Napredne računarske aplikacije

Predavanje broj: 03

Nastavne teme:

- INTEGRITET PODATAKA U RELACIONOM MODELU:
  Korisnička pravila integriteta; Identifikacioni integritet; Referencijalni integritet; Šema baze podataka; Redudansa; Anomalije
- NORMALIZACIJA:
  Funkcionalne zavisnosti; Prva normalna forma; Druga normalna forma; Treća normalna forma
- SQL - DDL:
  Create; Alter;

INTEGRITET PODATAKA U RELACIONOM MODELU

- **Integritet podataka** – konzistentnost baze podataka podrazumeva sve **mere kojima je cilj da se spreči unos neispravnih podataka.**
- **Pravila integriteta** su **ograničenja sadržaja baze podataka** i definišu se u okviru same baze podataka.
- Pravila integriteta delimo na:
  - **Korisnička pravila** integriteta;
  - **Identifikacioni** integritet;
  - **Referencijalni** integritet.

Korisnička pravila integriteta

- To su pravila koja se odnose na **ograničenja koja važe za pojedine atribute**
- **Integritet kolone / integritet domena.**
- Tip podatka (char, int, date..), dužina podatka (char(20)), uslov.
- Uslov:
  - Ocene iz pojedinih predmeta od 5 do 10.
  - Studentu mora da se unesu sve ocene, da bi pristupio odbrani diplomskog rada itd.

Identifikacioni integritet

- Odnosi se na opšta ograničenja za primarni ključ relacije.
- **Identifikacioni integritet** se odnosi na ograničenje da **ni jedna komponenta primarnog ključa relacije ne sme imati NULL vrednost.**
- Identifikacioni integritet predstavlja **integritet entiteta**.

Referencijalni integritet

- Suština referencijalnog integriteta je u **ograničavanju vrednosti spoljnog ključa**.
- **Ograničenja** sa stanovišta izmena **u relaciji koja sadrži spoljni ključ** (pozivajuća relacija):
  - Ne može se **uneti** zapis(instanca) ako vrednost stranog ključa ne odgovara nekoj od vrednosti primarnog ključa u ciljnoj relaciji ili NULL vrednosti;
  - Ne može se **izmeniti** zapis(instanca) ako vrednost stranog ključa ne odgovara nekoj vrednosti primarnog ključa u ciljnoj relaciji ili NULL vrednosti.

Referencijalni integritet

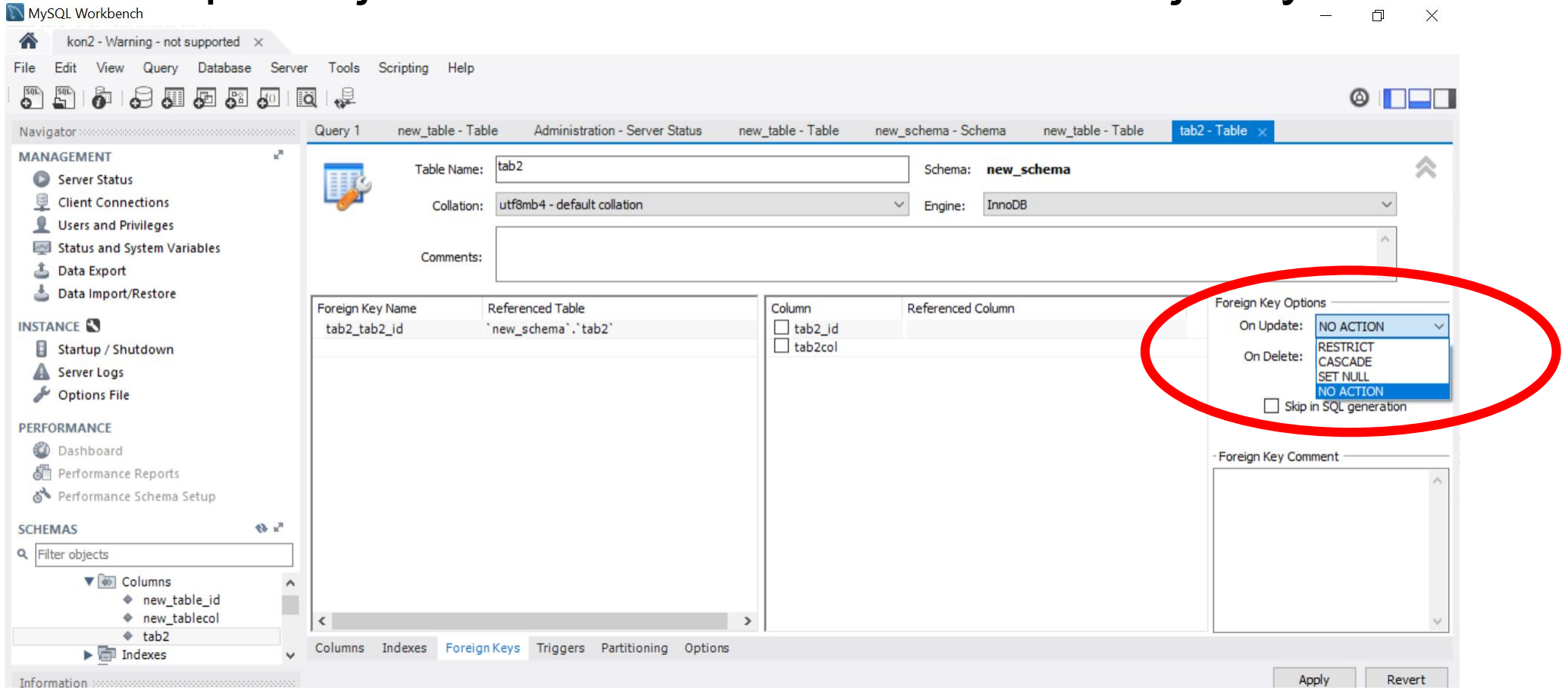
- **Ograničenja** sa stanovišta ciljne relacije – **relacije na koju ukazuje spoljni ključ** :
  - Dodavanje nove instance (u ciljnoj relaciji) ne narušava referencijalni integritet - nastaje samo nova vrednost primarnog ključa.
  - Uklanjanje instance dovodi do nestanka jedne vrednosti primarnog ključa. Ako bi se ta operacija izvršavala bezuslovno to bi narušilo referencijalni integritet.

Dinamička specifikacija referencijalnog integriteta

- Kako postupiti kada se vrši ažuriranje u ciljnoj relaciji, a da se ne naruši referencijalni integritet?
- Takva specifikacija se naziva **dinamička specifikacija referencijalnog integriteta.**
- Odnosi se na **kritične operacije**: operacija uklanjanja (**DELETE**) i operacija izmene (**UPDATE**).
- Neophodno je navesti jednu od sledeće tri klauzule: **RESTRICTED**, **CASCADES**,**NULLS**

Dinamička specifikacija referencijalnog integriteta

Postupak postavljanja opcija stranog ključa definisanjem specifikacija za kritične operacije UPDATE i DELETE u okruženju MySQL DBMS.



Dinamička specifikacija referencijalnog integriteta

- **RESTRICTED**: operacija se ne izvršava ako u pokaznoj relaciji postoji vrednost spoljašnjeg ključa koja odgovara vrednosti primarnog ključa u ciljnoj relaciji.
- **CASCADES**: operacija se uvek izvršava. Ako je uklonjena instanca iz ciljne relacije u pokaznoj relaciji se uklanjaju sve instance sa datim ključem. Ako je došlo do izmene, menjaju se sve vrednosti instanci u pokaznoj relaciji sa novim spoljašnjim ključem.
- **NULLS**: operacija se uvek izvršava. U pokaznoj relaciji se u svim instancama koje imaju promenjeni spoljašnji ključ, menja njegova vrednost u NULL.
  - **NULLS ne može ako je spoljašnji ključ u sastavu primarnog ključa!!!**

Struktura šeme baze podataka

- Šema ili **šema baze podataka** (engl. *database schema)* označava **strukturu** ili dizajn **baze podataka**, odnosno oblik koji baza podataka ima i **ne odnosi se na same podatke**.
- Šema je "nacrt strukture" podataka koji se čuvaju u bazi.

Vaš cilj ka projektanta i dizajnera baze podataka je da „skladištite podatke“ na **jednom mestu** koje je ujedno i **najbolje mesto**.

Struktura šeme baze podataka

- Kada projektujemo bazu podataka, moramo razmotriti dva važna pitanja:
  - Koje sve podatke treba čuvati, odnosno o kojim sve stvarima ili entitetima moramo da čuvamo podatke?
  - Koju ćemo vrstu pitanja postavljati bazi podataka? (Pitanja bazi podataka zovu se *upiti.)*
- Kada tražimo odgovore na ova pitanja, moramo uzeti u obzir poslovna pravila organizacije koju pokušavamo da modelujemo i koje su veze između tih podataka.
- Na osnovu odgovora na ova pitanja moramo da napravimo **bazu podataka** čija će **struktura** biti takva da **isključuje** strukturne probleme kao što su **redundantnost (dupliranje) podataka** i **anomalije podataka**.
- Ako neka baza podataka zadovoljava karakteristike dobrog dizajna ona ima **dobru strukturu**, tj. **u "dobroj" je formi**.

Redudantnost i gubljenje podataka

- **Redundantni podatak** znači da **se isti podatak ponavlja** u različitim redovima iste tabele ili u različitim tabelama baze podataka.
- Dobar dizajn baze podataka podrazumeva **svođenje redundantnosti** podataka **na minimum**, ali da pri tome **ne izgubimo nijedan podatak koji je nephodan**.

Šema relacije loše strukture

- Primer 1
  U jednoj šemi relacije se nalaze imena radnika, posao, naziv i lokacija odeljenja u kojima rade.

  Radnik (radnikID, ime, posao, odeljenjeID, nazivOdeljenja, mesto)

<radnikID, odeljenjeID> čine primarni ključ
  – Jedan radnik radi u jednom odeljenju.
  – U jednom odeljenju radi više radnika.

Šema relacije loše strukture

radnikID	ime	posao	odeljenjeID	nazivOdeljenja	mesto
1	Lazar	programer	10	IT sektor	Voždovac
2	Luka	vozač	20	Logistika	Novi Beograd
3	Aca	DBA dizajner	10	IT sektor	Voždovac

- Neka u jednom trenutku postoji prethodni sadržaj
- U budućnosti, ako se u odeljenju IT sektor ili Logistika zaposli još radnika onda:
  <10 IT sektor Voždovac > i <20 Logistika Novi Beograd > bi se pojavljivao više puta

Šema relacije loše strukture

radnikID	ime	posao	odeljenjeID	nazivOdeljenja	mesto
1	Lazar	programer	10	IT sektor	Voždovac
2	Luka	vozač	20	Logistika	Novi Beograd
3	Aca	DBA dizajner	10	NULL	NULL
4	Ana	administrator	10	NULL	NULL
5	Mirko	magacioner	20	NULL	NULL

- Ako se pokuša izbegavanje unosa naziva i mesta odeljenja više puta (samo prvi put se upisuje, a u ostalim slučajevima NULL), gube se neke informacije
- Izbegavanje unosa za atribut odeljenjeID nije moguće jer predstavlja deo primarnog ključa
- Nemogućnost postavljanja upita:
  - Npr: Naziv odeljenja i mesto u kome rade radnici Lazar, Ana i Mirko.
  - Npr: Naziv odeljenja i mesto za radnike koji obavljaju posao DBA dizajnera.

Šema relacije loše strukture

- Osnovni nedostatak relacije Radnik je:
  - **redudansa** – višestruko ponavljanje u relaciji
- Ovaj nedostatak izaziva probleme kod sva tri vida ažuriranja relacije:
  - **Višestruko unošenje**: naziv odeljenja se unosi se onoliko puta koliko ima radnika u tom odeljenju
  - **Višestruko menjanje**: eventualne promena naziva odeljenja vrši se na svim mestima
  - **Višestruko uklanjanje**: ako se želi potpuno uklanjanje podataka o odeljenju, vrši se onoliko puta koliko radnika je u tom odeljenju

Anomalije

- Anomalije su **problemi** koji se pojavljuju **među podacima zbog loše osmišljene strukture** baze podataka.

- Mogu se pojaviti tri vrste anomalija.
  - Anomalije pri dodavanju podataka
  - Anomalije pri brisanju podataka
  - Anomalije pri ažuriranju podataka

Anomalije pri dodavanju podataka

- **Anomalije pri dodavanju podataka** nastaju kada se u relaciju (tabelu) s loše osmišljenom strukturom **pokuša unos novih podataka**.
- Zamislite da imamo novog zaposlenog koji počinje da radi u preduzeću. Kada njegove podatke unosimo u tabelu *Radnici*, moramo da unesemo i id, naziv odeljenja u kome radi kao i lokaciju istog.
- Šta se događa ako unesemo podatke koji se razlikuju od postojećih u tabeli, npr. da za novog zaposlenog koji radi u odeljenju čiji je id=10 upišemo da je naziv odeljenja Računovodstvo ili da je lokacija odeljenja Novi Beograd?
- Neće biti jasno u kojim redovima su upisani ispravni podaci.

Anomalije pri brisanju podataka

- **Anomalije pri brisanju podataka** nastaju kada se **podaci brišu** iz tabele s loše osmišljenom strukturom.
- Zamislite da svi zaposleni iz odeljenja čiji je id 10 daju otkaz. Pošto izbrišemo sve zapise o tim zaposlenima, više nemamo nijedan zapis sa podacima o odeljenju sa id 10.
- Drugi slučaj bi bio ukoliko se neko odeljenje ugasi. Ako obrišemo redove sa podacima o tim odeljenjima obrisaćemo i podatke o radnicima iako i dalje rade u preduzeću ali su prebačeni u druga odeljenja.
- Ovo su primeri anomalije pri brisanju.

Anomalije pri ažuriranju podataka

- **Anomalije pri ažuriranju podataka** nastaju kada se **ažuriraju podaci** u tabeli s loše osmišljenom strukturom.
- Zamislite da je potrebno promeniti naziv odeljenja čiji je id=20 iz Logistika u Transport. U tom slučaju moramo izmeniti podatke pojedinačno za svakog zaposlenog koji radi u tom odeljenju. Lako se može dogoditi da nekog izostavimo.
- Ako se to desi, imaćemo anomaliju pri ažuriranju.

Funkcionalne zavisnosti

- Ako u relaciji postoji **funkcionalna zavisnost** između atributa A i atributa B onda se može reći da vrednost atributa A određuje vrednost atributa B i označava se kao:

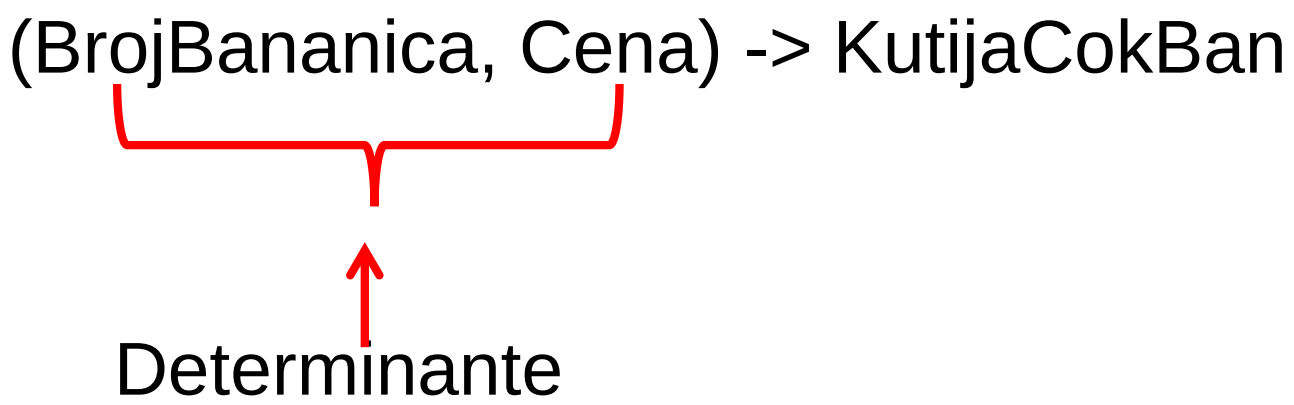
  A->B

- **Funkcionalna zavisnost** predstavlja **povezanost između atributa** u kojoj jedan atribut ili grupa atributa **određuje** vrednost drugog atributa u istoj tabeli.
- Vrednost jednog atributa može se upotrebi kako bi se pronašla vrednost drugih atributa.
- Primer: Cena jedne čokoladne bananice i broj komada u kutiji određuje cenu kutije čokoladnih bananica:

  (BrojBananica, Cena) -> KutijaCokBan

Funkcionalne zavisnosti

- Atribut (grupa atributa) koja se nalazi **sa leve strane** i predstavljaju početnu tačku funkcionalne zavisnosti naziva se **odrednica** (**determinanta**)



Kandidat/primarni ključ i funkcionalne zavisnosti

- Po definiciji....
  Kandidat ključ u tabeli treba da funkcionalno odredi ostale ne – ključne atribute u redu.

- Kako kandidat ključ može da bude izabran da bude primarni ključ onda važi...
  Primarni ključ u tabeli treba da funkcionalno odredi ostale ne – ključne atribute u redu.

Kandidat/primarni ključ i funkcionalne zavisnosti

(RadnikID) -> (PrezimeRadnika, ImeRadnika)

(ProjekatID) -> (NazivProjekta, DatumPocetka)

Normalizacija podataka

- **Normalizacija** predstavlja:
  - **Postupak analiziranja relacije** sa ciljem **ispravljanja loše strukture** i
  - Obezbeđivanje da relacija bude **dobro formirana**.

- Dobro formirana relacija (engl.well-formed) nije osetljiva na tri tipa anomalija.

- Normalizacija uključuje postupak razbijanja relacije na manje dobro formirane realacije.

Principi normalizacije

- Principi relacionog dizajna za normalizaciju relacija:
  - Da bi relacija bila dobro formirana, svaka determinanta mora biti kandidat ključ.
  - Svaka relacija koja nije dobro formirana mora biti razbijena u jednu ili više manjih dobro formiranih relacija.

- Dobro formirana relacija ne treba da obuhvata više od jednog poslovnog koncepta.

Primer normalizacije

(StudentID) -> (ImeStudenta, NazivDoma,CenaDoma)

(NazivDoma) -> (CenaDoma)

CenaDoma treba da bude u posebnoj relaciji

Rezultat normalizacije:

(StudentID) -> (ImeStudenta, NazivDoma)

(NazivDoma) -> (CenaDoma)

Normalizovane tabele:

StudentID	ImeStudenta	ID_doma
1	Miloš	10
2	Ana	10

StudentID - primarni ključ tabele Student

ID_doma – spoljni ključ tabele Student

ID_doma	NazivDoma	CenaDoma
10	4.april	500
20	Studenjak	1000

ID_doma - primarni ključ tabele Dom

Primer normalizacije

(AdvokatID, KlijentID) -> (ImeKlijenta,DatumSastanka,Trajanje)

(KlijentID) -> (ImeKlijenta)

ImeKlijenta treba da bude u posebnoj relaciji

Rezultat normalizacije:

(AdvokatID, KlijentID) -> (DatumSastanka,Trajanje)

(KlijentID) -> (ImeKlijenta)

Normalizovane tabele:

AdvokatID	Klijent_Id	DatumS	Trajanje
1	10	02.02.2019	60
2	20	07.08.2019	45

AdvokatID, KlijentID - primarni ključ tabele Sastanak
Klijent_Id - spoljni ključ tabele Sastanak

Klijent_Id	ImeKlijenta
10	Petar
20	Miloš

Klijent_Id - primarni ključ tabele Klijent

Koraci normalizacije

Prva normalna forma (1NF)

- **Prva normalna forma**, koja se naziva i 1NF, propisuje da **nema viševrednosnih atributa** u tabeli.
- Vrednost svakog atributa mora biti atomična.
- Svaki atribut mora sadržati samo jednu vrednost, a ne kombinaciju više vrednosti ili drugi red baze podataka.
- Da biste proverili **da li je tabela u 1NF**, potvrdite da **svaki atribut** ima **samo jednu vrednost za svaku instancu** (red, zapis).
- Sve relacije, po definiciji, su u 1NF.
- Ako tabela ispunjava specifične karaketristike koje se odnose na relaciju onda je u 1NF.

Tabela sa viševrednosnim atrubutima – nije u 1NF

PorID	DatumP	KupacID	ImeKupca	ArtikalID	ArtikalNaziv	Cena	Količina
1006	21.10.2018.	22	Marković Luka	7 5 8	Olovka Sveska Blok	50 din 120 din 70 din	5 2 1
1007	07.07.2019.	29	Lazović Ivana	10 1	Rezač Naliv pero	100 din 450 din	2 1

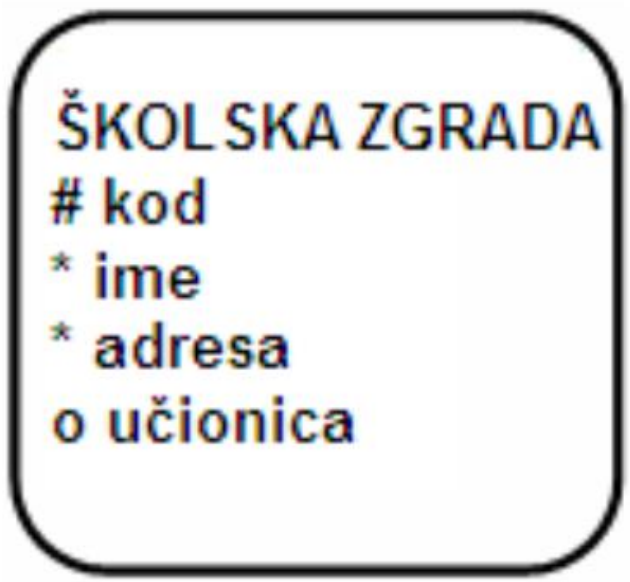
Tabela bez viševrednosnim atrubutima – u 1NF

PorID	DatumP	KupacID	ImeKupca	ArtikalID	ArtikalNaziv	Cena	Količina
1006	21.10.2018.	22	Marković Luka	7	Olovka	50 din	5
1006	21.10.2018.	22	Marković Luka	5	Sveska	120 din	2
1006	21.10.2018.	22	Marković Luka	8	Blok	70 din	1
1007	07.07.2019.	29	Lazović Ivana	10	Rezač	100 din	2
1007	07.07.2019.	29	Lazović Ivana	1	Naliv pero	450 din	1

Tabela je u 1NF ali je organizacija podataka daleko od savršene jer imamo veliku redudantnost podataka – ponavljaju se podaci PorID, DatumP, KupacID i ImeKupca za svaku stavku porudžbine.

Entitet sa viševrednosnim atrubutom – nije u 1NF

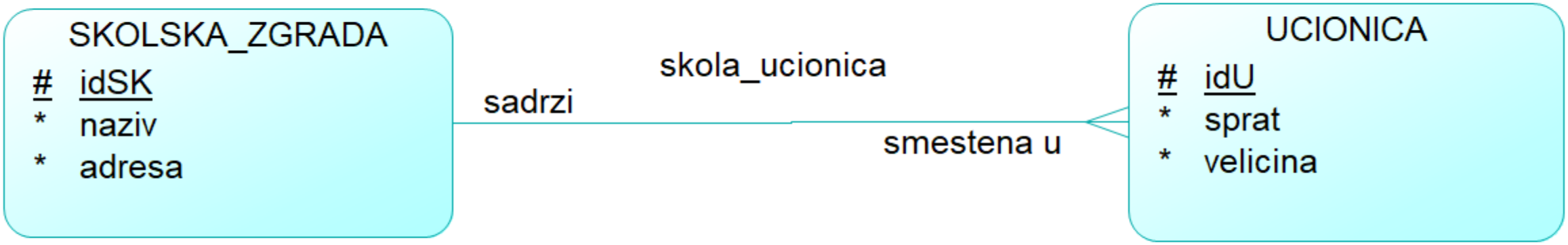
Postoji mnogo učionica u zgradi škole, dakle to je atribut ima više vrednosti za jednu instancu entiteta zgrada.
Atribut učionica će imati viševrednosti za jednu školsku zgradu.



Kod	Ime	Adresa	Učionica
123	VIŠER	Vojvode Stepe 283	SL SD 106 406
158	FON	Bulevar JNA	510 305 213

Entitet sa viševrednosnim atrubutom – rešenje

Ako entitet ima atribut koji može da ima više vrednosti za jednu instancu treba napravite novi entiteti i povezati ga sa početnim entitetom pomoću veze više



UCIONICA je sada poseban entitet. **Svi atributi imaju samo jednu vrednost u instanci**. **Oba entiteta** su u **prvoj normalnoj formi**.
Svaka ŠKOLSKA ZGRADA mora da sadrži jednu ili više UČIONICA.
Svaka UČIONICA mora da bude smeštena u jednoj SKOLSKOJ_ZGRADI

Tabela sa viševrednosnim atrubutom – nije u 1NF

radnikID	ime	posao	odeljenjeID	veštine
1	Lazar	programer	10	Java, C, PHP
2	Luka	Web dizajner	20	HTML, CSS, Photoshop
3	Aca	DBA dizajner	10	Modelovanje, SQL Server, SQL, Mongo
4	Ana	administrator	20	Linux, Windows

Tabela Radnici nije u 1NF - svaka vrednost u koloni veštine sastoji se od kombinacije drugih vrednosti, umesto da sadrži jednu prostu vrednost.

Tabela sa viševrednosnim atrubutom – rešenje

Napravili smo tri tabele: Radnik, Radnik_Vestine, Vestine kako bi rešili vezu M:N. Radnik može imati više veština. Veštinu ima više radnika. Tabele su povezane pomoću spoljnih ključeva koji su u tabeli Radnik_Vestine.

U tabeli Radnik_Vestine:

- primarni ključ je složen (radnikID,vestinaID)
- radnikID je spoljni ključ koji ukazuje tabelu Radnik.
- vestinaID spoljni ključ koji ukazuje na tabelu Vestine

radnikID	ime	posao
1	Lazar	programer
2	Luka	Web dizajner
3	Aca	DBA dizajner
4	Ana	administrator

radnikID	vestinaID
1	10
1	20
1	30
2	50
2	60
3	90
3	100

vestinaID	veštine
10	Java
20	C
30	PHP
40	HTML
50	CSS
60	Photoshop
70	Modelovanje
80	SQL Server
90	SQL
100	Mongo

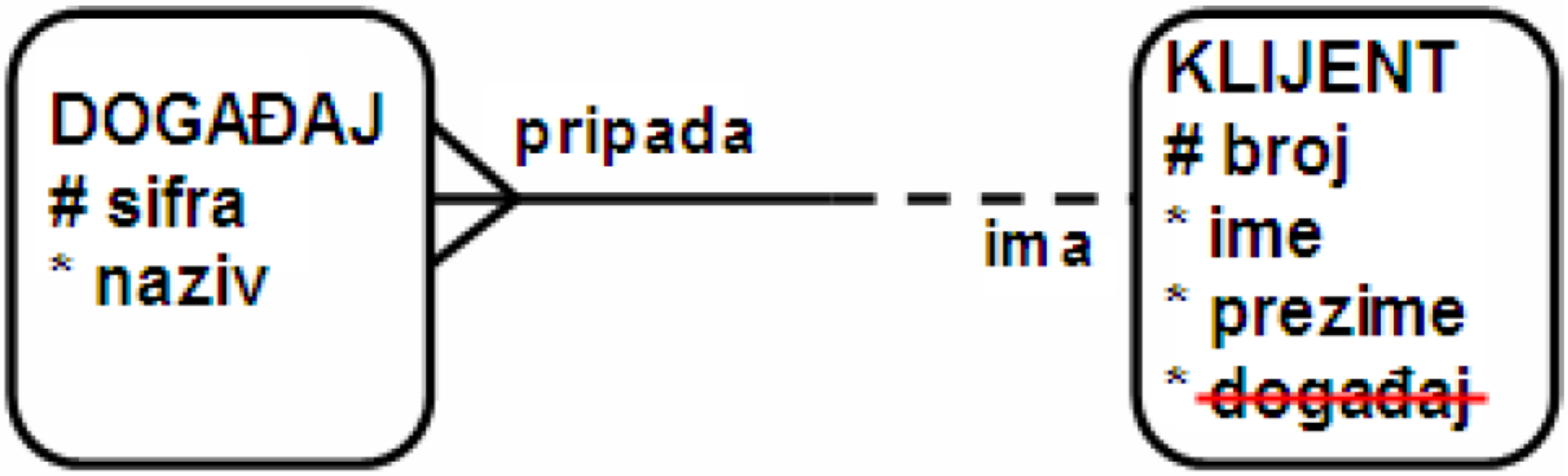
Entitet sa viševrednosnim atrubutom – nije u 1NF

Na slici je prikazan entitet KLIJENT sa atributima broj, ime, prezime i događaj koji je vezan za klijenta

Atribut događaj narušava prvu normalnu formu jer za jednog klijenta može biti vezano više događaja (rođendan, krštenje, godišnjica i sl...)

Entitet sa viševrednosnim atrubutom – rešenje

Atribut događaj postaje entitet DOGAĐAJ koji je povezan sa KLIJENTOM preko veze tipa N:1.



Druga normalna forma (2NF)

Druga normalna forma zahteva:

- da je zadovoljena prva normalna forma
- da svaki ne – ključni atribut (koji nije jedinstveni identifikator) je u potpunosti funkcionalno zavistan od **CELOG PRIMARNOG KLJUČA** (jedinstvenog identifikatora).

- Svaki neključni atribut mora da bude definisan i određen sa celim, a ne delom primarnog ključa.

- Nema delimičnih zavisnosti.

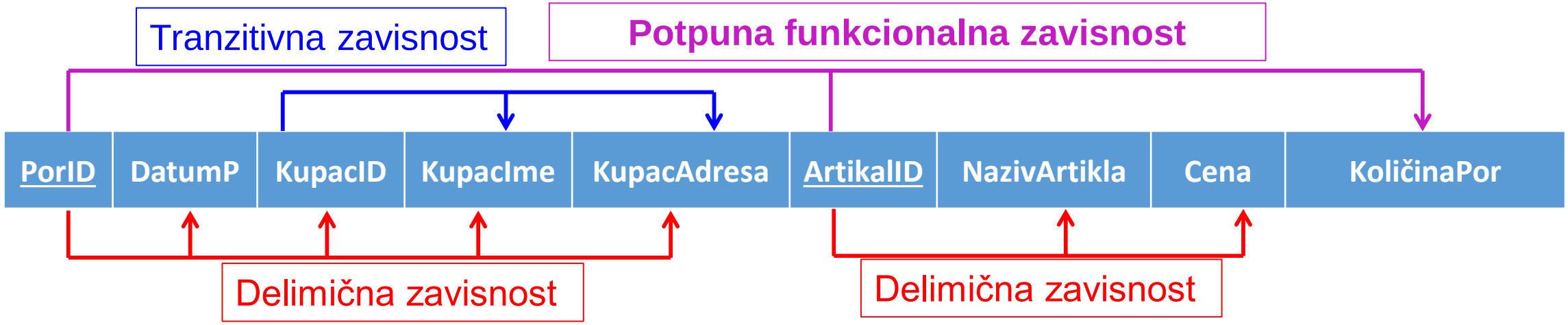
Druga normalna forma (2NF)

Svi atributi koji nisu deo PRIMARNOG KLJUČA (jedinstvenog identifikatora) moraju zavisiti od celog UID.

- To se posebno odnosi na entitete koji imaju složeni primarni ključ koji se sastoji od više atributa ili kombinacije atributa i veze (veza).To se zove **potpuna funkcionalna zavisnost** od primarnog ključa.

- Dakle, druga normalna forma zahteva potpunu funkcionalnu zavisnost od jedinstvenog identifikatora.

Dijagram zavisnosti



Relacija ima složeni primarni ključ: <PorID, ArtikalID>

Potpuna funkcionalna zavisnost

PorID, ArtikalID ->DatumP, KupacID, KupacIme,KupacAdresa, NazivArtikla, Cena, KoličinuPor

Delimične zavisnosti:

ArtikalID ->NazivArtikla, Cena

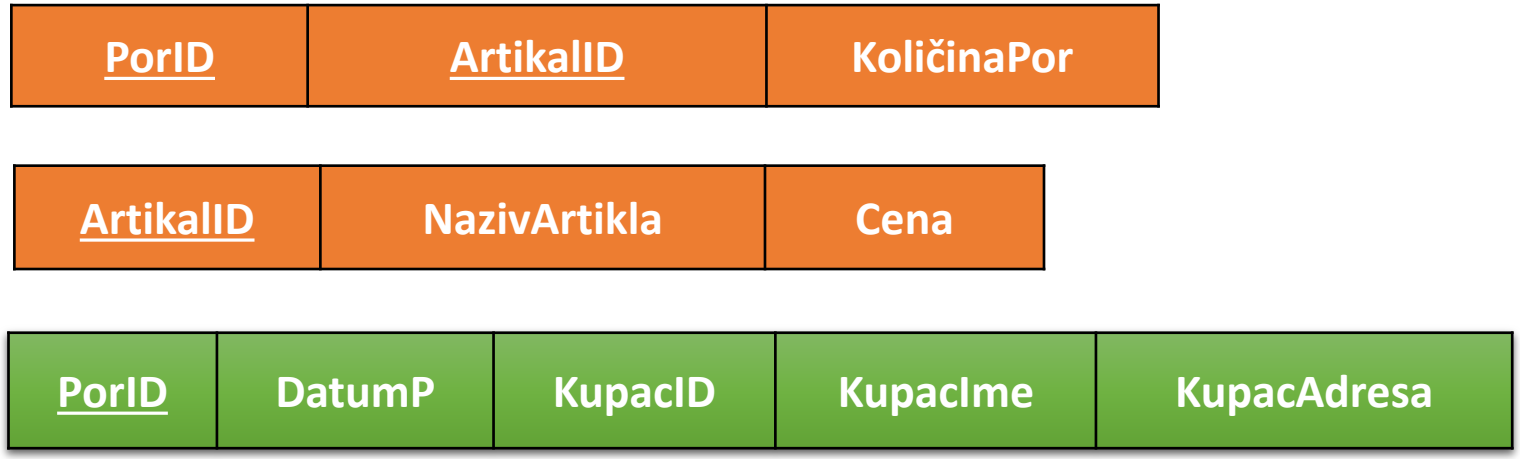
PorID -> DatumP, KupacID, KupacIme, KupacAdresa

Tranzitivna zavisnost:

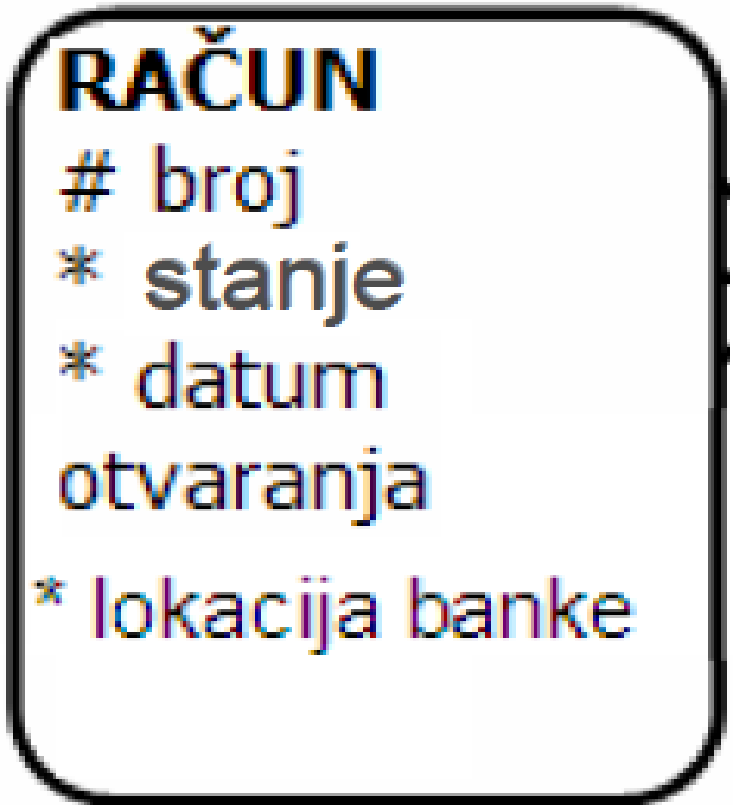
PorID->KupacID -> KupacIme, KupacAdresa

Uklanjanje delimičnih zavisnosti

- Strategija za prelazak relacije u 2NF podrazumeva da se iz relacije koja je u 1NF **uklone** **delimične funkcionalne zavisnosti.**
  - Razbijanje relacije u manje relacije



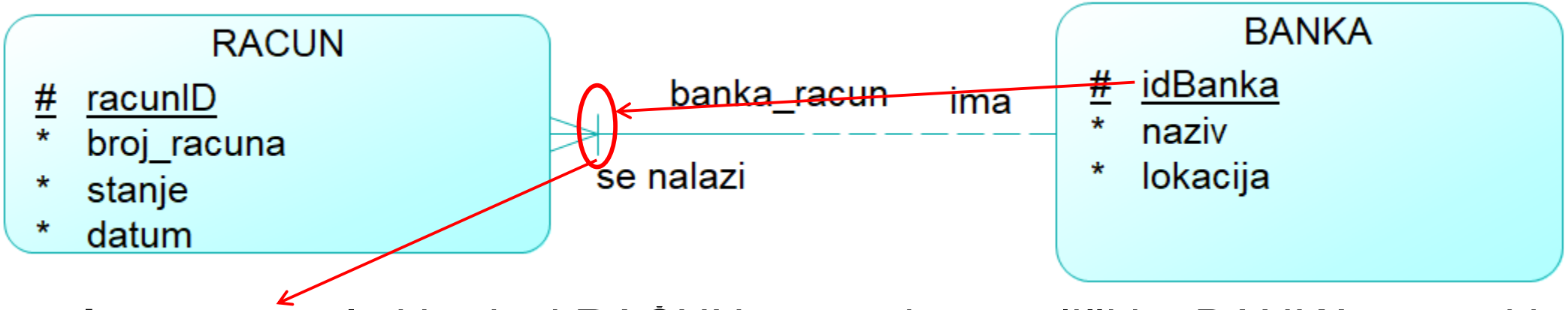
Druga normalna forma - Primer



U kojoj normalnoj formi je enitet RACUN?

Druga normalna forma - Primer

- Atribut lokacija banke zavisi SAMO od naziva banke.
- Lokacija banke nije modelovana na pravom mestu.
- Ovo je narušavanje druge normalne forme.
- Mesto atributa lokacija je u entitetu BANKA

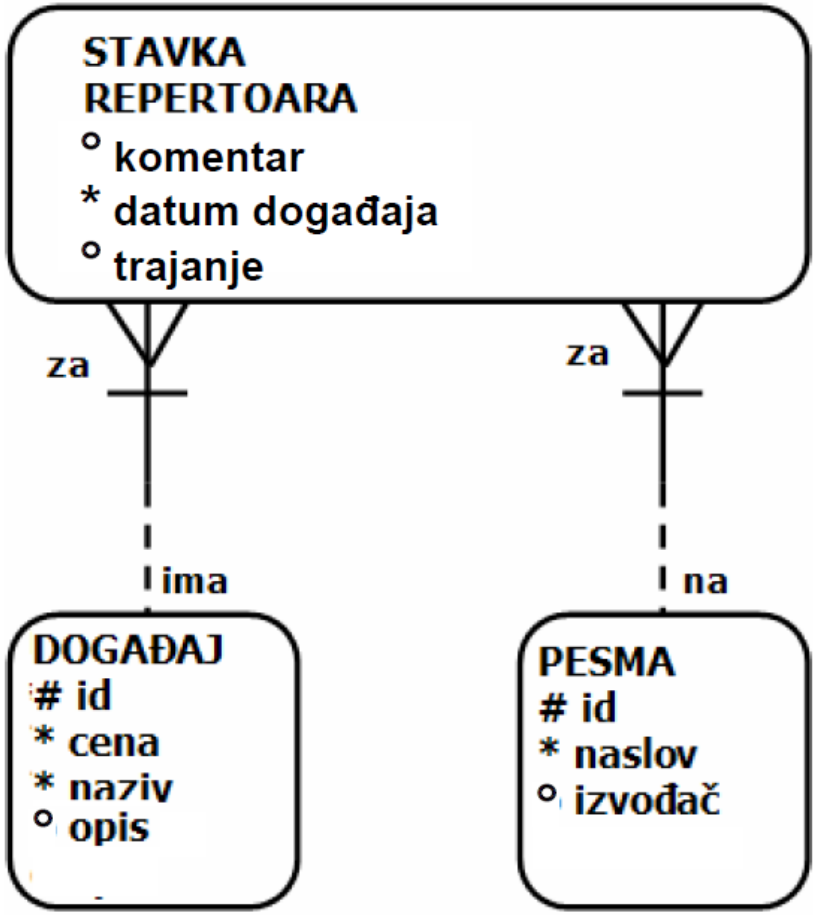


Barirana veza: Isti brojevi RAČUNa postoje u različitim BANKAma pa i je veza deo jedinstvenog identifikatora.

Druga normalna forma - Primer

Za svaki događaj, DJ je odgovoran za pripremanje kolekcije pesama koja će biti izvedena na tom događaju.

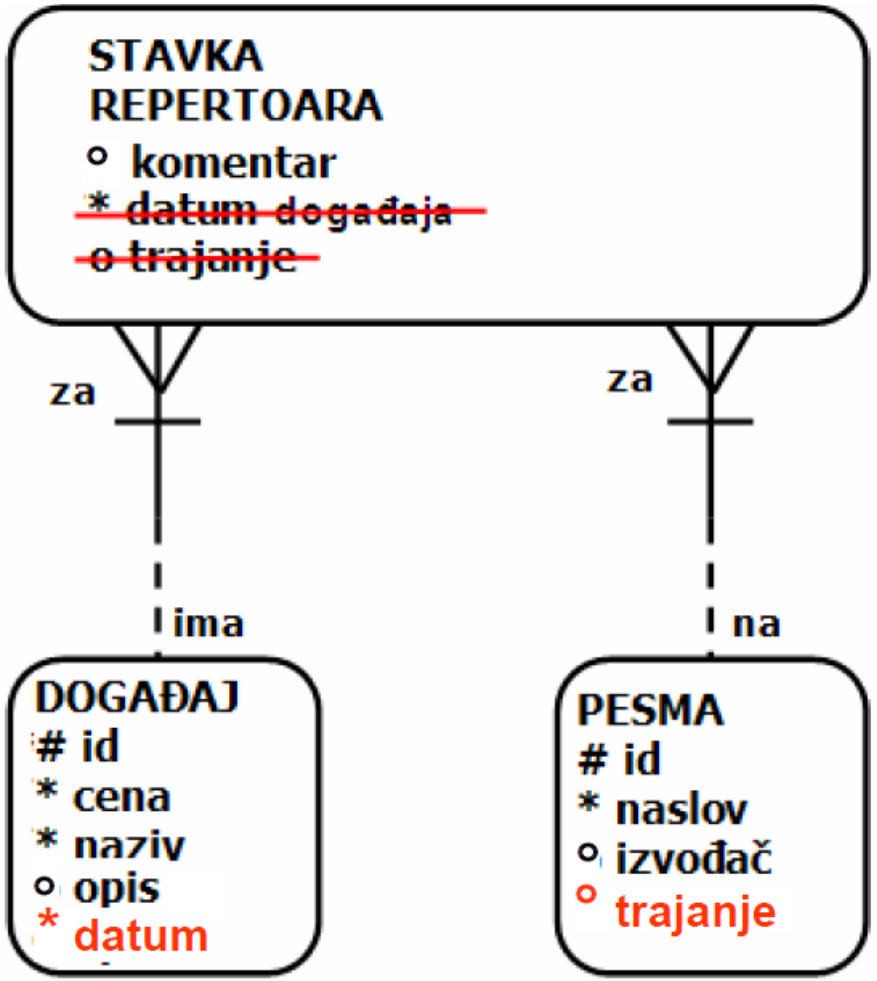
Svaka pesma može biti puštena na više od jednog događaja i na svakom događaju će biti pušteno više od jedne pesme.



Druga normalna forma - Primer

- Model ćemo normalizovati izmeštanjem atributa koji narušavaju II NF.

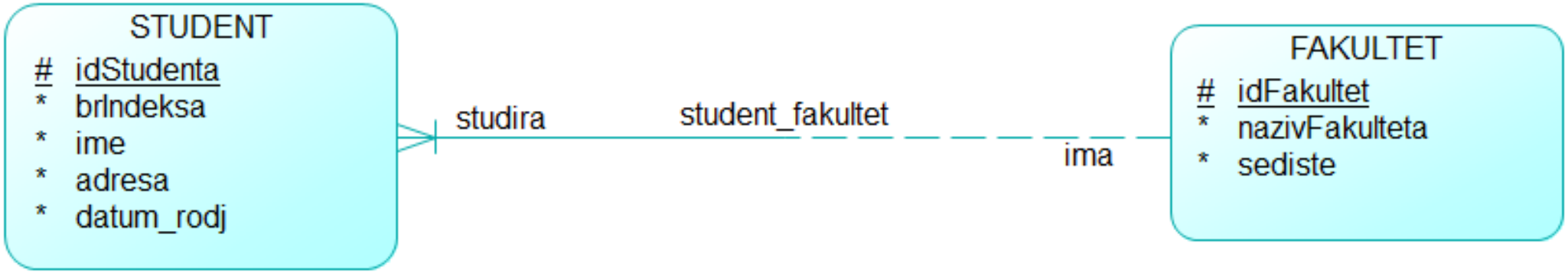
- **Izmeštanjem** atributa **datum** u entitet DOGAĐAJ i atributa **trajanje** u entitet PESMA preveli smo model u drugu normalnu formu.



Druga normalna forma - Primer

Druga normalna forma - Primer

- Postojanje **delimične zavisnosti** nazivFakulteta ->sediste
- Razbijanje entiteta STUDENT.
- Drugi entitet je FAKULTET.
- Atributi nazivFakulteta i sediste između kojih je utvrđena delimična zavisnost prebacuju se u entitet FAKULTET.

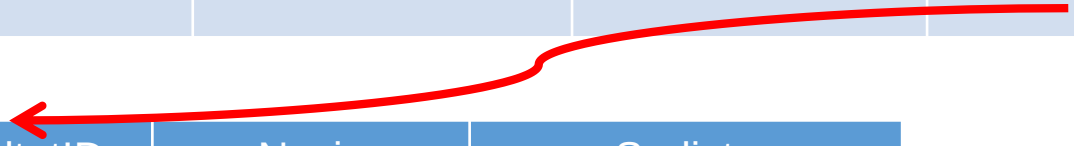


- Svaki STUDENT *mora* da *studira* jedan i samo jedan FAKULTET.
- Svaki FAKULTET *može* da *ima* jednog ili više STUDENTa.

Druga normalna forma

Relaciona baza podataka - implementacija

StudentID	BrIndeksa	Ime	Adresa	Datum_rodj	FakultetID
1	RT-123/13	Lazović Ivana	Višnjevačka 12	12.12.1978	11
2	RT-123/12	Mirković Mirko	Radnička 13	02.02.1989	14
3	TF-123/13	Milić Ana	Moračka 1	03.03.1989	12



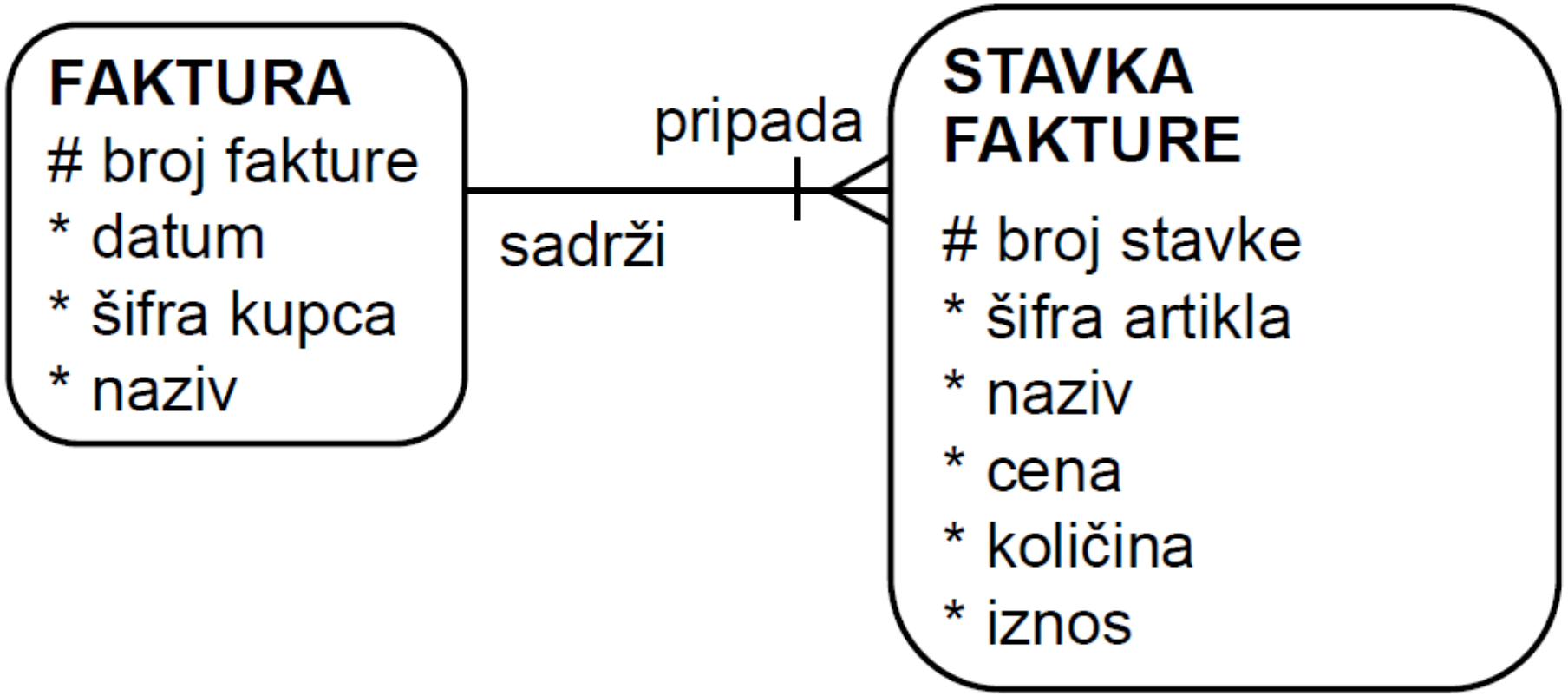
FakultetID	Naziv	Sediste
11	VIŠER	Beograd
12	FON	Beograd
13	ETF	Beograd
14	EI	Niš

Treća normalna forma

- Da bi entitet bio **u trećoj normalnoj formi**, treba da bude u 2NF i da zadovoljava pravilo 3NF.
- Treća normalna forma **zabranjuje** tranzitivnu zavisnost.
- **Tranzitivna zavisnost** predstavlja funkcionalnu zavisnost atributa od bilo kog drugog ne-ključnog atrubuta u tabeli (entitetu).
  - Ove zavisnosti se zovu tranzitivne jer je primarni ključ determinanta za atribut koji je determinanta za jedan ili više drugih atributa.

- Rešenje: Ne-ključni atributi (determinante) sa tranzitivnom zavisnošću prelaze u novu tabelu (entitet); Ne-ključni atributi determinante postaju primarni ključ u novoj tabeli, a ostaju kao strani ključ u staroj (polaznoj) tabeli.

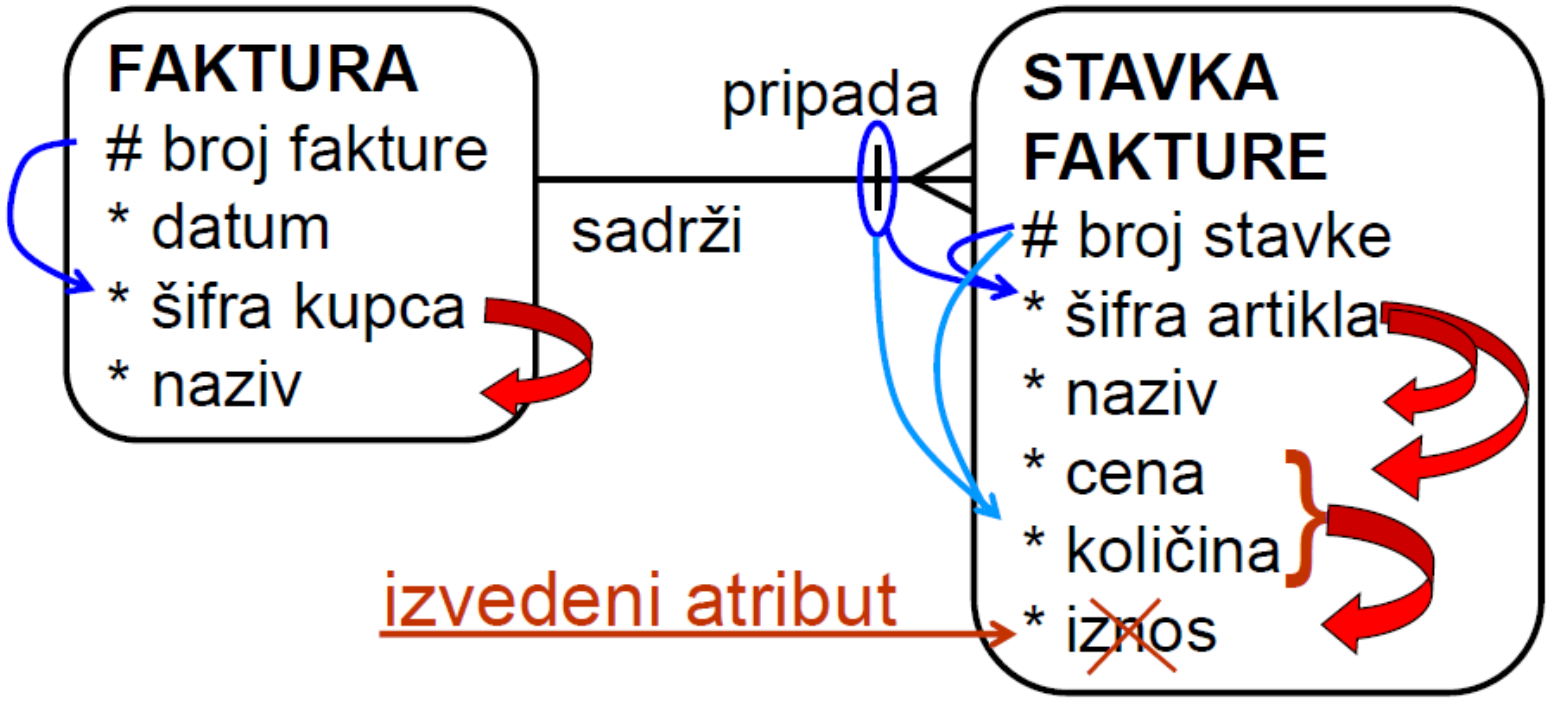
Treća normalna forma



Treća normalna forma

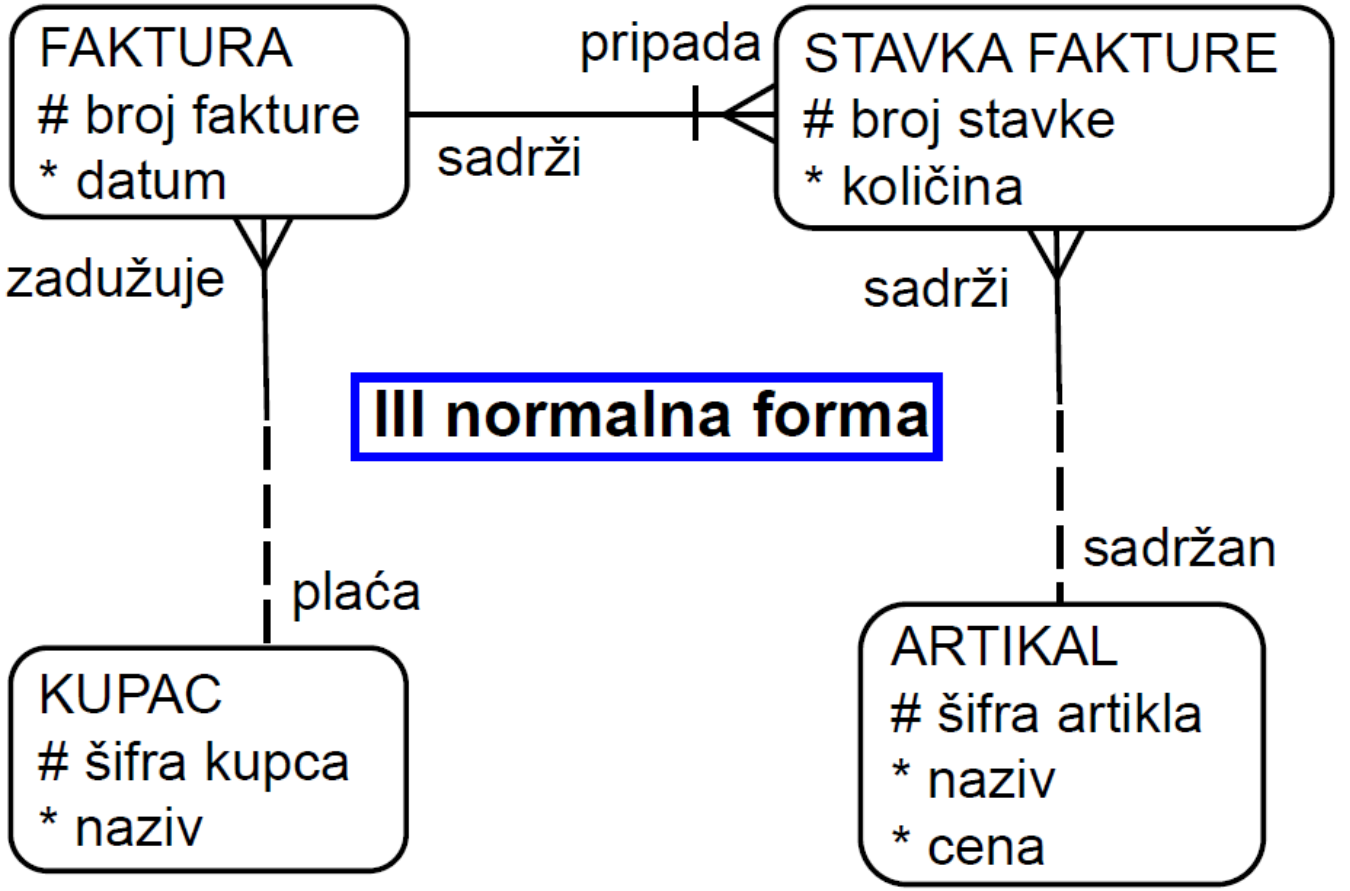
Oba entiteta su u 1NF ali nisu
u 2NF pa nisu ni u 3NF.

Delimične zavisnosti
sifra kupca ->naziv
sifra artikla ->naziv, cena
cena, kolicina -> iznos



Tranzitivne zavisnosti
Broj fakture ->sifra kupca ->naziv
Broj fakture, broj stavke ->sifra artikla->naziv, cena
Broj fakture, broj stavke ->kolicina, cena->iznos

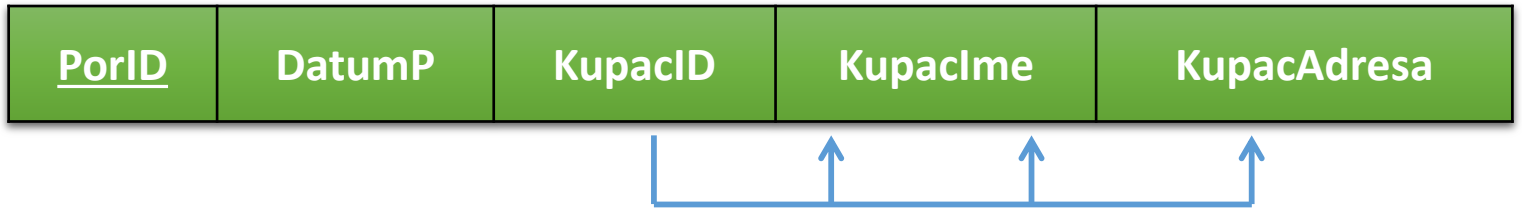
Treća normalna forma



Treća normalna forma

StavkePorudzbine (3NF)

Artikli (3NF)



Kupac_Porudzbenice (2NF) ali nije u 3NF

Tranzitivna zavisnost
PorID->KupacID->KupacIme,KupacAdresa

Porudzbinice (3NF)

Kupci (3NF)

Treća normalna forma

radnikOdeljenje(radnikID, ime, posao, odeljenjeID, nazivOdeljenja)

Atribut odeljenjeID ne ulazi u sastav ključa.

Funkcionalne zavisnosti:

radnikID->ime, posao, odeljenjeID, nazivOdeljenja

odeljenjeID -> nazivOdeljenja

Međutim, imamo i:

radnikID -> odeljenjeID

odeljenjeID -> nazivOdeljenja

Funkcionalna zavisnost *radnikID -> nazivOdeljenja* je *tranzitivna* jer postoji međukorak odeljenjeID -> nazivOdeljenja (ne- ključni atribut determiniše drugi atribut).

Treća normalna forma

Rešenje – da bi smo došli do 3NF.

Šemu radnikOdeljenje delimo na dve tabele, radnik i odeljenje:

radnik(radnikID, ime, posao, odeljenjeID)
odeljenje (odeljenjeID, nazivOdeljenja)

SQL - uvod

Create

Read

Update

Delate

SQL - uvod

- Standard Query Language
- Akronim: SQL
- Orginalno razvijen 1970.god. od strane IBM-a kao SEQUEL jezik.
- Razvijen kao podrška relacionom modelu E.Codda
- Zasnovan na relacionoj algebri
- ANSI / ISO standard
- Transact-SQL (T-SQL) podržava ANSI standard SQL-92.

SQL - uvod


Microsoft
SQL Server
Linux
IBM
DB2
Microsoft
Windows
android
UNIX®
MySQL
iOS
Mac
ORACLE

SQL - uvod

- SQL nije programski jezik, više je podjezik podataka
- SQL se sastoji od:
  - DDL – Data definition

    Definisanje resursa i logičkog modela relacione baze podataka. Tu spadaju naredbe za kreiranje tabela, pogleda, indeksa, naredba za izmenu definicije tabele, naredbe za uklanjanje fizičke tabele ili pogleda
  - DML –Data manipulation

    Ažuriranje podataka (izmenu, dodavanje, brisanje) i izveštavanje (izdavanje postojećih i izračunavanje novih informacija) iz baze podataka.
  - DCL – Data control

    Za kreiranje korisnika, upravljanje dozvolama, pravima pristupa, oporavak, konkurentnost, sigurnost i integritet relacione baze podataka.

SQL za definiciju podataka (DDL)

- **CREATE**
  - Kreiranje objekata baze podataka
- **ALTER**
  - Modifikovanje strukture i karakteristika postojećih objekata baze podataka
- **DROP**
  - Brisanje postojećih objekata baze podataka

SQL naredba za pravljenje baze podataka

```
CREATE {DATABASE | SCHEMA}
[IF NOT EXISTS] db_name
[DEFAULT] { CHARACTER SET [=] character_set_name |
COLLATE [=] collation_name }
```

Primer naredbe za pravljenje baze podataka čije ime je preduzece:

```
CREATE DATABASE preduzece
CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

SQL naredba za odabir baze podataka

Pre početka pravljenja tabela, indeksa, relacija ili upravljanja podacima u bazi podataka, neophodno je izdati MySQL serveru naredbu za odabir baze podataka na koju se odnose naredbe koje slede.

```
USE db_name;
```

U slučaju konkretnog primera koji je u ovom poglavlju obrađen, ime baze podataka je preduzeće, tako da je naredba koju koristimo

```
USE preduzece;
```

SQL naredba za pravljenje tabele - CREATE TABLE

```
CREATE TABLE radnik
        (
          radnik_id     int                 NOT NULL,
          ime           varchar(15)    NOT NULL
         );
```

SQL naredba za pravljenje tabele - CREATE TABLE

Kreiranje tabele i postavljanje ograničenja definisanjem primarnog ključa
SQL CREATE TABLE iskaz
SQL CONSTRAINT ključna reč

```
CREATE TABLE radnik
        (
          radnik_id          int                NOT NULL,
          ime                varchar(15)        NOT NULL,
          CONSTRAINT         radnikID_PK    PRIMARY KEY(radnik_id)
         );
```

SQL naredba za pravljenje tabele - CREATE TABLE

Kreiranje tabele i postavljanje ograničenja definisanjem složenog primarnog ključa
SQL CREATE TABLE iskaz
SQL CONSTRAINT ključna reč

```
CREATE TABLE ucesce
        (
          radnik_id          int              NOT NULL,
          projekat_id        int              NOT NULL,
          sredstva           float            NOT NULL,
          CONSTRAINT     ucesce_PK    PRIMARY KEY(radnik_id, projekat_id)
        );
```

SQL naredba za pravljenje tabele - CREATE TABLE

Kreiranje tabele i postavljanje ograničenja definisanjem složenog primarnog i stranog ključa
SQL CREATE TABLE iskaz
SQL CONSTRAINT i REFERENCES ključne reči

```
CREATE TABLE ucesce
      (
       radnik_id          int              NOT NULL,
       projekat_id        int              NOT NULL,
       sredstva           float            NOT NULL,
       CONSTRAINT     ucesce_PK    PRIMARY KEY(radnik_id, projekat_id),
       CONSTRAINT     radnik_FK    FOREIGN KEY(radnik_id)    REFERENCES radnik(radnik_id),
       CONSTRAINT     projekat_FK  FOREIGN KEY(projekat_id)  REFERENCES projekat(projekat_id)
      );
```

SQL naredba za pravljenje tabele - CREATE TABLE

RADNIK

radnikID	ime
1	Lazar
2	Aca
3	Marko
4	Luka

PROJEKAT

projekatID	naziv
100	Projektovanje igrica
101	DBA razvoj
102	Izvoz
103	Uvoz
104	Izgradnja

UCESCE

radnikID	projekatID	sredstva
1	100	300000
1	101	300000
2	102	249000
2	103	789999
3	104	123000
2	100	10000

SQL naredba za pravljenje tabele - CREATE TABLE

Kreiranje tabele i postavljanje ograničenja definisanjem složenog primarnog i stranog ključa
SQL CREATE TABLE iskaz
SQL CONSTRAINT i REFERENCES ključne reči
ON UPDATE CASCADE i ON DELETE CASCADE specifikacije referencijalnog integriteta

```
CREATE TABLE ucesce
      (
        radnik_id          int              NOT NULL,
        projekat_id        int              NOT NULL,
        sredstva           float            NOT NULL,
        CONSTRAINT     ucesce_PK    PRIMARY KEY(radnik_id, projekat_id),
        CONSTRAINT     radnik_FK    FOREIGN KEY(radnik_id)
           REFERENCES radnik(radnik_id) ON DELETE CASCADE,
        CONSTRAINT     projekat_FK  FOREIGN KEY(projekat_id)
           REFERENCES projekat(projekat_id) ON UPDATE CASCADE
      );
```

SQL naredba za pravljenje tabele - CREATE TABLE

radnikID	ime
1	~~Lazar~~
2	Aca
3	Marko

radnikID	projekatID	sredstav
1	100	~~300000~~
1	101	~~300000~~
2	102	249000
2	**109**	789999
3	104	123000

projekatID	naziv
100	Projektovanje igrica
101	DBA razvoj
102	Izvoz
109	Uvoz
104	Izgradnja

SQL naredba za promenu tabele - ALTER TABLE

Dodavanje ograničenja PRIMARY KEY u već kreiranu tabelu

SQL ALTER iskaz

```
ALTER TABLE radnik
        ADD CONSTRAINT  radnikID_PK   PRIMARY KEY(radnik_id);
```

SQL naredba za promenu tabele - ALTER TABLE

Dodavanje složenog primarnog ključa u već kreiranu tabelu

SQL ALTER iskaz

```
ALTER TABLE ucesce
        ADD CONSTRAINT   ucesceID_PK
           PRIMARY KEY(radnik_id, projekat_ID);
```

SQL naredba za promenu tabele - ALTER TABLE

Dodavanje stranog ključa u već kreiranu tabelu

SQL ALTER iskaz

```
ALTER TABLE radnik
        ADD CONSTRAINT   odeljenjeID_FK
          FOREIGN KEY(odeljenje_id)
            REFERENCES odeljenje(odeljenje_id);
```

TIPOVI PODATAKA

- Osnovni tipovi podataka koje se koriste u MySQL implementaciji SQL jezika su:
  - Tekstualni: kodirani i nekodirani
  - Numerički: označeni i neoznačeni
  - Vremenski

- **Kodirani tekstualni tipovi** podataka se koriste za čuvanje tekstualnih vrednosti, gde tekst nije kodiran ASCII kodnom mapom, već Unicode kodnom mapom promenljive širine. Najčešće se dužina vrednosti ovih tipova podataka izražava u broju karaktera.
- **Nekodirani ili binarni tipovi** podataka se koriste za čuvanje binarnih zapisa, kao što su sadržaji datoteka ili binarnih serijalizacija objekata iz programa. Najčešće se dužina vrednosti ovih tipova podataka izražava u broju bajtova.

Tekstualni tipovi podataka

Specifični kodirani tekstualni tipovi podataka su:	
CHAR	Dužina je ograničena na 255 karaktera
VARCHAR	Dužina je ograničena na 65.535 karaktera
TEXT	Dužina je ograničena na 65.535 karaktera
TINYTEXT	Dužina je ograničena na 256 karaktera
MEDIUMTEXT	Dužina je ograničena na 16.777.215 karaktera.
LONGTEXT	Dužina je ograničena na 4.294.967.295 karaktera.
Specifični binarni tipovi podataka su:	
BINARY	Dužina zapisa je ograničena na 255 bajtova.
VARBINARY	Dužina zapisa je ograničena na 65.535 bajtova.
BLOB	Dužina je ograničena na 65.535 bajtova.
MEDIUMBLOB	Dužina je ograničena na 16.777.215 bajtova.
LONGBLOB	Dužina zapisa je ograničena na 4.294.967.295 bajtova.

Tekstualni tipovi podataka

ENUM i SET tipovi podataka omogućavaju da se unapred definisane tekstualne vrednosti čuvaju u poljima tih tipova uz manju potrošnju prostora.

ENUM	Najviše 65.535 jedinstvenih vrednosti. Polje može da sadrži samo jednu od vrednosti
SET	Najviše 64 jedinstvene vrednosti. Polje može da sadrži najviše 64 jedinstvene vrednosti.

Numerički tipovi podataka

Označeni celobrojni numerički tipovi podataka su:	
INT	Opseg od -2147483648 do 2147483647
TINYINT	Opseg od -128 do 127.
SMALLINT	Opseg od -32768 do 32767
MEDIUMINT	Opseg od -8388608 do 8388607
BIGINT	Od -9223372036854775808 do 9223372036854775807.

Neoznačeni celobrojni numerički tipovi podataka su:	
INT	Opseg od 0 do 4.294.967.295
TINYINT	Opseg od 0 do 255
SMALLINT	Opseg od 0 do 65.535
MEDIUMINT	Opseg od 0 do 16.777.215
BIGINT	Opseg od 0 do 18.446.744.073.709.551.615.

Numerički tipovi podataka

Označeni i neoznačeni realni numerički tipovi sa fiksnom decimalnom tačkom su
DECIMAL(C, D)
NUMERIC

Označeni i neoznačeni realni numerički tipovi sa promenjivom decimalnom tačkom su
FLOAT
DOUBLE

Vremenski tipovi podataka

YEAR Opseg od 1901 do 2155. godine. Može da bude i 0000
DATETIME Opseg od 1. 1. 1000. u 00.00.01 do 19. 1. 2038. u 03.14.08
DATE Opseg datuma od 1. 1. 1000. do 31. 12. 9999. godine.
TIME Opseg vremena od 00.00.00 do 23.59.59
TIMESTAMP Opseg od 1. 1. 1000. u 00.00.01 do 19. 1. 2038. u 03.14.08.

Tipovi podataka DATETIME i TIMESTAMP podržavaju zapis do milionitog delova sekunde, od 0,000001 do 0,999999 sekundi

Specijalni tipovi podataka

- Prostorni tipovi, koji nalaze primenu u određenim oblastima kao što su geografski informacioni sistemi i drugi sistemi koji rade sa prostornim podacima.
- MySQL podržava JSON tipove podataka. JSON tip je specifičan, jer u određenoj meri narušava principe relacionih baza podataka, jer obezbeđuje mehanizam za čuvanje složenih podataka proizvoljne strukture u poljima tabele.
- JSON podaci se u MySQL bazama podataka beleže kao tekstualni, ali posebne jezičke konstrukcije omogućavaju korišćenje i referenciranje struktura podataka smeštenih unutar polja koje sadrži JSON vrednost.

Odabir tipova podataka

Odabir adekvatnog tipa podatka je važan, jer u slučaju prvobitnog pogrešnog odabira tipa, u slučaju kasnije potrebe za promenom tipa, neophodno je izvršiti proces migracije.

- Ime osobe može da bude VARCHAR tip podatka ograničene dužine, ali ne može da bude DECIMAL(10,2) ili INT;
- Cena artikla može da bude DECIMAL(10,4) ako je potrebno evidentirati cene koje su realne vrednosti, ali ako su cene uvek celobrojne, dovoljno je koristiti tip podatka INT;
- ID broj kartice može da bude INT UNSIGNED ako se obim vrednosti broja ID kartice uklapa u okvire najvećeg podržanog broja neoznačene celobrojne numeričke vrednosti tipa INT. Međutim, ako ID broj kartice sadrži specijalne karaktere ili, eventualno, slovne karaktere, onda je bolje koristiti CHAR ili VARCHAR potrebne dužine;
- Kada je potrebno evidentirati status paketa ili pošiljke, moguće je koristiti tip podatka TINYINT UNSIGNED dužine 1 u kojem je pomoću vrednosti 1 ili 0 moguće ukazati na to da li je pošiljka poslata ili ne. Međutim, bolja opcija je definisati ENUM tip sa podržanim vrednostima koje mogu da ukažu na više koraka u postupku dostave paketa ili pošiljke, npr. pending, sent, lost, returned, delivered itd.