Projektovanje baza podataka (programiranje CRUD operacija)

CRUD = Create, Read, Update, Delete

C= Create

Jezik za definisanje podataka (*Data Definition Language* (DDL)) omogućava formiranje strukture baze podataka.

DDL: Kreiranje baze

Nakon projektovanja strukture baze podataka, naredni korak je, izdavanje komande MySQL-u za kreiranje nove baze podataka. To se radi pomoću SQL-ove komande CREATE DATABASE, na sledeći način:

CREATE DATABASE [IF NOT EXISTS] *db_name*;

Uslov je da imate CREATE privilegiju za bazu.

CREATE DATABASE kreira bazu podataka sa datim nazivom. Ako već postoji baza podataka sa datim nazivom, a nije upotrebljeno IF NOT EXISTS, javiće se greška.
Kreiranje baze podataka poslovanje:
CREATE DATABASE *poslovanje*;
Ako korisnik želi da proveri da li je ova komanda uspešno izvršena može da izda komandu:
SHOW DATABASES;
Trebalo bi da se naziv nove baze podataka pojavi u spisku baza podataka na serveru. Sada na serveru postoji prazna baza podataka, a u MySQL data direktorijumu postoji direktorijum i i datoteka db.opt u direktorijumu baze. Kako se budu dodavale tabele, dodavaće se i datoteke koje odgovaraju tabelama.

Nazive baza podataka je najbolje pisati malim slovima. U nazivima baza ne treba koristiti razmake. Ako se želi efekat razmaka treba koristiti donju crtu (_).

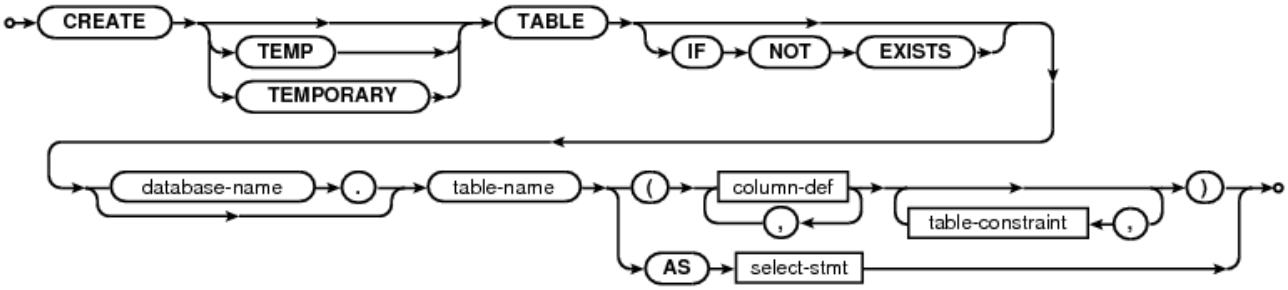
Komanda **CREATE SCHEMA je sinonym za CREATE DATABASE (MySQL 5.0.2)**
Proširenje komande ima sintaksu:
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] ***db_name***
[*create_specification*] ...
create_specification:
[DEFAULT] CHARACTER SET [=] *charset_name*
| [DEFAULT] COLLATE [=] *collation_name*

create_specification je opcija koja specifira karacteristi baze (one se čuvaju u datoteci db.opt u direktorijumu baze).
Klauza CHARACTER SET specificira karakterski skup
Klauza COLLATE specificira podrazumevanu kolaciju baze podataka. Collation (uparivanje) je skup pravila za poređenje karaktera u okviru seta karaktera.

Za kreiranje baze može se koristiti mysqladmin program ("mysqladmin — Client for Administering a MySQL Server").

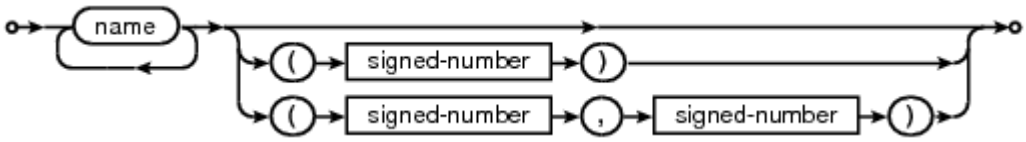
DDL: Kreiranje tabela (podsetnik)

NaredbaKreiranjaTabela::= CREATE TABLE Tabela(DefinicijaKolone, …[OgranicenjeTabele, …]);

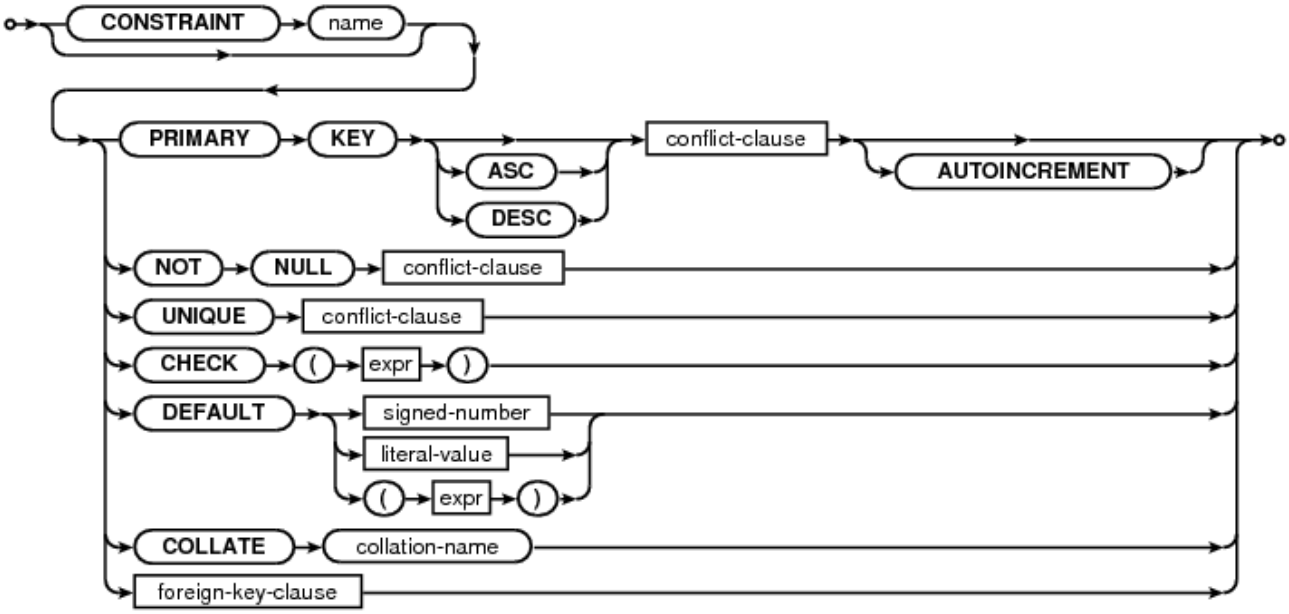


DefinicijaKolone::= Kolona{ Tip | Domen} OgranicenjeKolone…

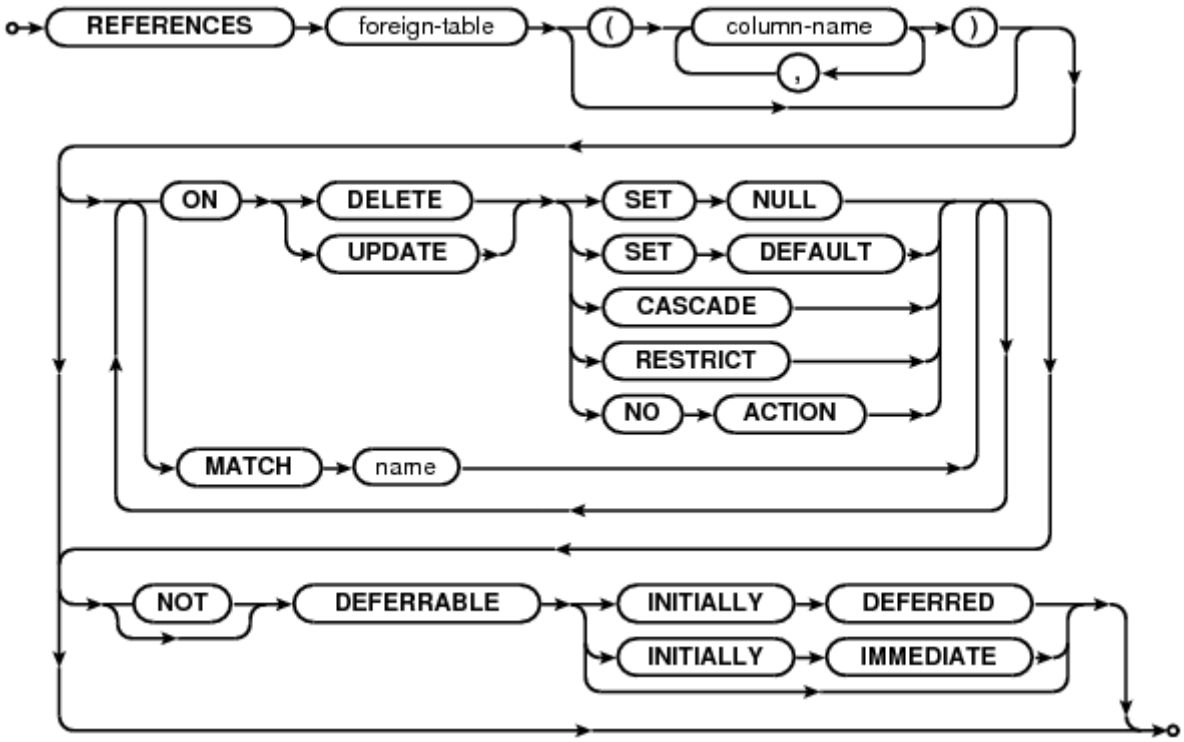
TipKolone::=



OgranicenjaKolone::=

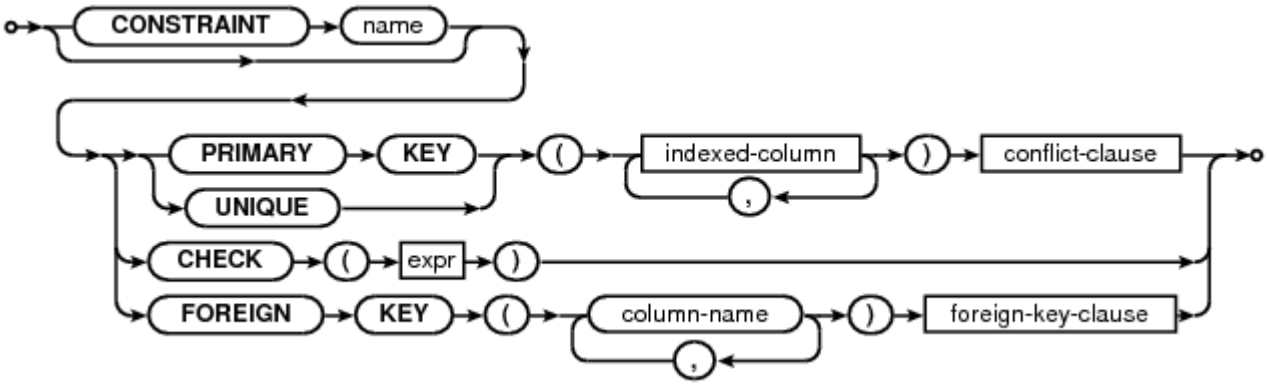


foreign-key-clause::=



OgraničenjaKolone	OgrančenjaTabele
NOT NULL **UNIQUE** **PRIMARY KEY** **CHECK(Predikat)** **DEFAULT=Const** **REFERENCES Tabela[(Kolona)]** **[ON UPDATE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}]** **[ON DELETE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}]**	**UNIQUE(Kolona, …)** **PRIMARY KEY(Kolona, …)** **CHECK(Predikat)** **FOREIGN KEY(Kolona, …)** **REFERENCES Tabela[(Kolona, …)]** **[ON UPDATE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}]** **[ON DELETE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}]**

OgraničenjaTabele::=



CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ***tbl_name***
(*create_definition*,...)
[*table_option* ...]

ili:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl_name*
[(*create_definition*,...)]
[*table_option* ...]
select_statement

ili:
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl_name*
{ LIKE *old_tbl_name* | (LIKE *old_tbl_name*) }

create_definition:
column_definition
| [CONSTRAINT [*symbol*]] PRIMARY KEY [*index_type*] (*index_col_name*,...)
| {INDEX|KEY} [*index_name*] [*index_type*] (*index_col_name*,...)
| [CONSTRAINT [*symbol*]] UNIQUE [INDEX|KEY]
[*index_name*] [*index_type*] (*index_col_name*,...)
| {FULLTEXT|SPATIAL} [INDEX|KEY] [*index_name*] (*index_col_name*,...)
| [CONSTRAINT [*symbol*]] FOREIGN KEY
[*index_name*] (*index_col_name*,...) [*reference_definition*]
| CHECK (*expr*)

column_definition:

col_name data_type [NOT NULL | NULL] [DEFAULT *default_value*]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
[COMMENT '*string*'] [*reference_definition*]

index_col_name:
col_name [(*length*)] [ASC | DESC]

index_type:
USING {BTREE | HASH}

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

table_option:
{ENGINE|TYPE} [=] *engine_name*
| AUTO_INCREMENT [=] *value*
| AVG_ROW_LENGTH [=] *value*
| [DEFAULT] CHARACTER SET *charset_name*
| CHECKSUM [=] {0 | 1}
| COLLATE *collation_name*
| COMMENT [=] '*string*'
| CONNECTION [=] '*connect_string*'
| DATA DIRECTORY [=] '*absolute path to directory*'
| DELAY_KEY_WRITE [=] {0 | 1}
| INDEX DIRECTORY [=] '*absolute path to directory*'
| INSERT_METHOD [=] { NO | FIRST | LAST }
| MAX_ROWS [=] *value*
| MIN_ROWS [=] *value*
| PACK_KEYS [=] {0 | 1 | DEFAULT}
| PASSWORD [=] '*string*'
| ROW_FORMAT [=]
{DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
| UNION [=] (*tbl_name*[,*tbl_name*]...)

select_statement:
[IGNORE | REPLACE] [AS] SELECT ...

Ograničenja (Constraints)

Ako se redovno sortiraju podaci u tabeli po jednoj istoj koloni ili kolonama potrebno je napraviti indeks za tu kolonu ili te kolone. Indeks je interna tabela vrednosti koja održava red, odnosno redosled zapisa. Na ovaj način, kada je potrebno da se sortiraju podaci ili kada je potrebno da se određeni podatak pronađe brzo MySQL može da vrši pretraživanje po indeks ključevima po poznatom redosledu, što je bolje od sekvencijalnog pretraživanja po podacima.

Pravljenje indeksa usporava unošenje podataka; svaki novi zapis, obrisani zapis ili promena vrednosti u indeksiranoj koloni zahteva promenu indeksa. Indeksirane kolone treba koristiti samo onda kada su stvarno neophodne.

KEY (ključ) je sinonim za INDEX (indeks) i znači da će zadata kolona (ili kolone) biti indeksirana. Indeksi se koriste da bi se brzo pronašli redovi sa određenom vrednošću u koloni. Atribut PRIMARY KEY se može specificirati i kao KEY kada je dat u definiciji kolone.

UNIQUE (jedinstveni) indeks kreira ograničenje takvo da sve vrednosti u indeksu (odnosno koloni koja je indeksirana) moraju biti različite. Ukoliko korisnik pokuša da doda novi red u tabelu koji ima istu vrednost u koloni za koju je definisan UNIQUE indeks kao neki drugi red javiće se greška. Ovo ograničenje se ne odnosi na NULL vrednosti sem za BDB mašine za smeštanje podataka. O mašinama biće reči dole u tekstu. Za ostale mašine za smeštanje podataka UNIQUE indeks dozvoljava višestruke NULL vrednosti za kolone koje mogu sadržati NULL vrednosti.

PRIMARY KEY (primarni ključ) je jedinstveni indeks gde sve kolone ključa moraju biti definisane kao NOT NULL. Ako nisu eksplicitno deklarisane kao NOT NULL, MySQL ih deklariše implicitno. Tabela može imati samo jedan primarni ključ. Na ovaj način se postavlja ograničenje nad tabelom. To znači da se upisom novog zapisa u kolonu koja je primarni ključ ne sme upisati vrednost koja već postoji. Za kolonu (ili kolone) koja je primarni ključ se automatski formira indeks.
U kreiranoj tabeli primarni ključ je smešten kao prvi, zatim dolaze svi UNIQUE (jedinstveni) indeksi i na kraju svi nejedinstveni indeksi.

Primani ključ može biti indeks nad više kolona. Međutim, nije moguće kreirati indeks nad više kolona korišćenjem atributa PRIMARY KEY u specifikaciji kolone. Mora se koristiti posebna PRIMARY KEY(*index_col_name, ...*) klauzula.
U MySQL-u naziv primarnog ključa je PRIMARY. Za ostale indekse, ako im se ne dodeli naziv, indeksu se daje naziv prve indeksirane kolone, zajedno sa opcionim sufiksom (_2,_3, ...) da bi bio jedinstven. Nazivi indeksa tabele se mogu pogledati sa SHOW INDEX FROM *tbl_name*.

Za mašine za skladištenje koje nisu tipa InnoDB je moguće prilikom definisanja kolone koristiti REFERENCES *tbl_name*(*col_name*) koja nema stvarni efekat i služi samo kao podsetnik ili komentar korisniku da se namerava da kolona koja se trenutno definiše ukazuje na kolonu neke druge tabele.

Neke mašine za skladištenje dozvoljavaju definisanje tipa indeksa prilikom kreiranja indeksa.

Specifikacija za *index_col_name* se može završavati sa ASC ili DESC. Ove ključne reči su dozvoljene za buduće ekstenzije za specificiranje smeštanja vrednosti u indeksu po rastućem ili opadajućem redosledu. Trenutno se ove ključne reči ignorišu; vrednosti u indeksu su uvek smeštene po rastućem redosledu.

Moguće je kreirati specijalne FULLTEXT indekse koji se koriste za Full-text (tekstualna) pretraživanja. Samo MyISAM mašina za skladištenje podržava FULLTEXT indekse. Oni mogu biti kreirani samo za CHAR, VARCHAR i TEXT kolone.

Moguće je kreirati SPATIAL indekse za prostorne tipove podataka (dozvoljavaju generisanje, smeštanje i analizu geografskih obeležja). Ovi tipovi podataka su podržani samo u MyISAM tabelama i indeksirane kolone moraju biti deklarisane kao NOT NULL.

InnoDB tabele podržavaju ograničenja tipa strani ključ (FOREIGN KEY). Strani ključ je skup kolona (jedna ili više kolona) iz jedne tabele (sekundarna tabela ili dete tabela) čije vrednosti moraju da se „slažu“, odnosno da odgovaraju vrednostima primarnog ključa u nekoj drugoj tabeli (primarna tabela ili roditelj tabela). Definisanjem stranog ključa se u stvari uspostavlja relacija između tabela. Sintaksa za definisanje stranog ključa kod ovog tipa tabela izgleda ovako:

```
[CONSTRAINT symbol] FOREIGN KEY [id] (index_col_name, ...)
REFERENCES tbl_name (index_col_name, ...)
[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
```

Prilikom definisanja stranog ključa obe tabele moraju biti InnoDB tabele i ne smeju biti privremene (TEMPORARY) tabele.
InnoDB odbija bilo koju INSERT ili UPDATE operaciju koja pokušava da kreira vrednost za strani ključ u dete tabeli ako ne postoji odgovarajuća vrednost ključa u roditelj tabeli. Akcija koju preduzima InnoDB za bilo koju UPDATE ili DELETE operaciju koja pokušava da promeni ili obriše vrednost ključa u roditelj tabeli koji ima povezane redove u dete tabeli zavisi od referencijalne akcije specificirane korišćenjem ON UPDATE i ON DELETE podklauzula u FOREIGN KEY klauzuli. Kada korisnik pokuša da obriše ili promeni red u roditelj tabeli, a postoje jedan ili više povezanih redova u dete tabeli, InnoDB podržava 5 opcija vezano za akciju koja će biti izvršena:

• CASCADE: Brisanjem ili izmenom reda u roditelj tabeli se automatski brišu ili menjaju povezani redovi u dete tabeli

• SET NULL: Brisanjem ili izmenom reda u roditelj tabeli se vrednost u koloni koja je strani ključ dete tabele za povezane redove postavlja na NULL. Ovo važi samo ako kolone koje su strani ključ nemaju specificiran NOT NULL kvalifikator.

• NO ACTION: U standardnom SQL-u NO ACTION znači da nema akcije u smislu da neće biti moguće obrisati ili promeniti vrednost primarnog ključa u roditelj tabeli ako postoji povezana vrednost stranog ključa u dete tabeli. InnoDB odbija operaciju brisanja ili izmene za roditelj tabelu.

• RESTRICT: Odbija se operacija brisanja ili izmene za roditelj tabelu

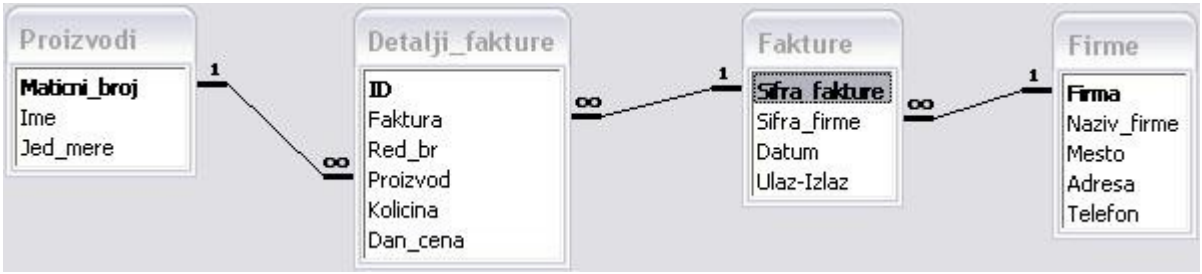
• SET DEFAULT: Ova akcija je prepoznata od strane parsera, ali InnoDB odbija definicije tabela koje sadrže ON DELETE SET DEFAULT ili ON UPDATE SET DEFAULT klauzule

Odgovarajuće kolone koje čine strani i primarni ključ moraju imati slične tipove podataka da bi mogle da budu poređene bez konverzije podataka. Veličina i znak celobrojnih tipova moraju biti isti. Dužine string tipova ne moraju biti iste.

Zadaci

1. Neka je kreirana baza podataka **poslovanje** koja služi za praćenje poslovanja jedne firme koja radi sa određenom grupom proizvoda i sarađuje sa određenim brojem drugih firmi.
Baza podataka **poslovanje** ima četiri tabele i to: priozvodi, firme, fakture i detalji_fakture.

Na sledećoj slici se mogu videti sve tabele sa nazivima kolona i primarnim ključevima (nazivi kolona ispisani polucrnim slovima) i uspostavljene relacije i strani ključevi u odgovarajućim tabelama. Može se videti da su sve uspostavljene relacije tipa jedan-prema više.



Kreirajte tabelu proizvodi

REŠENJE:

Verzija 1	Verzija 2
CREATE TABLE proizvodi (maticni_broj TINYINT(3) UNSIGNED NOT NULL, ime VARCHAR(50) NOT NULL, jed_mere VARCHAR(20) NOT NULL, PRIMARY KEY (maticni_broj));	CREATE TABLE proizvodi (maticni_broj TINYINT(3) UNSIGNED NOT NULL, ime VARCHAR(50) NOT NULL, jed_mere VARCHAR(20) NOT NULL); >>>Query OK, 0 rows affected (0.09 sec)

	ALTER TABLE proizvodi ADD CONSTRAINT pk_mb PRIMARY KEY (maticni_broj); >>>Query OK, 0 rows affected (0.06 sec) Records: 0 Duplicates: 0 Warnings: 0

2. Kreirajte tabele Fakture i Detalji_fakture koje su povezane preko stranog ključa. Naime, polje Faktura iz tabele Detalji_fakture može uzimati samo one vrednosti koje se pojavljuju u polju Sifra_fakture tabele Fakture.
Vodite računa da obezbedite deklarisanje kolona odgovarajućim opcijama PRIMARY KEY, KEY, UNIQUE ili INDEX, kako bi mogao da se automatski formira i indeks. Svi indeksi koji su neophodni najčešće će se kreirati automatski prilikom kreiranja tabele.

REŠENJE:

```
CREATE TABLE firme (
firma TINYINT(3) UNSIGNED NOT NULL,
naziv_firme VARCHAR(100) NOT NULL,
mesto VARCHAR(50) NOT NULL,
adresa VARCHAR(50) NOT NULL,
telefon VARCHAR(20) NOT NULL,
PRIMARY KEY (firma)
);

CREATE TABLE fakture (
sifra_fakture SMALLINT(5) UNSIGNED NOT NULL,
sifra_firme TINYINT(3) UNSIGNED NOT NULL,
datum DATE NOT NULL,
ulaz_izlaz CHAR(1) NOT NULL,
PRIMARY KEY (sifra_fakture),
KEY k_sifra_firme (sifra_firme),
CONSTRAINT fk_fakture_firme FOREIGN KEY (sifra_firme) REFERENCES firme (firma)
ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE detalji_fakture (
id MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
faktura SMALLINT(5) UNSIGNED NOT NULL,
red_br TINYINT(3) UNSIGNED NOT NULL,
proizvod TINYINT(3) UNSIGNED NOT NULL,
kolicina DECIMAL(8,2) NOT NULL,
dan_cena DECIMAL(8,2) NOT NULL,
PRIMARY KEY (id),
KEY k_proizvod (proizvod),
KEY k_faktura (faktura),
CONSTRAINT fk_detalji_fakture_proizvodi FOREIGN KEY (proizvod) REFERENCES
proizvodi (maticni_broj) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT fk_detalji_fakture_fakture FOREIGN KEY (faktura) REFERENCES fakture
(sifra_fakture) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Mašine za skladištenje i tipovi tabela

MySQL podržava nekoliko mašina za skladištenje koji rade kao manipulatori za različite tipove tabela. Među ovim mašinama postoje one koje rade sa transakcionim i one koje rade sa netransakcionim tabelama. Ovo su samo neke od ukupno 11 mašina za skladištenje.

• MyISAM upravlja netransakcionim tabelama. Obezbeđuje visoku brzinu smeštanja i preuzimanja podataka kao i mogućnost fulltext pretraživanja.

• MEMORY mašina za skladištenje radi sa tabelama koje su smeštene u memoriji računara i nikada se ne upisuju na disk. Zahvaljujući tome te tabele su veoma brze, ali veličina im je ograničena i podaci iz njih se ne mogu restaurirati

ukoliko dođe do kvara sistema.
MERGE mašina za skladištenje, takođe poznata i kao MRG_MyISAM mašina omogućava rad sa kolekcijom identičnih MyISAM tabela kao sa jednom tabelom. Pod identičnim tabelama se podrazumeva da tabele imaju identične kolone i indekse. To se može iskoristiti kada operativni sistem ograničava maksimalnu veličinu fajlova – pa zbog toga i tabela. MEMORY i MERGE rade sa netransakcionim tabelama.

• InnoDB mašina za skladištenje radi sa transakcionim tabelama i ima mogućnosti izvršavanja, rollback-a i oporavka prilikom havarije. Ova mašina radi zaključavanje na nivou redova prilikom transakcija. Dizajnirana je za maksimum performansi kada obrađuje velike količine podataka. Ova mašina podržava spoljne ključeve. **Ovo je podrazumevana mašina za skladištenje.**

• ARCHIVE mašina za skladištenje se koristi za smeštanje velikih količina podataka bez indeksa.

SQL-92: Opšti integritet

NaredbaKreiranjaPravila::= CREATE ASSERTION Pravilo CHECK(Ogranicenje);

NaredbaUklanjanjaPravila::=DROP ASSERTIONPravilo;

DDL: izvršavanje ograničenja

Naredba Imenovanja Pravila::=
CONSTRAINT Pravilo CHECK(Ogranicenje)
[{ INITIALLY DEFERRED| INITIALLY IMMEDIATE}]
[{ DEFERRABLE| NOT DEFERRABLE}] ;

NaredbaPostvljanjaModaPravila::= SET CONSTRAINT{ Pravilo, …| ALL} { DEFERRED| IMMEDIATE}

Zadaci
3. Data je šema relacione baze podataka fubalskog saveza za potrebe evidencije utakmica jedne sezone (pretpostavka je da fudbaleri ne mogu da menjaju tim u kome igraju, u toku sezone):
FUDBALER (SifF, Ime, SifT);
TIM (SifT, Naziv, Mesto);
UTAKMICA (SifU, SifTDomaci, SifTGost, Kolo, Ishod, Godina);
IGRAO (SifF, SifU, PozicijaIgraca);
GOL (SifG, SifU, SifF, RedniBrGola, Minut);
KARTON (SifK, SifU, SifF, Tip, Minut);

Sastaviti SQL skript kojim se formira tabela UTAKMICA, ukoliko je poznato da utakmicu igraju dva različita tima čije se šifre nalaze u tabeli TIM, da ishod utakmice može biti iz skupa vrednosti {X-nerešeno, 1-pobeda domaćih, 2-pobeda gostiju} i da postoji 42 kola u kojima se utakmice igraju. U toku sezone svaki par timova odigra dve utakmice, pri čemu je jedna na domaćem, a druga na gostujućem terenu.

Rešenje:

```
CREATE TABLE UTAKMICA
(
SifU  INT PRIMARY KEY,
SifTDomaci  INT NOT NULL REFERENCES TIM(SifT) ON UPDATE CASCADE,
SifTGost   INT NOT NULL REFERENCES TIM(SifT) ON UPDATE CASCADE,
Kolo   INT NOT NULL CHECK (Kolo BETWEEN 1 and 42),
Ishod   CHAR NOT NULL CHECK (Ishod  IN ( '1', '2', 'X')),
Godina  INT,
CHECK (SifTDomaci<> SifTGost),
UNIQUE (SifTDomaci, SifTGost)
);
```

4.
Data je šema relacione baze podataka za potrebe skladišta robe u toku jedne godine:

ROBA(SifR, Naziv, Opis, SifD);
DOBAVLJAC(SifD, Naziv, Adresa);
NABAVKA(SifN, Datum, Kolicina, Cena, SifR);

Sastaviti SQL skript koji formira tabelu NABAVKA ukoliko je poznato da se datum predstavlja kao celobrojna vrednost u opsegu od1 do365, i da cena predstavlja jediničnu cenu koja ne sme biti skuplja za više od10% od cene pri prethodnoj nabavci te robe. Takođe je poznato da jednog datuma može biti najviše jedna nabavka jedne vrste robe.

Rešenje:

```
CREATE TABLE NABAVKA
( SifN  INT PRIMARY KEY,
Datum INT NOT NULL CHECK (Datum BETWEEN 1 and 365),
Cena INT NOT NULL CHECK (Cena>0),
Kolicina  INT NOT NULL CHECK (Kolicina>0),
SifR   INT NOT NULL REFERENCES ROBA(SifR) ON UPDATE CASCADE,
UNIQUE (SifR, Datum),
CHECK (NOT EXISTS (SELECT *
                   FROM NABAVKA N1
                   WHERE N1.Cena >= 1.10 *(SELECT N.Cena
                                           FROM NABAVKA N2
                                           WHERE N2.SifR=N1.SifR
                                            AND N2.Datum = (SELECT MAX(N3.Datum)
                                                            FROM NabavkaN3
                                                            WHERE N3.SifR=N1.SifR
                                                             AND N3.Datum<N1.Datum
                                                            )
                                             )
                   )
        )
    );
```

5. Data je šema relacione baze podataka, za potrebe izračunavanja pomoću matrica:
MATRICA(SifM, Naziv, BrVrsta, BrKolona);
PODACI(SifP, I, J, Vrednost, SifM);

Sastaviti SQL skript koji formira tabelu PODACI ukoliko je poznato da svaka matrica mora imati za svaki element po tačno jednu celobrojnu vrednost.

Rešenje:

```
CREATE TABLE PODACI
(
SifP INT PRIMARY KEY,
I INT NOT NULL CHECK (I > 0),
J INT NOT NULL CHECK (J > 0),
Vrednost INT NOT NULL,
SifM  INT NOT NULL REFERENCES MATRICA(SifM) ON UPDATE CASCADE,
UNIQUE (SifM, I, J),
CHECK (NOT EXISTS (SELECT *
                   FROM PODACI N1
                   GROUP BY SifM
                   HAVING MAX (I) <> COUNT (I)
                  )
      ),
CHECK (NOT EXISTS (SELECT *
                   FROM PODACI N1
```

```
                        GROUP BY SifM
                        HAVING MAX (J) <> COUNT (J)
        )
    )
);

CREATE ASSERTION Dimenzije
CHECK (NOT EXISTS (SELECT *
                        FROM PODACI P, MATRICA M
                        WHERE P.SifM=M.SifM
                        GROUP BY M.SifM, M.BrVrsta, M.BrKolona
                        HAVING (BrVrsta<> COUNT(P.I)) OR (BrKolona<> COUNT(P.J))
        )
);
```

Izmena strukture postojeće tabele – ALTER TABLE sintaksa

```
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|KEY} [index_name] [index_type] (index_col_name,...)
| ADD [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
| ADD [CONSTRAINT [symbol]]
UNIQUE [INDEX|KEY] [index_name] [index_type] (index_col_name,...)
| ADD [FULLTEXT|SPATIAL] [INDEX|KEY] [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|KEY} 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_option ...

index_col_name:
col_name [(length)] [ASC | DESC]

index_type:
USING {BTREE | HASH}
```

ALTER TABLE omogućava promenu strukture postojeće tabele. Na primer, moguće je dodati ili izbrisati kolone, kreirati ili uništiti indekse, promeniti tip podataka za postojeće kolone ili preimenovati

kolone ili celu tabelu.

Sintaksa za mnoge dozvoljene promene je slična sintaksi za klauzule CREATE TABLE iskaza.

U većini slučajeva ALTER TABLE radi tako što pravi privremenu kopiju originalne tabele.
Izmene se vrše na kopiji, a zatim se originalna tabela briše, a privremena tabela se preimenuje u originalnu.

Ako se koristi ALTER TABLE *tbl_name* RENAME TO *new_tbl_name* bez dodatnih opcija, MySQL jednostavno preimenuje sve fajlove koji odgovaraju tabeli *tbl_name*. U ovom slučaju nema potrebe za pravljenjem privremene tabele.
Opcije CHANGE i MODIFY zapravo su jedna te ista opcija i omogućavaju izmenu definicije kolone ili njenog mesta u tabeli.

Opcija DROP COLUMN briše kolonu iz tabele, dok opcije DROP PRIMARY KEY i DROP INDEX brišu samo indeks pridružen koloni.
Opcija DISABLE KEYS nalaže MySQL-u da ne ažurira sadržaj nejedinstvenih indeksa, ali upotrebljiva je samo za MyISAM tabele. Opcija ENABLE KEYS uključuje ažuriranje indeksa.
Opcija ORDER BY će poređati zadatim redosledom redove tabele na koju je primenjena.
Taj redosled neće biti očuvan kasnije kada se budu dodavali ili brisali podaci.

Zadaci (imajte na umu tabele iz zadatka 1 i bazu podataka poslovanje)

6.
Kreirati SQL iskaz kojim se dodaje nova kolona sa nazivom *primio* i tipom podataka VARCHAR(50) tabeli *fakture*.

Rešenje:

```
ALTER TABLE fakture
ADD COLUMN primio VARCHAR(50);
```

7. Kreirati SQL iskaz kojim se briše prethodno dodata kolona iz zadatka 6.

Rešenje:

```
ALTER TABLE fakture
DROP COLUMN primio;
```

8. Kreirati SQL iskaz kojim se menja definicija kolone *kolicina* tabele *detalji_fakture* u smislu promene podešavanja za tip podataka.

```
ALTER TABLE detalji_fakture
MODIFY COLUMN kolicina DECIMAL(6,2);
```

9. Kreirati SQL iskaz kojim se menja definicija kolone *kolicina* tabele *detalji_fakture* u smislu vraćanja na prvobitnu definiciju.

```
ALTER TABLE detalji_fakture
MODIFY COLUMN kolicina DECIMAL(8,2);
```

10. Kreirati SQL iskaz kojim se menja definicija kolone *dan_cena* tabele *detalji_fakture* tako što se naziv kolone iz *dan_cena* menja u *cena* i pored svega se menjaju podešavanja za tip podataka (u prvobitnoj definiciji je bilo DECIMAL(8,2)).

ALTER TABLE *detalji_fakture*
CHANGE COLUMN *dan_cena cena* DECIMAL(10,2);

Zadaci za vežbanje:

11. Data je šema relacione baze podataka veleprodajnog lanca prodavnica:
PRODAVNICA(SifP, Adresa, SifM);
MESTO(SifM, Naziv);
KLIJENT(SifK, Naziv, SifM);
RACUN(SifR, SifK, SifP, SifRa, Datum);
PROIZVOD(SifPr, Naziv, Cena);
STAVKA_RACUNA(SifS, SifR, SifPr, RedniBr, Kolicina, Iznos);
RADNIK(SifRa, Ime, SifP);

Sastaviti SQL skript kojim se formira tabela STAVKA_RACUNA, ukoliko je poznato da se stavke jednog računa moraju unositi redom (sukcesivni RedniBr) i da Iznos mora biti definisan količinom i cenom proizvoda na koji se posmatrana stavka odnosi. Jedan proizvod sena računu ne sme pojavljivati u više stavki.

12. Data je šema relacione deo baze podataka fakulteta:
STUDENT (SifS, Ime, BrIndeksa);
PROFESOR (SifP, Ime, SifO);
ODSEK (SifO, Naziv);
KURS (SifK, Naziv, BrKredita, SifO) UČIONICA (SifU, BrMesta);
PREDUSLOV (SifK, SifKP) POHAĐA(SifS, SifR);
RASPORED (SifR, SifP, SifK, SifU, Termin, Dan, Br.Prijavljenih);

Sastaviti SQL skript kojim se formira tabela RASPORED, ukoliko je poznato da samo jedan profesor može držati po rasporedu predavanje u jednom terminu u jednoj učionici.
Pri tom broj prijavljenih mora biti manji od broja mesta u učionici koja je rasporedu predviđena za taj kurs, a takođe vrednost atributa termin mora biti celobrojna vrednost u opsegu od 1 do 7, a vrednost atributa dan iz skupa vrednosti {Pon, Uto, Sre, Cet, Pet}.