

# Lekcija 01

## SQL: DDL i DML

*dr Svetlana Cvetanović*



Uvod

01

02

03

04

Uvod

SQL

DDL-CREATE  
TABLE

DDL-CREATE  
INDEX

DDL-ALTER  
TABLE

- ❑ *CREATE TABLE-ograničenja: PRIMARY KEY, UNIQUE, NULL/NOT NULL*
- ❑ *CREATE TABLE-FOREIGN KEY*
- ❑ *CREATE TABLE-CHECK*
- ❑ *CREATE TABLE-case study*
- ❑ *Vežba: Primeri naredbe CREATE TABLE*

05

DDL-DROP TABLE

- *Vežba: Primeri ALTER TABLE i DROP TABLE naredbe*

06

DML-INSERT

*Vežba: Primeri INSERT naredbe*

07

DML UPDATE

- ❑ *Vežba: Primeri UPDATE naredbe*

08

DML DELETE

09

DML SELECT

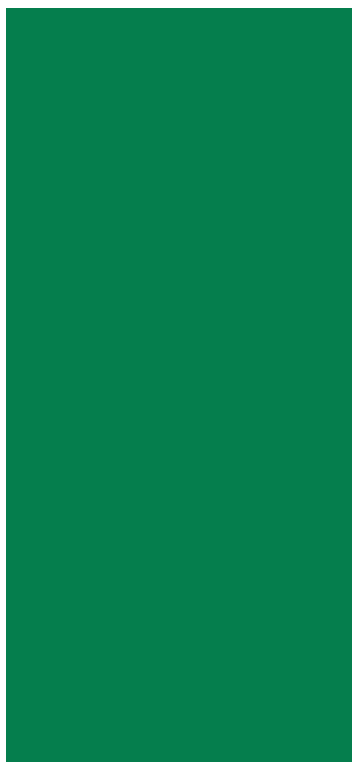
- ❑ *DML-Primena naredbe SELECT*
- ❑ *Vežba: Primeri bazicnih oblika SELECT naredbe*

10

Uvod u  
phpMyAdmin

11

Zaključak



# UVOD

## *Šta ćemo naučiti u ovoj lekciji?*

Naredbe **DDL-a** omogućuju definisanje resursa relacione baze podataka. Tu spadaju naredbe:

**CREATE TABLE** (kreiranje tabele baze podataka)

**CREATE INDEX** (kreiranje indeksa)

**ALTER TABLE** (izmena definicije tabele)

**DROP TABLE** (izbacivanje tabele iz baze podataka)

Naredbe za manipulaciju baza podataka (**DML**) omogućuju ažuriranje u širem smislu značenja te reči i izveštavanje iz **relacione baze podataka**. Ažuriranje u širem smislu značenja obuhvata dodavanje, izmenu sadržaja i brisanje reda ili redova tabele. Te osnovne operacije realizuju se SQL naredbama:

**INSERT** (dodavanje redova postojećoj tabeli)

**UPDATE** (izmena vrednosti kolona tabele)

**DELETE** (izbacivanje redova tabele)

U predavanju je opisan i osnovni oblik naredbe **SELECT**.

# SQL JE?

## *Standardni jezik za pristup bazi podataka.*

SQL je ANSI (American National Standards Institute) standardni računarski jezik za pristup i manipulaciju podataka u relacionim bazama podataka. Naredbe SQL-a se koriste za pretraživanje i ažuriranje podataka z bazama podataka.

Postoji mnogo različitih verzija SQL-a, ali da bi bile u skladu sa ANSI standardom, svaka od njih mora da podrži iste ključne reči na sličan način (kao što su SELECT, UPDATE, INSERT, DELETE, WHERE...).

S obzirom da se korišćenje SQL-a vezuje za korišćenje relacionih baza podataka, u ovom delu će biti data definicija relacionih baza podataka i osnovnih elemenata sistema za upravljanje relacionim bazama podataka (RDBMS).

Baza podataka se može definisati kao celokupnost međusobno povezanih podataka uskladištenih na spoljnoj memoriji uz minimalnu redundantnost (ponavljanje podataka), koja dozvoljava njihovo korišćenje od strane jedne ili više aplikacija.

# SQL TABELE

*Svaka baza se sastoji od tabela.*

Svaka baza podataka se sastoji od jedne ili više tabela, u kojoj se čuvaju podaci. Svaka tabela ima svoje jedinstveno ime i sastoji se od kolona i redova. Kolone tabela se zovu i polja tabela: one imaju svoje jedinstveno ime i unapred definisan tip podataka (na primer VARCHAR, NUMBER, INTEGER...).

<b>Ime</b>	<b>Prezime</b>	<b>EmailAdresa</b>	<b>Grad</b>
Hansen	Ola	John.Smith@yahoo.com	Sandnes
Svendson	Tove	goldfish@fishhere.net	Sandnes
Pettersen	Kari	jim@supergig.co.uk	Stavanger

Slika 1. Primer tabele kupac

Svaka kolona uzima jednu od skupa mogućih vrednosti definisanih kao domen. Na primer, domen kolone Prezime predstavlja skup svih prezimena koja mogu imati kupci. Vrednost kolone može biti i prazna što se označava sa NULL. Ukoliko kolona ne sme da bude prazna, uz nju se piše klauzula NOT NULL.

Kolone tabela se karakterišu i različitim svojstvima kojima se definiše funkcionalnost kolone. Tako kolona može biti primarni ključ, sekundarni ključ, predstavljati indeks, imati neku default vrednost itd.

Dok kolone tabela opisuju njihove tipove podataka, redovi tabela sadrže stvarne podatke kolona. U datom primeru, prvi red tabele je boldovan i sadrži imena kolona dok ostali redovi tabele sadrže vrednosti svake od kolona tabele.

# KLJUČEVI TABELA

*Ključ je kombinacija jedne ili više kolona.*

Ključ je kombinacija jedne ili više kolona koje se koriste za identifikaciju pojedinačnih redova u tabeli. Ključ može biti primarni ili sekundarni.

**Primarni ključ** jedinstveno identifikuje svaki red tabele. Može imati jednu kolonu ili biti složen, tj. sastojati se od više kolona.

**Sekundarni (strani) ključ** je kolona ili skup kolona koje predstavljaju primarni ključ neke druge tabele. To je ključ strane tabele u odnosu na onu u kojoj se ključ pojavljuje.

Primeri ključeva tabele:

**ODELJENJE (Odeljenjelme, IznosBudžeta, MenadžerIme)**  
**ZAPOSLENI (ZaposleniBroj, Zaposlenilme, *Odeljenjelme*)**

Primarni ključevi su podvučeni a strani ključ je označen italic formom.

Stranim ključevima se uspostavljaju relacije između redova tabela. U stranom ključu ZAPOSLENI-Odljenjelme se čuva relacija između zaposlenog i njegovog odeljenja. Vrednost stranog ključa mora da se podudara sa vrednošću primarnog ključa. U ovom slučaju se mora kreirati ograničenje:

**ZAPOSLENI.Odeljenjelme mora postojati u**  
**ODELJENJE.Odeljenjelme**



# PRIMER RELACIONE BAZE SA TRI TABELE

The image shows three database tables in a windowed application:

- STUDENT Table:**

StudentNumber	StudentName	EmailAddress
100	Cooke	Cooke@OurU.edu
200	Lau	Lau@OurU.edu
300	Harris	Harris@OurU.edu
400	Greene	Greene@OurU.edu
0		
- CLASS Table:**

ClassNumber	Name	Term	Section
10	Chem101	F04	1
20	Chem101	F04	2
30	Chem101	S05	1
40	Acct101	F04	1
50	Acct102	S05	1
0			
- GRADE Table:**

StudentNumber	ClassNumber	Grade
100	10	3.7
100	40	3.5
200	20	3.7
300	30	3.1
400	40	3.9
400	50	3.5
0	0	0

Slika 2. Relaciona baza sa 3 tabele

Slika prikazuje bazu podataka sa tri tabele: STUDENT, GRADE(Uspeh) i CLASS(Predmet).

Tabela STUDENT ima primarni ključ StudentNumber a tabela PREDMET ClassNumber. Tabela Ocene je u vezi sa tabelama STUDENT i PREDMET i ta relacija je ostvarena tako što se primarni ključevi ovih tabela nalaze u tabeli OCENE kao sekundarni ključevi.

# SISTEMI ZA UPRAVLJANJE RELACIONIM BAZAMA PODATAKA (RDBMS)

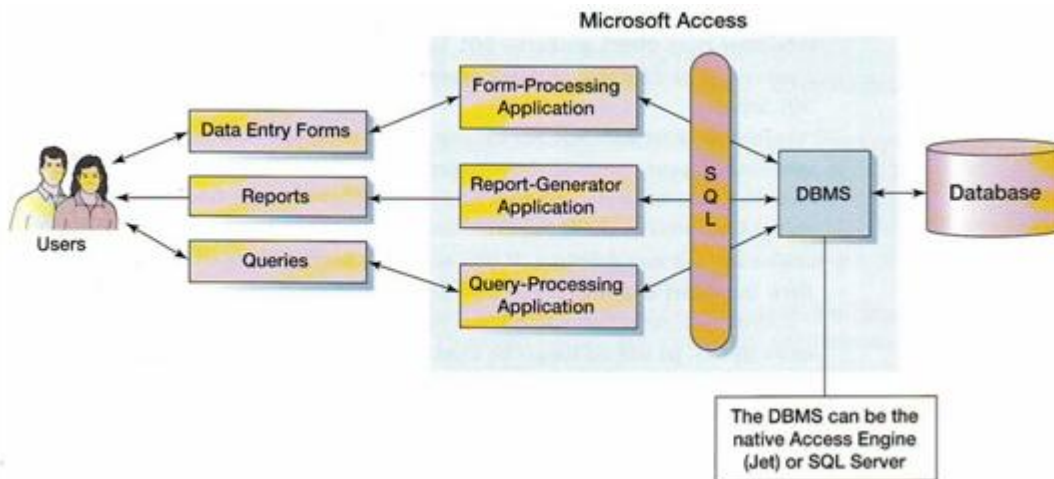
*Postoje tri komponente sistema.*

Ima tri komponente: aplikacije, SQL i RDBMS.

Aplikacije (za obradu formi, generisanje izveštaja i obranu upita) prihvataju podatke od korisnika, obrađujući ih prema zahtevima i koriste SQL za transfer podatak prema i od baze podataka.

SQL je standardni jezik koji razumeju svi komercijalni RDBMS sistemi.

RDBMS upravlja podacima smeštenim u relacionim bazama podataka.



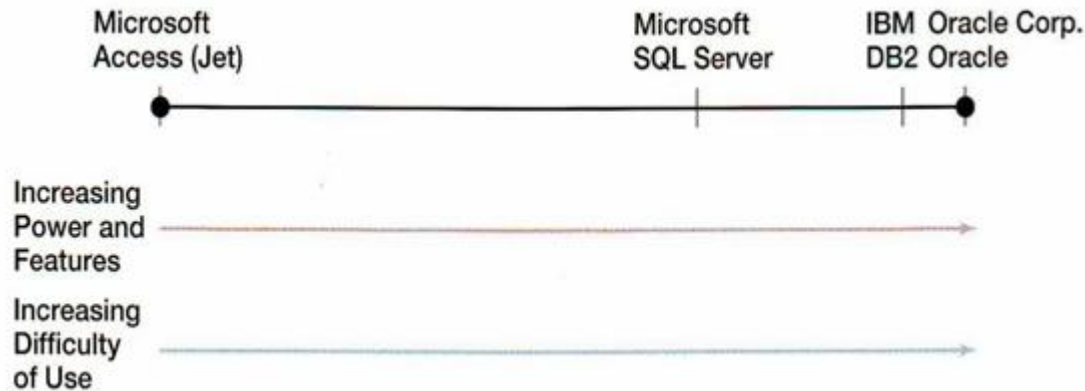
Slika 3. RDBMS

# NAJPOZNATIJI RDBMS SISTEMI

## *Jedan od najpoznatijih je MySQL*

Proizvodi su poređani po mogućnostima kojima raspolažu i lakoćom korišćenja. Tako se za Access može reći da je najlakši za korišćenje ali su i njegove mogućnosti najmanje, dok se Oracle može smatrati najmoćnijim DBMS-om koji je i najteži za korišćenje.

Jedan od nekomercijalnih RDBMS sistema, koji se može besplatno koristiti i kroz koji će biti prikazane osnovne naredbe SQL-a jeste MySQL.



Slika 4. Mogućnosti i lakoća korišćenja

# SQL


---

01

# ISTORIJA SQL-A

## *Prvi RDBMS koji je koristio SQL napravio je Oracle*

**Prvi komercijalni RDBMS koji je koristio SQL je napravila korporacija Oracle.** Mada su originalne verzije razvijane za VAX/VMS sisteme Oracle su bili jedni od prvih proizvođača koji su napravili DOS verziju RDBMS-a (Oracle je sada prisutan na skoro 70 platformi).

Sredinom osamdesetih Sybase je napravio RDBMS SQL server. Sa bibliotekama klijenata za pristup bazama podataka, podrškom za zapisane procedure i mogućnošću rada sa različitim mrežama, SQL Server je postao uspešan proizvod naročito u klijent/server okruženju. Jedna od najjačih tačaka oba snažna SQL Server sistema bila je pokretljivost kroz platforme. Kod u jeziku C (u kombinaciji sa SQL-om) pisan za Oracle na PC-u je virtuelno identičan kodu pisanom za Oracle bazu podataka koja se izvršava na VAX sistemu.

SQL je izgubio dosta od svoje osnovne karakteristike – struktuiranosti, jer je standard definisan kada su različiti proizvođači već imali u značajnoj meri razvijene sopstvene dijalekte SQL-a, tako da je standardni SQL praktično rezultat kompromisa najuticajnijih proizvođača. I pored toga **SQL je neophodno znati i to iz dva razloga:**

- u literaturi o relacionim bazama podataka često se ne objašnjava SQL, nego **se koristi kao standard, kao nešto što je poznato da bi se objasnili drugi koncepti relacionih baza podataka**
- SQL je implementiran u više od 50 komercijalno raspoloživih sistema za upravljanje bazama podataka (SUBP-ova) relacionog tipa (DB2, SQL/DS, ORACLE, ...). **Znajući SQL moguće je na gotovo identičan način raditi sa bilo kojim od njih.** Proizvođači relacionih RDBMS sa drugim upitnim jezicima prinuđeni su da omoguće alternativno korišćenje i SQL-a, kako bi opstali na tržištu.

# KARAKTERISTIKE SQL-A

## *Jednostavnost i jednoobraznost pri korišćenju, mogućnost interaktivnog i klasičnog (aplikativnog) programiranja, neproceduralnost*

SQL je jezik koji je okrenut korisniku. Uči se lako i brzo, a prethodno iskustvo u automatskoj obradi nije neophodno.

Njegove osnovne karakteristike su:

- **jednostavnost i jednoobraznost pri korišćenju**. Tabela (relacija) se kreira jednom izvršnom naredbom. Odmah po kreiranju tabela je raspoloživa za korišćenje. Svi podaci memorisani su u tabelama i rezultat bilo koje operacije se logički prikazuje u obliku tabele
- **mogućnost interaktivnog i klasičnog (aplikativnog) programiranja**. Koristeći SQL dobijaju se odgovori na trenutne, unapred ne predviđene zahteve ili se SQL blokovi "ugrađuju" u klasični viši programski jezik (FORTRAN, COBOL, PL/I, C) omogućujući klasičnu obradu
- **neproceduralnost** (tj. proceduralnost u minimalnom stepenu). Ni za jedan jezik se ne može reći da je potpuno neproceduralan, već da je neproceduralan u većem ili manjem stepenu. SQL je u velikoj meri neproceduralan jer definiše ŠTA, a ne KAKO: koji podaci se žele, koje tabele se referenciraju i koji uslovi treba da budu ispunjeni, bez specifikacije procedure za dobijanje željenih podataka.

Funkcija SQL-a je da omogući definisanje, korišćenje i kontrolu podataka relacione baze kroz:

- **Naredbe za definisanje podataka (data definition statements)**: Omogućuju definisanje resursa relacione baze podataka.
- **Naredbe za manipulisanje (rukovanje) podacima (data manipulation statements)**: Omogućuju ažuriranje u širem smislu značenja te reči i izveštavanje iz relacione baze podataka.
- **Naredbe za kontrolne (upravljačke) funkcije (data control functions)**: Omogućuju oporavak, konkurentnost, sigurnost i integritet relacione baze podataka.

# DDL-CREATE TABLE


- 
- CREATE TABLE- ograničenja: PRIMARY KEY, UNIQUE, NULL/NOT NULL*
  - CREATE TABLE-FOREIGN KEY*
  - CREATE TABLE-CHECK*
  - CREATE TABLE-case study*
  - Vežba: Primeri naredbe CREATE TABLE*

02

# ČEMU SLUŽI NAREDBA CREATE TABLE

*Za kreiranje novih tabela, definisanje kolona, ograničenja koja se odnose na kolone i kreiranje relacija*

Korišćenje naredbe CREATE TABLE spada u grupu naredbi za definisanje podataka (DDL).

Naredbe DDL-a:

- CREATE TABLE (kreiranje tabele baze podataka),
- CREATE INDEX (kreiranje indeksa),
- ALTER TABLE (izmena definicije tabele),
- DROP TABLE (izbacivanje tabele iz baze podataka)

omogućuju definisanje resursa relacione baze podataka.

**Naredba CREATE TABLE** se koristi za kreiranje novih tabela, definisanje kolona, ograničenja koja se odnose na kolone i kreiranje relacija. Ova naredba se na isti način može koristiti u svim RDBMS (SQL Server, DB2, Oracle, MySQL), razlike postoje samo u tipovima podataka koji se koriste za opis kolona tabele.

**CREATE TABLE** naredba ima pet tipova ograničenja: PRIMARY KEY, UNIQUE, NULL/NOT NULL, FOREIGN KEY i CHECK.

Svrha prva tri ograničenja je očigledna. FOREIGN KEY se koristi za definisanje ograničenja referencijalnog integriteta dok se CHECK koristi za definisanje ograničenja nad podacima.



CREATE TABLE-primary key, unique,  
null/not null

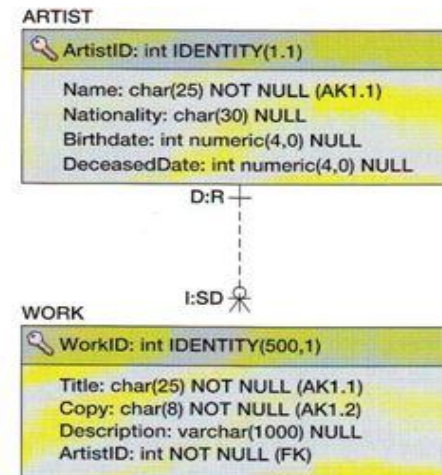

---

02

# PRIMERI TABELA KOJE TREBA KREIRATI

## Tabele ARTIST i WORK

Primer: Na slici 1. su prikazane dve tabele dizajna baze podataka za galeriju slika.



Slika 1. Tabele ARTIST i WORK

Slika prikazuje tipove podataka i null svojstva svih kolona kao i karakteristiku označenu kao IDENTITY koja se koristi za specificiranje surogat ključa.

# KREIRANJE TABELE ARTIST

*CREATE TABLE* sadrži ime tabele koje je praćeno listom definicija svih kolona i njihovih ograničenja

Slika 2. prikazuje kao treba kreirati tabelu ARTIST. Format naredbe CREATE TABLE sadrži ime tabele koje je praćeno listom definicija svih kolona i njihovih ograničenja koja se stavlja u zagradi.

```
CREATE TABLE ARTIST(  
  ArtistID          int          NOT NULL IDENTITY (1, 1)  
  Name              char (25)   NOT NULL,  
  Nationality       char (30)   NULL,  
  Birthdate         numeric (4, 0) NULL,  
  DeceasedDate     numeric (4, 0) NULL,  
  
  CONSTRAINT ArtistPK PRIMARY KEY (ArtistID),  
  CONSTRAINT ArtistAK1 UNIQUE (Name)  
);
```

Slika 2. Naredba CREATE TABLE za kreiranje tabele ARTIST

U prvom delu naredbe CREATE TABLE je svaka kolona definisana svojim imenom, tipom podataka i null statusom. Mada SQL-92 definiše veliki broj tipova podataka, svaki proizvođač DBMS-a koristi svoje sopstvene tipove.

U drugom delu naredbe se navode ograničenja – definisan primarni ključ i kandidat odnosno alternativni ključ. Primarna svrha alternativnog ključa je da se obezbedi jedinstvenost vrednosti neke kolone pa se alternativni ključ definiše ograničenjem UNIQUE.

Format za pisanje ograničenja je: reč CONSTRAINT, ime ograničenja koje daje sam programer, neko od prethodno navedenih ograničenja (u datom primeru PRIMARY KEY i UNIQUE) i jedna ili više kolona koje se stavljaju u zagradi.

# CREATE TABLE-FOREIGN KEY

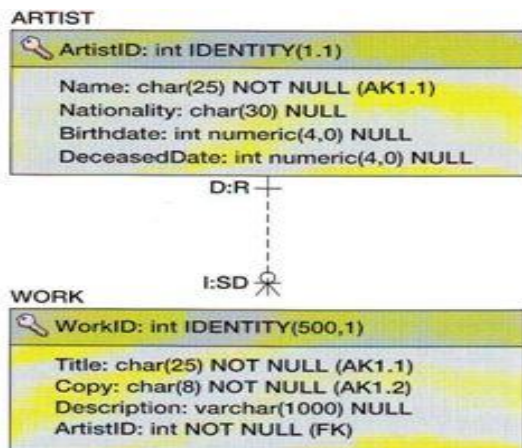

---

02

# KREIRANJE TABELE WORK

*Prikazuje upotrebu ograničenja FOREIGN KEY koje se koristi za definisanje referencijalnog integriteta*

Primer: Na slici 1. su prikazane dve tabelle dizajna baze podataka za galeriju slika.



Slika 1. Tabele ARTIST i WORK

Slika prikazuje tipove podataka i null svojstva svih kolona.

Slika 2. prikazuje kako treba kreirati tabelu WORK koja je u vezi sa tabelom ARTIST.

```
CREATE TABLE WORK (
  WorkID          int          NOT NULL IDENTITY (500, 1),
  Title           char (25)    NOT NULL,
  Copy            char (8)     NOT NULL,
  Description     varchar (1000) NULL,
  ArtistID       int          NOT NULL,
  CONSTRAINT WorkPK PRIMARY KEY (WorkID),
  CONSTRAINT WorkAK1 UNIQUE (Title, Copy),
  CONSTRAINT ArtistFK FOREIGN KEY (ArtistID) REFERENCES ARTIST (ArtistID)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
);
```

Slika 2. SQL naredba za kreiranje tabelle WORK i veze između tabelle WORK i ARTIST

U ovoj SQL naredbi se pojavljuje ograničenje FOREIGN KEY koje se koristi za definisanje referencijalnog integriteta.

Ograničenje FOREIGN KEY na slici ima isto značenje kao i ograničenje referencijalnog integriteta

WORK.ArtistID mora da postoji u ARTIST.ArtistID

# KLAUZULE DELETE (UPDATE) NO ACTION I DELETE (UPDATE) CASCADE UZ FOREIGN KEY

*Označavaju zabranjeno brisanje (ažuriranje) redova koji imaju decu odnosno kaskadno brisanje (ažuriranje)*

Ograničenje FOREIGN KEY sadrži klauzulu kojom se specificira da li je ažuriranje ili brisanje kaskadno.

Izraz DELETE NO ACTION označava da je zabranjeno brisanje redova koji imaju decu. Izraz DELETE CASCADE označava da je brisanje kaskadno. Default vrednost je DELETE NO ACTION.

Slično, izraz UPDATE NO ACTION označava da je zabranjeno ažuriranje primarnog ključa redova tabela koje imaju decu.

Izraz UPDATE CASCADE označava da je ažuriranje kaskadno. Default vrednost je UPDATE NO ACTION.

U prikazanom slučaju je iskorišćena klauzula UPDATE NO ACTION jer je primarni ključ table ARTIST surogat ključ koji se ne može nikada menjati.

Ponekad je pogodno kreirati strani ključ ali ne i specificirati ograničenje koje se na njega odnosi. To su slučajevi kada vrednosti stranog ključa ne mora da se podudaraju sa vrednostima primarnog ključa u redovima table roditelja. Takve relacije, koje se često nazivaju **uslovne relacije** se pojavljuju u aplikacijama koje obrađuju table u kojima neki podaci nedostaju. Na slici 3. su navedene sve tehnike kreiranja stranog ključa korišćenjem ograničenja FOREIGN KEY, NULL/NOT NULL i UNIQUE.

Tip relacije	Ograničenja komande CREATE TABLE
Relacija tipa 1:N, roditelj opcion	Za FOREIGN KEY treba specificirati ograničenje. Foreign key može biti NULL.
Relacija tipa 1:N, roditelj obavezan	Za FOREIGN KEY treba specificirati ograničenje. Foreign key mora biti NOT NULL.
Relacija tipa 1:1, roditelj opcion	Za FOREIGN KEY treba specificirati ograničenje. Za FOREIGN KEY treba specificirati ograničenje UNIQUE. Foreign key može biti NULL.
Relacija tipa 1:1, roditelj obavezan	Za FOREIGN KEY treba specificirati ograničenje. Za FOREIGN KEY treba specificirati ograničenje UNIQUE. Foreign key mora biti NOT NULL.
Uslovna relacija	Za FOREIGN KEY ne treba specificirati ograničenje. Ako je relacija tipa 1:1 treba specificirati ograničenje UNIQUE.

Slika 3. Tip relacije/ Ograničenja komande CREATE TABLE

# CREATE TABLE-case study

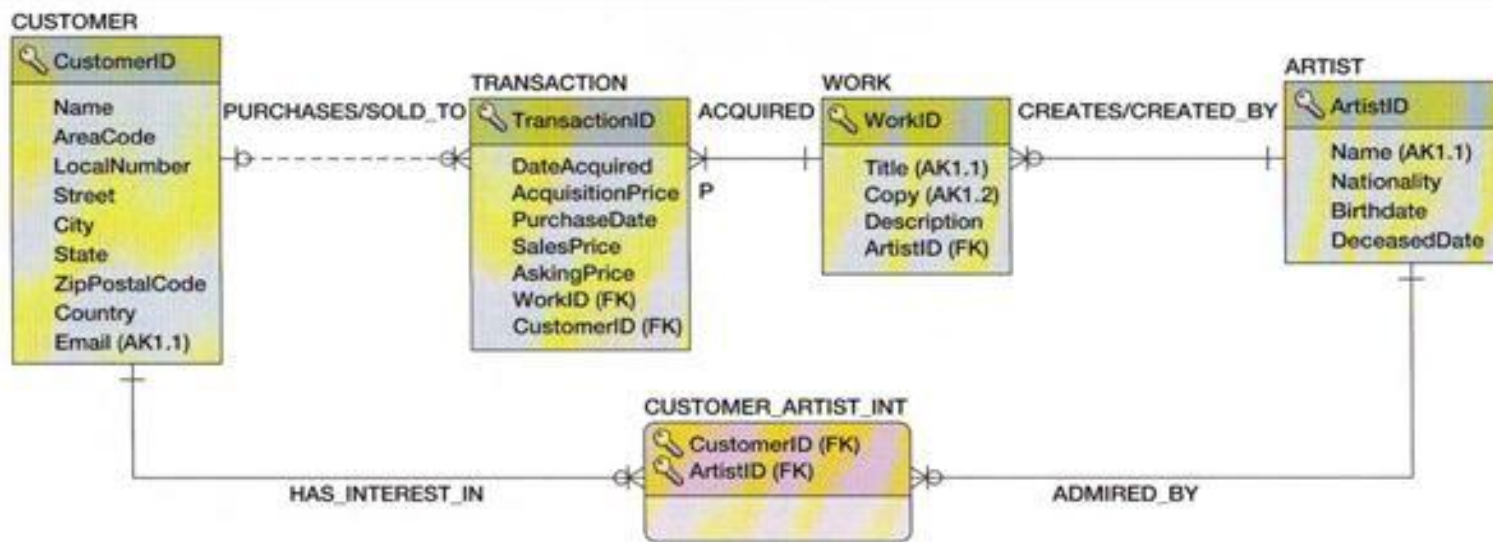

---

02

# DIZAJN BAZE PODATAKA GALERIJE SLIKA

*Logički model ove baze prikazan E-R dijagramom*

Na slici 1. je prikazan logički model ove baze prikazan E-R dijagramom



Slika 1. Dizajn baze podataka galerije slika



# KREIRANJE TABELA ARTIST, WORK I CUSTOMER

*Pored naziva i tipova atributa, CREATE TABLE naredbe sadrže i odgovarajuće CONSTRAINT-e*

Na slici 2. je prikazana CREATE TABLE naredba za tabelu ARTIST, na slici 3. za tabelu WORK a na slici 4. za tabelu CUSTOMER.

```
CREATE TABLE ARTIST(  
  ArtistID      int           NOT NULL IDENTITY (1, 1),  
  Name          char (25)      NOT NULL,  
  Nationality   char (30)      NULL,  
  Birthdate     numeric (4, 0)   NULL,  
  DeceasedDate  numeric (4, 0)   NULL,  
  
  CONSTRAINT ArtistPK PRIMARY KEY (ArtistID),  
  CONSTRAINT ArtistAK1 UNIQUE (Name),
```

Slika 2. CREATE TABLE naredba za tabelu ARTIST

```
CREATE TABLE [WORK] (  
  WorkID      int           NOT NULL IDENTITY (500, 1),  
  Title       char (25)      NOT NULL,  
  Copy        char (8)       NOT NULL,  
  Description  varchar (1000)  NULL DEFAULT 'Unknown provenance',  
  ArtistID    int           NOT NULL,  
  
  CONSTRAINT WorkPK PRIMARY KEY (WorkID),  
  CONSTRAINT WorkAK1 UNIQUE (Title, Copy),  
  
  CONSTRAINT ArtistFK FOREIGN KEY(ArtistID) REFERENCES ARTIST (ArtistID)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION  
);
```

Slika 3. CREATE TABLE naredba za tabelu WORK

```
CREATE TABLE CUSTOMER (  
  
  CustomerID    int           NOT NULL IDENTITY (1000, 1),  
  Name          char (25)      NOT NULL,  
  Street        char (30)      NULL,  
  City          char (35)      NULL,  
  State         char (2)       NULL,  
  ZipPostalCode char (9)       NULL,  
  Country       varchar (50)   NULL,  
  AreaCode      char (3)       NULL,  
  PhoneNumber   char (8)       NULL,  
  Email         char (100)     NULL,  
  
  CONSTRAINT CustomerPK PRIMARY KEY (CustomerID),  
  CONSTRAINT EmailAK1 UNIQUE (Email)  
);
```

Slika 4. CREATE TABLE naredba za tabelu CUSTOMER

# KREIRANJE TABELA TRANSACTION I CUSTOMER\_ARTIST\_INT

*Pored naziva i tipova atributa, CREATE TABLE naredbe sadrže i odgovarajuće CONSTRAINT-e*

Na slici 5. je prikazana CREATE TABLE naredba za tabelu TRANSACTION a na slici 6. za tabelu CUSTOMER\_ARTIST\_INT

```
CREATE TABLE [TRANSACTION] (  
  
    TransactionID      int          NOT NULL IDENTITY (100, 1) ,  
    DateAquired        datetime     NOT NULL,  
    AcquisitionPrice    numeric (8, 2) NOT NULL,  
    PurchaseDate        datetime     NULL,  
    SalesPrice          numeric (8, 2) NULL,  
    AskingPrice         numeric (8, 2) NULL,  
    CustomerID         int          NULL,  
    WorkID             int          NOT NULL,  
  
    CONSTRAINT          TransactionPK PRIMARY KEY (TransactionID) ,  
  
    CONSTRAINT TransactionWorkFK FOREIGN KEY (WorkID) REFERENCES WORK (W  
        ON UPDATE NO ACTION  
        ON DELETE NO ACTION,  
  
    CONSTRAINT TransactionCustomerFK  
        FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (CustomerID)  
        ON UPDATE NO ACTION  
        ON DELETE NO ACTION,
```

Slika 5. CREATE TABLE naredba za tabelu TRANSACTION

```
CREATE TABLE CUSTOMER_ARTIST_INT (  
  
    ArtistID           int          NOT NULL,  
    CustomerID         int          NOT NULL,  
  
    CONSTRAINT CustomerArtistPK PRIMARY KEY (ArtistID, CustomerID) ,  
  
    CONSTRAINT Customer_Artist_Int_ArtistFK  
        FOREIGN KEY (ArtistID) REFERENCES ARTIST (ArtistID)  
        ON UPDATE NO ACTION  
        ON DELETE CASCADE,  
    CONSTRAINT Customer_Artist_Int_CustomerFK  
        FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (CustomerID)  
        ON UPDATE NO ACTION  
        ON DELETE CASCADE
```

Slika 6. CREATE TABLE naredba za tabelu CUSTOMER\_ARTIST\_INT

# Vežba: Primeri naredbe CREATE TABLE


---

02

# PRIMER 1. KREIRANJE TABELE IMENIK

*Prilikom kreiranja tabele Imenik se koristi samo ograničenje za PRIMARY KEY*

Centralni objekat svake relacione baze podataka jeste tabela. Za kreiranje tabele koristi se SQL naredba CREATE TABLE.

Osnovni format CREATE TABLE:

**CREATE TABLE <ime\_tabele>**

**(<lista\_deklaracija\_kolona>**

**[,<lista\_deklaracija\_ograničenja\_tabele>]);**

:

Kreiranje tabele Imenik:

```
CREATE TABLE Imenik(
```

```
  ID                int          NOT NULL IDENTITY(1,1)
```

```
  BROJ_TELEFONA    char(30)
```

```
  EMAIL            char(50)     NULL,
```

```
  IME              char(30)     NULL,
```

```
  PREZIME          char(30)     NULL,
```

```
  GRAD             char(40)     NULL,
```

```
  CONSTRAINT ImenikPK PRIMARY KEY (ID),
```

```
);
```

S obzirom da je kreirana tabela prazna, tj nema inicijalne podatke još uvek, ona će izgledati kao na slici 1.

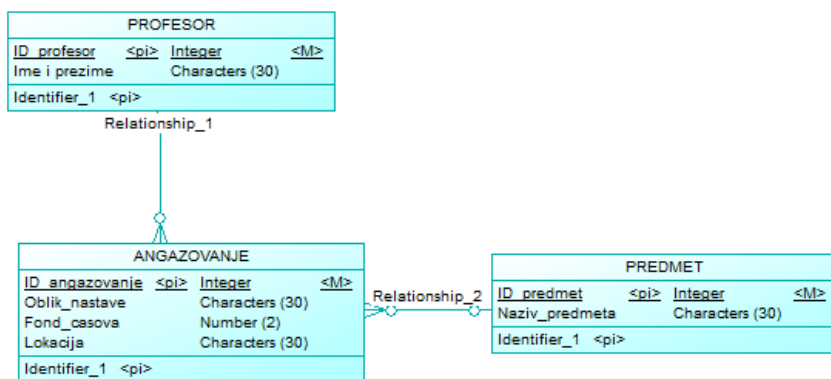
ID	EMAIL	BROJ_TELEFON	IME	PREZIME	GRAD
		A			

Slika 1. Sadržaj kreirane tabele Imenik

# PRIMER 2. KREIRANJA NAREDBE CREATE TABLE

## Na primeru modela za angažovanje nastavnika

Za dijagram predstavljen na slici 2. napiši naredbe za kreiranje tabela



Slika 2. Dijagram za kreiranje tabela

```
CREATE TABLE PROFESOR (  
ID_profesor int NOT NULL IDENTITY (1, 10000)  
Ime i prezime char (30)  
CONSTRAINT ProfesorPK PRIMARY KEY (ID_profesor),  
CONSTRAINT ProfesorUN UNIQUE (Ime i prezime);
```

```
CREATE TABLE PREDMET (  
ID_predmet int NOT NULL IDENTITY (1, 10000)  
Naziv_predmeta char (30)  
CONSTRAINT PredmetPK PRIMARY KEY (ID_prdmet),  
CONSTRAINT PredmetUN UNIQUE (Naziv_predmeta);
```

```
CREATE TABLE ANGAZOVANJE(  
ID_angazovanje int NOT NULL IDENTITY (1, 10000)  
Oblik_nastave char (30)  
Fond_casova number (2)  
Lokacija char (30)  
CONSTRAINT AngazovanjePK PRIMARY KEY  
(ID_angazovanje),  
CONSTRAINT ProfesorFK FOREIGN KEY  
REFERENCES (ID_profesor) ON DELETE  
NO ACTION, ON UPDATE NO ACTION,  
CONSTRAINT PredmetFK FOREIGN KEY  
REFERENCES (ID_predmet) ON DELETE  
NO ACTION, ON UPDATE NO ACTION;
```

# DDL-CREATE INDEX


---

03

# ČEMU SLUŽE INDEKSI?

*Indeksi se kreiraju kao bi se naglasila jedinstvenost kolona, olakšalo sortiranje i kako bi se omogućilo brže pretraživanje po vrednostima nekih kolona*

Indeksi se kreiraju kao bi se naglasila jedinstvenost kolona, olakšalo sortiranje i kako bi se omogućilo brže pretraživanje po vrednostima nekih kolona. Kolone koje se često koriste u WHERE klauzuli predstavljaju dobre kandidate za indekse. One se mogu koristiti kao jednostavni uslovi u WHERE klauzuli ili se mogu pojaviti u join-u. Oba slučaja su prikazana u sledeća dva primera:

```
SELECT *  
FROM TABELA  
WHERE Kolona1 = 100;  
  
i  
SELECT *  
FROM TABELA1, TABELA2  
WHERE TABELA1.Kolona1 = TABELA2.Kolona2;
```

Ako se naredbe kao što su ove često izvršavaju, Kolona1 i Kolona2 su dobri kandidati za indeks. Sledećom naredbom se može kreirati indeks nad kolonom Name u tabeli CUSTOMER.

```
CREATE INDEX Kupaclmidx ON KUPAC (ime);
```

Indeks je nazvan Kupaclmidx. Ime indeksa daje programer i ono nema značaja za RDBMS. Da bi se kreirao jedinstveni indeks, pre ključne reči INDEX treba dodati ključnu reč UNIQUE

*Na primer: da bi se obezbedilo da se jedean posao ne doda dva puta u tabelu POSAO, treba kreirati jedinstveni indeks nad poljima tabele (Naziv, Kopija, UmetnikID) na sledeći način:*

```
CREATE UNIQUE INDEX JedinstveniPosaoIndex ON POSAO  
(Naziv, Kopija, UmetnikID);
```

# DDL-ALTER TABLE


---

04



# KADA SE KORISTI NAREDBA ALTER TABLE ?

## *Ukoliko želimo da promenimo strukturu postojeće tabele*

Koristi se ukoliko želimo da promenimo strukturu postojeće tabele. Može se koristiti za dodavanje, izbacivanje ili menjanje kolona kao i za dodavanje i izbacivanje ograničenja nad tabelama.

*Primer: Ukoliko želimo da postojećoj tabeli KUPAC dodamo novu kolonu, naredbe je:*

```
ALTER TABLE KUPAC ADD NovaKolona Char(5) NULL;
```

Dodavanje nove kolone sa NOT NULL opcijom postojećoj tabeli koja nije prazna nije moguće direktno izvršiti. Najpre treba naredbom ALTER TABLE ... ADD ... dodati novu kolonu bez klauzule NOT NULL. Zatim treba dodati vrednosti za novu kolonu za sve n-torke. Na kraju, naredbom ALTER TABLE ... MODIFY ... treba dodati klauzulu NOT NULL prethodno definisanoj koloni.

*Primer: Ukoliko hoćemo da izmenimo definiciju postojeće kolone tabela (dužinu, tip, not null), naredbe je:*

```
ALTER TABLE KUPAC ALTER COLUMN StaraKolona date NOT NULL;
```

*Primer: Ukoliko hoćemo da izbrišemo postojeću kolonu tabele, naredbe je:*

```
ALTER TABLE KUPAC DROP COLUMN StaraKolona;
```

*Primer: Ukoliko želimo da dodamo ili obrišemo ograničenja:*

```
ALTER TABLE KUPAC ADD CONSTRAINT NovoOgraničenje  
CHECK ([Ime] NOT IN ('Robert no pay'));
```

```
ALTER TABLE KUPAC DROP CONSTRAINT  
StarOgraničenje;
```

# DDL-DROP TABLE


---

*Vežba: Primeri ALTER TABLE i DROP TABLE naredbe*

05

# KADA SE KORISTI NAREDBA DROP TABLE?

*Ukoliko hoćemo da izbacimo definiciju tabele, zajedno sa podacima koje sadrži*

Ukoliko hoćemo da izbacimo definiciju tabele, zajedno sa podacima koje sadrži, iz baze podataka koristi se DROP TABLE naredba. Opšti oblik naredbe je:

## **DROP TABLE tabela**

*Primer: Izbaciti definiciju tabele PREMIJA iz baze podataka.*

**DROP TABLE PREMIJA;**

Obratiti pažnju na različito dejstvo naredbi DROP TABLE i DELETE. Naredbom DELETE se može brisati kompletan sadržaj tabele, ali sama tabela ostaje u bazi podataka i sa njom se može dalje raditi. Naredbom DROP TABLE tabela sa sadržajem se izbacuje iz baze podataka i više nije dostupna.

Ukoliko, pri korišćenju konkretnog sistema za upravljanje bazom podataka, hoćemo da izbrišemo kompletan sadržaj tabele, često se to brže postiže uzastopnom primenom naredbi DROP TABLE i CREATE TABLE, nego primenom naredbe DELETE.

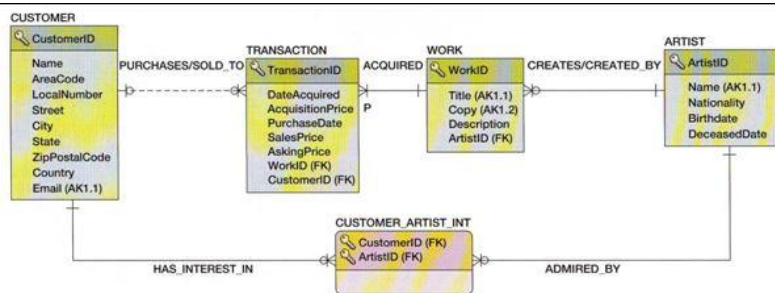
DBMS neće dozvoliti brisanje tabele koja je roditelj i ima ograničenje FOREIGN KEY. Brisanje se može izvršiti samo ako ona nema decu ili ako je naznačeno kaskadno brisanje (DELETE CASCADE). Takođe, prilikom brisanja takve tabele se može najpre izbrisati ograničenje FOREIGN KEY ili tabela koja predstavlja dete date tabele.

# NAREDBE DROP TABLE I ALTER TABLE

## Na primeru baze podataka za galeriju slika

Primer: Na slici 1. je prikazan je dizajn baze podataka za jednu galeriju slika.

Dva moguća načina za brisanje tabele CUSTOMER iz baze podataka galerije slika su prikazana na slici 2. i slici 3. Važno je naglasiti da se brisanje može izvršiti samo ako ona tabela nema decu



Slika 1. Dizajn baze podataka za galeriju slika

```
DROP TABLE CUSTOMER_ARTIST_INT;  
DROP TABLE TRANSACTION;  
DROP TABLE CUSTOMER;
```

Slika 2. Brisanje tabele CUSTOMER tako što su prethodno izbrisane tabele CUSTOMER\_ARTIST\_INT i TRANSACTION

```
ALTER TABLE CUSTOMER_ARTIST_INT  
DROP CONSTRAINT  
Customer_Artist_Int_CustomerFK;  
ALTER TABLE TRANSACTION  
DROP CONSTRAINT TransactionCustomerFK;  
DROP TABLE CUSTOMER;
```

Slika 3. Brisanje tabele CUSTOMER tako što su prethodno izbrisani strani ključevi u CUSTOMER\_ARTIST\_INT i TRANSACTION i

# Vežba: Primeri ALTER TABLE i DROP TABLE


---

05

# 1. PRIMER

## *ALTER TABLE i DROP TABLE nad tabelom Imenik*

ALTER TABLE služi za izmenu definicije tabele. DROP TABLE služi za izbacivanje tabele iz baze podataka.

Recimo da imamo tabelu Imenik sa kolonama (ID, EMAIL, BROJ\_TELEFONA, IME, PREZIME)-slika 1.

ID	EMAIL	BROJ_TELEFON A	IME	PREZIME
----	-------	-------------------	-----	---------

Slika 1. Prikaz kolona tabele Imenik

Tabeli želimo da dodamo kolonu GRAD. Kod za dodavanje kolone će glasiti:

**ALTER TABLE Imenik ADD GRAD Char(20) NULL;**

Na prikazanu tabelu Imenik dodaće se kolona GRAD čiji je tip podataka Char gde može da se unesu 20 znakova (slika 2.).

ID	EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
----	-------	-------------------	-----	---------	------

Slika 2. Prikaz kolona tabele Imenik nakon ALTER TABLE naredbe

Ukoliko želimo da obrišemo tabelu iz baze:

**DROP TABLE Imenik;**

Ukoliko obrisemo DROP tabela se briše iz baze, ukoliko DELETE koristimo brišemo podatke iz tabele (inicijalne podatke).

Ukoliko želimo da izbrišemo neku određenu kolonu iz tabele:

**ALTER TABLE Imenik DROP COLUMN GRAD;**

Samim tim izbrisaće se kolona GRAD.

IT2008-IM-DQL-SQDD-DDL-V-PrimeriALTER-DROP-Slika1

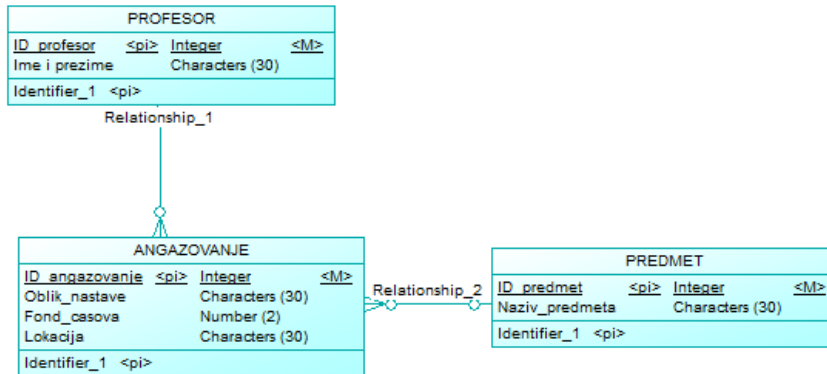
## 2. PRIMER

### *ALTER TABLE I DROP TABLE nad tabelama PROFESOR i PREDMET*

Za model prikazan na slici 3.

- Izbriši tabelu PROFESOR
- Izbriši tabelu PREDMET

na dva moguća načina



Sika 3. Model za brisanje tabela

a.

I način:

```
DROP TABLE ANGAZOVANJE;
DROP TABLE PROFESOR
```

II način:

```
ALTER TABLE ANGAZOVANJE DROP CONSTRAINT
ProfesorFK FOREIGN KEY;
DROP TABLE PROFESOR;
```

b.

I način:

```
DROP TABLE ANGAZOVANJE;
DROP TABLE PREDMET
```

II način:

```
ALTER TABLE ANGAZOVANJE DROP CONSTRAINT
PredmetFK FOREIGN KEY;
DROP TABLE PREDMET;
```

# DML-INSERT


---

□ *Vežba: Primeri INSERT naredbe*

06



# ČEMU SLUŽI NAREDBA INSERT?

*Za dodavanje redova tabele; Naredba INSERT ima svoju standardnu verziju ali i različite opcije*

Naredba INSERT se koristi za dodavanje redova tabele.

Standardna verzija naredbe INSERT je ime tabele, imena kolona u koje se unose podaci i lista podataka u sledećem formatu:

```
INSERT INTO TABELA_UMETNIK (Ime, Nacionalnost,  
Datum_rodjenja, Datum_smrti) VALUES
```

```
('Tomy', 'Meksikanac', '1927', '1998');
```

Treba napomenuti da se i lista naziva kolona i lista podataka nalaze unutar zagrada.

INSERT naredba može imati različite opcije od kojih će se razmotriti sledeći slučajevi:

Ubacivanje vrednosti SVIH kolona za slučaj kada su podaci u istom redosledu kao i kolone u tabeli a pri tom ne postoji surogat ključ.

U ovom slučaju nije potrebno specificirati nazive atributa, pa insert naredba ima oblik:

```
INSERT INTO naziv_tabele
```

```
VALUES (vrednost_1, vrednost_2, ...);
```

Za svaki atribut koji je specificiran kao NOT NULL, MORA postojati vrednost.

*Primer: Ubaciti podatke o Bojanu, pravniku, koji počinje da radi u RJ 10, 7.04.1984 sa platom od 450000 i bez prava na premiju. Neposredni rukovodilac još nije poznat.*

```
INSERT INTO RADNIK VALUES
```

```
(3545, 'BOJAN', 'PRAVNIK', NULL, '07-APR-84', 450000,  
NULL, 10);
```

# RAZLIČITE OPCIJE NAREDBE INSERT

*Za ubacivanje vrednosti kolona kada redosled podataka i kolona nije isti ili za ubacivanje podataka iz jedne u drugu tabelu*

Ubacivanje vrednosti NEKIH kolona ili svih kolona ali redosled podataka i kolona u tabeli nije isti.

U ovom slučaju nazivi tih kolona se moraju eksplicitno navesti pa naredba insert ima oblik:

```
INSERT INTO naziv_tabele (kolona1, kolona2,...)
VALUES (vrednost_1, vrednost_2,...);
```

Vrednosti za NOT NULL attribute moraju biti unete.

*Primer: Uneti podatke o Srđanu, planeru, koji je primljen u RJ 20.*

```
INSERT INTO RADNIK (IME, RADNIK, S_RJ, POSAO)
VALUES ('SRDJAN', 3562, 20, 'PLANER');
```

Ubacivanje podataka iz jedne tabele u drugu. Ukoliko obe tabele imaju isti broj atributa i ukoliko su atributi identično definisani, naredba INSERT je oblika:

```
INSERT INTO tabela1 SELECT * FROM tabela2;
```

inače:

```
INSERT INTO tabela1 (kolona1, kolona2,...)
SELECT atribut, izraz
FROM tabela2
WHERE kriterijum selekcije;
```

*Primer: Dati svim analitičarima i savetnicima premiju u iznosu od 10% njihovog ličnog dohotka. Te informacije uneti u tabelu premija zajedno sa datumom zaposlenja.*

```
INSERT INTO PREMIJA (RADNIK, PREMIJA, POSAO,
DATZAP)
SELECT RADNIK, LD * .10, POSAO, DATZAP
FROM RADNIK
WHERE POSAO IN ('ANALITICAR', 'SAVETNIK');
```

# Vežba: Primeri INSERT naredbe


---

06

# FORMAT NAREDBE INSERT

*Za slučaj kada treba uneti više zapisa u tabelu i kada treba dodati samo jedan zapis*

Naredba INSERT INTO dodaje zapise u tabelu. Poznatiji je kao upit za dodavanje.

Ukoliko postoje više zapisa naredba INSERT izgleda:

```
INSERT INTO cilj [(prvoPolje [, drugoPolje[, ...]])] [IN  
spoljnaBazaPodataka]
```

```
SELECT [izvor.]prvoPolje[, drugoPolje[, ...]]
```

```
FROM izrazTabele
```

Upit za dodavanje sa jednim zapismo izgleda:

```
INSERT INTO cilj [(prvoPolje[, drugoPolje[, ...]])]
```

```
VALUES (prvaVrednost[, drugaVrednost[, ...]])
```

Znači naredba INSERT INTO se sastoji iz:

- cilj - je ime tabele ili upita u koji treba da se dodaju zapisi;
- prvoPolje, drugoPolje - polja koja treba da se dodaju podaci ako slede nakon argumenta ili imena polja iz kojih treba da se preuzmu podaci
- spoljnaBazaPodataka - putanja do spoljne baze podataka
- spoljnaBazaPodataka - putanja do spoljne baze podataka
- izvor - ime tabele ili upita iz kog treba da se kopiraju zapisi
- izrazTabele - ime tabele ili grupe iz koje se umeću zapisi
- prvaVrednost, drugaVrednost- vrednosti koje treba da se umetnu u određena polja u zapisu.

# PRIMER 1. NAREDBE INSERT

## Naredbe INSERT nad tabelom Imenik

Primer:

```
INSERT INTO Imenik
```

```
VALUES ('milos@pera.com', '064 98 39 354', 'Milos', 'Misić,', 'Niš');
```

U tabeli Imenik su ubačene navedene vrednosti i više tabela nije prazna, i nema samo imena kolona (slika 1.).

EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
milos@pera.com	064 98 39 354	Milos	Misić	Niš

Slika 1. Sadržaj tabelle Imenik nakon naredbe INSERT

Primer: Drugi način unošenja podataka je da se navedu kolone u kojima ce se unositi vrednosti određene tabelle za isti primer upit bi glasilo ovako:

```
INSERT INTO Imenik (EMAIL,  
BROJ_TELEFONA,IME,PREZIME,GRAD)
```

```
VALUES ('milos@pera.com', '064 98 39 354', 'Milos', 'Misić,', 'Niš');
```

Ukoliko se navode imena kolone, vrednosti će automatski biti redom unešene, ali ukoliko se unesu imena kolone mora da se unese sa istim redosledom. Ukoliko ne navedemo neku kolonu, vrednost za tu kolonu ce biti jednostavno izostavljena i neće se pojaviti kada se SELECT naredbom pozove, tacnije bice prazna.

Primer:Necemo da unesemo mail,

```
INSERT INTO Imenik
```

```
(BROJ_TELEFONA,IME,PREZIME,GRAD)
```

```
VALUES ('064 98 39 354', 'Milos', 'Misić,', 'Niš');
```

Kada SELECT komandom pozovemo prikaz izgledaće kao na slici 2.

EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
	064 98 39 354	Milos	Misić	Niš

Slika 2. Prikaz unešenih podataka u tabelu Imenik

## PRIMER 2. NAREDBE INSERT

### *Naredbe INSERT nad tabelom Imenik i Radnik*

Pogrešno je unositi ukoliko ste naveli određene kolone a niste naveli podatke za tu kolonu:

```
INSERT INTO Imenik (EMAIL,  
BROJ_TELEFONA,IME,PREZIME,GRAD)  
VALUES ('milos@pera.com', '064 98 39 354', 'Milos', 'Misić',  
'Beograd');
```

Naveden primer nema za kolonu GRAD nikakve podatke i na monitoru će se pojaviti error poruka.

Još jedan primer gde se ubaciju vrednosti milos@pera.com i 064 98 39 354 u novi red tabele.

```
INSERT INTO IMENIK (EMAIL, BROJ_TELEFONA)  
VALUES ('milos@pera.com', '064 98 39 354', NULL, NULL,  
NULL);
```

Još jedan primer je:

Email, broj telefona, ime, prezime i grad iz tabele Imenik prepisati u tabelu RADNIK za sve stanovnike Beograda pri čemu tabela RADNIK ima kolone: **IME,PREZIME, EMAIL, BROJ\_TELEFONA, GRAD**

```
INSERT INTO RADNIK (IME,PREZIME, EMAIL,  
BROJ_TELEFONA, GRAD) SELECT ime, prezime, email,  
broj_telefona, grad FROM IMENIK where grad = 'Beograd';
```

# DML-UPDATE


---

*Vežba: Primeri UPDATE naredbe*

07

# ČEMU SLUŽI NAREDBA UPDATE?

## Za menjanje sadržaja tabele

Opšti oblik naredbe je:

**UPDATE tabela**

**SET kolona1=izraz1 [,kolona2=izraz2]**

**[WHERE kriterijum selekcije];**

odnosno:

**UPDATE tabela**

**SET(kolona1, kolona2, ...) = (podupit)**

**[WHERE kriterijum selekcije]**

Podupit mora vratiti najviše po jednu vrednost za svaki od kolona u listi kolona iza SET klauzule.

*Primer: Novo primljeni pravnik Bojan imaće predsednika za neposrednog rukovodioca. Realizovati tu odluku.*

**UPDATE RADNIK**

**SET RUKOV=3539**

**WHERE IME='BOJAN';**

*Primer: Srđan počinje sa radom 19.09.84 sa ličnim dohotkom od 350000 i ima neposrednog rukovodioca sa šifrom 3266.*

*Realizovati te informacije.*

**UPDATE RADNIK**

**SET RUKOV=3266, DATZAP='19-SEP-84', LD=350000**

**WHERE IME = 'SRDJAN';**

*Primer: Dati svim analitičarima i savetnicima u RJ 20 15% povišicu na lični dohodak.*

**UPDATE RADNIK**

**SET LD = LD\*1.15**

**WHERE POSAO IN ('ANALITICAR', 'SAVETNIK')**

**AND S\_RJ = 20;**



# DRUGI NAČIN UPOTREBE NAREDBE UPDATE

*Komandom UPDATE se vrednost kolone može postaviti tako da bude jednaka vrednosti kolone neke druge tabele*

SQL komandom UPDATE se može vrednost kolone postaviti tako da bude jednaka vrednosti kolone neke druge tabele

*Primer: Pretpostavimo da postoji tabela TAX\_TABLE (Tax, City) gde je Tax odgovarajuća vrednost poreza na nivou jednog grada (City). Takodje pretpostavimo da imamo tabelu PURCHASE\_ORDER koja uključuje kolone TaxRate i City. Mi sve kolone iz porudžbina (PURCHASE\_ORDER) u jednom gradu, na primer Beogradu, možemo ažurirati sledećom SQL rečenicom:*

```
UPDATE PURCHASE_ORDER
SET Tax_Rate =
(SELECT Tax from TAX_TABLE WHERE TAX_TABLE.City =
'Beograd')
WHERE PURCHASE_ORDER.City = 'Beograd';
```

*Primer: slučaj kada želimo da ažuriramo vrednost takse za narudžbine a da pri tom ne specificiramo grad. Na primer želimo da ažuriramo TaxRate za narudžbinu 1000. U tom slučaju se može koristiti sledeća SQL rečenica:*

```
UPDATE PURCHASE_ORDER
SET Tax_Rate =
(SELECT Tax from TAX_TABLE
WHERE TAX_TABLE.City = PURCHASE_ORDER.City )
WHERE PURCHASE_ORDER.Number = 1000;
```

# Vežba: Primeri UPDATE naredbe


---

07

# PRIMER 1. NAREDBE UPDATE

## *Naredba UPDATE nad tabelom Imenik*

Upit koji služi za menjanje vrednosti u poljima u određenoj tabeli na osnovu navedenih kriterijuma. Sinaksa za UPDATE naredbu glasi:

**UPDATE imeTabele**

**SET novaVrednost**

**WHERE kriterijumi;**

Delovi UPDATE naredbe su:

- imeTabele - tabela koja sadrži podatke koji se menjaju
- novaVrednost - određuje vrednost koja treba da se unese u određeno polje u ažuriranim zapisima
- kriterijumi - određuje koji zapisi će biti ažurirani.

Ova naredba se koristi kada se menja mnogo zapisa ili kada se zapisi koje želimo da promenimo nalaz u više tabela.

Ukoliko uzmemo primer table Imenik sa slike 1. i primenimo naredbu

**UPDATE Imenik set IME='Petrija'**

**WHERE ID=1;**

ID	EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
1	<a href="mailto:milos@pera.com">milos@pera.com</a>	064 98 39 354	Milos	Misić	Niš
2	<a href="mailto:krompirovazlatica@gmail.com">krompirovazlatica@gmail.com</a>	063 55 66 455	Marko	Nakić	Kruševac
3	<a href="mailto:asdf@mail.com">asdf@mail.com</a>	061 45 45 666	Milić	Perić	Kragujevac
4	<a href="mailto:mail.mail@mail.com">mail.mail@mail.com</a>	065 789 54 66	Milenko	Popović	Niš
5	<a href="mailto:mislisa@yahoo.com">mislisa@yahoo.com</a>	067 752 33 55	Milica	Martić	Beograd

Slika 1. Prikaz table Imenik pre UPDATE-a

Promena podatka u koloni IME gde je ID 1. Nakon UPDATE-a tabela će izgledati kao na slici 2.

ID	EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
1	<a href="mailto:milos@pera.com">milos@pera.com</a>	064 98 39 354	Petrija	Misić	Niš
2	<a href="mailto:krompirovazlatica@gmail.com">krompirovazlatica@gmail.com</a>	063 55 66 455	Marko	Nakić	Kruševac
3	<a href="mailto:asdf@mail.com">asdf@mail.com</a>	061 45 45 666	Milić	Perić	Kragujevac
4	<a href="mailto:mail.mail@mail.com">mail.mail@mail.com</a>	065 789 54 66	Milenko	Popović	Niš
5	<a href="mailto:mislisa@yahoo.com">mislisa@yahoo.com</a>	067 752 33 55	Milica	Martić	Beograd

Slika 2. Prikaz table Imenik posle UPDATE

# PRIMER 2. NAREDBA UPDATE SA NAREDBOM SELECT

## *Nad tabelama CUSTOMER i SUPPLIERS*

### 1. Ažuriranje tabele sa podacima iz druge tabele:

```
UPDATE customers SET c_details = (SELECT contract_date  
FROM suppliers WHERE suppliers.supplier_name =  
customers.customer_name) WHERE customer_id < 1000;
```

### 2. Ažuriranje više tabela:

```
UPDATE suppliers, contacts  
SET suppliers.status = 'Active', contacts.note = 'Also Supplier'  
WHERE suppliers.supplier_id = contacts.contact_id
```

### 3. Korišćenje klauzule exist

```
UPDATE suppliers SET supplier_name = (SELECT  
customers.customer_name FROM customers WHERE  
customers.customer_id = suppliers.supplier_id) WHERE EXISTS  
(SELECT customers.customer_name FROM customers WHERE  
customers.customer_id = suppliers.supplier_id);
```

### 4. Predhodna naredba se može napisati i na drugi način:

```
UPDATE suppliers, customers SET suppliers.supplier_name =  
customers.customer_name WHERE suppliers.supplier_id =  
customers.customer_id;
```

# PRAKTIČNA VEŽBA 1:

## *Primeri naredbi CREATE, INSERT i UPDATE*

Na osnovu tabele Suppliers koja sadrži sledeće podatke, ažurirajte city sa "Santa Clara" za sve slogove kod kojih je supplier\_name "NVIDIA".

```
UPDATE suppliers SET city = 'Santa Clara' WHERE  
supplier_name = 'NVIDIA';
```

```
CREATE TABLE suppliers ( supplier_id number(10) not null,  
supplier_name varchar2(50) not null, city varchar2(50),  
CONSTRAINT suppliers_pk PRIMARY KEY (supplier_id) );
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5001, 'Microsoft', 'New York');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5002, 'IBM', 'Chicago');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5003, 'Red Hat', 'Detroit');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5004, 'NVIDIA', 'New York');
```

# PRAKTIČNA VEŽBA 2:

## Primeri naredbi *CREATE*, *INSERT* i *UPDATE*

Na osnovu tabela *suppliers* i *customers* koje sadrže sledeće podatke, ažuriraj *city* u tabeli *suppliers* sa *city* u tabeli *customers* kada se *supplier\_name* u tabeli *suppliers* podudara sa *customer\_name* u tabeli *customers*.

```
CREATE TABLE suppliers ( supplier_id number(10) not null,  
supplier_name varchar2(50) not null, city varchar2(50),  
CONSTRAINT suppliers_pk PRIMARY KEY (supplier_id) );
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5001, 'Microsoft', 'New York');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5002, 'IBM', 'Chicago');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5003, 'Red Hat', 'Detroit');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)  
VALUES (5005, 'NVIDIA', 'LA');
```

```
CREATE TABLE customers ( customer_id number(10) not null,  
customer_name varchar2(50) not null, city varchar2(50),  
CONSTRAINT customers_pk PRIMARY KEY (customer_id) );
```

```
INSERT INTO customers (customer_id, customer_name, city)  
VALUES (7001, 'Microsoft', 'San Francisco');
```

```
INSERT INTO customers (customer_id, customer_name, city)  
VALUES (7002, 'IBM', 'Toronto');
```

```
INSERT INTO customers (customer_id, customer_name, city)  
VALUES (7003, 'Red Hat', 'Newark');
```

```
UPDATE suppliers SET city =
```

```
(SELECT customers.city FROM customers  
WHERE customers.customer_name = suppliers.supplier_name)  
WHERE EXISTS (SELECT customers.city FROM customers  
WHERE customers.customer_name = suppliers.supplier_name);
```

# Naredba DELETE


---

08

# ČEMU SLUŽI NAREDBA DELETE

## Za brisanje sadržaja dela ili cele tabele

Upit koji služi za uklanjanje zapisa iz tabela navedenih u odredbi FROM koji zadovoljavaju odredbu WHERE.

**DELETE**

**FROM** tabela

**WHERE** kriterijumi

**Tabela** je ime tabele iz koje se bršu zapisi a **kriterijumi** je izraz koji određuje koji zapisi treba da se obrišu.

DELETE se koristi ako se potencira na brisanju više zapisa. Ukoliko želimo da izbrišemo celu tabelu koristi se naredba DROP. Međutim ako se izbriše tabela, izbrisće se struktura. DELETE briše samo podatke. Struktura, atributi, indeksi ostaće netaknuti.

Uzmimo na primer tabelu IMENIK (slika 1.). Izvršimo upit

**DELETE FROM Imenik WHERE ID=5;**

Nakon toga se dobija sadržaj tabele kao na slici 2.

ID	EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
1	milos@pera.com	064 98 39 354	Milos	Misić	Niš
2	krompirovazlatica@g mail.com	063 55 66 455	Marko	Nakić	Kruševac
3	asdf@mail.com	061 45 45 666	Milić	Perić	Kragujevac
4	mail.mail@mail.com	065 789 54 66	Milenko	Popović	Niš
5	mislisa@yahoo.com	067 752 33 55	Milica	Martić	Beograd

Slika 1. Izgled tabele Imenik pre DELETE upita

Rezultat je prikazan na slici 2. Obrisan je ceo red.

ID	EMAIL	BROJ_TELEFON A	IME	PREZIME	GRAD
1	milos@pera.com	064 98 39 354	Petrija	Misić	Niš
2	krompirovazlatica@g mail.com	063 55 66 455	Marko	Nakić	Kruševac
3	asdf@mail.com	061 45 45 666	Milić	Perić	Kragujevac
4	mail.mail@mail.com	065 789 54 66	Milenko	Popović	Niš
5	mislisa@yahoo.com	067 752 33 55	Milica	Martić	Beograd

Slika 2. Prikaz tabele Imenik posle DELETE



# DML-SELECT-osnovni oblik


---

□ *DML-Primena naredbe SELECT*

□ *Vežba: Primeri bazicnih oblika SELECT naredbe*

09

# ČEMU SLUŽI NAREDBA SELECT?

*Za izveštavanje iz relacije baze podataka.*

Naredba SELECT spada u grupu naredbi za manipulisanje podacima (data manipulation statements-DML). Naredbe DML-a (SELECT, INSERT, UPDATE, DELETE) omogućavaju ažuriranje u širem smislu značenja te reči i izveštavanje iz relacije baze podataka.

Naredba SELECT služi za izveštavanje iz baze podataka.

Osnovni oblik naredbe SELECT je:

```
SELECT    < lista atributa >
FROM      < lista relacija >
WHERE     < kvalifikacioni izraz >;
```

Listom atributa zadaje se operacija PROJEKCIJE.

Kvalifikacionim izrazom zadaju se uslovi RESTRIKCIJE (SELEKCIJE) i SPAJANJA, odnosno iskazi slični iskazima u relacionom računu.

Klauzule SELECT i FROM su obavezne, dok klauzula WHERE nije.

# DML-Primena naredbe SELECT


---

09

# TABELE ZA PRIKAZ RAZLIČITIH PRIMENA NAREDBE SELECT

## Tabela S\_RJ i RADNIK

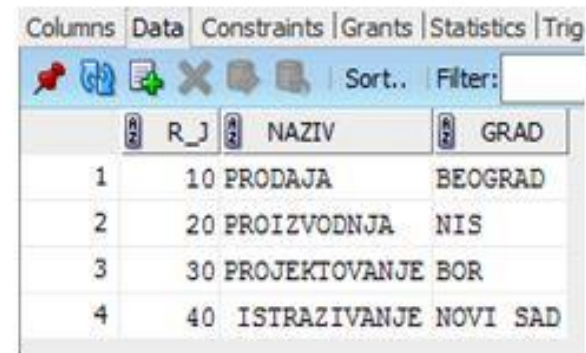
Mogućnosti naredbe SELECT za prikaz sadržaja tabela su ilustrovane na primeru relacionog modela sastavljenog od dve relacije: RADNIK i S\_RJ (RADNA\_JEDINICA).

**Relacija S\_RJ** ima prost ključ S\_RJ (šifra radne jedinice) i opisne atribute NAZIV i GRAD.

**Relacija RADNIK** ima prost ključ S\_RADNIK (šifra radnika) i opisne atribute IME, POSAO, S\_RUKOV (šifra neposrednog rukovodioca), DAT\_ZAP (datum zaposlenja), LD (lični dohodak), PREMIJA i S\_RJ (šifra RJ-a u kojoj radnik radi).

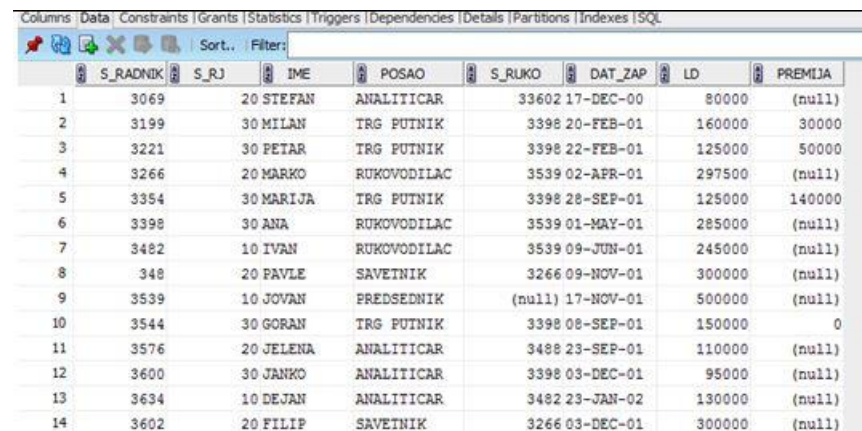
Atribut S\_RJ relacije RADNIK je spoljni, sekundarni ključ i omogućuje povezivanje te relacije sa relacijom S\_RJ.

Odgovarajuće tabele su prikazane na slici 1. i slici 2



	R_J	NAZIV	GRAD
1	10	PRODAJA	BEOGRAD
2	20	PROIZVODNJA	NIS
3	30	PROJEKTOVANJE	BOR
4	40	ISTRAZIVANJE	NOVI SAD

Slika 1. Tabela: S\_RJ



	S_RADNIK	S_RJ	IME	POSAO	S_RUKOV	DAT_ZAP	LD	PREMIJA
1	3069	20	STEFAN	ANALITICAR	33602	17-DEC-00	80000	(null)
2	3199	30	MILAN	TRG PUTNIK	3398	20-FEB-01	160000	30000
3	3221	30	PETAR	TRG PUTNIK	3398	22-FEB-01	125000	50000
4	3266	20	MARKO	RUKOVODILAC	3539	02-APR-01	297500	(null)
5	3354	30	MARIJA	TRG PUTNIK	3398	28-SEP-01	125000	140000
6	3398	30	ANA	RUKOVODILAC	3539	01-MAY-01	285000	(null)
7	3482	10	IVAN	RUKOVODILAC	3539	09-JUN-01	245000	(null)
8	348	20	PAVLE	SAVETNIK	3266	09-NOV-01	300000	(null)
9	3539	10	JOVAN	PREDSEDNIK	(null)	17-NOV-01	500000	(null)
10	3544	30	GORAN	TRG PUTNIK	3398	08-SEP-01	150000	0
11	3576	20	JELENA	ANALITICAR	3488	23-SEP-01	110000	(null)
12	3600	30	JANKO	ANALITICAR	3398	03-DEC-01	95000	(null)
13	3634	10	DEJAN	ANALITICAR	3482	23-JAN-02	130000	(null)
14	3602	20	FILIP	SAVETNIK	3266	03-DEC-01	300000	(null)

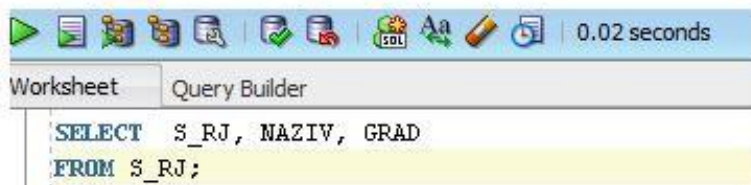
Slika 2. Tabela RADNIK

# PRIMER 1.

## Primeri za prikaz neizmenjenog sadržaja tabele

Naredbom SELECT.....FROM se može prikazati neizmenjen sadržaj tabele

Primer: Prikazati šifre, nazive i lokacije svih radnih jedinica iz relacije o RADNA\_JEDINICA (slika 3. i slika 4.)



```
SELECT S_RJ, NAZIV, GRAD
FROM S_RJ;
```

Slika 3. SELECT naredba za prikaz sadržaja tabele S\_RJ

Iz naredbe SELECT se dobija sledeći izlaz:

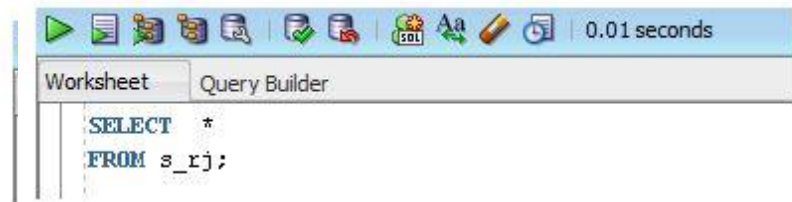
S_RJ NAZIV	GRAD
10 Prodaja	Beograd
20 Proizvodnja	MIS
30 Projektovanje	BOR
40 Istrazivanje	NOVI SAD

Slika 4. Neizmenjeni sadržaj tabele S\_RJ

U primeru je prikazano korišćenje naredbe SELECT za prikaz svih kolona iz table RADNA\_JEDINICA, navođenjem njihovih imena.

Kada se traže svi atributi neke relacije, umesto navođenja svakog atributa pojedinačno, moguće je koristiti znak "\*"sa istim dejstvom (slike 5. i 6.)

Primer:



```
SELECT *
FROM s_rj;
```

Slika 5. Drugi način za prikaz neizmenjenog sadržaja tabele S\_RJ

Iz naredbe SELECT se dobija sledeći izlaz:

S_RJ NAZIV	GRAD
10 Prodaja	Beograd
20 Proizvodnja	MIS
30 Projektovanje	BOR
40 Istrazivanje	NOVI SAD

Slika 6. Prikaz sadržaja svih atributa tabele

U primeru je prikazano korišćenje naredbe SELECT kojom se dobijaju svi atributi neke relacije korišćenjem znaka "\*".

## PRIMER 2.

### *Prikaz sadržaja samo nekih atributa*

Kada želimo da prikazemo samo neke attribute relacije, nakon reči SELECT navodimo nazive tih atributa ( slika 7. i slika 8.)

*Primer: Prikazati sve poslove radnika.*



Slika 7. Naredba SELECT za prikaz atributa POSAO

Iz naredbe SELECT se dobija sledeći izlaz:

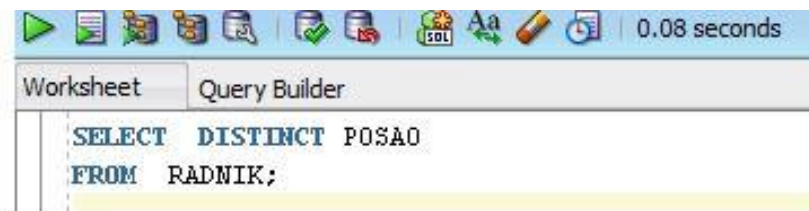
```
POSAO
-----
ANALITICAR
TRG PUTNIK
TRG PUTNIK
RUKOVODILAC
TRG PUTNIK
RUKOVODILAC
RUKOVODILAC
SAVETNIK
PREDSEDNIK
TRG PUTNIK
ANALITICAR
ANALITICAR
ANALITICAR
SAVETNIK
14 rows selected
```

Slika 8. Izlaz iz naredbe SELECT

U primeru je prikazno korišćenje naredbe SELECT za prikaz samo nekih atributa relacije, navođenjem naziva tih atributa.

**Koristeći klauzulu DISTINCT** obezbeđujemo prikazivanje samo različitih vrednosti nekog atributa (u ovom slučajku različitih poslova-slika 9 i slika 10).

*Primer:*



Slika 9. Korišćenje klauzule DISTINCT

Iz naredbe SELECT se dobija sledeći izlaz:

```
POSAO
-----
PREDSEDNIK
SAVETNIK
ANALITICAR
TRG PUTNIK
RUKOVODILAC
```

Slika 10. Izlaz iz naredbe SELECT

# Vežba: Primeri bazicnih oblika SELECT naredbe

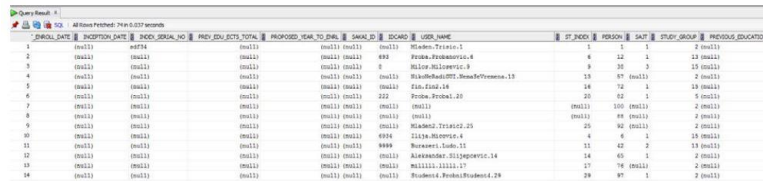

# NAREDBA SELECT BEZ WHERE KLAUZULE

## Nad tabelom STUDENT

Nekvalifikovano pretraživanje SELECT naredbe je kada ne sadrži WHERE.

Primer za nekvalifikovano pretraživanje:

SELECT \* FROM student;



Query Result 1 | 48 Rows Fetched: 794 0.027 seconds

ID	PERSON_ID	PERSON	SAT	STUDY_GROUP	PREVIOUS_EDUCATION
1	Mladen.Trisic.1	1	1	1	2
2	Proba.Probanovic.6	6	12	1	13
3	Milos.Milosevic.9	9	20	3	15
4	NikolcRadiGUI.NemaSeVremena.13	13	87	1	2
5	fin.fin.2.16	16	72	1	13
6	Proba.Probal.20	20	52	1	5
7	(null)	(null)	(null)	(null)	(null)
8	(null)	(null)	(null)	(null)	(null)
9	Mladen2.Trisic2.25	25	92	2	2
10	Ilija.Micovic.4	4	6	1	10
11	Burazeri.Ludo.11	11	42	2	13
12	Aleksandar.Slijepcevic.14	14	49	1	2
13	mililil.lililil.17	17	78	(null)	2
14	Student4.ProbniStudent4.29	29	87	1	2

Slika 1. Selektovanje svih podataka iz tabele student

SELECT user name, st index, idcard FROM student;



Query Result 1 | 35 Rows Fetched: 794 0.007 seconds

ID	USER_NAME	ST_INDEX	IDCARD
1	Mladen.Trisic.1	1	(null)
2	Proba.Probanovic.6	6	693
3	Milos.Milosevic.9	9	8
4	NikolcRadiGUI.NemaSeVremena.13	13	(null)
5	fin.fin.2.16	16	(null)
6	Proba.Probal.20	20	222
7	(null)	(null)	(null)
8	(null)	(null)	(null)
9	Mladen2.Trisic2.25	25	(null)
10	Ilija.Micovic.4	4	6934
11	Burazeri.Ludo.11	11	9999
12	Aleksandar.Slijepcevic.14	14	(null)
13	mililil.lililil.17	17	(null)
14	Student4.ProbniStudent4.29	29	(null)
15	(null)	(null)	(null)
16	Proba4.Proba4.23	23	(null)
17	Ponedeljak2.Ponedeljnik2.35	35	(null)
18	Ponedeljak3.Ponedeljnik3.36	36	(null)
19	Ponedeljak4.Ponedeljnik4.37	37	(null)
20	Utorak.Utornik.40	40	(null)
21	Cetvrtak.Cetvrtkovic.41	41	(null)
22	Cetvrtak2.Cetvrtkovic2.42	42	(null)
23	Petak.Petkovic.43	43	(null)
24	TestiranjeUtorak2.TestiranjeUtornik2.47	47	(null)
25	TestiranjeUtorak3.TestiranjeUtornik3.48	48	(null)
26	TestiranjeUtorak6.TestiranjeUtornik6.51	51	(null)
27	TestiranjeUtorak12.TestiranjeUtornik12.59	59	(null)
28	TestiranjeUtorak14.TestiranjeUtornik14.60	60	(null)
29	TestiranjeUtorak16.TestiranjeUtornik16.62	62	(null)
30	TestiranjeUtorak19.TestiranjeUtornik19.65	65	(null)
31	TestiranjeUtorak20.TestiranjeUtornik20.66	66	(null)
32	TestiranjeSreda.TestiranjeSredaPresime.67	67	(null)
33	Sreda3.Prezime3.69	69	(null)
34	Deseta.proba.71	71	(null)
35	Milos.Ijubinkovic.2	2	22225

Slika 2. Selektovanje određenih kolona iz tabele student



# Uvod u phpMyAdmin


---

10

# ČEMU SLUŽI PHPMYADMIN?

*Za upravljanje MySQL bazama podataka; Može da kreira i briše baze podataka, izvodi operacije nad tabelama i poljima, izvršava SQL upite, rukovodi ključevima*

PHPMyAdmin je besplatan alat koji je napisan u PHP-u i služi za upravljanje MySQL bazama podataka. Može da stvara i uklanja baze podataka, izvodi operacije nad tabelama i poljima, izvršava SQL upite, rukovodi ključevima nad poljima. Kreirao ga je Tobias Račiler (Tobias Ratschiller) je započeo rad na 1998 godine inspirisan MySQL-Webadmin-om. Softver je trenutno dostupan u 47 različitih jezika.

PHPMyAdmin može da se instalira na više načina. Može doći kao dodatak prilikom instalacije lokalnih servera WAMP (<http://www.wampserver.com/en/>) ili XAMPP (<http://www.apachefriends.org/en/xampp.html>). Osim toga može se preuzeti i sa adrese [http://www.phpmyadmin.net/home\\_page/index.php](http://www.phpmyadmin.net/home_page/index.php) ali s obzirom da su PHP fajlovi u pitanju potrebno je postojati server koji ima mogućnost da ih pokreće. Preporuka je da se koristi WAMP server i u daljem tekstu biće korišćen.

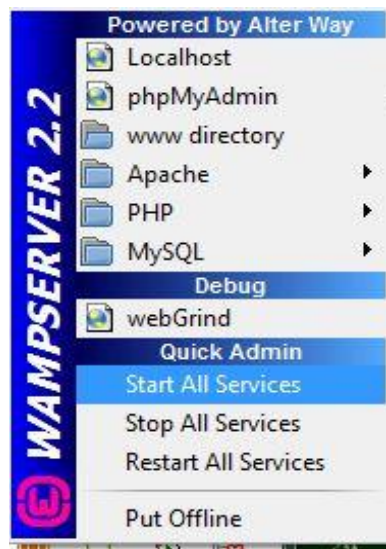
Nakon preuzimanja i instalacije potrebno je obratiti pažnju na neke stvari kako ne bi imali problema:

- 1.) Potrebno je voditi računa o tome da li je port 80 zauzet (Skype zauzima port 80, i samim tim potrebno je isključiti skype ili u opcijama podesiti da se koristi drugi port)
- 2.) Da li postoje još neki serveri instalirani (phpDEV, XAMPP etc, preporučuje se samo korišćenje jednog od navedenih, jer će Apache server biti zauzet).

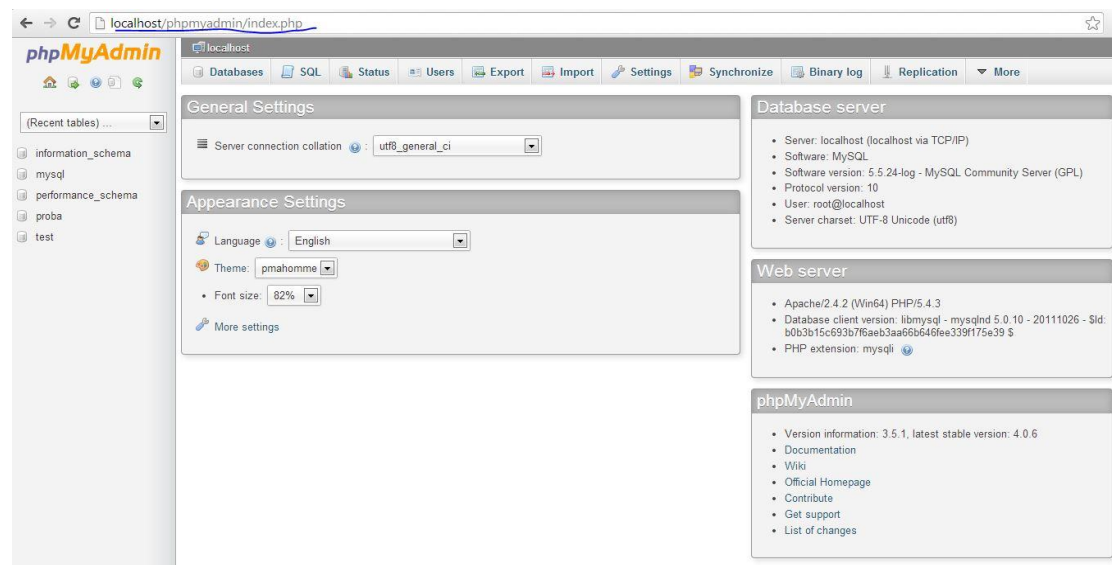
# POKRETANJE PHPMYADMIN

*Nakon pokretanja servisa, WAMP je u desnom uglu zelene boje.*

Nakon pokretanja WAMP-a očekujemo da se u donjem desnom uglu pojavi ikonica WAMP servera koja mora biti zelene boje. Ona može biti narandžaste ili crvene. Narandžaste je boje ukoliko postoji neki problem (može biti problem u Apache, MySQL ili PHP). Prikaz pokretanja celokupnog WAMP servera dat je na slici.



Nakon pokretanja servisa WAMP bi u desnom uglu trebao biti zelene boje. Posle toga, potrebno je u web čitaču kucati sledeće: localhost/phpmyadmin. Nakon toga ukoliko je servis pokrenut, ukoliko nije bilo nikakvog problema pokrenuće se aplikacija.



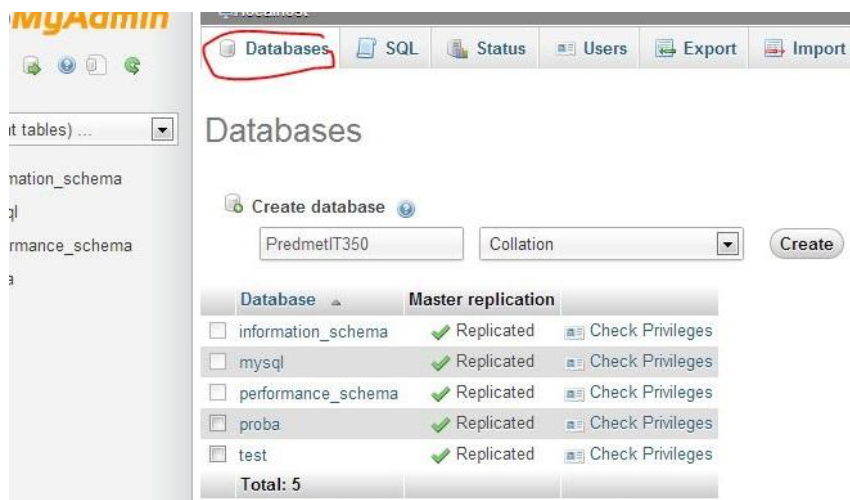
Slika 2. Prikaz phpMyAdmin početne strane

Na slici 2. su prikazani i meniji koji se nalaze u phpMyAdmin-u i to za: kreiranje i upravljanje privilegijama baza (**Databases**), za pisanje SQL upita (**SQL**), statistički podaci (**Status**), korisnici sa svojim privilegijama (**Users**), izvoz-eksportovanje SQL koda (**Export**), ubacivanje-importovanje SQL koda (**Import**), opšta podešavanja (**Settings**), sinhronizacija (**Synchronize**).

# KREIRANJE BAZE

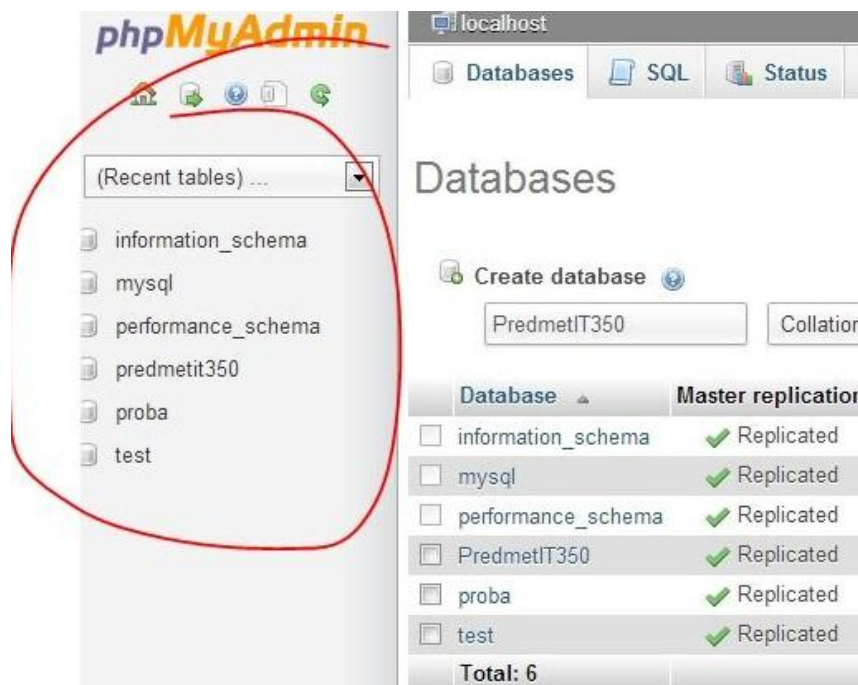
*Imena svih kreiranih baza se mogu videti u levom delu phpMyAdmin-a*

Potrebno je kliknuti na tab Databases, a zatim uneti ime tabele kao što je prikazano na slici 3. Encoding koji biramo kako bi bila prikazana ćirilčna slova i specifična slova naše latinice potrebno je izabrati UTF8-Encoding



Slika 3. Kreiranje baze

Nakon kreiranja, imena svih baza se mogu videti u levom delu phpMyAdmin-a. Potrebno je izabrati željenu bazu i dva puta kliknuti na nju (slika 4.)



Slika 4. Prikaz svih baza

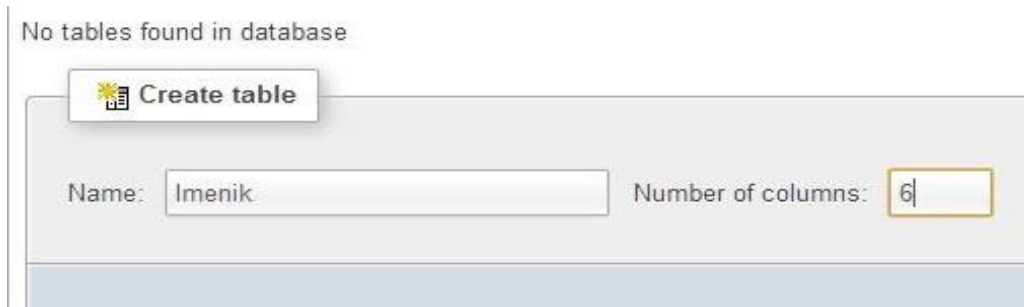
# KREIRANJE TABELA

*Prilikom kreiranja tabela potrebno je dodati imena kolona, tipove podataka kolona, specificirati njihovu duzinu i podrazumevanu vrednost (prazno polje, null...) itd.*

Nakon ulaska u određenu bazu ukoliko ne postoje tabele, otvoriće se opcija za kreiranje tabele. Na slici je prikazano kreiranje tabele Imenik koja sadrži 6 kolone (slika 5.)

Nakon kreiranja tabele, potrebno je dodati imena kolona, odrediti tipove podataka kolona, ukoliko je potrebno i specificirati kolika je duzina ili velicina tipa podataka (npr. 300 karaktera), zatim podrazumevana vrednost (prazno polje, null...) charset, komentari itd. Pre svega postoje puno tipova podataka i potrebno je razjasniti kada se koji koristi kako bi znali koji tip podataka ćemo upotrebiti određenu kolonu (slika 6.) Više o tome može se pročitati na adresi:

<http://www.homeandlearn.co.uk/php/php12p3.html>.



No tables found in database

Create table

Name: Imenik Number of columns: 6

Slika 5. Kreiranje tabele Imenik

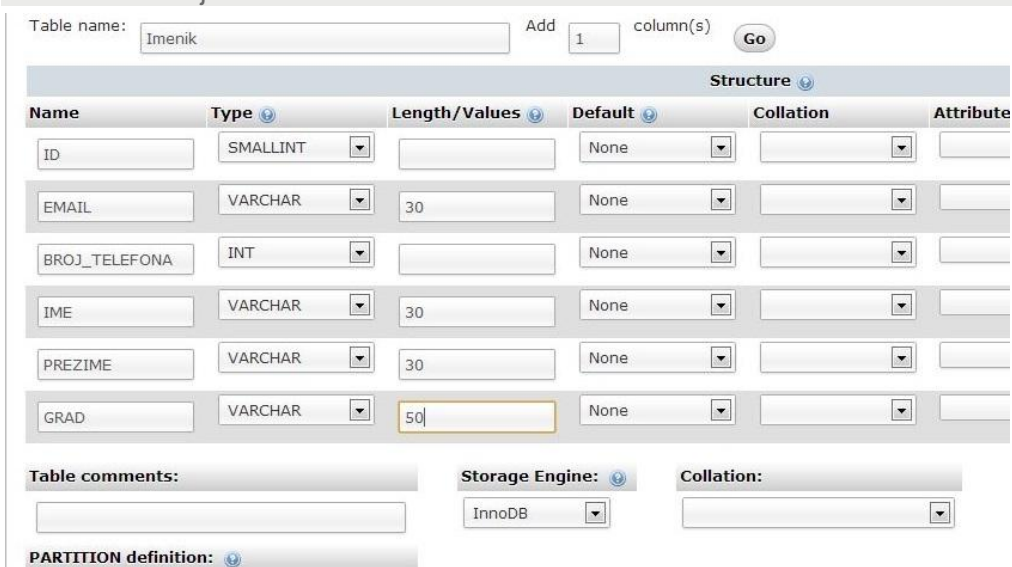


Table name: Imenik Add 1 column(s) Go

Structure

Name	Type	Length/Values	Default	Collation	Attribute
ID	SMALLINT		None		
EMAIL	VARCHAR	30	None		
BROJ_TELEFONA	INT		None		
IME	VARCHAR	30	None		
PREZIME	VARCHAR	30	None		
GRAD	VARCHAR	50	None		

Table comments:

Storage Engine: InnoDB

Collation:

PARTITION definition:

Slika 6. Unošenje kolona u tabelu Imenik

# PRIKAZ KREIRANE TABELE

*Prikaz je moguć na način prikazan na slici*

Nakon toga kreirana tabela može se pogledati kao što je prikazano na sledećoj slici 6.

Table name:  Add  column(s)

Name	Type	Length/Values	Default	Collation	Attribute
<input type="text" value="ID"/>	SMALLINT	<input type="text"/>	None	<input type="text"/>	<input type="text"/>
<input type="text" value="EMAIL"/>	VARCHAR	30	None	<input type="text"/>	<input type="text"/>
<input type="text" value="BROJ_TELEFONA"/>	INT	<input type="text"/>	None	<input type="text"/>	<input type="text"/>
<input type="text" value="IME"/>	VARCHAR	30	None	<input type="text"/>	<input type="text"/>
<input type="text" value="PREZIME"/>	VARCHAR	30	None	<input type="text"/>	<input type="text"/>
<input type="text" value="GRAD"/>	VARCHAR	50	None	<input type="text"/>	<input type="text"/>

Table comments:

Storage Engine:

Collation:

PARTITION definition:

Slika 6. Prikaz kreirane tabele Imenik

# SQL NAREDBE

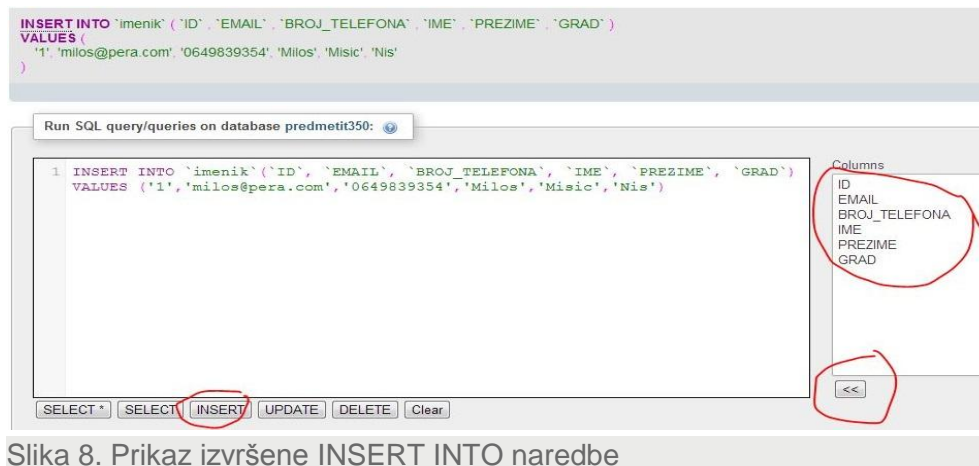
*Unos SQL naredbi je olakšan jer editor omogućava dobijanje standardnog oblika naredbe koju želi da unese*

Ukoliko želimo da kreiramo neke SQL naredbe potrebno je kliknuti na tab SQL. Dosta je olakšan unos naredbi pa se može videti tabovi SELECT\*, SELECT, INSERT, UPDATE, DELETE koji nakon odabira baze tabele sa kojom se radi ispišu sa desne strane i kolone nad kojima se mogu vršiti upiti.

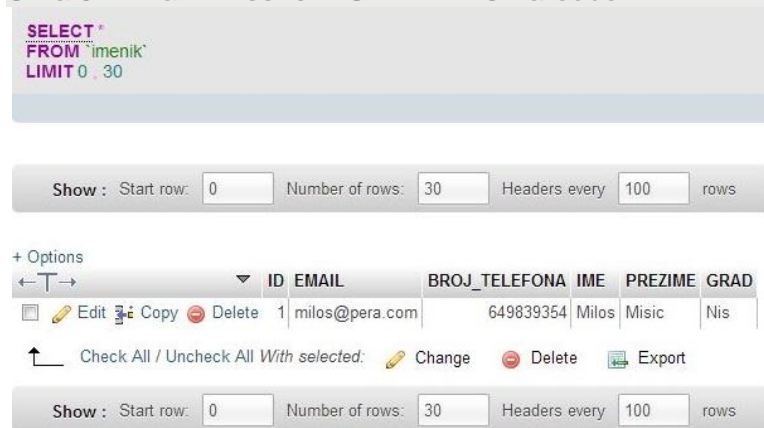
Ukoliko želimo da koristimo INSERT naredbu, možemo samo kliknuti na tab INSERT i dobićemo kod:

```
INSERT INTO `imenik` (`ID`, `EMAIL`, `BROJ_TELEFONA`, `IME`, `PREZIME`, `GRAD`) VALUES ([value-1],[value-2],[value-3],[value-4],[value-5],[value-6]).
```

Ukoliko se određena samo kolona odredi da se unos vrši samo u njoj može se izvesti odbariom imena kolone. Ovim je olakšano kucanje osnovnih upita.



Slika 8. Prikaz izvršene INSERT INTO naredbe



Slika 9. Prikaz izvršavanja SELECT\* naredbe.

# Zaključak

---



# ZAKLJUČAK

## *Šta smo naučili u ovoj lekciji?*

U ovom predavanju se govori o dve grupe naredbi SQL-a: naredbe za definisanje podataka (DDL) i naredbe za manipulisanje podacima (DMS). U grupu naredbi za definisanje podataka spadaju naredbe čija je osnovna namena kreiranje i izmena strukture ili brisanje osnovnih elemenata baze podataka kao što su tabele i indeksi nad tabelama kao i naredbe za dodeljivanje i oduzimanje prava korišćenja tabela drugim korisnicima.

U grupu naredbi za manipulisanje podacima (DMS) spadaju naredbe kojima se menjaju sadržaji osnovnih elemenata baze podataka tako što omogućavaju unošenje novih podataka u tabele, izmenu sadržaja podataka u tabelama ili brisanje sadržaja iz tabela.

Korišćenje svih ovih naredbi je potkrepljeno odgovarajućim primerima koji studentima treba da omoguće lakše razumevanje navedenih mogućnosti DDL-a i DMS-a.