1. SQL TIPOVI PODATAKA

MySQL DBMS podržava veći broj tipova podataka koje možemo svrstati u tri osnovne kategorije:

- Numerički tipovi podataka
- Vremenski tipovi podataka
- Binarni i tekstualni tipovi podataka

MySQL podržava i proširenja za podršku podataka većeg obima. Takođe, osim tipa podataka skup vrednosti se može dodatno definisati i korišćenjem atributa koji mogu biti opšti (npr. da je unos vrednosti obavezan) i specifični za određeni tip podataka (npr. Signed/Unsigned za tip Integer).

1.1. Numerički tipovi podataka

MySql u potpunosti podržava SQL standard vezan za numeričke tipove podataka. U numeričke tipove podataka spadaju celobrojne vrednosti (INTEGER, SMALLINT, DECIMAL i NUMERIC) kao i aproksimativne vrednosti (FLOAT, REAL i DOUBLE PRECISION). Sinonim za INTEGER je INT dok je sinonim za DECIMAL DEC.

Tip	**Bajtova**	**Min. vrednost**	**Maks. vrednost**
		Signed/Unsigned	Signed/Unsigned
tinyint	1	-128	127
		0	255
smallint	2	-32768	32767
		0	65535
mediumint	3	-8388608	8388607
		0	16777215
int	4	-2147483648	2147483647
		0	4294967295
bigint	8	-9223372036854775808	92233720368547758
		0	18446744073709551 615

Tabela 4.1 – Integer tipovi podataka

Najbitniji atributi numeričkih tipova podataka su "Signed/Unsigned" i "Zerrofil". "Signed/Unsigned" atribut označava da li će se u poljima tog tipa čuvati samo pozitivne ili i pozitivne i negativne vrednosti. U zavisnosti od ovog parametra se određuje i donja/gornja granica unetih vrednosti (videti tabelu 2.1.1). Parametar "Zerofill" svih vrednostima dodaje nule do definisane dužine (npr. unos vrednosti "4" u polje tipa INT(5) će se odraziti kao vrednost "00004") i automatski postavlja i parametar "Unsigned".

Tipovi FLOAT i DOUBLE se koriste za predstavljanje približnih vrednosti. Preciznost je opcionalni parametear i ona za vrednosti od 0-23 podrazumeva tip FLOAT a za vrednosti od 24-53 podrazumeva tip DOUBLE. MySql automatski podržava zaokruživanje tako da će se vrednost *999.00009* uneta u polje definisano kao *ime_polja DOUBLE(5,4)* sačuvati kao *999.0001*.

Tipovi DECIMAL i NUMERIC se koriste za predstavljanje tačnih numeričkih vrednosti. Razlika između ovih tipova i tipova FLOAT i DOUBLE je u tome što se kod tipova DECIMAL i NUMERIC parametar preciznost zahteva kao statički (npr. *kurs_dinara DECIMAL(3,2)*).

1.2. Tipovi podataka vezani za datum i vreme

U tipove podataka namenjenih skladištenju vrednosti vezanih za vreme spadaju:

- DATETIME
- DATE
- TIME
- TIMESTAMP
- YEAR

Svaki od navedenih tipova podataka ima sopstveni skup validnih vrednosti kao i "nula" vrednost koja zamenjuje unešene vrednosti koje nisu validne. Tip TIMESTAMP poseduje automatsko ponašanje tj. automatski unosi vrednost sa sistemskog časovnika. U Tabeli 2.2.1 je prikazan format podataka vezanih za datum i vreme.

Tip podataka	**"nula" vrednost**
DATETIME	`'0000-00-00 00:00:00'`
DATE	`'0000-00-00'`
TIME	`'00:00:00'`
TIMESTAMP	`'0000-00-00 00:00:00'`
YEAR	`0000`

Tabela 4.2 – Format podataka vezanih za datum i vreme

Navedeni tipovi podataka vezani za datum i vreme nemaju specifične parametre osim tipova TIMESTAMP i YEAR koji mogu imati parametar "length".

Podrazumevana vrednost parametar "length" kod tipa TIMESTAMP je 14 a može biti i dodatno navedena:

- 14 (format YYYYMMDDHHMMSS)
- 12 (format YYYYMMDDHHMM)
- 10 (format YYYYMMDDHH)
- 8 (format YYYYMMDD)
- 6 (format YYYYMM)
- 4 (format YYYY)
- 2 (format YY)

Podrazumevana vrednost parametar "length" kod tipa YEAR je 4 a može biti i dodatno navedena kao vrednost 4 (format YYYY) ili 2 (format YY). U slučaju kada je parametar "length" postavljen na vrednost 4 kod YEAR tipa podataka, unešene vrednosti u opsegu 00-69 se konvertuju u 2000-2069 dok se vrednosti 70-99 konvertuju u 1970-1999.

1.3. Binarni i tekstualni tipovi podataka

U tipove podataka namenjenih skladištenju binarnih i tekstualnih vrednosti spadaju:

- CHAR i VARCHAR
- BINARY i VARBINARY
- BLOB i TEXT
- ENUM i SET

CHAR i VARCHAR tipovi podataka su namenjeni za skladištenje kraćih nizova karaktera. Oba tipa imaju parametar “length” (npr. ime CHAR(20), prezime VARCHAR(30)) koji kod CHAR tipa može biti numerička vrednost u opsegu od 0 do 256 a kod VARCHAR tipa od 0 do 65.536. Osim razlike u dužini stringa koji se skladišti, osnovna razlika između između ovih tipova je način na koji se podaci skladište u bazi. Naime, vrednost parametra “length” se kod tipa CHAR koristi statički tj. razlika između maksimalne dužine i unete dužine stringa se popunjava znakom razmaka. Kod VARCHAR tipa se dužina prilagođava unetoj dužini stringa.

BINARY i VARBINARY tipovi su veoma slični CHAR i VARCHAR tipovima sa tom razlikom što su BINARY i VARBINARY tipovi namenjeni skladištenju binarnih podataka. Ova dva tipa podataka su namenjena uglavnom za skladištenje manjih količina podataka. Za skladištenje fajlova u bazi podataka se radije koristi BLOB tip podataka.

BLOB tip podataka (Binary Large OBject) i njegove podvarijante (TINYBLOB, BLOB, MEDIUMBLOB, i LONGBLOB) su namenjene za skladištenje binarnih nizova. TEXT tip podataka i njegove podvarijante (TINYTEXT, TEXT, MEDIUMTEXT, i LONGTEXT) su namenjene za skladištenje nizova karaktera različite veličine. BLOB i TEXT tipovi podataka ne mogu imati "default" vrednosti.

ENUM tip podataka je namenjen za skladištenje tekstualnih vrednosti s tom razlikom da se vrednosti koje se unose moraju biti na listi vrednosti koja se kreira pri definisanju polja (npr. godina ENUM("prva", "druga", "treca")). Lista dozvoljenih elemenata kod ENUM tipa podataka je ograničena na maksimalno 65.536 elemenata. SET tip podataka je sličan ENUM tipu podataka s tom razlikom da je lista dozvoljenih elemenata ograničena na 64 elementa kao i da postoje određene razlike pri indeksiranju elemenata liste.

1.4. Zahtevi pri skladištenju

Numerički tipovi:

Tip podataka	**Zahtev pri skladištenju**
TINYINT	1 bajt
SMALLINT	2 bajta
MEDIUMINT	3 bajta
INT, INTEGER	4 bajta
BIGINT	8 bajtova
FLOAT(p)	4B za 0 <= *p* <= 24; 8B za 25 <= p <= 53
FLOAT	4 bajta
DOUBLE [PRECISION], REAL	8 bajta
DECIMAL(M,D), NUMERIC(M,*D*)	Varira; dato je naknadno objašnjenje
BIT(M)	približno (*M*+7)/8 bajtova

Tipovi podataka vezani za datum i vreme:

Tip podataka	**Zahtev pri skladištenju**
DATE	3 bajta
DATETIME	8 bajtova
TIMESTAMP	4 bajta
TIME	3 bajta
YEAR	1 bajt

Binarni i tekstualni tipovi podataka:

Tip podataka	**Zahtev pri skladištenju**
CHAR(M)	M bajtova, 0 <= M <= 255
VARCHAR(*M*)	L + 1 bajtova, gde je L <= *M* i 0 <= M <= 255 ili L + 2 bajtova, gde je *L* <= M i 256 <= *M* <= 65535.
BINARY(M)	M bajtova, 0 <= *M* <= 255
VARBINARY(*M*)	L + 1 bajtova, gde je L <= *M* i 0 <= *M* <= 255 ili *L* + 2 bajtova, gde je L <= M and 256 <= M <= 65535.
TINYBLOB, TINYTEXT	L+1 bajtova, gde je $L < 2^8$
BLOB, TEXT	L+2 bajtova, gde je $L < 2^{16}$
MEDIUMBLOB, MEDIUMTEXT	*L*+3 bajtova, gde je $L < 2^{24}$
LONGBLOB, LONGTEXT	L+4 bajtova, gde je $L < 2^{32}$
ENUM('*vrednost1*','*vrednost2*',...)	1 ili 2 bajta, u zavisnosti od broja članova
SET('*vrednost1*','*vrednost2*',...)	1, 2, 3, 4, ili 8 bajtova, u zavisnosti od broja članova

1.5. Odabir adekvatnog tipa podataka

Pri modeliranju baze podataka veoma je važno odabrati adekvatne tipove podataka tj. definisati tipove i atribute koji odgovaraju potrebama rada sa bazom podataka. U slučaju da kreirani model ne odgovara realnim potrebama u daljem radu se mogu javiti netačne vrednosti u bazi kao i nemogućnost korišćenja svih prednosti DBMS-a.

Primer 1: Polje *starost* (pod kojim podrazumevamo broj godina osobe) je u bazi moguće čuvati i u polju tipa *Integer* i u polju tipa *String*. Međutim, odabir tekstualnog tipa podatka onemogućava korišćenje upita tipa:

- prikaži sve osobe mlađe od 30 godina
- prikaži sve osobe stare između 20 i 50 godina
- ...

Primer 2: Polje *datum* (pod kojim planiramo čuvanje informacije o datumu i vremenu nekog događaja) je u bazi moguće čuvati i u polju tipa *Integer* i u polju tipa *DATETIME* ili *TIMESTAMP*. Odabir numeričkog ili tekstualnog tipa podatka umesto vremenskog onemogućava funkcija vezanih za vremenski tip podatka. Međutim, odabir *INTEGER* tipa u ovakvim situacijama nije redak slučaj jer se izgubljena funkcionalnost DMBS-a nadoknađuje korišćenjem nekog od programskih jezika koji imaju mogućnost rada sa pomenutim DBMS-om.

Takođe, pri odabiru podataka treba imati u vidu i zahteve za skladištenjem određenih tipova kao i brzinu rada.

1.6. Sinhronizacija tipova podataka sa drugim DBMS sistemima

Da bi se olakšalo importovanje modela rađenih za druge DBMS sisteme u MySql DBMS, MySql podržava sledeće mapiranje tipova:

Ostali DBMS	**MySql DBMS**
BOOL	TINYINT
BOOLEAN	TINYINT
CHAR VARYING(*M*)	VARCHAR(*M*)
DEC	DECIMAL
FIXED	DECIMAL
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT

Ostali DBMS	MySql DBMS
MIDDLEINT	MEDIUMINT
NUMERIC	DECIMAL

Tabela 4.3 - Mapiranje tipova za sinhronizaciju sa ostalim DBMS sistemima

U praksi ovo znači da je moguće iskoristiti strane definicije tipova (navedenih u levoj koloni) za kreiranje tabela s tim da će se one automatski prevesti u MySql tipove podataka (navedene u desnoj koloni).

1.7. Unos različitih tipova podataka

U zavisnosti od tipa podataka, različiti tipovi podataka imaju različit način unosa pomoću SUBP-a u bazu podataka.

- Numerički tipovi podataka
  Unose se na klasičan način, na primer:

```
INSERT INTO Artikl
    (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
    VALUES
    (123, 'Artikl br. 1', 199.99, 100, 'Germany');
```

U prethodnom primeru polja **Magacin_rb, Cena i Količina** su numeričkog tipa I unose se na klasičan način, bez ikakvih specijalnih znakova.

- Vremenski tipovi podataka
  Unose se na klasičan način, na primer:

```
INSERT INTO Magacin
    (Magacin_rb, Naziv, Cena, Datum)
    VALUES
    (12, 'Disketa', 59.99, 12/09/2005);
```

U prethodnom primeru polje **Datum** je vremenskog tipa.

- Binarni i tekstualni tipovi podataka

Unose se obavezno između jednostrukih znakova novoda, na primer:

```
INSERT INTO Artikl
    (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
    VALUES
    (123, 'Artikl br. 1', 199.99, 100, 'Germany');
```

U prethodnom primeru polja **Naziv i ZamljaPorekla** su tekstualnog tipa i unose se između jednostrukih znakova navoda.

2. DDL (Data Definition Language)

2.1. Naredbe za definisanje podataka

Omogućuju definisanje resursa relacione baze podataka. Pod resursima BP se najčešće podrazumevaju:

1. struktura BP,
2. tabele,
3. atributi,
4. tipovi podataka,
5. ograničenja,
6. pomoćni indeksi za direktan pristup itd.

Efikasan sistem za upravljanje bazom podataka SUBP-a treba da omogući da izvršimo sledeće operacije:

1. kreiranje baze podataka
2. kreiranje tebele baze podataka
3. kreiranje indeksa nad kombinacijom kolona tabele
4. kreiranje virtuelne tabele - "pogleda"
5. izmena definicije tabele
6. izmena pogleda u bazi podataka
7. promena imena tabeli u bazi podataka
8. brisanje tabele iz baze podataka
9. uklanjanje indeksa iz tabele
10. uklanjanje baze podataka

2.2. Kreiranje baze podataka

Funkcija kreiranja baze podataka putem SUBP-a se odnosi na kreiranje nove prazne baze sa svim potrebnim elementima da je posle toga moguć rad sa njom. Kreiranje baze podataka se u SQL jeziku vrši pomoću naredbe **CREATE DATABASE**.

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification [, create_specification] ...]
create_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

5.1 - Dokumentovana struktura naredbe CREATE DATABASE

Da bi se kreirala baza podataka u skladu sa strukturom naredbe potrebno je ukucati sledeće:

```
CREATE DATABASE PRODAJA;
```

Ovom naredbom kreirana je nova prazna baza podataka pod imenom **PRODAJA**. Sada je moguće koristiti ovu bazu podataka za dalji rad. Sve što je potrebno jeste izabrati bazu sa kojom želite da radite, samim kreiranjem baze ne podrazumeva se da vi automatski hoćete sa njom da nastavite da radite pa je zato potrebno ukucati sledeću naredbu:

```
USE PRODAJA;
```

Ovom naredbom izabrali smo bazu podataka pod imenom **PRODAJA** sa kojom želimo da radimo (ovo je analogno kada u nekom programu izaberemo opciju Open pa izaberemo određeni fajl sa kojim želimo da radimo).

2.3. Kreiranje tebele u bazi podataka

Funkcija kreiranja nove tabele u bazi podataka putem SUBP-a se odnosi na kreiranje nove prazne tabele u bazi podataka. Kreiranje nove tabele u bazi podataka se u SQL jeziku vrši pomoću naredbe **CREATE TABLE**.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [() LIKE old_tbl_name ()];
create_definition:
    column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
  | KEY [index_name] [index_type] (index_col_name,...)
  | INDEX [index_name] [index_type] (index_col_name,...)
  | [CONSTRAINT [symbol]] UNIQUE [INDEX]
        [index_name] [index_type] (index_col_name,...)
  | [FULLTEXT|SPATIAL] [INDEX] [index_name] (index_col_name,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY
        [index_name] (index_col_name,...) [reference_definition]
  | CHECK (expr)
column_definition:
    col_name type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
        [reference_definition]
type:
    TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  | INT[(length)] [UNSIGNED] [ZEROFILL]
  | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
  | NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
  | DATE
  | TIME
  | TIMESTAMP
  | DATETIME
  | CHAR(length) [BINARY | ASCII | UNICODE]
  | VARCHAR(length) [BINARY]
  | TINYBLOB
  | BLOB
  | MEDIUMBLOB
  | LONGBLOB
  | TINYTEXT
  | TEXT
  | MEDIUMTEXT
  | LONGTEXT
  | ENUM(value1,value2,value3,...)
  | SET(value1,value2,value3,...)
  | spatial_type
index_col_name:
    col_name [(length)] [ASC | DESC]
reference_definition:
```

```
    REFERENCES tbl_name [(index_col_name,...)]
               [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
               [ON DELETE reference_option]
               [ON UPDATE reference_option]
reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

5.2 - Dokumentovana struktura naredbe CREATE TABLE

Ukucajte sledeće primere i napravićete nekoliko tabela u bazi podataka **PRODAJA** koja je kreirana u prvom primeru.

```
CREATE TABLE Artikl (
  rb INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Magacin_rb INTEGER UNSIGNED NOT NULL,
  Naziv VARCHAR(255) NULL,
  Cena FLOAT NULL,
  Kolicina INTEGER UNSIGNED NOT NULL DEFAULT 0,
  ZemljaPorekla VARCHAR(50) NULL,
  PRIMARY KEY(rb),
  INDEX Artikl_FKIndex1(Magacin_rb)
);

CREATE TABLE Dobavljac (
  rb INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Naziv VARCHAR(255) NOT NULL,
  Adresa VARCHAR(255) NULL,
  Telefon VARCHAR(30) NULL,
  PRIMARY KEY(rb)
);


CREATE TABLE Dobavljac_Artikl (
  Dobavljac_rb INTEGER UNSIGNED NOT NULL,
  Artikl_rb INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(Dobavljac_rb, Artikl_rb),
  INDEX Dobavljac_has_Artikl_FKIndex1(Dobavljac_rb),
  INDEX Dobavljac_has_Artikl_FKIndex2(Artikl_rb)
);

CREATE TABLE Kupac (
  rb INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Artikl_rb INTEGER UNSIGNED NOT NULL,
  Ime VARCHAR(25) NOT NULL,
  Prezime VARCHAR(25) NOT NULL,
  Adresa VARCHAR(50) NOT NULL,
  Telefon VARCHAR(30) NOT NULL,
  email VARCHAR(30) NULL,
  PRIMARY KEY(rb),
  INDEX Kupac_FKIndex1(Artikl_rb)
);

CREATE TABLE Magacin (
  rb INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Adresa VARCHAR(50) NOT NULL,
  Telefon VARCHAR(30) NOT NULL,
  Kapacitet INTEGER UNSIGNED NOT NULL DEFAULT 0,
  PRIMARY KEY(rb)
);
```

U prethodnim primerima ukucali ste naredbe kojima je kreirano pet različitih tabela u bazi podataka **PRODAJA** koje će mo koristiti i u daljim primerima za vežbanje.

2.4. Kreiranje indeksa u bazi podataka

Funkcija kreiranja indeksa u tabeli u baze podataka putem SUBP-a se odnosi na kreiranje novog indeksa u tabeli u bazi podataka (iako se on uglavnom kreira pri samom kreiranju tabela baze podataka (pogledajte prethodne primere gde će te primetiti da se često na kraju navodi primarni ključ i ako postoje i ostali indeksi), koji će omogućiti lakše pretraživanje i ostale funkcije koje se dobijaju indeksiranjem nekog od polja u tabeli baze podataka. Kreiranje novog indeksa u tabele baze podataka se u SQL jeziku vrši pomoću naredbe **CREATE INDEX**.

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
    [USING index_type]
    ON tbl_name (index_col_name,...)
index_col_name:
    col_name [(length)] [ASC | DESC]
```

5.3 - Dokumentovana struktura naredbe CREATE INDEX

Kreiraćemo indeks u tabeli **Magacin** nad poljem **Telefon** (strukturu tabele pogledajte tamo gde je ona kreirana) sledećom naredbom:

```
CREATE INDEX K2 ON Magacin(Telefon);
```

2.5. Kreiranje virtuelne tabele - "POGLEDA"

Funkcija kreiranja pogleda (VIEW) u bazi podataka putem SUBP-a se odnosi na kreiranje pogleda tj. virtualne tabele koja se koristi u slučajevima kada se često izvršavana neki isti upit nad određenom tabelom ili više tabela u bazi podataka. Tada se kreira pogled tj. *view* koji se kasnije koristi u radu sa bazom podataka. Kreiranje pogleda bazi podataka se u SQL jeziku vrši pomoću naredbe **CREATE VIEW.**

```
CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

5.4 - Dokumentovana struktura naredbe CREATE VIEW

Kao što u gore navedenoj strukturi možete primetiti da ovde postoji select_statement što je u stvari `SELECT` naredba koja je posebno i detaljno objašnjena u četvrtom delu o **DML-u** a ovde neće biti više reči o tome (za više informacija pogledajte poglavlje 4.3 gde će te videti detaljno objašnjenje ove naredbe i svih njenih opcija.

Kreiraćemo pogled nad tabelom magacin (najjednostavnijim oblikom SELECT naredbe) koji će prikazivati sve podatke koji se nalaze u istoimenoj tabeli sledećom naredbom:

```
CREATE VIEW SPISAK AS SELECT * FROM MAGACIN;
```

2.6. Izmena definicije tabele

Funkcija izmene tabele u bazi podataka putem SUBP-a se odnosi na dodavanje, uklanjanje i modifikovanje kolona u tabeli baze podataka. Prethodno navedene operacije nad tabelom u bazi podataka se u SQL jeziku vrši pomoću jedinstvene naredbe **ALTER TABLE:**

```
ALTER [IGNORE] TABLE tbl_name
    alter_specification [, alter_specification] ...
alter_specification:
    ADD [COLUMN] column_definition [FIRST | AFTER col_name ]
  | ADD [COLUMN] (column_definition,...)
  | ADD INDEX [index_name] [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        PRIMARY KEY [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [index_name] [index_type] (index_col_name,...)
  | ADD [FULLTEXT|SPATIAL] [index_name] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
        [reference_definition]
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name column_definition
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] column_definition [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | table_options
```

5.5 - Dokumentovana struktura naredbe ALTER TABLE

U zavisnosti od toga šta želimo da promenimo u tabeli, koritićemo različite oblike prethodno navedene naredbe u strukturi koja je prikazana.

Za dodavanje nove kolone koristi se **ADD**:

```
ALTER TABLE Kupac
ADD ime_oca CHAR(25);
```

Za uklanjanje kolone iz tabele koristi se **DROP**:

```
ALTER TABLE Kupac
DROP ime_oca;
```

Za izmenu svojstava kolone iz tabele koristi se **MODIFY**:

```
ALTER TABLE Kupac
MODIFY email VARCHAR(40);
```

2.7. Izmena pogleda u bazi podataka

Funkcija izmene pogleda u bazi podataka putem SUBP-a se odnosi na izmenu pogleda koji je prethodno kreiran u bazi podataka. Izmena pogleda bazi podataka se u SQL jeziku vrši pomoću naredbe **ALTER VIEW.**

```
ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

5.6 - Dokumentovana struktura naredbe ALTER VIEW

Primenićemo gore navedenu naredbu da bi promenili pogled pod nazivom **SPISAK** tako da prikazuje sve iz tabele **Dobavljač** a ne **Magacin** kako je ranije bilo, u skladu sa prethodno navedenom strukturom potrebno je ukucati:

```
ALTER VIEW SPISAK AS SELECT * FROM Dobavljac;
```

2.8. Promena imena tabeli u bazi podataka

Funkcija promene imena tabele u bazi podataka putem SUBP-a se odnosi na promenu inema neke od tabela koja je prethodno kreirana u bazi podataka. Izmena imena tabele u bazi podataka se u SQL jeziku vrši pomoću naredbe **RENAME TABLE.**

```
RENAME TABLE tbl_name TO new_tbl_name
    [, tbl_name2 TO new_tbl_name2] ...
```

5.7 - Dokumentovana struktura naredbe RENAME TABLE

Iskoristićemo prethodno opisanu naredbu da bi promenili ime tabele Kupac u Klijent, u skladu sa prethodnom strukturom potrebno je ukucati:

```
RENAME TABLE Kupac TO Klijent;
```

2.9. Brisanje tabele iz baze podataka

Funkcija brisanja tabele iz baze podataka putem SUBP-a se odnosi na uklanjenje neke od tabela koja je prethodno kreirana u bazi podataka tako da ona više fizički ne postoji u bazi podataka.. Ovde je potrebno uočiti bitnu razliku između naredbe **DROP TABLE** i naredbe **DELETE FROM TABLE**. Naredbom **DROP TABLE** briše se tabela iz baze podataka (**napomena**: korišćenjem ove naredbe nad tabelom u kojoj se nalaze podaci doći će do gubljenja tih podataka) dok se naredbom **DELETE FROM TABLE** brišu se svi podaci iz tabele ali se prazna tabela i dalje čuva u bazi podataka tako da se kasnije ponovo može koristiti. Uklanjanje tabele u bazi podataka se u SQL jeziku vrši pomoću naredbe **DROP TABLE.**

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

5.8 - Dokumentovana struktura naredbe DROP TABLE

Korišćenjem prethodno navedene naredbe uklonićemo tabelu Klijent iz baze podataka, u skladu sa prethodno navedenost strukturom potrebno je ukucati:

```
DROP TABLE Klijent;
```

2.10. Uklanjanje indeksa iz tabele

Funkcija uklanjanja indeksa iz tabele u bazi podataka putem SUBP-a se odnosi na uklanjenje indeksa u nekoj od tabela koji je prethodno kreiran u bazi podataka. Uklanjanje indeksa u nekoj od tabele u bazi podataka se u SQL jeziku vrši pomoću naredbe **DROP INDEX.**

```
DROP INDEX index_name ON tbl_name
```

5.9 - Dokumentovana struktura naredbe DROP INDEX

Korišćenjem prethodne naredbe uklonićemo indeks pod nazivom K2, koji je kreiran nad tabelom Magacin, u skladu sa prethodno navedenom strukturom potrebno je ukucati:

```
DROP INDEX K2 ON Magacin;
```

2.11. Uklanje baze podataka

Funkcija uklanjanja baze podataka putem SUBP-a se odnosi na uklanjenje kompletne baze podataka koja je prethodno kreirana (Ovom naredbom uklanjamo čitavu bazu podataka i sve njene resurse (tabele, atribute, tipove podataka, ograničenja, indekse itd.)). Uklanjanje baze podataka se u SQL jeziku vrši pomoću naredbe **DROP DATABASE.**

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_namež
```

5.10 - Dokumentovana struktura naredbe DROP DATABASE

Korišćenjem prethodne naredbe obrisaćemo čitavu bazu podataka koju smo na početku kreirali (**PAŽNJA:** posle izvršenja ove naredbe baza će biti uništena i dalji rad neće biti moguć), u skladu sa gore navedenom strukturom potrebno je ukucati:

```
DROP DATABASE PRODAJA;
```

3. DML (Data Manipulation Language)

3.1. Manipulisanje podacima

Osnovni razlog korišćenja Sistema za Upravljanje Bazama Podataka je jednostavan i lak rad sa podacima koji se nalaze u bazi podataka. Pod manipulisanjem (rukovanjem) podacima se podrazumeva:

- Unos (dodavanje) podataka
- Pregled (korišćenje) podataka
- Izmena podataka
- Uklanjanje podataka

Efikasnost sistema za upravljanje bazama podataka (SUBP) se može meriti preko sledećih kriterijuma:

- jednostavnost upotrebe
- preciznost podataka
- održavanje integriteta podataka
- brzina rada (izvršavanja upita)
- podržana količina podataka u bazi
- specifične funkcije i mogućnosti

3.2. Unos podataka

Funkcija unosa podataka u bazu putem SUBP-a se odnosi na unos podataka u neku (jednu ili više) od tabela u bazi podataka. Unos podataka se u SQL jeziku vrši pomoću naredbe **INSERT**.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
      [INTO] tbl_name [(col_name,...)]
      VALUES ({expr | DEFAULT},...),(...),...
      [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

4.2.1 - Dokumentovana struktura naredbe INSERT

Da bi ste u tabelu **Artikl** uneli novi zapis, u skladu sa strukturom naredbe **INSERT**, to možete uraditi na sledeća dva načina:

```
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
      (123, 'Artikl br. 1', 199.99, 100, 'Germany');
```

ili

```
INSERT INTO Artikl
      VALUES
```

```
(1, 123, 'Artikl br. 1', 199.99, 100, 'Germany');
```

Postoje dve osnovne razlike između ove dve navedene varijante:

- U prvoj varijanti smo pre vrednosti koje želimo da unesemo u tabelu **Artikl** naveli kolone u koje želimo da smestimo te vrednosti dok u drugoj varijanti nismo eksplicitno navodili imena kolona već smo samo naveli vrednosti po redosledu u kom se kolone nalaze u tabeli.
- U drugoj varijanti smo zadali vrednost i za polje **rb** da bi smo ispoštovali broj i redosled kolona pošto ih nismo eksplicitno naveli. Međutim, na ovaj način smo izgubili funkcionalnost parametra **AUTO_INCREMENT** koji je postavljen na ovu kolonu.

Druga varijanta predstavlja kraći oblik naredbe koji nije poželjno koristiti iz sledećih razloga:

- Gubi se funkcionalnost parametra **AUTO_INCREMENT** i **DEFAULT**.
- Ukoliko naredbu u ovakvom obliku sačuvamo u nekoj internoj ili eksternoj proceduri, nakon izmene tabele (dodavanja/uklanjanja neke od kolona) ta procedura neće biti validna.
- Moraju se unositi vrednosti za sva polja i to u odgovarajućem redosledu.

Na prethodno opisan način uneti i sledeće podatke:

rb	Magacin_rb	Naziv	Cena	Kolicina	ZemljaPorekla
2	124	Artikl br. 2	299.99	102	Serbian
3	127	Artikl br. 3	190.00	170	USA
4	129	Artikl br. 4	599.05	340	Germany
5	132	Artikl br. 5	179.97	140	Switzerland
6	135	Artikl br. 6	397.00	390	France
7	137	Artikl br. 7	269.07	250	Dansk

To ćemo uraditi uz pomoć sledećih naredbi:

```
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
      (124, 'Artikl br. 2', 299.99, 103, 'Serbian');
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
```

```
      (127, 'Artikl br. 3', 190.00, 170, 'USA');
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
      (129, 'Artikl br. 4', 599.05, 340, 'Germany');
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
      (132, 'Artikl br. 5', 179.97, 140, 'Switzerland');
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
      (135, 'Artikl br. 6', 397.00, 390, 'France');
INSERT INTO Artikl
      (Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla)
      VALUES
      (137, 'Artikl br. 7', 269.07, 250, 'Dansk');
```

3.3. Pregled podataka

Funkcija pregleda podataka u bazi putem SUBP-a se odnosi na uzimanje podataka iz neke (jedne ili više) od tabela u bazi podataka. Pregled podataka se u SQL jeziku vrši pomoću naredbe **SELECT**.

```
[ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
      select_expr, ...
      [FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      [PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name'
      | INTO @var_name [, @var_name]]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

4.3.1 - Dokumentovana struktura naredbe SELECT

Da bi smo izvršili pregled svih podataka iz tabele **Artikl** iskoristićemo najosnovniji oblik naredbe **SELECT**:

```
SELECT * FROM Artikl;
```

Ovom naredbom smo preuzeli sve podatke (sve zapise sa svim kolonama) iz tabele **Artikl**.

```
+----+------------+--------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+----+------------+--------------+--------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 199.99 |      100 | Germany       |
|  2 |        124 | Artikl br. 2 | 299.99 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 |    190 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 599.05 |      340 | Germany       |
|  5 |        132 | Artikl br. 5 | 179.97 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 |    397 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 269.07 |      250 | Dansk         |
+----+------------+--------------+--------+----------+---------------+
7 rows in set (0.00 sec)
```

Identična naredba (tj. naredba sa identičnim rezultatom) u ovom slučaju bi bila:

```
SELECT rb, Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla FROM Artikl;
```

Iako su rezultati isti:

```
+----+------------+--------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+----+------------+--------------+--------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 199.99 |      100 | Germany       |
|  2 |        124 | Artikl br. 2 | 299.99 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 |    190 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 599.05 |      340 | Germany       |
|  5 |        132 | Artikl br. 5 | 179.97 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 |    397 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 269.07 |      250 | Dansk         |
+----+------------+--------------+--------+----------+---------------+
7 rows in set (0.00 sec)
```

razlika između ove i prethodne naredbe je u tome što smo u prethodnoj naredbi koristili simbol '*****' da bi smo označili sve kolone (sva polja zapisa). Ukoliko želimo da iz prikaza izbacimo kolonu **rb**, potrebno je samo da je izostavimo iz liste kolona:vom slučaju bi bila:

```
SELECT Magacin_rb, Naziv, Cena, Kolicina, ZemljaPorekla FROM Artikl;
```

Rezultat ove naredbe je isti kao i kod prethodne naredbe ali je izostavljen parametar **rb**:

```
+------------+--------------+--------+----------+---------------+
| Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+------------+--------------+--------+----------+---------------+
|        123 | Artikl br. 1 | 199.99 |      100 | Germany       |
|        124 | Artikl br. 2 | 299.99 |      102 | Srbija        |
|        127 | Artikl br. 3 |    190 |      170 | USA           |
```

```
|        127 | Artikl br. 4 | 599.05 |      340 | Germany       |
|        132 | Artikl br. 5 | 179.97 |      140 | Switzerland   |
|        135 | Artikl br. 6 |    397 |      390 | France        |
|        137 | Artikl br. 7 | 269.07 |      250 | Dansk         |
+------------+--------------+--------+----------+---------------+
7 rows in set (0.00 sec)
```

Rezultate možemo sortirati vertikalno i horizontalno. Vertikalno sortiranje se izvodi preko navođenja kolona u redosledu u kom želimo da se prikazuju parametri zapisa:

```
SELECT Kolicina, Cena, Naziv, Magacin_rb, ZemljaPorekla FROM Artikl;
```

U rezultat ove naredbe:

```
+----------+--------+--------------+------------+---------------+
| Kolicina | Cena   | Naziv        | Magacin_rb | ZemljaPorekla |
+----------+--------+--------------+------------+---------------+
|      100 | 199.99 | Artikl br. 1 |        123 | Germany       |
|      102 | 299.99 | Artikl br. 2 |        124 | Srbija        |
|      170 |    190 | Artikl br. 3 |        127 | USA           |
|      340 | 599.05 | Artikl br. 4 |        127 | Germany       |
|      140 | 179.97 | Artikl br. 5 |        132 | Switzerland   |
|      390 |    397 | Artikl br. 6 |        135 | France        |
|      250 | 269.07 | Artikl br. 7 |        137 | Dansk         |
+----------+--------+--------------+------------+---------------+
7 rows in set (0.00 sec)
```

ulaze svi zapisi vertikalno poređani po redosledu u kom su unošeni u tabelu ali je redosled njihovih parametara promenjen u **Kolicina, Cena, Naziv, Magacin_rb, ZemljaPorekla**.

Horizontalno sortiranje se vrši zadavanjem jedne ili više kolona na osnovu čijeg sadržaja će sortiranje biti obavljeno u zadatom smeru:

```
SELECT * FROM Artikl ORDER BY ZemljaPorekla ASC;
```

U rezultat ove naredbe:

```
+----+------------+--------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+----+------------+--------------+--------+----------+---------------+
|  7 |        137 | Artikl br. 7 | 269.07 |      250 | Dansk         |
|  6 |        135 | Artikl br. 6 |    397 |      390 | France        |
|  1 |        123 | Artikl br. 1 | 199.99 |      100 | Germany       |
|  4 |        127 | Artikl br. 4 | 599.05 |      340 | Germany       |
|  2 |        124 | Artikl br. 2 | 299.99 |      102 | Srbija        |
|  5 |        132 | Artikl br. 5 | 179.97 |      140 | Switzerland   |
|  3 |        127 | Artikl br. 3 |    190 |      170 | USA           |
+----+------------+--------------+--------+----------+---------------+
```

7 rows in set (0.00 sec)

ulaze svi zapisi sortirani po parametru **ZemljaPorekla** u rastućem redosledu. Ukoliko želimo da zapise poređamo u opadajućem redosledu parametar **ASC** ćemo zameniti sa **DESC**.

Osim vertikalnog ograničavanja rezultata (ograničavanja kolona/parametara zapisa koji ulaze u rezultat) SQL jezik podržava i horizontalno ograničavanje (ograničavanje zapisa koji ulaze u rezultat) preko sledećih parametara naredbe **SELECT**:

1. **WHERE**
2. **DISTINCT**
3. **LIMIT**
4. **GROUP BY / HAVING**

3.3.1. WHERE

Ova klauzula se koristi za zadavanje odredjenog uslova koji zapis mora da ispuni da bi ušao u rezultat:

```
SELECT * FROM Artikl
      WHERE rb=5;
```

Ovakvim korišćenje **WHERE** klauzule smo rezultat ograničili samo na zapise kod kojih je parametar **rb** (vrednost u koloni **rb**) jednak 1:

```
+----+------------+--------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+----+------------+--------------+--------+----------+---------------+
|  5 |        132 | Artikl br. 5 | 179.97 |      140 | Switzerland   |
+----+------------+--------------+--------+----------+---------------+
1 row in set (0.00 sec)
```

Osim poređenja po numeričkoj jednakosti moguće je koristiti i sledeće numeričke operatore:

=	`SELECT * FROM Artikl WHERE rb=1;`	Jednako
<>	`SELECT * FROM Artikl WHERE rb<>1;`	Nije jednako
<	`SELECT * FROM Artikl WHERE rb<1;`	Veće od
>	`SELECT * FROM Artikl WHERE rb>1;`	Manje od
<=	`SELECT * FROM Artikl WHERE rb<=1;`	Veće ili jednako
>=	`SELECT * FROM Artikl WHERE rb>=1;`	Manje ili jednako
BETWEEN	`SELECT * FROM Artikl WHERE rb BETWEEN 1 AND 5;`	Između dve vrednosti
LIKE	`SELECT * FROM Artikl WHERE Naziv LIKE 'Artik%';`	Simboličko poređenje
IN	`SELECT * FROM Artikl WHERE rb IN (1,3,5);`	Univerzalno poređenje

Dodatna objašnjenja:

- Simboličko poređenje

- Klauzula **LIKE** omogućuje pretraživanje na osnovu "UZORKA" odnosno dobijanje informacija i kada ne znamo potpun naziv (tj. vrednost) odredjenog atributa tipa character. Ona koristi dva specijalna karaktera ("%","_") sa sledećim značenjem:

  "%" predstavlja string od 0 ili više karaktera

  "_" predstavlja poziciju jednog karaktera.

  Ostali karakteri imaju ucbičajeno značenje.

Uslov u **WHERE** klauzuli navedenog upita kaže da IME treba da liči na uzorak naveden u jednostrukim navodnicima.

Primeri:

... gde se ime završava sa N.
WHERE IME LIKE '%N'

... gde je treći karakter imena R.
WHERE IME LIKE '__R%'

... gde je ime dugačko 5 karaktera
WHERE IME LIKE '_____'

... gde ime nije dugačko 5 karaktera.
WHERE IME NOT LIKE '_____ '

... gde je u imenu slovo G posle R. WHERE IME LIKE '%R%G%'

- Univerzalno poređenje

- **IN** koristimo da prikažemo zapise koje pripadaju skupu vrednosti (ovo je zamena za više **OR** klauzula)

Klauzula **WHERE** nije ograničena na postavljanje samo jednog uslova (što se može primetiti već kod parametra **IN**). Više uslova se može kominovati pomoću logičkih operatora **AND** i **OR**:

```
SELECT * FROM Artikl
   WHERE (
      (rb>1 AND rb<3)
      OR
      (rb>3 AND rb<5)
      AND
      Naziv LIKE 'Artik%');
```

Ovom naredbom smo dobili rezultat u koji su uključeni svi zapisi kod kojih parametar **rb** ima vrednost 2 ili 4 i kod kojih parametar **Naziv** počinje sa *Artik*.

```
+----+------------+-------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv       | Cena   | Kolicina | ZemljaPorekla |
+----+------------+-------------+--------+----------+---------------+
|  2 |        124 | Artikl br. 2 | 299.99 |      102 | Srbija        |
|  4 |        127 | Artikl br. 4 | 599.05 |      340 | Germany       |
+----+------------+-------------+--------+----------+---------------+
2 rows in set (0.02 sec)
```

3.3.2. DISTINCT

Ova klauzula se koristi za eliminisanje duplih zapisa iz prikaza:

```
SELECT DISTINCT ZemljaPorekla FROM Artikl;
```

Rezultat ove komande sadrži listu svih zemalja iz kojih artikli potiču ali bez ponavljanja.

```
+---------------+
| ZemljaPorekla |
+---------------+
| Germany       |
| Srbija        |
| USA           |
| Switzerland   |
| France        |
| Dansk         |
+---------------+
6 rows in set (0.00 sec)
```

3.3.3. LIMIT

Ova klauzula se koristi za ograničavanje broja zapisa koji ulaze u rezultat bez obzira da li su ispunili eventualni uslov. Parametar **LIMIT** može imati jedan ili dva naknadna parametra (prvi koji definiše koliko prvih zapisa treba preskočiti i drugi koji definiše koliko zapisa treba prikazati). Ukoliko se zada samo jedan parametar on se odnosi na broj zapisa koje treba prikazati počev od prvog zapisa.

Naredba sa jednim parametrom:

```
SELECT * FROM Artikl LIMIT 5;
```

vraća prvih 5 zapisa:

```
+----+------------+--------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+----+------------+--------------+--------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 199.99 |      100 | Germany       |
|  2 |        124 | Artikl br. 2 | 299.99 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 |    190 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 599.05 |      340 | Germany       |
|  5 |        132 | Artikl br. 5 | 179.97 |      140 | Switzerland   |
+----+------------+--------------+--------+----------+---------------+
5 rows in set (0.00 sec)
```

dok ista naredba sa dva parametra:

```
SELECT * FROM Artikl LIMIT 5,10;
```

vraća zapise od 6. do 15.:

```
+----+------------+--------------+--------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena   | Kolicina | ZemljaPorekla |
+----+------------+--------------+--------+----------+---------------+
|  6 |        135 | Artikl br. 6 |    397 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 269.07 |      250 | Dansk         |
+----+------------+--------------+--------+----------+---------------+
2 rows in set (0.00 sec)
```

3.3.4. GROUP BY / HAVING

Klauzula **GROUP BY** se koristi za grupisanje podataka koji se dobijaju kao rezultat upita, a klauzula **HAVING** se koristi za selekciju određenih grupa koje su već kreirane GROUP BY klauzulom.

```
SELECT ZemljaPorekla, COUNT(*) FROM Artikl
GROUP BY ZemljaPorekla;
```

Korišćenjem naredbe u ovakvom obliku kao rezultat dobićemo spisak svih zemalja i broj artikala koji potiče iz koje zemlje.

```
+--------------+----------+
| ZemljaPorekla | COUNT(*) |
+--------------+----------+
| Dansk         |        1 |
| France        |        1 |
| Germany       |        2 |
| Srbija        |        1 |
| Switzerland   |        1 |
| USA           |        1 |
+--------------+----------+
6 rows in set (0.02 sec)
```

Ukoliko želimo da iz rezultata prikažemo samo podatke koji ispunjavaju određeni kriterijum, to ćemo uraditi uz pomoć klauzule **HAVING,** na sledeći način:

```
SELECT ZemljaPorekla, COUNT(*) FROM Artikl
GROUP BY ZemljaPorekla
HAVING COUNT(*)=1;
```

Na ovaj način kao rezultat dobićemo spisak zemalja iz kojih dolazi samo po jedan proizvod.

```
+--------------+----------+
| ZemljaPorekla | COUNT(*) |
+--------------+----------+
| Dansk         |        1 |
| France        |        1 |
| Srbija        |        1 |
| Switzerland   |        1 |
| USA           |        1 |
+--------------+----------+
5 rows in set (0.02 sec)
```

3.4. Izmena podataka

Funkcija izmene podataka u bazi putem SUBP-a se odnosi na izmenu podataka u nekoj (jednoj ili više) od tabela u bazi podataka. Moguće je izmeniti jedan ili više parametara (kolona) jednog ili više zapisa (reda). Izmena podataka se u SQL jeziku vrši pomoću naredbe **UPDATE**.

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

4.4.1 - Dokumentovana struktura naredbe UPDATE za jednu tabelu

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_condition]
```

4.4.2 - Dokumentovana struktura naredbe UPDATE za više tabela

Osnovni oblik komande **UPDATE** izgleda ovako:

```
UPDATE Artikl
      SET Cena=1000.00;
```

i njome su je vrednost svih vrednosti u koloni **Cena** postavljena na 1000.00.

Ukoliko želimo da izmenu primenimo samo na određene zapise (redove) moramo koristiti klauzulu **WHERE** (*videti deo 4.3.1*):

```
UPDATE Artikl
      SET Cena=1000.00
      WHERE ZemljaPorekla='Germany';
```

```
+----+------------+--------------+------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena | Kolicina | ZemljaPorekla |
+----+------------+--------------+------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 1000 |      100 | Germany       |
|  2 |        124 | Artikl br. 2 | 1000 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 | 1000 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 1000 |      340 | Germany       |
|  5 |        132 | Artikl br. 5 | 1000 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 | 1000 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 1000 |      250 | Dansk         |
+----+------------+--------------+------+----------+---------------+
7 rows in set (0.00 sec)
```

Ukoliko želimo da više parametara (kolona) izmenimo jednim upitom, navešćemo više kolona i vrednosti odvojenih zarezom:

```
UPDATE Artikl
      SET Cena=2000.00, Kolicina=46
      WHERE ZemljaPorekla='Germany';
```

```
+----+------------+--------------+------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena | Kolicina | ZemljaPorekla |
+----+------------+--------------+------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 2000 |       46 | Germany       |
|  2 |        124 | Artikl br. 2 | 1000 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 | 1000 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 2000 |       46 | Germany       |
|  5 |        132 | Artikl br. 5 | 1000 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 | 1000 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 1000 |      250 | Dansk         |
+----+------------+--------------+------+----------+---------------+
7 rows in set (0.00 sec)
```

Ukoliko želimo da u za vrednost koristimo već postojeću vrednost u nekoj od kolona za vrednost ćemo navesti kolonu zapisa čiju vrednost želimo da primenimo:

```
UPDATE Artikl
      SET Kolicina=Cena
      WHERE ZemljaPorekla='Germany';
```

```
+----+------------+--------------+------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena | Kolicina | ZemljaPorekla |
+----+------------+--------------+------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 2000 |     2000 | Germany       |
```

```
|  2 |        124 | Artikl br. 2 | 1000 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 | 1000 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 2000 |     2000 | Germany       |
|  5 |        132 | Artikl br. 5 | 1000 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 | 1000 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 1000 |      250 | Dansk         |
+----+------------+--------------+------+----------+---------------+
7 rows in set (0.00 sec)
```

Ukoliko želimo da u za vrednost koristimo već postojeću vrednost uz primenu matematičke operacije:

```
UPDATE Artikl
       SET Cena=Cena*1.18;
+----+------------+--------------+------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena | Kolicina | ZemljaPorekla |
+----+------------+--------------+------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 2360 |     2000 | Germany       |
|  2 |        124 | Artikl br. 2 | 1180 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 | 1180 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 2360 |     2000 | Germany       |
|  5 |        132 | Artikl br. 5 | 1180 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 | 1180 |      390 | France        |
|  7 |        137 | Artikl br. 7 | 1180 |      250 | Dansk         |
+----+------------+--------------+------+----------+---------------+
7 rows in set (0.00 sec)
```

Ili

```
UPDATE Artikl
       SET Cena=Kolicina+10;
+----+------------+--------------+------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena | Kolicina | ZemljaPorekla |
+----+------------+--------------+------+----------+---------------+
|  1 |        123 | Artikl br. 1 | 2010 |     2000 | Germany       |
|  2 |        124 | Artikl br. 2 |  112 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 |  180 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 2010 |     2000 | Germany       |
|  5 |        132 | Artikl br. 5 |  150 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 |  400 |      390 | France        |
|  7 |        137 | Artikl br. 7 |  260 |      250 | Dansk         |
+----+------------+--------------+------+----------+---------------+
7 rows in set (0.00 sec)
```

Parametar **LIMIT** (*videti deo 4.3.3*) služi za ograničavanje broja kolona na koje ćemo primeniti izmenu:

```
UPDATE Artikl
       SET Cena=100
```

```
      LIMIT 3;
+----+------------+--------------+------+----------+---------------+
| rb | Magacin_rb | Naziv        | Cena | Kolicina | ZemljaPorekla |
+----+------------+--------------+------+----------+---------------+
|  1 |        123 | Artikl br. 1 |  100 |     2000 | Germany       |
|  2 |        124 | Artikl br. 2 |  100 |      102 | Srbija        |
|  3 |        127 | Artikl br. 3 |  100 |      170 | USA           |
|  4 |        127 | Artikl br. 4 | 2010 |     2000 | Germany       |
|  5 |        132 | Artikl br. 5 |  150 |      140 | Switzerland   |
|  6 |        135 | Artikl br. 6 |  400 |      390 | France        |
|  7 |        137 | Artikl br. 7 |  260 |      250 | Dansk         |
+----+------------+--------------+------+----------+---------------+
7 rows in set (0.00 sec)
```

3.5. Uklanjanje podataka

Funkcija uklanjanja podataka u bazi putem SUBP-a se odnosi na uklanjanje podataka iz neke (jedne ili više) od tabela u bazi podataka. Moguće je ukloniti samo kompletne redove tabela (uklanjanje vrednosti parametara zapisa se vrši pomoću naredbe **UPDATE**). Uklanjanje podataka se u SQL jeziku vrši pomoću naredbe **DELETE**.

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
      [WHERE where_condition]
      [ORDER BY ...]
      [LIMIT row_count]
```

4.5.1 Dokumentovana struktura naredbe DELETE za jednu tabelu

Osnovni oblik komande **DELETE** izgleda ovako:

```
DELETE FROM Artikl;
```

i njome su uklonjeni svi zapisi iz tabele Artikl.

Ukoliko želimo da uklanjenje ograničimo samo na određene zapise (redove) moramo koristiti klauzulu **WHERE** (videti deo 6.3.1):

```
DELETE FROM Artikl
      WHERE ZemljaPorekla='Germany';
```

Parametar **LIMIT** (videti deo 6.3.3) služi za ograničavanje broja kolona na koje ćemo primeniti uklanjanje:

```
DELETE FROM Artikl
      LIMIT 10;
```

4. DCL (Data Control Language)

Osim manipulacije podacima (DML), jedna od bitnih karakteristika SUBP-a je kontrola podataka (DCL), pre svega kontrola pristupa podacima. Kontrola pristupa podacima obično podrazumeva mogućnost kreiranja više korisničkih profila (naloga) sa određenim pristupnim podacima (korisničko ime i lozinka) i pridruženim privilegijama. Neki SUBP imaju mogućnost povezivanja internih korisničkih naloga i privilegija sa nekim spoljnim sistemom za proveru autentičnosti korisnika (npr. povezivanje sa sistemskim nalozima, Active Directory-jem i sl.). MySql SUBP poseduje interni sistem korisničkih naloga (sa parametrima: korisničko ime, lozinka, mrežna lokacija klijenta) sa privilegijama (izlistani u tabeli 5.1) na nivou jedne ili više baza, tabela i/ili kolona. Podaci vezani za korisničke naloge i privilegije su skladišteni u sistemskoj bazi "mysql".

Osnovne naredbe za rad sa korisničkim nalozima:

- **CREATE USER**
- **RENAME USER**
- **DROP USER**
- **SET PASSWORD**

Kreiranje novih korisničkih naloga nije neophodno zasebno obaviti pre izmene privilegija već se novi nalog (ili nova varijanta postojećeg naloga) može kreirati samom dodelom privilegija. Prve tri naredbe su dostupne počev od verzije MySql-a 5.0 sa određenim izmenama u verziji 5.0.2.

Osnovne naredbe za rad sa privilegijama su:

- **GRANT**
- **REVOKE**
- **FLUSH PRIVILEGES**

4.1. CREATE USER

Naredba `CREATE USER` služi za kreiranje novih korisničkih naloga na SUBP. Pravo na korišćenje ove naredbe ima administrator sistema kao i korisnici koji imaju globalno pravo na ovu privilegiju i korisnici koji imaju INSERT privilegiju za sistemsku **mysql** bazu podataka.

```
CREATE USER user [IDENTIFIED BY [PASSWORD] 'password']
    [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
```

8.1 - Dokumentovana struktura naredbe CREATE USER

Osnovni oblik korišćenja ove naredbe je:

```
CREATE USER 'korisnik';
```

Ukoliko želimo da korisničkom nalogu pridružimo i hostname računara sa koga je dozvoljen pristup, naredba je:

```
CREATE USER 'korisnik@racunar1.mreza.edu';
```

Ukoliko želimo da korisničkom nalogu pridružimo i lozinku, naredba je:

```
CREATE USER 'korisnik' IDENTIFIED BY 'lozinka';
```

Ukoliko želimo da istovremeno kreiramo više korisničkih naloga, naredba je:

```
CREATE USER 'korisnik1' IDENTIFIED BY 'lozinka1',
'korisnik2' IDENTIFIED BY 'lozinka2';
```

4.2. RENAME USER

Naredba `RENAME USER` služi za izmenu postojećih korisničkih naloga na SUBP. Pravo na korišćenje ove naredbe ima administrator sistema kao i korisnici koji imaju globalno pravo na `CREATE USER` privilegiju i korisnici koji imaju UPDATE privilegiju za sistemsku **mysql** bazu podataka.

```
RENAME USER old_user TO new_user
    [, old_user TO new_user] ...
```

8.2 - Dokumentovana struktura naredbe RENAME USER

Osnovni oblik korišćenja ove naredbe je:

```
RENAME USER 'korisnik1' TO 'korisnik2';
```

4.3. DROP USER

Naredba `DROP USER` služi za izmenu postojećih korisničkih naloga na SUBP. Pravo na korišćenje ove naredbe ima administrator sistema kao i korisnici koji imaju globalno pravo na `CREATE USER` privilegiju i korisnici koji imaju DELETE privilegiju za sistemsku **mysql** bazu podataka.

```
DROP USER user [, user] ...
```

8.3 - Dokumentovana struktura naredbe DROP USER

Osnovni oblik korišćenja ove naredbe je:

```
DROP USER 'korisnik1';
```

Ukoliko želimo da istovremeno uklonimo više korisničkih naloga, naredba je:

```
DROP USER 'korisnik1', 'korisnik2';
```

4.4. SET PASSWORD

Naredba `SET PASSWORD` služi za promenu pristupne lozinke postojećih korisničkih naloga na SUBP. Ovom naredbom je moguće promeniti lozinku za sopstveni nalog kao i lozinku za ostale korisničke naloge ukoliko aktivni korisnički nalog ima UPDATE privilegiju za sistemsku **mysql** bazu podataka.

```
SET PASSWORD [FOR user] = PASSWORD('some password')
```

8.4 - Dokumentovana struktura naredbe SET PASSWORD

Osnovni oblik korišćenja ove naredbe je:

```
SET PASSWORD = PASSWORD('nova lozinka');
```

Ukoliko želimo da promenimo lozinku za tuđi korisnički nalog, naredba je:

```
SET PASSWORD FOR 'korisnik2' = PASSWORD('nova lozinka');
```

4.5. Dodela privilegija (GRANT)

Naredba GRANT omogućava administratorima SUBP-a da određenom koričkom nalogu dodele određene privliegije nad određenim objektima baze podataka. S obzirom na to da ukoliko se privilegije dodeljuju nalogu koji ne postoji na SUBP on se automatski kreira, ovom naredbom je moguće i kreirati nove korisničke naloge. Za korišćenje ove naredbe neophodno je biti administrator sistema ili imati **GRANT OPTION** privilegiju kojom je moguće dodeliti privilegije iz skupa privilegija koje su dodeljene aktivnom korisničkom nalogu.

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [REQUIRE
        NONE |
        [{SSL| X509}]
        [CIPHER 'cipher' [AND]]
        [ISSUER 'issuer' [AND]]
        [SUBJECT 'subject']]
    [WITH with_option [with_option] ...]

object_type =
    TABLE
  | FUNCTION
  | PROCEDURE

with_option =
    GRANT OPTION
  | MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
```

8.5 - Dokumentovana struktura naredbe GRANT

Osnovna dodela privilegija se vrši naredbom:

```
GRANT ALL ON database_name.table_name TO 'korisnik@hostname'     IDENTIFIED BY
'lozinka';
```

Dodela privilegija na globalnom nivou (svim objektima - bazama, tabelama, kolonama -u SUBP) se vrši naredbom:

```
GRANT ALL ON *.* TO 'korisnik' IDENTIFIED BY 'lozinka'
```

Dodela privilegija na globalnom nivou baze podataka se vrši naredbom:

```
GRANT ALL ON db_name.* TO 'korisnik' IDENTIFIED BY 'lozinka'
```

Dodela privilegija na globalnom nivou tabele u bazi se vrši naredbom:

```
GRANT ALL ON db_name.table_name TO 'korisnik' IDENTIFIED BY 'lozinka'
```

Takođe, moguća je (mada se retko koristi) dodela privilegija i na nivou kolone u tabeli kao i rutine.

Privilegije	**Značenje**
`ALL [PRIVILEGES]`	Sets all simple privileges except `GRANT OPTION`
`ALTER`	Allows use of `ALTER TABLE`
`CREATE`	Allows use of `CREATE TABLE`
`CREATE TEMPORARY TABLES`	Allows use of `CREATE TEMPORARY TABLE`
`CREATE VIEW`	Allows use of `CREATE VIEW`
`DELETE`	Allows use of `DELETE`
`DROP`	Allows use of `DROP TABLE`
`EXECUTE`	Allows the user to run stored procedures (MySQL 5.0)
`FILE`	Allows use of `SELECT ... INTO OUTFILE` and `LOAD DATA INFILE`
`INDEX`	Allows use of `CREATE INDEX` and `DROP INDEX`
`INSERT`	Allows use of `INSERT`
`LOCK TABLES`	Allows use of `LOCK TABLES` on tables for which you have the `SELECT` privilege
`PROCESS`	Allows use of `SHOW FULL PROCESSLIST`
`REFERENCES`	Not yet implemented
`RELOAD`	Allows use of `FLUSH`
`REPLICATION CLIENT`	Allows the user to ask where the slave or master servers are
`REPLICATION SLAVE`	Needed for replication slaves (to read binary log events from the master)
`SELECT`	Allows use of `SELECT`
`SHOW DATABASES`	`SHOW DATABASES` shows all databases
`SHOW VIEW`	Allows use of `SHOW CREATE VIEW`
`SHUTDOWN`	Allows use of **mysqladmin shutdown**
`SUPER`	Allows use of `CHANGE MASTER`, `KILL`, `PURGE MASTER LOGS`, and `SET GLOBAL` statements, the **mysqladmin debug** command; allows you to connect (once) even if `max_connections` is reached
`UPDATE`	Allows use of `UPDATE`
`USAGE`	Synonym for ``no privileges''
`GRANT OPTION`	Allows privileges to be granted

Tabela 8.1 - Moguće vrednosti za dodelu odnosno oduzimanje `privilegija`

4.6. Oduzimanje privilegija (REVOKE)

Naredba REVOKE omogućava administratorima SUBP-a da određenom koričkom nalogu oduzmu određene privliegije nad određenim objektima baze podataka. Za korišćenje ove naredbe neophodno je biti administrator sistema ili imati GRANT OPTION privilegiju kojom je moguće oduzeti privilegije iz skupa privilegija koje su dodeljene aktivnom korisničkom nalogu.

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    FROM user [, user] ...
```

8.6 - Dokumentovana struktura naredbe REVOKE

Osnovno oduzimanje privilegija se vrši naredbom:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'korisnik';
```

4.7. Primena izmena (FLUSH)

Naredba FLUSH služi za ponovo učitavanje parametara sistema koji se nalaze keširani u memoriji. Za izvršavanje ove naredbe potrebno je imati RELOAD privilegiju.

```
FLUSH [LOCAL | NO_WRITE_TO_BINLOG] flush_option [, flush_option] ...
```

8.7 - Dokumentovana struktura naredbe FLUSH

Flush_option:	Objašnjenje:
HOSTS	Pražnjenje keša host tabele
DES_KEY_FILE	Ponovno učitavanje DES ključa
LOGS	Zatvaranje i otvaranje svih log fajlova
MASTER	Brisanje i ponovno kreiranje svih log fajlova
PRIVILEGES	Ponovno učitavanje svih privilegija iz GRANT tabele
QUERY CACHE	Defragmentacija keširanih upita
SLAVE	Resetovanje svih replikacija sekundardnih parametara
STATUS	Resetovanje svih statusnih promenljivu na 0
USER_RESOURCES	Resetovanje svih resursa iz prethodnih sati na 0

Tabela 8.2 - Moguće vrednosti za flashi_option parametar

5. DODATAK 1 (Izrazi i funkcije)

Pored prikazivanja prostih vrednosti memorisanih u tabelama baze podataka, SQL ima više funkcija koje mogu biti korišćene za dobijanje sumarnih informacija, kao i mogućnost primene aritmetičkih funkcija i funkcija nad stringovima (nizovima karaktera).

Aritmetičke funkcije

Funkcije za dobijanje sumarn.ih informacija su:

AVG (atribut)	- izračunava srednju vrednost
SUM (atribut)	- izračunava ukupnu vrednost
MIN (atribut)	- nalazi minimalnu vrednost
MAX (atribut)	- nalazi maksimalnu vrednost

Ove funkcijesu definisane nad numeričkim kolonama.
Funkcija COUNT definisana je nad kolonama bilo kog tipa. Ona ima tri oblika:

COUNT (*)	- nalazi broj n-torki u grupi
COUNT (atribut)	- nalazi NOT-NULL vrednosti kolone

Funkcije nad nizom karaktera

U SQL-u su definisane brojne funkcije nad kolonama tipa character. Najčešće se primenjuju sledeće:

- stringi || string2 - spaja stringove karaktera
- **LENGTH** (string) - nalazi dužinu stringa
- **UPPER** (str) - menja sva mala slova u velika
- **LOWER** (str) - menja sva velika slova u mala
- **TO_NUM** (str) - pretvara niz karaktera (numeričkih) u broj
- **TO_CHAR** (str) - pretvara broj u niz karaktera

SQL podržava sledeće aritmetičke funkcije:

POWER (broj. e)	- diže broj na e-ti stepen
ROUND (broj [.d])	- zaokružuje broj na d decimala
TRUNC (broj [.d])	- odbacuje ostatak od d-tog decimalnog mesta
ABS (broj)	- naiazi apsolutnu vrednos; broja
SIGN (broj)	- daje +1 ako je broj >0. 0 ako je broj =0, -1 ako je broj <0.
MOD (broji, broj2)	- izračunava broji moduo broj2
SQRT (broj)	- nalazi pozitivan kvadratni koren broja

6. DODATAK 2 (Zadaci za vežbu)

6.1. Zadaci za vežbu DDL

1. Kreirati bazu podataka univerzitet.
2. Kreirati tabelu student sa sledećim poljima:
    * broj_indeksa - varchar, 15 karaktera, primarni ključ
    * ime - varchar, 25 karaktera
    * prezime - varchar, 25 karaktera
    * adresa - varchar, 50 karaktera
    * datum_rođenja - date
    * jmbg - varchar, 13
    * fakultet_rb - integer (spoljni ključ)
3. Kreirati tabelu predmet sa sledećim poljima:
    * rb - integer, auto_increment, primarni ključ
    * naziv - varchar, 50
    * datum - date
    * ocena - integer
    * broj_indeksa - varchar, 15 karaktera (spoljni ključ)
4. Kreirati tabelu fakultet sa sledećim poljima:
    * redni_broj - integer, auto_increment, primarni ključ
    * pun_naziv - varchar, 40 karaktera
    * skraćenica - varchar, 8 karaktera
5. Kreirati indeks faks u tabeli fakultet nad poljem skraćenica
6. Kreirati pogled studenti koji će da prikazuje sve studente koji su upisani na univerzitet
7. Izmeniti tabelu ispit tako da polje naziv ne bude 50 nego 40 karaktera
8. Izmeniti naziv tabele predmet u ispit
9. Izmeniti pogled studenti tako da prikazuje samo studente koji su upisani na FPI
10. Ukloniti indeks faks
11. Ukloniti tabelu fakultet
12. Ukloniti pogled studenti
13. Ukloniti bazu podataka univerzitet

6.2. Zadaci za vežbu DML

1. Uneti podatke iz sledeće tabele u tabelu **Dobavljac**:

```
+----+------------+------------------+-----------+
| rb | Naziv      | Adresa           | Telefon   |
+----+------------+------------------+-----------+
|  1 | Dobavljac1 | Beogradska 5     | 64123456  |
|  2 | Dobavljac2 | Bac49            | 11256782  |
|  3 | Dobavljac3 | Uzicka 42        | 62564778  |
|  4 | Dobavljac4 | Knez Mihajlova 2 | 112345812 |
|  5 | Dobavljac5 | Nehruova 62b     | 637469923 |
+----+------------+------------------+-----------+
             5 rows in set (0.00 sec)
```

2. Uneti podatke iz sledeće tabele u tabelu **Magacin:**

```
+-----+---------------+-----------+-----------+
| rb  | Adresa        | Telefon   | Kapacitet |
+-----+---------------+-----------+-----------+
| 124 | bac26         | 11145856  |      2000 |
| 127 | Beogradska 5  | 635844535 |      1000 |
| 129 | Dalmatinska43 | 633463    |      1500 |
| 132 | Vojvodanska2  | 642888158 |       460 |
| 135 | Francuska6    | 11136845  |      1200 |
| 137 | Nusiceva24    | 23425698  |       900 |
+-----+---------------+-----------+-----------+
            6 rows in set (0.00 sec)
```

3. Prikazati sve podatke sa svim poljima iz tabele **Magacin**
4. Prikazati sve podatke sa svim poljima iz tabele **Dobavljac**
5. Prikazati sve podatke iz tabele **Dobavljac** u sledećem redosledu polja:**Naziv**, **Rb**, **Telefon**, **Adresa**
6. Prikazati sve podatke iz tabele **Magacin** sortirane po polju **Kapacitet** po opadajućem redosledu
7.Prikazati iz tabele **Dobavljac** sledeća polja: **Rb**, **Adresa**, **Telefon,** za dobavljača koji se nalazi u **Dalmatinskoj 43**
8. Prikazati iz tabele **Magacin** prva četiri zapisa
9.Prikazati iz tabele **Dobavljac** zapise od **2** do **4**
10.Izmeniti tabelu **Magacin** tako da polje **Kapacitet** kod svih zapisa bude **2000**
11.Izmeniti tabelu **Magacin** tako da polje **Kapacitet** bude **500** za magacin koji se nalazi u **Francuskoj 6**
12.Izmeniti tabelu **Magacin** tako da polje **Kapacitet** kod svih zapisa bude bude povećano za **30 %**
13.Izmeniti tabelu **Magacin** tako da polje **Kapacitet** bude umanjeno za **20 %** prva dva zapisa
14.Izmeniti tabelu **Magacin** tako da polje **Kapacitet** bude povećano za **50** jedinica kod magacina sa rednim brojem **2**, **3** i **4**
15.Ukloniti poslednji zapis iz tabele **Dobavljac**
16.Ukloniti sve zapise iz tabele **Magacin**

6.3. Zadaci za vežbu DCL

1. Kreirati korisnički nalog **student** koji serveru baze podataka može da pristupi samo sa računara **workstation1** koji se nalazi u mreži **praktikumbp.local**. Nalogu pridružiti šifru **amsterdam**.

2. Preimenovati korisnički nalog **student** u **student1**.

3. Promeniti pristupnu lozinu za nalog **student1** u **akapulko_t_shirt**.

4. Dodeliti sve privilegije nad bazom **univerzitet** i svim njenim tabelama korisniku **student1**.

5. Oduzeti privilegiju **DELETE** od korisnika **student1**.

6. Primeniti prethodne izmene privilegije.

7. Ukloniti korisnički nalog **student1**.