smestene knjige.

Radnik koji radi u knjizari, ne moze da doda zapise o nekom korisniku osim ako taj korisnik ne uzme neku knjigu. Zasto?

Zato sto se to sve nalazi u jednoj tabeli, a zamislite kako bi izgledala ta jedna tabela kada bi pola zapisa bilo prazno i jos bi se i ponavljali neki. Stim sto se pri projektovanju baze, za neka polja stavlja da ne mogu da prime null vrednost. Drugim recima, MORATE da upisete nesto.

Takvo polje bi bilo za ime knjige, grupu. Korisnik koji je uzeo knjigu je opciono.

Kako bi se zapis dodao u bazu, sva tri polja moraju biti popunjena (pored onih ostalih).

Isto to vazii za knjige. Ne mozemo dodati novu knjigu ako nemamo kome da je iznajmimo.

2. Anomalije pri azuriranju zapisa:

Kada promenimo ime knjige (iz bilo kod razloga) npr. tek posle nekoliko godina 😊 smo shvatili da nismo lepo napisali naslov knjige.

Sve te izmene moramo da vrsimo onoliko koliko je neko puta uzimao knjigu sto moze da dovede do greske i samim tim ne validnih podataka.

To isto vazii za grupe i korisnike.....

3. Anomalije pri brisanju zapisa:

Ako nam je neka knjiga slucajno ispala sa mosta 😊 naravno unistena, ljudi u knjizari (pored naplate stete 😊) moraju i da obrisu knjigu koja je bila prva knjiga u biblioteci staroj nekoliko

vekova 😊.

Da bi obrisali tu knjigu, moraju da brisu onoliko puta koliko je neko iznajmio tu knjigu. Samim tim (brisanjem) gube se informacije o korisnicima koji su uzimali tu knjigu.

Ista stvar je i sa grupama i korisnicima.....

'Tri tabele' To the rescue 😊

Ipak, postoji nacin da se resi sve ovo.

U nasem slucaju (knjige, grupe i korisnici) trebamo da napravimo tri tabele.

Jedna tabela za spisak svih knjiga sa svim njenim informacijama (sta god vam padne na pamet).

Tabela sa spiskom svih mogucih grupa (beletristika, kompjuteri, kompjuterska filozofija 😊...)

I tabela sa spiskom korisnika koji bi trebalo da uzimaju te knjige sa svim njihovim informacijama.

Kontrolni broj, broj dece...

U tabeli za knjige treba da se dodaju jos dva atributa (pored onih sa informacijama za knjige) za grupe i korisnike preko kojih ce se tabela knjige povezivati sa abelom grupe i tabelom

korisnici.

Sve tri tabele imaju atribut ID koji je primarni kljuc. Ime knjige je obavezno polje (ono not null) kao i grupaID dok je korisnikID opciono.

Kod:

```
knjige
+-----+
| id | ime knjige | grupaID | korisnikID |
+-----+
| 01 | kompjuterska fil. | 01 | |
| 02 | tom sojer | 02 | |
```

03	dig. integr. el.	03	
----	------------------	----	--

Kod:

```

grupe
+-----+
| id | ime grupe |
+-----+
| 01 | kompjuteri |
| 02 | lektira |
| 03 | elektronika |
+-----+

```

Kod:

```

korisnici
+-----+
| id | ime korisnika |
+-----+
| 01 | Mika |
| 02 | Pera |
| 03 | Sima |
+-----+

```

Tabele izgledaju ovako.

Kada zelimo da dodamo novog korisnika, Sve njegove informacije dodajemo u tabelu korisnici.

Kada zelimo da dodamo grupu, sve informacije o grupi dodajemo u tabelu grupe.

Isto tako i za knjige.

Kada dodje neki korisnik (Mika npr.) i zeli da uzme knjigu 'kompjuterska filozofija', operater zapisuje te podatke u tabelu knjige i to za tu knjigu koju je taj korisnik uzeo.

Tabela izgleda ovako:

Kod:

```

knjige
+-----+
| id | ime knjige | grupaID | korisnikID |
+-----+
| 01 | kompjuterska fil. | 01 | 01 |
| 02 | tom sojer | 02 | |
| 03 | dig. integr. el. | 03 | |
+-----+

```

Vidimo da je Mika povezan sa knjigom koju je uzeo preko njegovog ID (iz tabele korisnici) i ID iz tabele knjige tako da se zna ko je uzeo koju knjigu.

Kad bi ovo trebali da ispisemo, dobili bismo ne bas razumne podatke (mislim ovo, 01, 03, 10003, 383....). Za ispisivanje pravih podataka, treba nam malo komplikovaniji SQL upit, ali radi posao.

Teorijski opis SQL upita:

SQL upit treba da selektuje sve iz tabele knjige gde je grupaID=ID id tabele grupe I gde je korisnikID=ID iz tabele korisnici.

Prakticni opis SQL upita:

Kod:

```
SELECT * FROM knjige,grupe,korisnici WHERE knjige.grupaID=grupe.ID AND  
knjige.korisniciID=korisnici.ID;
```

Posle ovog SQL upita, normalno mozemo da pristupamo poljima i da ih ispisujemo gde treba.
Ako niko nije uzeo ni jednu knjigu, SQL upit nece vratiti ni jedan zapis.

To bi bilo to. Za sada. Ako imate nekih pitanja, slobodno pitajte.

Izvinjavam se zbog neurednosti tabela. Primetio sam tek kada sam postovao ovde, a sada me mrzi da ih ispravljam 😊.