UNIVERZITET METROPOLITAN BEOGRAD

IT250 - BAZE PODATAKA

SQL za definiciju podataka (DDL)

Lekcija 08

PRIRUČNIK ZA STUDENTE

IT250 - BAZE PODATAKA

Lekcija 08

SQL ZA DEFINICIJU PODATAKA (DDL)

UVOD

Šta ćemo naučiti u ovoj lekciji?

SQL pored ostalih sadrži i grupe naredbi koje omogućuju definisanje resursa relacione baze podataka (eng. Data Definition Language -DDL) kao i naredbe za manipulaciju podacima u bazi podataka (eng. Data Manipulation Language -DML) koje omogućuju ažuriranje podataka u širem smislu značenja te reči i izveštavanje iz relacione baze podataka.

U DDL, spadaju naredbe:

1. CREATE TABLE (kreiranje tabele baze podataka)
2. CREATE INDEX (kreiranje indeksa)
3. ALTER TABLE (izmena definicije tabele)
4. DROP TABLE (izbacivanje tabele iz baze podataka)

Osnovne operacije vezane za manipulaciju podataka realizuju se DML naredbama:

1. INSERT (dodavanje redova postojećoj tabeli)
2. U. U PDATE (izmena vrednosti kolona tabele)
3. DELETE (izbacivanje redova tabele)
4. SELECT (Nalaženje podataka u bazi podataka)

U lekciji je opisana upotreba i sintaksa svih naredbi DDL-a kao i naredbi DML-a INSERT, UPDATE i DELETE.

Poglavlje 1

SQL

ŠTA JE SQL (STANDARD QUERY LANGUAGE)?

SQL je standardni jezik za pristup bazama podataka

SQL (eng.Structured Query Language) predstavlja standardizovani jezik za upravljanje i manipulaciju podacima u relacionim bazama podataka.

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

Osnovne karakteristike su:

Relacioni model: SQL je zasnovan na relacionom modelu podataka, koji predstavlja podatke u obliku tabela sa redovima i kolonama. Ove tabele se međusobno povezuju prema definisanim relacijama, što omogućava efikasno organizovanje podataka.

Upiti (eng.queries): SQL omogućava izvršavanje različitih vrsta upita za dobijanje i filtriranje podataka iz baze. Upiti se koriste za pretragu, izmenu, brisanje i unos novih podataka u bazu.

Kreiranje i upravljanje baza podataka: SQL omogućava kreiranje novih baza podataka, tabela i indeksa. Takođe, korisnici mogu definisati pravila i ograničenja koja se primenjuju na podatke kako bi se održao integritet baze.

Bezbednost : SQL pruža mehanizme za upravljanje pristupom podacima i sprovodi različite nivoe bezbednosti kako bi se zaštitili podaci od neovlašćenog pristupa i modifikacija.

Transakcije: SQL podržava transakcije, što omogućava izvršavanje skupa operacija kao jedne celine. Transakcije obezbeđuju da se promene na podacima izvrše u celosti ili se uopšte ne izvrše ukoliko neka od operacija ne uspe.

Višekorisnička podrška: SQL omogućava rad sa više korisnika koji pristupaju istoj bazi podataka istovremeno, čime se omogućava efikasno deljenje informacija u organizaciji.

Podrška za upravljanje metapodacima: SQL omogućava upravljanje metapodacima koji opisuju strukturu baze podataka, što je korisno za automatsko generisanje upita i izveštaja.

Portabilnost: SQL je standardizovan od strane međunarodnih organizacija, što znači da je moguće pisati SQL upite koji će raditi na različitim sistemima za upravljanje bazama podataka (Oracle, MySQL, SQL Server itd.).

JEZIK SQL

Nastavlja se opis jezika SQL

Deklarativni jezik : SQL je deklarativni jezik, što znači da programer definiše šta želi postići, a ne kako to tačno postići. Umesto naredbi koje detaljno opisuju korake, programer daje upite koji opisuju šta želi dobiti iz baze podataka.

Višestruka DBMS podrška: SQL je standardizovan jezik, što znači da je kompatibilan sa različitim sistemima za upravljanje bazama podataka (DBMS). To omogućava da SQL upiti budu prenosivi između različitih baza podataka, što olakšava migraciju ili rad na različitim platformama.

Ove osnovne karakteristike SQL-a čine ga moćnim i univerzalnim alatom za upravljanje podacima, koji se koristi širom sveta u različitim industrijskim granama i organizacijama. Bez obzira da li koristite SQL za upravljanje malom bazom podataka ili složenim sistemom sa velikim količinama podataka, razumevanje SQL-a je od ključne važnosti za efikasno rukovanje informacijama.

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:

1. 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.
2. Mogućnost interaktivnog i klasičnog (aplikativnog) programiranja. Koristeći SQL dobijaju se odgovori na trenutne, unapred nepredviđene zahteve ili se SQL blokovi "ugrađuju" u klasični viši programski jezik (Java, PHP) omogućujući klasičnu obradu.
3. **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 (eng.Data Definition Statements - DDL): Omogućuju definisanje resursa relacione baze podataka.

- Naredbe za manipulisanje podacima (eng.Data Manipulation Statements - DML):Omogućuju ažuriranje u širem smislu značenja te reči i izveštavanje iz relacione baze podataka.
- Naredbe za kontrolne funkcije (eng.Data Control Functions - DCF): Omogućavaju oporavak, konkurentnost, sigurnost i integritet relacione baze podataka.

NAREDBE ZA DEFINISANJE I MANIPULACIJU PODACIMA - DDL; DML

Naredbe DDL-a omogućuju definisanje resursa relacione baze podataka; Naredbe DML omogućuju ažuriranje podataka i izveštavanje iz baze

U DDL spadaju naredbe:

1. **CREATE DATABASE** (kreiranje baze podataka)
2. **CREATE USER / ALTER USER** (kreiranje/izmena korisnika)
3. **CREATE SCHEMA** (kreiranje šeme)
4. **CREATE TABLE** (kreiranje tabele baze podataka)
5. **CREATE INDEX** (kreiranje indeksa)
6. **ALTER TABLE** (izmena definicije tabele)
7. **DROP TABLE** (izbacivanje tabele iz baze podataka)

U DML spadaju naredbe:

1. **INSERT** (dodavanje redova postojećoj tabeli)
2. **UPDATE** (izmena vrednosti kolona tabele)
3. **DELETE** (izbacivanje redova tabele)
4. **SELECT** (izveštavanje iz baze podataka)

Poglavlje 2

DDL - Naredbe za rad sa bazom podataka

KREIRANJE, SELEKTOVANJE I BRISANJE BAZA PODATAKA

Naredbe: CREATE DATABASE; SHOW DATABASES; DROP DATABASE

Da bi kreirali novu SQL bazu podataka, koristimo naredbu:

CREATE DATABASE imeBaze;
Ipak pri kreiranju baze podataka moramo voditi računa da ime baze mora biti jedinstveno (unique) u okviru RDBMS. U suprotnom, javiće se greška.
Ukoliko baza podataka sa datim imenom već postoji, javlja se greška, pa morate dati neko drugo ime bazi.
Da bi olakšali ovaj proces, u MySQL-u postoji klauzula kojom možete navesti da se kreira baza podataka sa datim imenom, ali samo ukoliko ne postoji .
CREATE DATABASE IF NOT EXISTS imeBaze;

Da biste kreirali bazu podataka, takođe morate imate privilegije administratora na sistemu.

Ukoliko koristite konzolu za pristup serveru i bazama podataka, ili ne možete da vidite koje su baze podataka (ili sa kojim imenima) već kreirane i postoje, postoji komanda koja će to olakšati. Da bi prikazali sve baze podataka koristimo komandu:

SHOW DATABASES;

Nakon uspešnog kreiranja baze podataka, možete početi sa njenim korišćenjem i kreiranjem tabela u njoj. Da bismo pristupili novoj bazi (ili prebacili rad u već postojeću bazu podataka koritimo naredbu:

USE imeBaze;
Ukoliko želimo da u potpunosti obrišemo bazu podataka, koristimo naredbu:
DROP DATABASE imeBaze;

Ono na šta treba obratiti pažnju pri korišćenju ove naredbe je da naredba DROP DATABASE imeBaze briše kompletnu zadatu bazu podataka, uključujući i podatke koji su sačuvani u okviru navedene baze.

Takođe, kao i pri kreiranju morate imati administratorske privilegije nad datom bazom podataka.
Da biste proverili da li je baza podataka koju ste obrisali stvarno obrisana, možete koristiti naredbu SHOW DATABASES;.

KREIRANJE KORISNIKA (CREATE USER)

Naredba za kreiranje novih korisnika baze podataka

Naredba za kreiranje novih korisnika CREATE USER ima sledeću sintaksu:

```
CREATE USER username IDENTIFIED {BY password | EXTERNALLY | GLOBALLY AS
'external_name'} options;

options:

DEFAULT TABLESPACE tablespace

TEMPORARY TABLESPACE tablespace

QUOTA int {K | M} ON tablespace

QUOTA UNLIMITED ON tablespace

PROFILE profile_name

PASSWORD EXPIRE ACCOUNT {LOCK|UNLOCK}
```

Značenje klauzula koje se koriste u naredbi je sledeće:

1. ***IDENTIFIED klauzula*** označava kako će se korisnik autentifikovati :
2. ***BY password***: omogućava kreiranje lokalnog korisnika koji mora da unese svoj pasvord radi logovanja na bazu podataka.
3. ***EXTERNALLY klauzula***: omogućava kreiranje eksternog korisnika koji mora biti autentifikovan nekim eksternim servisom kao što je operativni servis
4. ***GLOBALLY klauzula****:* omogućava kreiranje globalnog korisnika koji mora biti autentifikovan enterprise directory servisom
5. ***TABLESPACE klauzula****:* Specificira tablespace u kojem će se smeštati objekti koje korisnik kreira a koji može biti DEFAULT ili privremeni (TEMPORARY). Ako nije kreiran DEFAULT tablespace, objekti će biti smeštani u SYSTEM tablespace.
6. ***QUOTA klauzula****:* Koristi se da se specificira maksimalni prostor na kojem korisnik može da smešta svoje tabele. QUOTA Clause se može odnositi na različite tablespace-ove. UNLIMITED omogućava korisniku da ima neograničen prostor.
7. ***PASSWORD EXPIRE klauzula***: Prisiljava korisnike da promene password pre logovanja na bazu podataka
8. ***ACCOUNT klauzula***: ACCOUNT LOCK zaključava račun korisnika i onemogućava mu da se loguje na bazu podataka; ACCOUNT UNLOCK otključava račun korisnika i omogućava mu da se loguje na bazu podataka

IZMENA KORISNIKA (ALTER USER)

Služi za izmenu prethodno kreiranih korisnika

1. Primer naredbe CREATE USER:

```
CREATE USER sidney

IDENTIFIED BY out_standing1
DEFAULT TABLESPACE example
QUOTA 10M ON example
TEMPORARY TABLESPACE temp
QUOTA 5M ON system
PROFILE app_user PASSWORD EXPIRE;
```

2. primer naredbe CREATE

```
2. CREATE USER MySchemaOwner

IDENTIFIED BY ChangeThis
DEFAULT TABLESPACE data
TEMPORARY TABLESPACE temp
QUOTA UNLIMITED ON data;
```

Nakon kreiranja korisnika, može postajati potreba da izmenite neke parametre korišćene prilikom kreiranja kao što su password ili promenite neke atribute.

Promena parametara za već kreirane korisnike se vrši korišćenjem naredbe

ALTER USER

Ovde su dati neki primeri:

```
1. ALTER USER myuser IDENTIFIED BY new_password;
2. ALTER USER myuser DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp QUOTA 100M
ON users QUOTA 0 ON my_data;
3. ALTER USER myuser ACCOUNT LOCK;
4. ALTER USER myuser ACCOUNT UNLOCK;
5. ALTER USER myuser PASSWORD EXPIRE;
```

NAREDBA CREATE ZA KREIRANJE ŠEMA

Koncept SQL šeme je inkorporiran kako bi se sakupile tabele i druge konstrukcije koji pripadaju istoj aplikaciji baze podataka.

Rane verzije SQL-a nisu uključivale koncept šeme relacione baze podataka; sve tabele (relacije) smatrane su delom iste šeme. Koncept SQL šeme je inkorporiran počevši od SQL2 kako bi se sakupile tabele i druge konstrukcije koje pripadaju istoj aplikaciji baze podataka. SQL šema je identifikovana imenom šeme i uključuje identifikator za autorizaciju koji ukazuje na korisnike ili račune koji su vlasnici šeme, kao i deskriptore za svaki element u šemi.

Elementi šeme su tabele, ograničenja, pogledi, domeni i drugi elementi (kao što su grantovi za autorizaciju) koji opisuju šemu.

Šema se kreira korišćenjem naredbe

CREATE SCHEMA,
koja može da sadrži definicije svih elemenata šeme.

Alternativno, šemi se može dodeliti ime i identifikacioni identifikator, a elementi se mogu definisati kasnije. Na primer, naredni izraz kreira šemu pod nazivom COMPANY, koja je u vlasništvu korisnika sa autorizacijom "Jsmith".

CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

Generalno, nisu svi korisnici ovlašćeni da kreiraju šeme i elemente šeme. Ovlašćenje za kreiranje šema, tabela i drugih elemenata mora biti eksplicitno dodeljeno odgovarajućim korisničkim nalozima od strane administratora sistema ili DBA.

Poglavlje 3

Kreiranje tabele (CREATE TABLE)

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).

Kao što je rečeno, naredbama DDL-a koje služe za kreiranje resursa baze podataka, su pored naredbe *CREATE TABLE* obuhvaćene i sledeće naredbe:

1. *CREATE INDEX* (kreiranje indeksa),
2. *ALTER TABLE* (izmena definicije tabele),
3. *DROP TABLE* (izbacivanje tabele iz baze podataka)

Naredba *CREATE TABLE* se koristi za:

1. **kreiranje novih tabela,**
2. **definisanje kolona,**
3. **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:

1. **PRIMARY KEY,**
2. **UNIQUE,**
3. **NULL/NOT NULL,**
4. **FOREIGN KEY**
5. **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 u kolonama tabela.

Naredba CREATE TABLE se koristi za specificiranje nove relacije davanjem imena i navođenjem njenih atributa i početnih ograničenja. Prvo su navedeni atributi, a svakom atributu je dodeljeno ime, tip podataka koji specificira domen vrednosti, i sva ograničenja atributa, kao što je NOT NULL. Ključ, integritet entiteta i ograničenja referencijalnog integriteta su u naredbi CREATE TABLE specificirani nakon deklaracije atributa, ili se mogu dodati kasnije pomoću komande ALTER TABLE.

KREIRANJE TABELE KORISNIK

Prikazuje upotrebu naredbe CREATE TABLE za kreiranje tabele KORISNIK

Pretpostavićemo da želimo da kreiramo tabelu "Korisnik" koja će sadržati informacije o korisnicima naše aplikacije.

```
CREATE TABLE Korisnik (
    id INT PRIMARY KEY,
    ime VARCHAR(50) NOT NULL,
    prezime VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    godine INT,
    pol CHAR(1) CHECK (pol IN ('M', 'Ž')),
    datum_registracije DATE DEFAULT CURRENT_DATE
);
```

Objašnjenje kreiranog primera:

CREATE TABLE: označava da kreiramo novu tabelu u bazi podataka.
Korisnik: naziv tabele koju kreiramo.
id INT PRIMARY KEY: definiše kolonu "id" u tabeli. Tip podataka je "INT" (celobrojna vrednost), a ključna reč "PRIMARY KEY" označava da je ovo primarni ključ tabele, što znači da će vrednosti u ovoj koloni biti jedinstvene i koristiće se za identifikaciju svakog zapisa u tabeli.

ime VARCHAR(50) NOT NULL: definiše kolonu "ime" u tabeli. Tip podataka je "VARCHAR" (niska promenljive dužine), sa ograničenjem dužine od 50 karaktera. Ključna reč "NOT NULL" označava da ova kolona ne može sadržati NULL vrednost, odnosno mora imati vrednost za svaki zapis.
prezime VARCHAR(50) NOT NULL: definišemo kolonu "prezime" sa istim ograničenjima kao i za kolonu "ime".
email VARCHAR(100) UNIQUE: definiše kolonu "email" u tabeli. Tip podataka je takođe "VARCHAR" sa ograničenjem dužine od 100 karaktera. Ključna reč "UNIQUE" označava da vrednosti u ovoj koloni moraju biti jedinstvene za svaki zapis u tabeli, što znači da ne može biti više korisnika sa istom email adresom.
godine INT: definiše kolonu "godine" u tabeli sa tipom podataka "INT" (celobrojna vrednost).

pol CHAR(1) CHECK (pol IN (’M’, ’Ž’)): definiše kolonu "pol" u tabeli sa tipom podataka "CHAR" i dužinom od 1 karaktera. Ključna reč "CHECK" postavlja uslov da vrednost u ovoj koloni može biti samo ’M’ ili ’Ž’, što ograničava moguće vrednosti na ove dve opcije.

datum_registracije DATE DEFAULT CURRENT_DATE: definiše kolonu "datum_registracije" sa tipom podataka "DATE" (datum). Ključna reč "DEFAULT CURRENT_DATE" postavlja podrazumevanu vrednost za ovu kolonu na trenutni datum kada se novi zapis ubaci u tabelu.

KREIRANJE TABELE NARUDZBINE SA OGRANIČENJEM FOREIGN KEY

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

Primer kreiranja nove tabele "Narudzbina" koja će sadržati informacije o narudžbinama, uključujući i strani ključ koji je referenciran na tabelu "Korisnik":

```
CREATE TABLE Narudzbina (
    id INT PRIMARY KEY,
    korisnik_id INT,
    proizvod VARCHAR(100) NOT NULL,
    kolicina INT,
    datum_narudzbine DATE,
    FOREIGN KEY (korisnik_id)
        REFERENCES Korisnik(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CHECK (kolicina > 0)
);
```

Objašnjenje kreiranog primera:

CREATE TABLE: označava da kreiramo novu tabelu "Narudzbina".
id INT PRIMARY KEY: definiše kolonu "id" kao primarni ključ tabele "Narudzbina".
korisnik_id INT: definiše kolonu "korisnik_id" koja će sadržati vrednosti stranog ključa, a tip podataka je "INT".

proizvod VARCHAR(100) NOT NULL: definiše kolonu "proizvod" sa tipom podataka "VARCHAR" (niska promenljive dužine) i ograničenjem dužine od 100 karaktera. Ključna reč "NOT NULL" označava da ova kolona ne sme sadržati NULL vrednosti.
kolicina INT: definiše kolonu "kolicina" sa tipom podataka "INT".
datum_narudzbine DATE: Definiše kolonu "datum_narudzbine" sa tipom podataka "DATE" (datum).
FOREIGN KEY (korisnik_id) REFERENCES Korisnik(id): definiše strani ključ "korisnik_id" koji je referenciran na kolonu "id" u tabeli "Korisnik". Ovo uspostavlja vezu između tabele "Narudzbina" i tabele "Korisnik", tako da "korisnik_id" u tabeli "Narudzbina" mora biti vrednost koja postoji u koloni "id" tabele "Korisnik".

ON DELETE CASCADE: Ova klauzula se dodaje na strani ključ "korisnik_id" u tabeli "Narudzbina". Kada se red u tabeli "Korisnik" obriše, svi redovi u tabeli "Narudzbina" koji imaju istu vrednost "korisnik_id" će biti automatski obrisani. To osigurava da ne ostanu nekonzistentni podaci u tabeli "Narudzbina" nakon brisanja korisnika.

OBJAŠNJENJE OGRANIČENJA FOREIGN KEY

Nastavak objašnjenja

ON UPDATE CASCADE: Ova klauzula se takođe dodaje na strani ključ "korisnik_id" u tabeli "Narudzbina". Kada se vrednost primarnog ključa "id" u tabeli "Korisnik" ažurira, sve vrednosti "korisnik_id" u tabeli "Narudzbina" koje su bile vezane za tu vrednost primarnog ključa će biti automatski ažurirane. To osigurava da podaci u tabeli "Narudzbina" ostaju u skladu sa ažuriranim vrednostima u tabeli "Korisnik".

CHECK (kolicina > 0): Ovo je dodatno ograničenje (CHECK constraint) koje definiše da vrednost u koloni "kolicina" mora biti veća od 0. Ovo sprečava unošenje negativnih ili nula vrednosti za količinu narudžbine.

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* može da sadrži klauzule kojima se specificira da li je ažuriranje ili brisanje kaskadno i to:

1. **Izraz *DELETE NO ACTION* -** označava da je zabranjeno brisanje redova tabele čiji se primarni ključevi koriste kao strani ključevi (foreign key) u drugim tabelama.
2. **Izraz *DELETE CASCADE*** -označava da je brisanje kaskadno što znači da se prilikom brisanja redova jedne tabele brišu i redovi drugih tabela u kojima su primarni ključevi date tabele koriste kao strani ključevi (foreign key). Predefinisana vrednost je DELETE NO ACTION.
3. **Izraz *UPDATE NO ACTION*** - označava da je zabranjeno ažuriranje primarnog ključa redova tabela ukoliko se on koristi kao strani ključ u drugim tabelama.
4. **Izraz *UPDATE CASCADE*** označava da je ažuriranje kaskadno. Predefinisana vrednost je UPDATE NO ACTION.

U prikazanom slučaju je iskorišćena klauzula UPDATE NO ACTION jer je primarni ključ tabele 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 tabele roditelja. Takve relacije, koje se često nazivaju uslovne relacije se pojavljuju u aplikacijama koje obrađuju tabele u kojima neki podaci nedostaju. Na slici 3.1 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 3.1 Tip relacije/ Ograničenja komande CREATE TABLE [Izvor: NM IT350-2020/2021.]

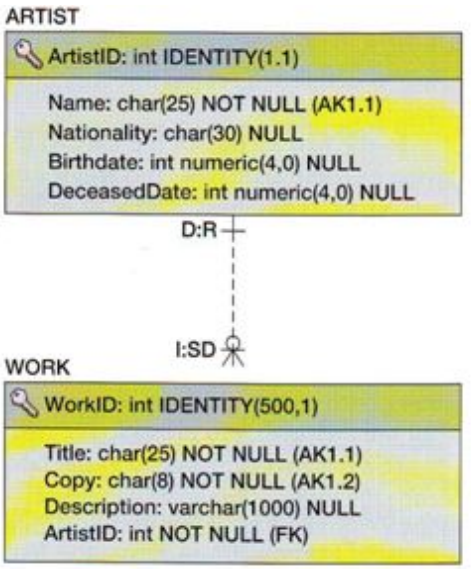
KLAUZULA CHECK

Služi za definisanje ograničenja koja se mogu primeniti na vrednosti podataka

Default vrednost se kreira specificiranjem ključne reči *DEFAULT* u definiciji kolone, odmah posle specifikacije NULL/NOT NULL.

Ograničenja podataka se kreiraju korišćenjem ključnih reči *CHECK CONSTRAINTS*.

Format je: CONSTRAINTS, ime ograničenja koje daje programer, reč CHECK, specifikacija ograničenja stavljena između zagrada.
Primer: *Na slici 3.2 su prikazane dve tabele baze podataka za galeriju slika. U tabeli na slici 3.3 je prikazan primer default vrednosti i primer ograničena koji se mogu primeniti na vrednosti podataka u bazi podataka galerije slika.*



Slika 3.2 Tabele ARTIST i WORK [Izvor: NM IT350-2020/2021.]

Tabela	Kolona	Default vrednost	Ograničenje
WORK	Description (opis)	Unknown provenance (Nepoznato poreklo)	
ARTIST	Nationality (nacionalnost)		IN ('Canadian', 'English', 'French', 'German', 'Mexican','Russian','US')
ARTIST	Birthdate (Datum rođenja)		Pre DeceasedDate
ARTIST	Birthdate (Godina rođenja)		4 cifre: prva cifra je 1 ili 2 ostale tri cifre od 0 do 9
ARTIST	DeceasedDate (Datum smrti)		4 cifre: prva cifra je 1 ili 2 ostale tri cifre od 0 do 9
TRANSACTION	SalesPrice (Prodajna cena)		> 1000 i <= 200000
TRANSACTION	PuechaseDate (Datum nabavke)		=>DateAcquire (Datum sticanja)

Slika 3.3 Ograničenja na vrednosti polja u bazi podataka galerije slika [Izvor: NM IT350-2020/2021.]

PRIMERI KREIRANJA OGRANIČENJA ZA VREDNOSTI PODATAKA

Kreiranje ograničenja za tabelu ARTIST

Prikazane su komande za kreiranje tabela WO RK i ARTIST sa ograničenjima za vrednosti podataka datih u tabeli na slici 3.3.

```
CREATE TABLE WORK (
    WorkID        int            NOT NULL IDENTITY (500, 1),
    Title         char(25)       NOT NULL,
    Copy          char(8)        NOT NULL,
    Description   varchar(100)   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 (ArtisID)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
```

```
CREATE TABLE ARTIST (
    ArtistID        int            NOT NULL IDENTITY (1, 1),
    Title           char(25)       NOT NULL,
    Nationality     char(30)       NULL,
    Birthdate       numeric(4, 0)  NULL,
    DeceasedDate    numeric(4, 0)  NULL,

    CONSTRAINT ArtistPK PRIMARY KEY (ArtisID),
    CONSTRAINT ArtistAK1 UNIQUE (Name),
    CONSTRAINT NationalityValues CHECK
        (Nationality IN ('Canadian', 'English', 'German',
         'Mexican', 'Russian', 'Spanish', 'US')),
    CONSTRAINT BirthValuesCheck CHECK (Birthdate < DeceasedDate),
```

```
    CONSTRAINT ValidBirthYear CHECK (Birthdate LIKE '[1-2] [0-9] [0-9] [0-9]'),
    CONSTRAINT ValidDeathYear CHECK (DeceasedDate LIKE '[1-2] [0-9] [0-9] [0-9]')
 );
```

KREIRANJE TABELA ZA BAZU PODATAKA COMPANY: EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS

Primer kreiranja tabela za bazu podataka COMPANY: EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS

U daljem tekstu su date relacije za bazu podataka COMPANY:

EMPLOYEE (Fname, Minit, Lname, **Ssn**, Bdate, Address, Sex, Salary, *Super_ssn, Dno*)

DEPARTMENT (Dname, **Dnumber**, *Mgr_ssn*, Mgr_start_date)

DEPT_LOCATIONS (**Dnumber, Dlocation**)

PROJECT (**Pname**, Pnumber, Plocation, *Dnum*)

TABLE WORKS_ON (Essn, Pno, Hours)

DEPENDENT (Essn, Dependent_name, Sex, Bdate DATE)

```
CREATE TABLE DEPT_LOCATIONS
( Dnumber INT NOT NULL,
Dlocation VARCHAR(15) NOT NULL,
PRIMARY KEY (Dnumber, Dlocation),
FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

```
CREATE TABLE EMPLOYEE
( Fname VARCHAR(15) NOT NULL,
Minit CHAR,
Lname VARCHAR(15) NOT NULL,
Ssn CHAR(9) NOT NULL,
Bdate DATE,
Address VARCHAR(30),
Sex CHAR,
Salary DECIMAL(10,2),
Super_ssn CHAR(9),
Dno INT NOT NULL,
PRIMARY KEY (Ssn),
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
```

```
CREATE TABLE DEPARTMENT
( Dname VARCHAR(15) NOT NULL,
Dnumber INT NOT NULL,
Mgr_ssn CHAR(9) NOT NULL,
Mgr_start_date DATE,
```

```
PRIMARY KEY (Dnumber),
UNIQUE (Dname),
FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

KREIRANJE TABELA ZA BAZU PODATAKA COMPANY: PROJECT, WORKS_ON

Primer kreiranja tabela za bazu podataka COMPANY: PROJECT, WORKS_ON

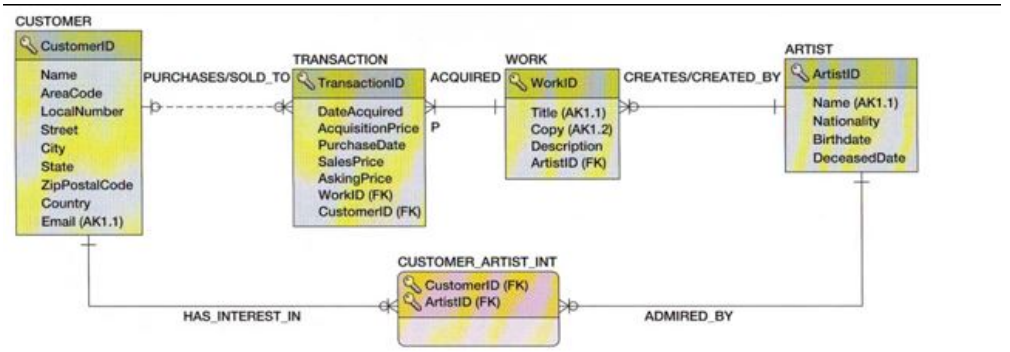
```
CREATE TABLE PROJECT
( Pname VARCHAR(15) NOT NULL,
Pnumber INT NOT NULL,
Plocation VARCHAR(15),
Dnum INT NOT NULL,
PRIMARY KEY (Pnumber),
UNIQUE (Pname),
FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
```

```
CREATE TABLE WORKS_ON
( Essn CHAR(9) NOT NULL,
Pno INT NOT NULL,
Hours DECIMAL(3,1) NOT NULL,
PRIMARY KEY (Essn, Pno),
FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
( Essn CHAR(9) NOT NULL,
Dependent_name VARCHAR(15) NOT NULL,
Sex CHAR,
Bdate DATE,
Relationship VARCHAR(8),
PRIMARY KEY (Essn, Dependent_name),
FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn)
```

CREATE TABLE-SLUČAJ IZUČAVANJA

Dizajn baze podataka galerije slika

Na slici 3. 4 je prikazan logički model baze podataka galerije slika ER dijagramom.



Slika 3.4 Logički model baze podataka galerije slika [Izvor: NM IT350-2020/2021.]

KREIRANJE TABELA ARTIST, WORK I CUSTOMER

Pored naziva i tipova atributa, CREATE TABLE naredbe sadrže i odgovarajuće CONSTRAINT-e za tabele ARTIST, WORK i CUSTOMER

```
CREATE TABLE WORK (
    WorkID        int            NOT NULL IDENTITY (500, 1),
    Title         char(25)       NOT NULL,
    Copy          char(8)        NOT NULL,
    Description   varchar(100)   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 (ArtisID)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
```

```
CREATE TABLE ARTIST (
    ArtistID        int            NOT NULL IDENTITY (1, 1),
    Title           char(25)       NOT NULL,
    Nationality     char(30)       NULL,
    Birthdate       numeric(4, 0)  NULL,
    DeceasedDate    numeric(4, 0)  NULL,

    CONSTRAINT ArtistPK PRIMARY KEY (ArtisID),
    CONSTRAINT ArtistAK1 UNIQUE (Name),
    CONSTRAINT NationalityValues CHECK
        (Nationality IN ('Canadian', 'English', 'German',
         'Mexican', 'Russian', 'Spanish', 'US')),
    CONSTRAINT BirthValuesCheck CHECK (Birthdate < DeceasedDate),
    CONSTRAINT ValidBirthYear CHECK (Birthdate LIKE '[1-2] [0-9] [0-9] [0-9]'),
    CONSTRAINT ValidDeathYear CHECK (DeceasedDate LIKE '[1-2] [0-9] [0-9] [0-9]')
 );
```

```
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)
 );
```

KREIRANJE TABELA TRANSACTION I CUSTOMER_ARTIST_INT

Pored naziva i tipova atributa, CREATE TABLE naredbe za tabele TRANSACTION i CUSTOMER_ARTIST_INT sadrže i odgovarajuće CONSTRAINT-e

Na slici 3.5 je prikazana CREATE TABLE naredba za kreiranje tabele TRANSACTION, a na slici 3.6 tabele 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,

        CONSTRAINT          SalesPriceRange CHECK
                            ( (SalesPrice > 1000) AND (SalesPrice <=200000) ),
        CONSTRAINT          ValidTransDate CHECK (PurchaseDate => DateAcquired)
);
```

Slika 3.5 CREATE TABLE naredba za tabelu TRANSACTION [Izvor: NM IT350-2020/2021.]

```
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 3.6 CREATE TABLE naredba za tabelu CUSTOMER_ARTIST_INT [Izvor: NM IT350-2020/2021.]

NAREDBA CREATE SEQUENCE

Sekvencom se generiše sekvencijalna serija jedinstvenih brojeva za potrebe surogat ključa

Sve tabele koje čine strukturu baze podataka galerije slika osim tabele CUSTOMER_ARTIST_WORK imaju surogat ključeve. Sistem za upravljanje bazom podataka kakav je na primer Oracle, ne podržava definiciju takvih ključeva, već se za to koristi Oracle sekvenca.

Naredbom CREATE SEQUENCE se generiše sekvencijalna serija jedinstvenih brojeva.

Primer: *Sledeća naredba definiše sekvencu koja je nazvana CustID koja počinje od 1000 i povećava se za 1 svaki put kada se iskoristi.*

Create Sequence CustID Increment by 1 start with 1000;

Za nas su važne dve metode. Metod *NextVal* koja obezbeđuje sledeću vrednost u sekvenci i metod *CurrVal* koja obezbeđuje tekuću vrednost sekvence. Tako CustID.NextVal obezbeđuje sledeću vrednost CustID sekvence.

Red u tabelu CUSTOMER korišćenjem sekvence se može uneti na sledeći način:

INSERT INTO CUSTOMER (CustomerID, Name, AreaCode, PhoneNumber) VALUES (CustID.NextVal, 'Mary Jones', '350', '555-1234');

Red za tabelu CUSTOMER će biti kreiran sa sledećom vrednošću u sekvenci kao vrednosti za CustomerID.

Pošto se naredba izvrši, može se naći red koji je upravo kreiran korišćenjem metoda CurrVal na sledeći način:

*SELECT * FROM CUSTOMER WHERE CustomerID = CustID.CurrVal;*

Ovde CustID.CurrVal vraća tekuću vrednost sekvence koja predstavlja upravo iskorišćenu vrednost.

NAREDBA CREATE INDEX

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

Naredbom CREATE INDEX se kreiraju indeksi kako bi se:

1. naglasila jedinstvenost kolona,
2. olakšalo sortiranje i
3. omogućilo brže pretraživanje po vrednostima nekih kolona.

Kolone koje se često koriste u WHERE klauzuli predstavljaju dobre kandidate za kreiranje indeksa. 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:

Primer 1: *SELECT * FROM TABELA WHERE Kolona1 = 100; i*

Primer 2: *SELECT * FROM TABELA1, TABELA2 WHERE TABELA1.Kolona1 = TABELA2.Kolona2;*

Ako se naredbe kao što su ove često izvršavaju, Kolona1 i Kolona2 su dobri kandidadti za indeks.
Primer 3: *Sledećom naredbom se može kreirati indeks nad kolonom Name u tabeli CUSTOMER.*
CREATE INDEX KupacImeIdx ON KUPAC (ime);

Indeks je nazvan KupacImeIdx. 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.
Primer 4: *D bi se obezbedilo da se jedan 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);

Poglavlje 4

Izmena tabele (ALTER TABLE)

KADA SE KORISTI NAREDBA ALTER TABLE ?

Ukoliko želimo da promenimo strukturu postojeće tabele

Naredba ALTER TABLE se koristi ukoliko želimo da promenimo strukturu postojeće tabele.

Može se koristiti za:

1. dodavanje, izbacivanje ili menjanje kolona kada se koriste klauzule ADD, DROP, ALTER respektivno iza kojih stoji ime kolone
2. dodavanje i izbacivanje ograničenja nad tabelama kada se koriste klauzule ADD CONSTRAINT i DROP CONSTRAINT respektivno iza kojih stoji ime ograničenja

Primer 1: *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 ko ja 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 2: Ukoliko hoćemo da izmenimo definiciju postojeće kolone tabela (dužinu, tip, not null), naredba je:
ALTER TABLE KUPAC ALTER COLUMN StaraKolona date NOT NULL;
Primer 3: Ukoliko hoćemo da izbrišemo postojeću kolonu tabele, naredbe je:
ALTER TABLE KUPAC DROP COLUMN StaraKolona;

Primer 4: Ukoliko želimo da dodamo ograničenje:
ALTER TABLE KUPAC ADD CONSTRAINT NovoOgraničenje CHECK ([Ime] NOT IN ('Robert no pay'));

Primer 5: Ukoliko želimo da izbrišemo ograničenje:
ALTER TABLE KUPAC DROP CONSTRAINT StaroOgraničenje;

Poglavlje 5

Brisanje tabele (DROP TABLE)

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 imeTabele;
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 os taje 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 ima ograničenje FOREIGN KEY. Brisanje se može izvršiti samo ako njen primarni ključ nije spušten ni u jednu drugu tabelu, ili ako je naznačeno kaskadno brisanje (DELETE CASCADE). Takođe, prilikom brisanja takve tabele se može najpre izbrisati ograničenje FOREIGN KEY ili tabela u kojoj je spušten ključ.

NAREDBE DROP TABLE I ALTER TABLE

Na primeru baze podataka za galeriju slika

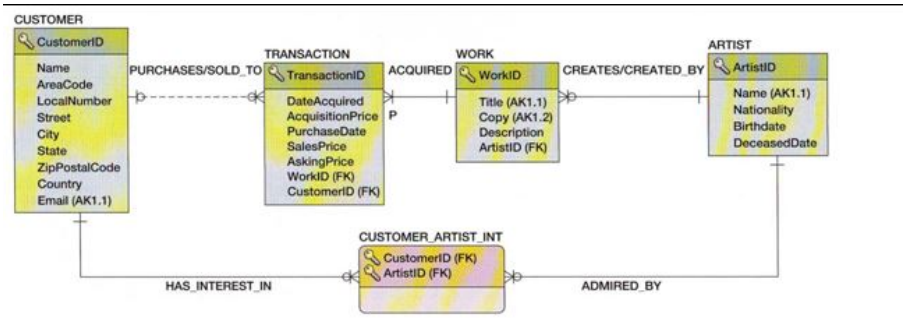
Primer: *Na slici 5. 1 je prikazana baza podataka za jednu galeriju slika.*

Dva moguća načina za brisanje tabele CUSTOMER iz baze podataka galerije slika su data u nastavku.

Najpre je *prikazano brisanje tabele CUSTOMER tako što su prethodno izbrisane tabele CUSTOMER_ARTIST_INT i TRANSACTION koje su imale strani ključ spušten iz tabele CUSTUMER.*

Zatim je *prikazano brisanje tabele CUSTUMER tako što su prethodno izbrisani strani ključevi u CUSTOMER_ARTIST_INT i TRANSACTION dobijeni spuštanjem primarnog ključa iz tabele CUSTOMER.*

Važno je naglasiti da se brisanje tabele može izvršiti samo ako njen primarni ključ nije spušten ni u jednu drugu tabelu.



Slika 5.1 Dizajn baze podataka za galeriju slika [Izvor: NM IT350-2020/2021.]

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

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

KADA MOŽEMO KORISTITI NAREDBU TRUNCATE TABLE?

Ukoliko hoćemo da obrišemo sve podatke koje tabela sadrži, ali da ne obrišemo i definiciju tabele

Kada hoćemo da obrišemo sve podatke jedne tabele (sav sadržaj) iz baze podataka, ali ne i da obrišemo samu definiciju tabele (kako bi je kasnije ponovo koristili), koristi se naredba TRUNCATE TABLE;
Opšti oblik naredbe je:
TRUNCATE TABLE imeTabele;
Primer: *Izbaciti sve podatke tabele PREMIJA iz baze podataka.*
Rešenje: *TRUNCATE TABLE PREMIJA;*

1. **Razlika između** DROP TABLE**i TRUNCATE TABLE**: razlika između naredbi DROP TABLE i TRUNCATE TABLE se može jasno uočiti. **Naredbom DROP TABLE iz baze podataka se briše kompletan sadržaj tabele, ali i sama definicija date tabele u bazi podataka. Nakon izvršenja ove naredbe tabela više nije dostupna za rad. Naredbom**

TRUNCATE TABLE iz tabele se brišu svi podaci, ali sama definicija tabele ostaje nepromenjena. Nakon izvršenja ove naredbe, tabela je i dalje dostupna za rad i u nju se mogu upisivati novi podaci.

2. **Razlika između DELETE FROM i TRUNCATE** TABLE: Takođe, naredbu TRUNCATE možemo uporediti i sa naredbom DELETE.

Naredba TRUNCATE TABLE imeTabele je logički (ali ne i fizički) ekvivalentna naredbi DELETE FROM imeTabele (bez WHERE klauzule). WHERE klauzulu ne možemo koristiti uz TRUNCATE, što bi značilo da sa TRUNCATE brišemo ili sve ili ništa. TRUNCATE je DDL naredba, dok je DELETE DML naredba, što bi značilo da TRUNCATE, u cilju da brže obavi operaciju, zaobilazi pokretanje okidača i celokupno izvršenje restrikcija. Sama naredba TRUNCATE TABLE se ne može koristiti kada postoje strani ključevi koji se referenciraju na podatke u tabeli. Pošto naredba TRUNCATE TABLE ne pokreće trigger-e (okidače), to može dovesti do nekonzistentnosti podataka, jer se okidači ON DELETE (ON UPDATE) neće pokrenuti. Zbog toga podaci obrisani na ovaj način ne mogu se povratiti pomoću rollback operacije.

Poglavlje 6

Dodavanje redova tabele (INSERT INTO TABLE)

ČEMU SLUŽI NAREDBA INSERT INTO TABLE?

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

Naredba INSERT INTO TABLE se koristi za dodavanje redova tabele.

Standardna verzija naredbe INSERT je:

INSERT INTO ime_tabele (imena kolona u koje se unose podaci) VALUES(lista podataka)

Na primer:

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:

1. **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, 7.04.2023. sa platom od 45000 i bez prava na premiju. Neposredni rukovodilac još nije poznat:*
INSERT INTO RADNIK VALUES (3545, 'BOJAN', 'PRAVNIK', NULL .'07-APR-23', 45000, 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

2. 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 atribute 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');

3. **Prepisivanje podataka iz jedne tabele u drugu (ukoliko obe tabele imaju isti broj atributa i ukoliko su atributi identično definisani)**

*INSERT INTO tabela1 SELECT * FROM tabela2 ;*
Međutim, ukoliko želimo da prepišemo podatke iz jedne tabele u drugu, a obe tabele nemaju isti broj atributa i atributi nisu identično definisani treba koristiti sledeći oblik naredbe INSERT INTO:

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')*;

Poglavlje 7

Menjanje sadržaja tabele (UPDATE TABLE)

ČEMU SLUŽI NAREDBA UPDATE?

Za menjanje sadržaja tabele

Naredba UPDATE TABLE služi za menjanje postojećeg sadržaja tabele. **Opšti oblik naredbe UPDATE je:**

UPDATE tabela SET kolona1=izraz1 [,kolona2=izraz2] [WHERE kriterijum selekcije];

ili:

UPDATE tabela SET(kolona1, kolona2, ...) = (pod upit) [WHERE kriterijum selekcije]

Pod upit mora vratiti najviše po jednu vrednost za svaku od kolona u listi kolona iza SET klauzule.
Primer 1: *Novo primljeni pravnik Bojan imaće predsednika za neposrednog rukovodioca. Realizovati tu odluku.*
UPDATE RADNIK SET RUKOV=3539 WHERE IME='BOJAN';
Primer 2: *Srđan počinje sa radom 19.09.22 sa ličnim dohotkom od 350000 i ima neposrednog rukovodioca sa šifrom 3266. Realizovati te informacije.*
UPDATE RADNIK SET RUKOV=3266, DATZAP='19-SEP-22', LD=350000 WHERE IME = 'SRDJAN';
Primer 3: *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 1: *Pretpostavimo da postoji tabela TAX_TABLE (Tax, City) gde je Tax odgovarajuća vrednost poreza na nivou jednog grada (City). Takođe 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 2: *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 naredba:*

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

Poglavlje 8

Brisanje sadržaja tabele (DELETE FROM TABLE)

ČEMU SLUŽI NAREDBA DELETE

Za brisanje sadržaja dela ili cele tabele

Naredba DELETE TABLE služi za uklanjanje zapisa iz tabela navedenih u odrebdi FROM koji zadovoljavaju odredbu WHERE.

Opšti oblik naredbe je:

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, korišćenjem klauzule DROP, i zbrisaće se i struktura cele tabele.

DELETE briše samo podatke. Struktura, atributi, indeksi ostaće netaknuti.
Pretpostavimo da želimo kreirati dve tabele - tabelu "Student" i tabelu "Ocena", gde će tabela "Ocena" sadržati ocene koje su dodeljene studentima. Studenti će biti identifikovani preko ID-a, a ocene će biti vezane za odgovarajućeg studenta preko stranog ključa "student_id".

```
CREATE TABLE Student (
    id INT PRIMARY KEY,
    ime VARCHAR(50) NOT NULL,
    prezime VARCHAR(50) NOT NULL,
    indeks VARCHAR(10) UNIQUE,
    datum_upisa DATE
);

CREATE TABLE Ocena (
    id INT PRIMARY KEY,
    student_id INT,
    predmet VARCHAR(100) NOT NULL,
    ocena INT CHECK (ocena >= 5 AND ocena <= 10),
    FOREIGN KEY (student_id)
        REFERENCES Student(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Evo primera kako bismo mogli uneti neke podatke u ove tabele:

```
INSERT INTO Student (id, ime, prezime, indeks, datum_upisa)
VALUES (1, 'Marko', 'Marković', '12345', '2023-07-15');

INSERT INTO Student (id, ime, prezime, indeks, datum_upisa)
VALUES (2, 'Jovan', 'Jovanović', '23456', '2023-07-20');

INSERT INTO Ocena (id, student_id, predmet, ocena)
VALUES (1, 1, 'Matematika', 9);

INSERT INTO Ocena (id, student_id, predmet, ocena)
VALUES (2, 1, 'Programiranje', 8);

INSERT INTO Ocena (id, student_id, predmet, ocena)
VALUES (3, 2, 'Fizika', 7);
```

Da bismo prikazali primenu naredbe DELETE na ove tabele, recimo da želimo obrisati studenta sa ID vrednošću 1 iz tabele "Student". Kada obrišemo ovog studenta, kaskadno brisanje (CASCADE) će automatski obrisati sve ocene koje su bile vezane za ovog studenta iz tabele "Ocena".

```
-- Prikaz trenutnog sadržaja tabela pre brisanja
SELECT * FROM Student;
SELECT * FROM Ocena;

-- Brisanje studenta sa ID vrednošću 1 iz tabele "Student"
DELETE FROM Student WHERE id = 1;

-- Prikaz sadržaja tabela nakon brisanja
SELECT * FROM Student;
SELECT * FROM Ocena;
```

Poglavlje 9

Slučaj korišćenja: Modifikacija baze o filmovima

ŠEMA BAZE PODATAKA O FILMOVIMA

Šema baze podataka sadrži relacije koje se odnose na filmove, glumačke zvezde koje igraju u tim filmovima, producetima filmova i studijima u kojima se filmovi snimaju.

```
Movies(
    title:string,
    year:integer,
    length:integer,
    genre:string,
    studioName:string,
    producerC# :integer
 )
 MovieStar(
       name:string,
       address:string,
       gendre:char,
       birthdate:date
 )
 StarsIn(
     movieTitle:string,
     movieYear:integer,
     starName:string
 )
 MovieExec(
     name:string,
     address:string,
     cert# :integer
     netWorth:integer
 )
 Studio(
     name:string,
     address:string,
     presC# :integer
 )
```

Filmovi (Movies): Relaciju čine primarni ključ title i year i atributi length i genere. Preostala dva atributa su sekundarni ključevi studioName koji govori kom studiju pripada film i producerC koji je spušteni ključ iz relacije MovieExec

Zvezda filma (MovieStar): Ova relacija nam govori o glumcima. Primarni ključ ove relacije je name

IgrajuU (StarsIn): Ova relacija povezuje glumce sa filmovima u kojima se pojavljuju. Može se primetiti da se sastoji od kolona title i year koje predstavljaju primarne ključeve tabele Movie i StarName koji je primarni ključ tabele MovieStar. Na kraju se može primetiti da su potrebna sva tri atributa kako bi se formirao ključ

Producent (MovieExec)*:* Relacija koja govori o producentima filmova. Sadrži polja name, adress i networth koje predstavlja njegova primanja. Za ovaj primer je kreiran primarni ključ cert# koji predstavlja broj licence i producente povezuje sa studijom u kom su direktori.
Studio: Ova relacija nam daje informaciju o studijima. Oslanjamo se na to da ne postoje dva studija sa istim imenom pa je atribut name primarni ključ. Ostali atributi su adresse u kojem se čuva adresa studija i cert# koji predstavlja spušteni ključ tabele MovieExec.

MODIFIKACIJA BAZE PODATAKA O FILMOVIMA

Modifikacija se vrši primenom naredbi INSERT, DELETE i UPDATE

U ovom primeru ćemo se fokusirati na tri grupe naredbi koje nam omogućavaju:

1. Unos torki u relaciju
2. Brisanje nekih torki iz relacije
3. Ažuriranje vrednosti nekih kolona nekih od postojećih torki.

Ove tipove operacija ćemo nazvati **operacije modifikacije.**
1. **Unos (INSERT):** Osnovna forma naredbe za unos je:
INSERT INTO R ($A_1, \ldots, A_n$) VALUES ($u_1, \ldots, u_n$);

Torka je kreirana korišćenjem vrednosti atributa A_i, for i = 1,2 ,,.. , n. Ukoliko lista atributa ne uključuje sve atribute relacije R, tada će kreirane torke imati default vrednosti za nedostajuće atribute.

Primer 1: *Pretpostavimo da želimo da unesemo Sydney Greenstreet u listu glumačkih zvezda za film The Maltese Falcon.*
Kako su svi atributi StarsIN spomenuti u liniji (1), nema potrebe za
korišćenjem default vrednosti. Redosled vrednosti u liniji (2) se podudaraju sa redosledom atributima u liniji (1) tako da će *The Maltese Falcon* postati vrednost atributa *movieTitle* i tako dalje.

```
INSERT INTO Starsln(movieTitle, movieYear, starName)
VALUES('The Maltese Falcon', 1942, 'Sydney Greenstreet');
```

U ovom slučaju kada navodimo vrednosti svih atributa u odgovarajućem redosledu, možemo izostaviti listu atributa pa će naredba izgledati:

```
INSERT INTO Starsln
VALUES('The Maltese Falcon', 1942, 'Sydney Greenstreet');
```

MODIFIKACIJA BAZE PODATAKA O FILMOVIMA - INSERT, DELETE

Primeri naredbi INSERT i DELETE nad bazom podataka o filmovima

Primer 2: *Pretpostavimo da relaciji*

Studio(name, address, presC#) želimo da dodamo studija svih filmova koji su spomenuti u relaciji

Movies (title, year, length , genre, studioName, producerC#) a ne postoje u relaciji Studio. Kako ne postoji način da odredimo adresu ili predsednika za te studije, za vrednosti atributa address i presC# ćemo pretpostaviti da imaju NULL vrednost. ***Naredba INSERT za ovaj slučaj izgleda:***

```
INSERT INTO Studio(name)
    SELECT DISTINCT studioName
        FROM Movies
    WHERE studioName NOT IN
    (SELECT name FROM Studio );
```

Kao i mnoge SQL naredbe sa ugnježdenim upitima, prethodnu INSERT naredbu je najlakše analizirati od kraja: Linije (4) i (5) generišu sva imena studija iz relacije Studio. U liniji (2) se testira da li imena studija već postoje u relaciji Movies. Vidimo da linije (2) do (5) proizvode skup imena studija iz Movies koja nisu u Studio. Korišćenje klauzule DISTINCT u liniji (2) obezbeđuje da se svaki studio u tom skupu pojavljuje samo jedanput. Na kraju linija (1) unosi svako ime studija sa NULL vrednostima za atribute address i presC#, u relaciju Studio.
2. ***Forma naredbe za brisanje je:***
DELETE FROM R WHERE <condition> ;

Efekat izvršenja ove naredbe je da će svaka torka koja zadovoljava uslov biti izbrisana iz relacije R.

Primer 3: *Želimo da iz relacije*
Starsln (movie Title , movieYear, starName)
izbrišemo činjenicu da je Sydney Greenstreet bila zvezda u filmu The Maltese Falcon. To možemo uraditi naredbom:

```
DELETE FROM Starsln
    WHERE movieTitle = 'The Maltese Falcon ' AND
movieYear = 1942 AND
starName = 'Sydney Greenstreet' ;
```

Napomenimo da nasuprot naredbi za unos u prethodnom primeru, mi ne možemo jednostavno specificirati torku koju želimo da obrišemo, već da torku **moramo eksplicitno specificirati WHERE klauzulom.**

MODIFIKACIJA BAZE PODATAKA O FILMOVIMA - DELETE, UPDATE

Primeri naredbi DELETE i UPDATE nad bazom poataka o filmovima

Primer 4: *Ovoga puta, želimo da iz relacije*
MovieExec(name, address, cert# , netWorth) obrišemo nekoliko torki istovremeno korišćenjem uslova koji može biti zadovoljen od strane više od jedne torke. Naredba:
DELETE FROM MovieExec
WHERE netWorth < 10000000;

briše sve producente čiji je net worth manji od milion dolara.

3. Mada o unošenju i brisanju torki u bazu podataka možemo razmišljati kao i ažuriranju baze podataka, naredba UPDATE u SQL-u je veoma specifična vrste izmene baze podataka: jednoj ili viši torki koje već postoje u bazi podataka se mogu promeniti vrednosti nekih atributa.
Generalna forma naredbe za ažuriranje je
UPDATE R SET <new-value assignments> WHERE <condition>;

Svaka novododeljena vrednost (eng. new-value assignments) je atribut, znak jednakosti ili izraz. Ukoliko je potrebno izvršiti više od jednog dodeljivanja, ona se razdvajaju kolonama. Efekat ove naredbe je da se pronađu sve torke u R koje zadovoljavaju uslov.

Svaka od tih torki se menja dodeljivanjem vrednosti odgovarajućem atributu relacije sračunatom na osnovu izraza sadržanog u novo dodeljena vrednost.

Primer 5: *Želimo da ažuriramo relaciju*
MovieExec(name, ad d ress, c e rt# , netWorth)
dodavanjem naziva Pres. na početku imena svakog producenta koji je predsednik studija. Uslov koji torke treba da zadovolje je da se broj sertifikata producenta cert# pojavi u atributu presC# torki relacije Studio. Naredba ažuriranja glasi.

```
UPDATE MovieExec
SET name = ’Pres. ’ I I name
WHERE cert# IN (SELECT presC# FROM Studio );
```

Linijom (3) se testira da li se certificate number iz torke MovieExec podudara sa nekom od vrednošću koja se pojavljuje kao certificate number predsednika u STUDIO.
Linija (2) vrši ažuriranje selektovanih torki. Operator I I označava konkatenaciju stringova tako da se znakom = u liniji (2) stavljaju karakteri Pres. i blank na početku stare vrednosti atributa name u toj torci.

Poglavlje 10

Pokazna vežba

NAČIN ORGANIZACIJE POKAZNIH VEŽBI

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata.

1. U uvodnom delu pokaznih vežbi se daje pokazni primer koji studentima treba da pomogne u samostalnom rešavanju zadataka.
2. Zadatke koji su zadati za samostalni rad student samostalno rešava uz pomoć asistenta.

10.1 Pokazni primer

ČEMU SLUŽI PHPMYADMIN? -(3 MIN.)

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

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 tabelam a i poljima, izvršava SQL upite, rukovodi ključevima nad poljima. Kreirao ga je Tobias Račiler (Tobias Ratschiller) koji je na njemu započeo rad 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 ili XAMPP. Osim toga može se preuzeti i sa adrese http://www.phpmyadmin.net/home_page/index.php ali s obzirom da su PHP fajlovi u pitanju potrebno je da postoji 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 je port 80 zauzet (Skype zauzima port 80, i samim tim potrebno je isključiti skype ili u opcijama podesiti da se koristi drugi port)

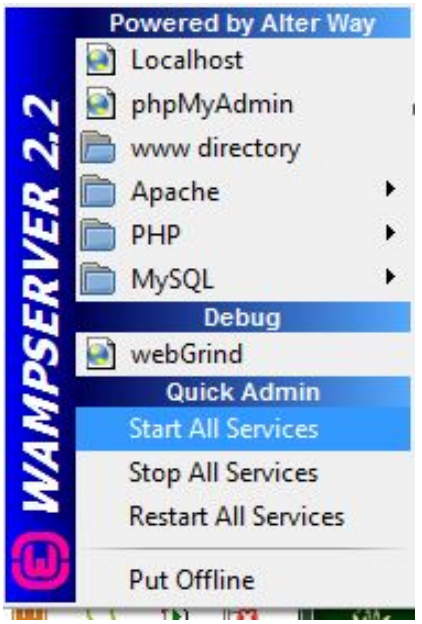
2. Proveriti da li su instalirani još neki serveri (phpDEV, XAMPP etc). Preporučuje se samo korišćenje jednog od navedenih, jer će Apache server biti zauzet).

POKRETANJE PHPMYADMIN - (4 MIN.)

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

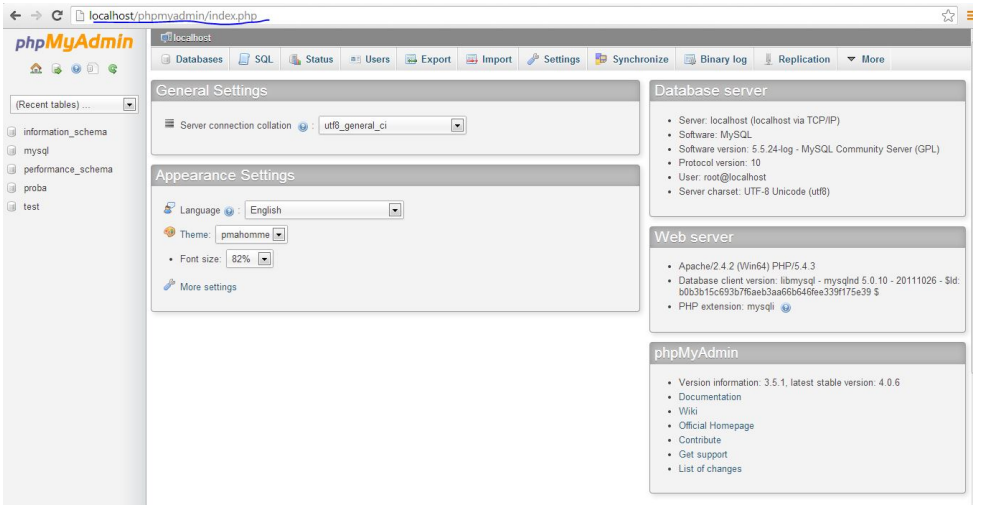
Nakon pokretanja *WAMP-a* očekujemo da se u donjem desnom uglu pojavi ikonica *WAMP* koja nakon pokretanja koja mora biti zelene boje. Ona može biti narandžaste ili crvene što znači da postoji neki problem (može biti problem u Apache, MySQL ili PHP).

Prikaz pokretanja celokupnog WAMP servera dat je na slici 11.1.



Slika 10.1.1 Pokretanje phpMyAdmin [Izvor: NM IT350-2020/2021.]

Nakon pokretanja servisa WAMP potrebno je u web čitaču kucati sledeće: localhost/ phpmyadmin. Nakon toga ukoliko je servis pokrenut, ukoliko nije bilo nikakvog problema pokrenuće se aplikacija.



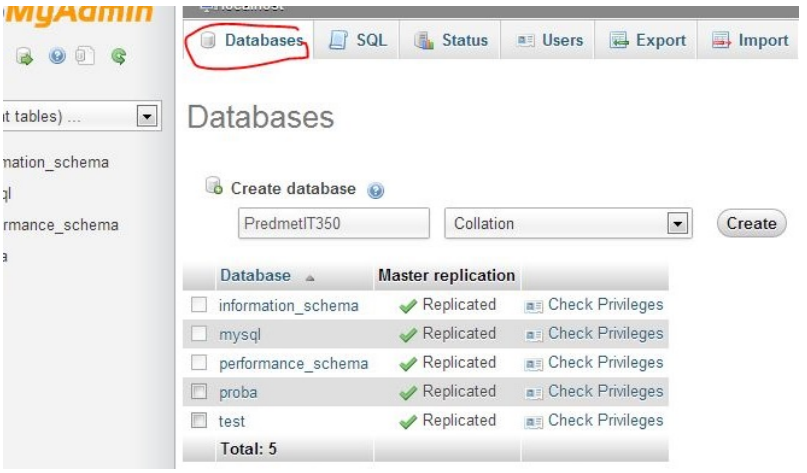
Slika 10.1.2 Meniji koji se nalaze u phpMyAdmin-u [Izvor: NM IT350-2020/2021.]

Na slici 11.2 su prikazani meniji koji se nalaze u phpMyAdmin-u i to za: kreiranje i upravljanje privilegijama baza (Databases), za pisanje SQL upita (SQL), statusne podatke (Status), korisnike sa svojim privilegijama (Users), izvoz-eksportovanje SQL koda (Export), ubacivanje-importovanje SQL koda (Import), opšta podešavanja (Settings), sinhronizaciju (Synchronize) itd.. Postoje još neki meniji o koji trenutno nisu od značaja.

KREIRANJE BAZE - (4 MIN.)

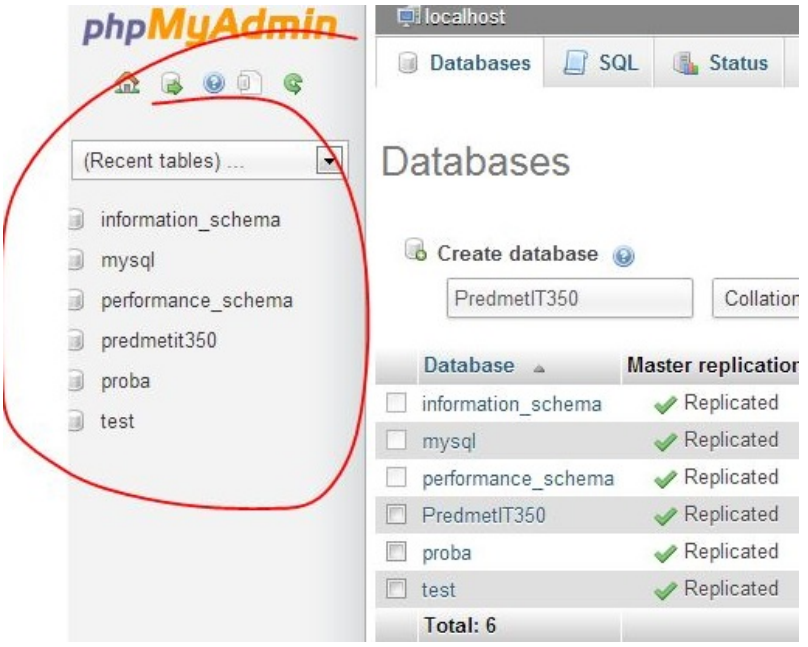
Imena svih kreiranih baza se mogu videti u levom delu phpMyAdmin-a

Potrebno je kliknuti na tab Databases, a zatim uneti ime tabele kao što je prikazano na slici 11. 3. Za Encoding koji biramo kako bi bila prikazana ćirilična slova i specifična slova naše latinice potrebno je izabrati UTF8-Encoding



Slika 10.1.3 Kreiranje baze [Izvor: NM IT350-2020/2021.]

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 11. 4)



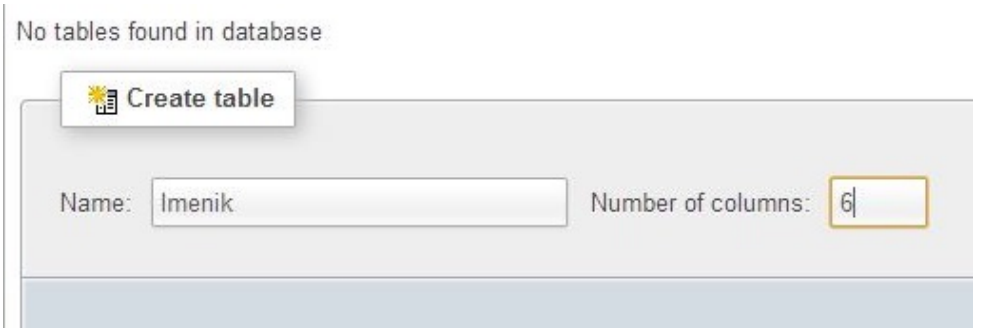
Slika 10.1.4 Pregled imena svih kreiranih baza [Izvor: NM IT350-2020/2021.]

KREIRANJE TABELA - (6 MIN.)

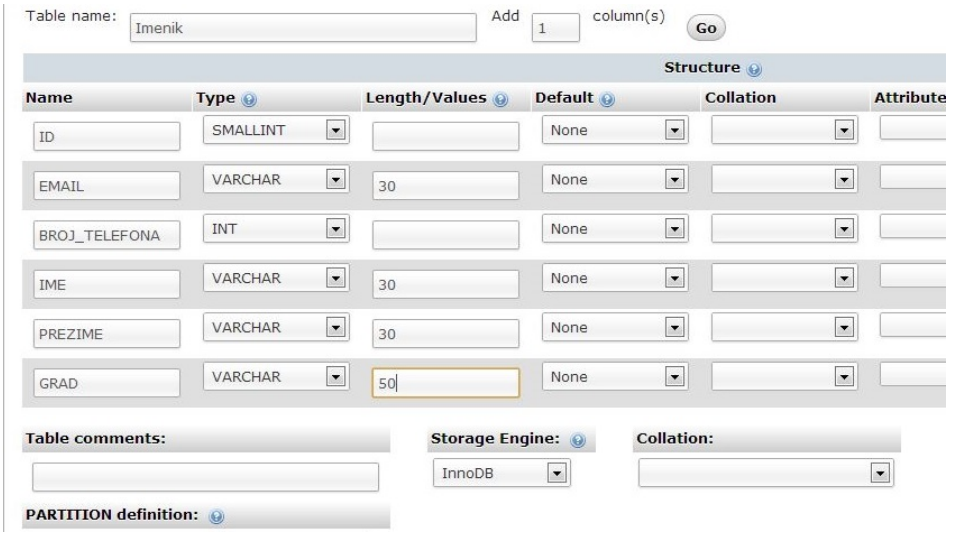
Prilikom kreiranja tabela potrebno je dodati imena kolona, tipove podataka kolona, specificirati njihovu duzinu i podrazumevanu vrednost

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 kolona (slika 11.5)

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 11.6) Više o tome može se pročitati na adresi: http://www.homeandlearn.co.uk/php/php12p3.html.



Slika 10.1.5 Kreiranje tabele IMENIK [Izvor: NM IT350-2020/2021.]

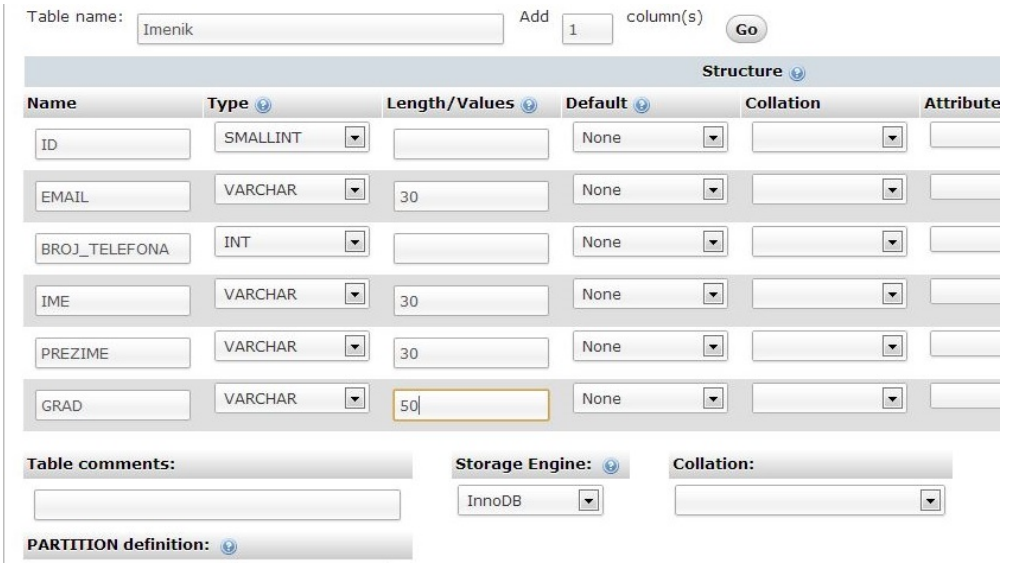


Slika 10.1.6 Tipovi podataka koji se koriste pri kreiranju tabela [Izvor: NM IT350-2020/2021.]

PRIKAZ KREIRANE TABELE - (3 MIN.)

Prikaz je moguć na način prikazan na slici

Nakon toga kreirana tabela se mo že pogledati kao što je prikazano na slici 11. 7.



Slika 10.1.7 Prikaz kreirane tabele [Izvor: NM IT350-2020/2021.]

UNOS SQL NAREDBI U PHPMYADMIN - (7 MIN.)

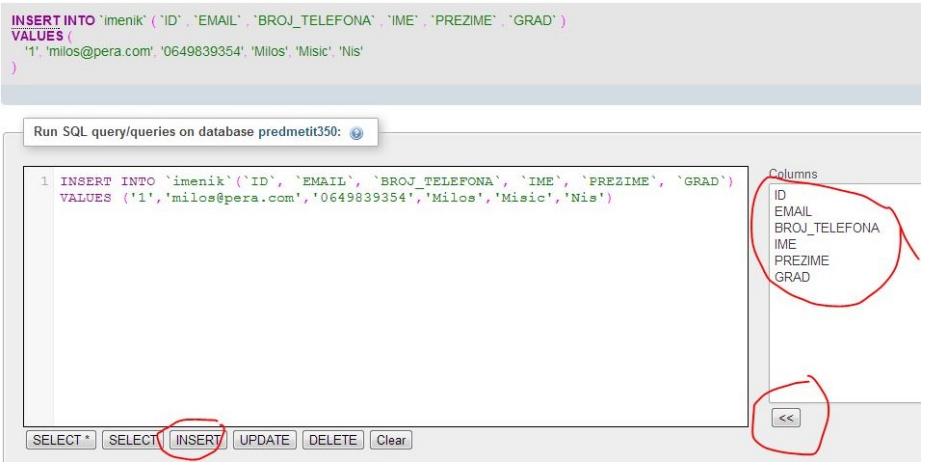
Unos SQL naredbi je olakšan jer editor omogućava dobijanje standardnog oblika naredbe koju želi da unese

Ukoliko želimo da izvršimo neke SQL naredbe potrebno je kliknuti na tab SQL kada je unos naredbi:
SELECT, INSERT, UPDATE, DELETE dosta lakši.

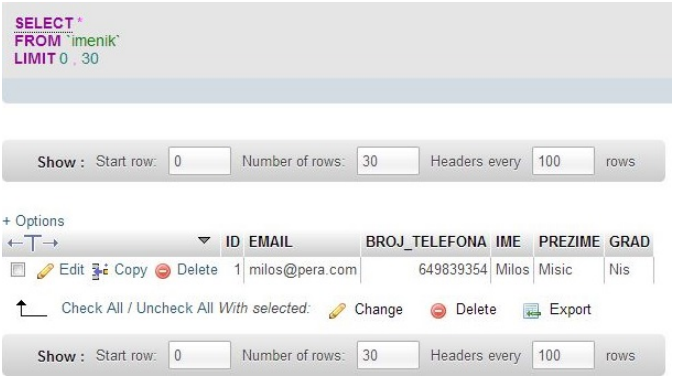
Nakon što se odabere baza sa kojom se radi, sa desne strane se ispisuju tabele sa kojom se radi kao i kolone nad kojima se mogu vršiti upiti.
Ukoliko želimo da koristimo INSERT naredbu, možemo samo kliknuti na tab INSERT i dobićemo kod:

INSERT INTO imenik (ID , EMAIL , BROJ_TELEFONA , IME , PREZIME , GRAD`) VALUES ([value-1],[value-2],[value-3],[value-4],[value-5],[value-6]). (slika 11.8)

Ukoliko se odabere samo određena kolona, unos se vrši samo u njoj. To se može izvesti odbariom imena te kolone. Ovim je olakšano kucanje osnovnih upita (slika 11.9)



Slika 10.1.8 Kucanje naredbe INSERT. Izvor:[NM IT350-2020/2021.]



Slika 10.1.9 Kucanje naredbe SELECT. Izvor:[NM IT350-2020/2021.]

KREIRANJE TABELE (CREATE TABLE) - PRIMER 1. - (6 MIN.)

Primer naredbe za kreiranje tabele IMENIK sa primarnim ključem

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

Osnovni format CREATE TABLE:

CREATE TABLE <ime_tabele> (<lista_deklaracija_kolona>[,<lista_deklaracija_organičenja_tabele>]);

NAPOMENA:Da bismo postavili neki atribut kao primarni ključ to radimo deklaracijom za ograničenja u tabeli, na kraju deklaracije kolona:

Primer: CONSTRAINT ImenikPK PRIMARY KEY (ID)

Primer 1: Kreiranje tabele Imenik:

S obzirom da je kreirana tabela prazna, tj nema inicijalne podatke još uvek, ona će izgledati kao na slici 10 .

ID	EMAIL	BROJ_TELEFONA	IME	PREZIME	GRAD

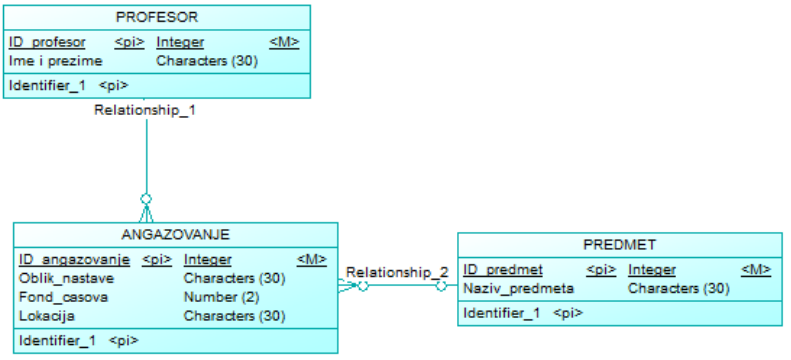
Slika 10.1.10 Kreirana tabela IMENIK koja je na početku prazna

```
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) );
```

KREIRANJE TABELE (CREATE TABLE) - PRIMER 2. - (7 MIN.)

Za dijagram predstavljen na slici napisati naredbe za kreiranje tabela

Primer 2: Za dijagram predstavljen na slici 11.11 napisati naredbe za kreiranje tabela



Slika 10.1.11 Dijagram na osnovu kojeg treba kreirati tabele baze podataka [Izvor: NM IT3250-2020/2021.]

REŠENJE ZADATKA 2:

1. Kreiranje tabele PROFESOR:

```
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));
```

NAPOMENA: CONSTRAINT ProfesorUN UNIQUE (Ime i prezime)

nam omogućava da kombinacija IME - PREZIME bude jedinstvena u tabeli

2. Kreiranje tabele PREDMET:

```
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));
```

NAPOMENA:Klauzulama

CONSTRAINT PredmetPK PRIMARY KEY (ID_prdmet),
CONSTRAINT PredmetUN UNIQUE (Naziv_predmeta));
u tabeli PREDMET određujemo primarni ključ ID_PREDMETA i postavljamo da je polje Naziv_predmeta jedinstveno, što znači da neće postojati dva predmeta sa istim imenom

KREIRANJE TABELE (CREATE TABLE) - PRIMER 2. NASTAVAK - (4 MIN.)

Naredbe za kreiranje tabela predmet i angažovanje

3. Kreiranje tabele ANGAZOVANJE

```
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 FOREING KEY REFERENCES (ID_profesor)

ON DELETE NO ACTION, ON UPDATE NO ACTION,

CONSTRAINT PredmetFK FOREING KEY REFERENCES (ID_predmet)

ON DELETE NO ACTION, ON UPDATE NO ACTION);
```

NAPOMENA:

Naredbom CONSTRAINT ProfesorFK FOREING KEY REFERENCES (ID_profesor) definišemo strane ključeve

IZMENA I BRISANJE TABELE (ALTER I DROP TABLE) - (5 MIN.)

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

Tabeli IMENIK sa kolonoma (ID, EMAIL, BROJ_TELEFONA,IME,PREZIME)- čija je struktura prikazana na slici 11.12. želimo da dodamo kolonu GRAD. NAREDBA za dodavanje kolone će glasiti:

ALTER TABLE Imenik ADD GRAD Char(20) NULL;

U tabeli IMENIK sa slike 12. dodaće se kolona GRAD čiji je tip podataka Char gde moze da se unesu 20 znakova (slika 11.13).

ID	EMAIL	BROJ_TELEFONA	IME	PREZIME

Slika 10.1.12 Tabela IMENIK pre izmene [Izvor: NM IT350-2020/2021.]

ID	EMAIL	BROJ_TELEFONA	IME	PREZIME	GRAD

Slika 10.1.13 Tabela IMENIK nakon izmene [Izvor: NM IT350-2020/2021.]

Ukoliko želimo da obrišemo tabelu IMENIK iz baze koristimo:

DROP TABLE IMENIK;

Ukoliko želimo da izbrišemo kolonu GRAD iz tabele IMENIK koristimo:

ALTER TABLE Imenik DROP COLUMN GRAD;

NAPOMENA 1:

Kada menjamo definiciju tabele, broj kolona koje ona ima, dodajemo nove ili uklanjamo postojeće kolone, menjamo osobine kolona ili dodajemo neka ograničenja koristimo ALTER TABLE naredbu.

NAPOMENA 2:

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

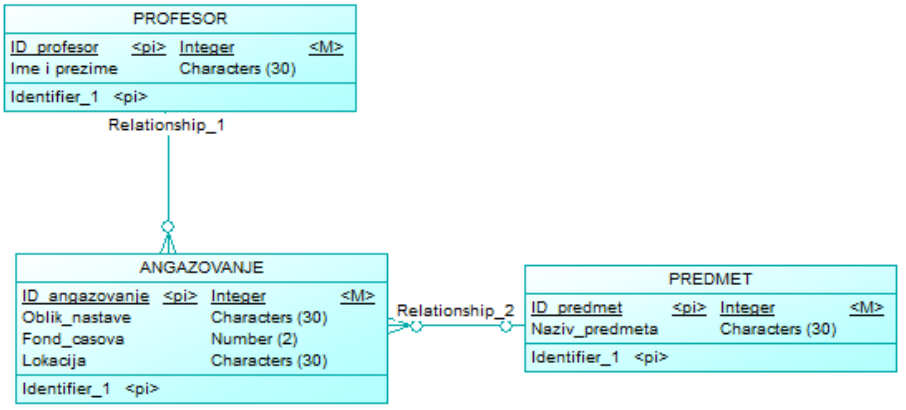
IZMENA I BRISANJE TABELE (ALTER I DROP TABLE) - PRIMER - (6 MIN.)

ALTER TABLE i DROP TABLE nad tabelam PROFESOR i PREDMET

Primer: Za model prikazan na slici 11.14 .

1. *Izbriši tabelu PROFESOR*
2. *Izbriši tabelu PREDMET*

To je moguće uraditi izvršenjem sledećeg skupa naredbi u definisanom redosledu:



Slika 10.1.14 Model baze nad kojim treba primeniti ALTER TABLE i DROP TABLE [Izvor: NM IT350-2020/2021.]

1. *ALTER TABLE ANGAZOVANJE DROP CONSTRAINT ProfesorFK FOREING KEY;*
2. *DROP TABLE PROFESOR;*
3. *ALTER TABLE ANGAZOVANJE DROP CONSTRAINT PredmetFK FOREING KEY;*
4. *DROP TABLE PREDMET;*

NAPOMENA:

Pre nego što obrišemo tabelu PROFESOR, potrebno je da iz tabela koje imaju strani ključ koji se odnosi na tu tabelu obrišemo ograničenja. U ovom slučaju, iz tabele ANGAZOVANJEuklanjamo strani ključ koji se referencira na tabelu PROFESOR, pa tek onda brišemo tabelu PROFESOR.

Analogno tome brišemo i tabelu PREDMET.

UNOŠENJE REDOVA TABELE (INSERT INTO) - (4 MIN.)

Naredba INSERT INTO dodaje zapise u tabelu.

Naredba INSERT INTO dodaje zapise u tabelu.

Ukoliko želimo da unesmo više zapisa u tabelu CILJ, prepisivanjem zapisa iz tabele IZVOR naredba INSERT izgleda:

```
INSERT INTO CILJ [(prvoPolje [, drugoPolje[, …]])]
[IN spoljnaBazaPodataka]
SELECT prvoPolje[, drugoPolje[, …]
FROM IZVOR;
```

Naredba za dodavanje jednog zapisa izgleda:

```
INSERT INTO CILJ [(prvoPolje[,drugoPolje[, …]])]
VALUES (prvaVrednost[, drugaVrednost[, …])
```

Naredba INSERT INTO koristi sledeće parametre:

1. CILJ- je ime tabele u koju treba da se dodaju zapisi;
2. prvoPolje, drugoPolje - polja u koja treba da se dodaju podaci ili imena polja iz kojih treba da se preuzmu podaci
3. spoljnaBazaPodataka - putanja do spoljne baze podataka
4. IZVOR - ime tabele iz koje treba da se kopiraju zapisi
5. prvaVrednost, drugaVrednost- vrednosti koje treba da se umetnu u određena polja u zapisu.

UNOŠENJE REDOVA TABELE (INSERT INTO) - PRIMERI 1-3 - (5 MIN.)

Naredbe INSERT nad tabelom Imenik

Primer 1: *Prvi način unošenja podataka u tabelu IMENIK je da se navedu imena kolona u koje ce se unositi vrednosti kao i same vrednosti kolona: Primer naredbe bi glasio ovako:*
INSERT INTO Imenik (EMAIL, BROJ_TELEFONA,IME,PREZIME,GRAD)
VALUES ('milos@pera.com' *, '064 98 39 354', 'Milos', 'Misić,', 'Niš');*

Primer 2: *Drugi način unošenja podataka u tabelu IMENIK je bez navođenja kolona u kojima će se unositi vrednosti kolona.*
INSERT INTO Imenik
VALUES ('milos@pera.com' *, '064 98 39 354', 'Milos', 'Misić,', 'Niš');*

U tabelu Imenik su ubačene navedene vrednosti i tabela više ni je prazna (slika 111.15).

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

Slika 10.1.15 Sadržaj tabele Imenik nakon izvršenja naredbe INSERT [Izvor: NM IT350-2020/2021.]

NAPOMENA:
Ukoliko se navode imena kolone, vrednosti će biti unošene u kolone tako što se prva vrednost unosi u prvu kolonu,
druga u drugu itd. Ako se navodi redosled kolona,
tada se podrazumeva da su vrednosti kolona navedene u istom redosledu kako su navedene i prilikom kreiranja tabele.

Ukoliko ne navedemo neku kolonu, vrednost za tu kolonu ce biti jednostavno izostavljena i nece se pojaviti kada se SELECT naredbom pozove, tačnije bice prazna.
Primer 3: *Necemo da unesemo mail (to je moguće samo ako je kolona mail kreirana kao NOT NULL)*
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 11.16 .

EMAIL	BROJ_TELEFONA	IME	PREZIME	GRAD
	064 98 39 354	Milos	Misić	Niš

Slika 10.1.16 Sadržaj tabele Imenik nakon izvršenja naredbe INSERT [Izvor: NM IT350-2020/2021.]

UNOŠENJE REDOVA TABELE (INSERT INTO) - PRIMERI 4-6 - (5 MIN.)

Naredbe INSERT nad tabelom IMENIK i RADNIK

Primer 4: *Ukoliko se navedu određene kolone, a ne navedu se podaci za te kolone, javiće se greška.*
INSERT INTO IMENIK (EMAIL, BROJ_TELEFONA,IME,PREZIME,GRAD) VALUES ('milos@pera.com' *, '064 98 39 354', 'Milos', 'Misić', 'Beograd');*

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

Primer 5: *Unošenje 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);*

Primer 6: *Email, broj telefona, ime, prezime i grad iz tabele Imenik prepisati u tabelu RADNIK za sve stanovnike Beograda pri čemu tabela RA DNIK 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';
```

IZMENA REDOVA TABELE (UPDATE) - (5 MIN.)

Naredba UPDATE služi za menjanje vrednosti u poljima u određenoj tabeli na osnovu navedenih kriterijuma

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

UPDATE imeTabele SET novaVrednost WHERE kriterijumi;

Delovi UPDATE naredbe su:

1. imeTabele - tabela ko ja sadrži podatke koji se menjaju
2. novaVrednost - određuje vrednost koja treba da se unese u određeno polje u ažuriranim zapisima
3. 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. U koliko uzmemo primer tabele IMENIK sa slike 11.17 i primenimo naredbu

UPDATE ImenikSET IME='Petrija' WHERE ID=1;

dobite se sadržaj tabele prikazan na slici 11.18.

Promena podatka u koloni IME gde je ID 1. nakon UPDATE-a tabela

ID	EMAIL	BROJ_TELEFONA	IME	PREZIME	GRAD
1	milos@pera.com	064 98 39 354	Milos	Misić	Niš
2	krompirovazlatica@gmail.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 10.1.17 Tabela Imenik [Izvor: NM IT350-2020/2021.]

ID	EMAIL	BROJ_TELEFONA	IME	PREZIME	GRAD
1	milos@pera.com	064 98 39 354	Petrija	Misić	Niš
2	krompirovazlatica@gmail.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 10.1.18 Prikaz tabele Imenik posle UPDATE-a [Izvor: NM IT350-2020/2021.]

IZMENA REDOVA TABELE (UPDATE) - PRIMERI 1-3 - (7 MIN.)

Nad tabelam CUSTOMER i SUPPLIERS

Primer 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;
```

Primer 2: Ažuriranje više tabela:

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

Primer 3. Korišćenje klauzule exist u naredbi UPDATE

```
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);
```

Prethodna 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;
```

POKAZNI PRIMER 1: - (4 MIN.)

Primer 1. naredbi CREATE, INSERT i UPDATE

```
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) );
```

2. KORAK: dodajemo podatke u tabelu SUPPLIERS

```
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');
```

```
UPDATE suppliers

SET city = 'Santa Clara'

WHERE supplier_name = 'NVIDIA';
```

POKAZNI PRIMER 2: - (7 MIN.)

Primer 2. naredbi CREATE, INSERT i UPDATE

Na osnovu tabela SUPPLIERS koja sadrži kolone supplier_id, supplier_name i city i CUSTOMERS koja sadrži kolone customer_id, customer_name, city), ažuriraj city u tabeli su SUPPLIERS sa city u tabeli CUSTOMERS, kada se supplier_name u tabeli SUPPLIERS podudara sa customer_name u tabeli CUSTOMERS .

1. KORAK - kreiramo tabelu SUPPLIERS i dodajemo podatke

```
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');
```

2. KORAK - kreiramo tabelu CUSTOMERS i dodajemo podatke

```
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');
```

3. KORAK - ažuriramo podatke u skladu sa zahtevima

```
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);
```

10.2 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENATA - 75 MIN.

Zadaci za samostalno vežbanje i utvrdjivanje gradiva

Uraditi sledeće zadatke:

1. Kreirati bazu podataka PODACI. **(10 minuta)**
2. Kreirati tabelu COUNTRY sa kolonama country_id, country_name i region_id. **(8 minuta)**

3. Postaviti country_id kao primarni ključ za tabelu COUNTRY. **(5 minuta)**
4. Postaviti polje country_name da bude jedinstveno. **(5 minuta)**
5. Dodati u tabeli COUNTRY kolonu country_note što predstavlja skraćenizu za državu (SRB,IT, UK ...) **(10 minuta)**
6. Napuniti tabelu podacima. **(15 minuta)**
7. Ažurirati podatke za Srbiju, tako da svuda bude region Balkan. **(7 minuta)**
8. Obrisati unos koji se odnosi na Italiju. **(5 minuta)**
9. Isprazniti tabelu. **(5 minuta)**
10. Obrisati tabelu. **(5 minuta)**

VIDEO I

SQL Data Definition Language

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO II

SQL: Data Manipulation Language INSERT / UPDATE / SELECT / DELETE

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 11

Domaći zadatak

DOMAĆI ZADATAK 8 - VREME IZRADE 90 MIN.

Zadatak 8 treba uraditi nad različitim specifikacijama baza podataka

BAZA PODATAKA 1: Svrha baze podataka za evidenciju zdravstvenih kartona pacijenata jeste da se prate istorije bolesti i terapija pacijenata, kao i rad lekara neke zdravstvene ustanove. Zadatak rešite praćenjem instrukcije koje su navedene. Baza se sastoji od sledećih tabela:
Pacijent (Ime , prezime, JMBG, adresa, telefon, *Doktor.Ime*)
Doktor (Ime, prezime, JMBG, Uža specijanost)
Medikament (Lek) (Naziv, Proizvođač, šifra, *Proizvodjac.naziv*)
Bolest (Naziv, Opis, Slika)
Proizvođač (naziv, adresa, telefon)
BolujeOd (*Pacijent.Ime, Bolest.Naziv*, datum_dijgnoze))

BAZA PODATAKA 2: Svrha ove baze podataka je evidentiranje iznajmljenih knjiga u nekoj biblioteci (koja knjiga je kom korisniku biblioteke izdata od strane kog zaposlenog lica). Baza se sastoji od sledećih tabela:
Zaposleni (Ime , prezime, JMBG, adresa, telefon, *Odeljenje.Ime*)
Korisnik (Ime , prezime, JMBG, adresa, telefon)
Odeljenje (Ime, Mesto)
Knjiga (Naziv, Opis, Slika, *Autor.ime*)
Autor (Ime , prezime, JMBG, adresa, telefon)
Iznajmljivanje (*Korisnik.ime, Knjiga.Naziv, Zaposleni.Ime*, datum_izd)

BAZA PODATAKA 3: Cilj ove baze podataka je evidencija podataka o nastavnicima, studentima i predmetima na jednom univerzitetu. Potrebno je znati koji nastavnik je angažovan na kom predmetu u datoj školskoj godini i koji student je polagao koji predmet. Baza se sastoji od sledećih tabela:
Student (Ime, prezime, JMBG, adresa, telefon, slika, *Smer.Ime*)
Smer (Ime)
Profesor (Ime, prezime, JMBG, adresa, telefon, slika, *Smer.Ime*)
Predmet (Naziv, fond_casova_predavanja, fond_casova_vežbi)
Predaje (*Profesor.Ime, Predmet.Naziv*, skolska_god)
Ispit (*Profesor.ime, Predmet.naziv, Student.Ime*, skolska_god, ocena)

BAZA PODATAKA 4: Cilj ove baze podataka je evidencija rezervisanih soba u jednom hotelu. Potrebno je znati koji gosti su izvršili rezervaciju soba u kom periodu kao i ko je izvršio pripremu soba za izdavanje. Baza se sastoji od sledećih tabela:

Zaposleni (Ime , prezime, JMBG, adresa, telefon, *Odeljenje.Ime*)
Gost (Ime , prezime, JMBG, adresa, telefon)
Odeljenje (Ime, Mesto)
Soba (Broj, Opis, Slika)
RezervacijaSobe (*Gost.ime, Soba.Broj,* datum_od, datum_do)
PripremaSobe (*Soba.broj, Zaposleni.Ime*, datum)

DOMAĆI ZADATAK 8 - NAČIN IZBORA ZADATKA

Izbor baze podataka 1-5 izvršiti na sledeći način

BAZA PODATAKA 5: Svrha ove baze podataka je evidencija proizvoda koji se mogu prodavati u različitim gradovima i zemljama različitim kupcima. Pored mesta gde je prodat za proizvod je potrebno znati i materijale od kojih je proizveden i količinu materijala koja ulazi u proizvod. Baza se sastoji od sledećih tabela:
Kupac (Ime , prezime, JMBG, adresa, telefon, *Mesto.naziv*)
Mesto (Naziv, ZIP, Zemlja)
Materijal (Naziv, Opis, Slika)
Proizvod (Naziv, Opis, Slika)
Sastavnica (Proizvod.Naziv, Matrijal.Naziv, količina)
KupovinaProizvoda (*Proizvod.Naziv, Mesto.Naziv, Kupac.Ime*, količina, datum)

Raspodelu baza podataka vrši asistent tako što:

1. bazu podataka br. 1 dodeljuje studentima čiji se broj indeksa završava na 0 i 5.
2. bazu podataka br. 2 dodeljuje studentima čiji se broj indeksa završava na 1 i 6.
3. bazu podataka br. 3 dodeljuje studentima čiji se broj indeksa završava na 2 i 7.
4. bazu podataka br. 4 dodeljuje studentima čiji se broj indeksa završava na 3 i 8.
5. bazu podataka br. 5 dodeljuje studentima čiji se broj indeksa završava na 4 i 9.

Asistent ili profesor su u obavezi da studentima daju sve potrebne dodatne informacije vezano zabaze podatakakoji analiziraju pri rešavanju domaćih zadataka.

TEKST ZADATKA

Rešiti zadatak prema postavljenom tekstu

Za izabranu bazu podataka uraditi sledeće:

1. Napišite skripte za kreiranje ovih tabela korišćenjem naredbe CREATE
2. Izmenite tabele korišćenjem naredbe ALTER TABLE tako što ćete za svaku od njih:
   - definisati jedinstveni identifikator
   - dodati strane ključeve, tamo gde postoje (označeni sa italic u naziv_tabele.naziv:kolone), sa pratećim restrikcijama.
   - Za brojeve JMBG i broj telefona proveriti da li sadrže samo cifre.
   - Iz tabela koje sadrže atribut Slika, izbrisati taj atribut

3. Za svaku bazu podataka kreirati po izboru novu tabelu koja treba da sadrži strane ključeve iz postojeće dve tabele
4. U svaku od tabela uneti po nekoliko redova poštujući referencijalne integritete ove baze podataka.

Prilikom slanja domaćih zadatka, neophodno je da ispunite sledeće:

- Subject mail-a mora biti IT250-DZbr (u slučaju kada šaljete domaći za ovu nedelju to je IT250-DZ08)
- U prilogu mail-a treba da se nalazi arhiviran projekat koji se ocenjuje imenovan na sledeći način: IT250-DZbr-BrojIndeksa-Ime Prezime. Na primer, IT250-DZ08-1234-VeljkoGrkovic
- Telo mail-a treba da ima pozdravnu poruku
- Arhivu sa zadatkom poslati na adresu predmetnog asistenta: milica.vlajkovic@metropolitan.ac.rs (studenti u Beogradu i online studenti) ili tamara.ukadinovic@metropolitan.ac.rs (studenti u Nišu).

Svi poslati mail-ovi koji ne ispunjavaju navedene uslove NEĆE biti pregledani. Za sva pitanja ili nedoumice u vezi zadatka, možete se obratiti asistentu

ZAKLJUČAK

Šta smo naučili u ovoj lekciji?

U ovom predavanju se govori o dve grupe naredbi SQL-a:

1. naredbe za definisanje podataka (DDL) i
2. naredbe za manipulisanje podacima (DML).

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 Z. .

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

LITERATURA

Za pisanje ove lekcije je korišćena sledeća literatura:

1. C. J. Date, An introduction to Database Systems, Addison-Wesley Publishing Company, 1990
2. https://www.tutorialspoint.com/sql_certificate/using_ddl_statements.htm
3. http://ramkedem.com/en/oracle-dml-statements/