

```
source = datasource.getConnection();
connection.createStatement();
SQL = "SELECT * FROM ";
statement.executeUpdate();
ResultSet next()) {
```

# BAZE PODATAKA

Konceptualno modelovanje

Goran Banjac  
goran.banjac@etf.unibl.org  
3/21/2023

# Konceptualno modelovanje

---

- ▶ **Projektovanje baze podataka** je jedan od najznačajnijih segmenta razvoja sistema sa bazama podataka
- ▶ **Cilj** je da se dođe do detaljne specifikacije sveukupne strukture baze podataka za odgovarajući DBMS (*Database Management System*)
- ▶ Početna faza projektovanja baze podataka je **konceptualno modelovanje**, koje omogućava specifikaciju sveukupne strukture baze podataka na visokom nivou apstrakcije

# Konceptualno modelovanje

---

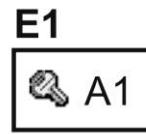
- ▶ Notacije:
  - ▶ Najčešće se koristi **ER/EER** (*Entity-Relationship/Enhanced Entity-Relationship*)
  - ▶ U praksi se često koriste alternativne, konceptualno slične notacije, kao što je npr. **IE** (*Information Engineering*)
  - ▶ Sve širu primenu u konceptualnom modelovanju baze podataka ima i **UML dijagram klasa** (*class diagram*)
- ▶ **IE notacija:**
  - ▶ **Često korištena notacija za konceptualno modelovanje u softverskim alatima za projektovanje baza podataka**
  - ▶ Iako raspolaze većinom semantički ekvivalentnih koncepata, ipak postoje neki ER/EER koncepti koji nisu podržani (npr. *n-arni* tip veze, unija, agregacija, atributi veznog tipa, više značni atributi itd.)

# IE notacija

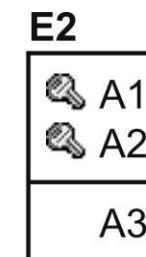
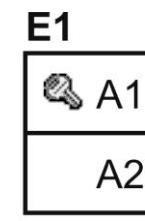
- ▶ **Entitetski tipovi** u IE zasnovanom ERwin konceptualnom modelu mogu da se prikažu na različitim nivoima detaljnosti:
  - ▶ bez atributa (a)
  - ▶ samo primarni ključevi (b)
  - ▶ svi atributi (c)
- ▶ Atributi koji čine **primarni ključ** (atributi sa simbolom ključa) prikazuju se u gornjoj sekciji pravougaonika, odvojeno od ostalih atributa



(a)



(b)



(c)

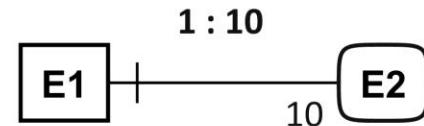
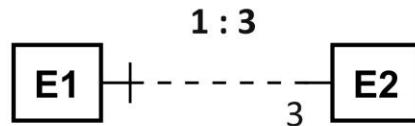
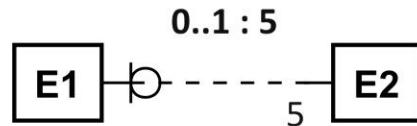
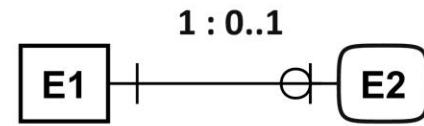
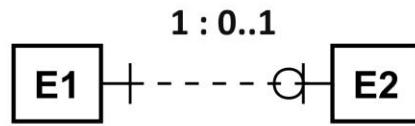
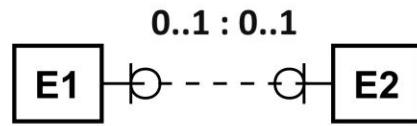
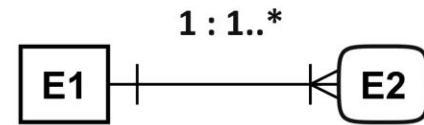
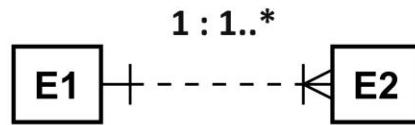
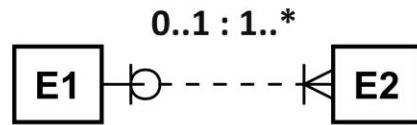
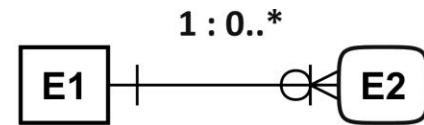
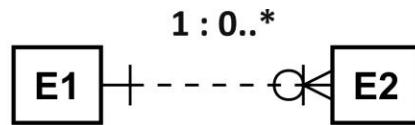
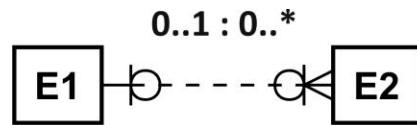
# IE notacija

---

- ▶ Neidentificujući vezni tipovi prikazuju se isprekidanim linijama, a identificujući vezni tipovi neisprekidanim linijama
- ▶ Za specifikaciju **kardinalnosti mapiranja** koristi se notacija ***ptičije stopalo***:
  - ▶ kardinalnost više prikazuje se u obliku višestrukog završetka
  - ▶ kardinalnost *jedan* prikazuje se u obliku jednostrukog završetka
- ▶ **Ograničenja učešća entiteta u vezama:**
  - ▶ Specifikuju se na potpuno drugačiji način u odnosu na ER notaciju
  - ▶ Parcijalna participacija entiteta nekog tipa specifikuje se kružićem na suprotnoj strani veznog tipa

# IE notacija

- ▶ Neidentifikujući (a) i identifikujući (b) vezni tipovi:

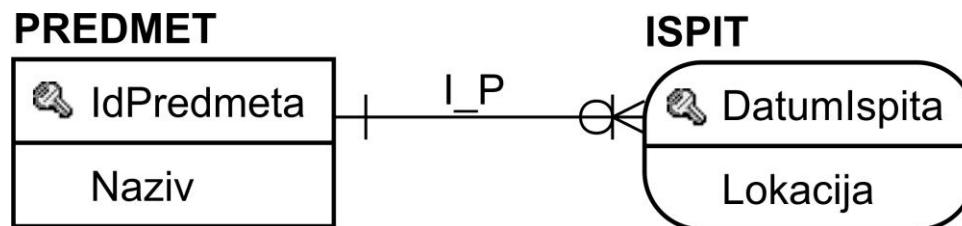


(a)

(b)

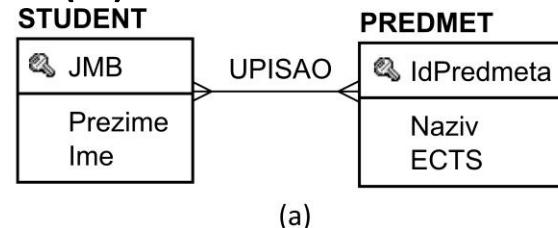
# IE notacija

- ▶ Za razliku od **jakih entitetskih tipova**, koji se reprezentuju odgovarajućim pravougaonikom, **slabi entitetski tipovi** reprezentuju se pravougaonikom sa zaobljenim vrhovima
- ▶ Atributi koji predstavljaju **diskriminator slabog entitetskog tipa** reprezentuju se identično atributima primarnog ključa jakog entitetskog tipa, tj. označavaju se istim simbolom i prikazuju u gornjoj sekciji pravougaonika

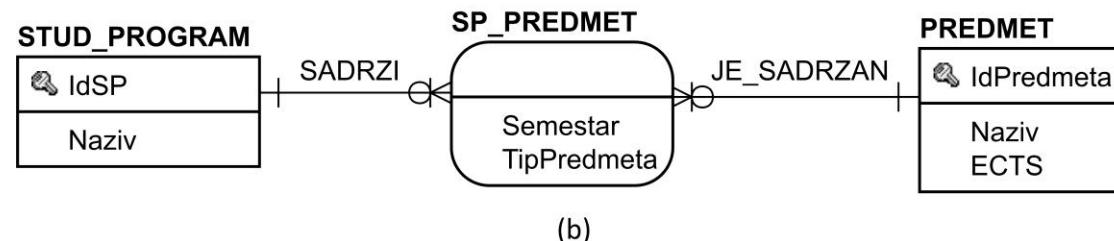


# IE notacija

- ▶ Vezni tip sa kardinalnošću mapiranja M:M, ako nema opisne atributе, može da se predstavi kao vezni tip (a)
- ▶ Međutim, ako se u toku projektovanja identificuju atributi veznog tipa sa kardinalnošću mapiranja M:M, tada takav vezni tip ne može da se predstavi kao vezni tip (jer IE notacija ne omogućava da se veznim tipovima specifikuju atributi), već mora da se reprezentuje odgovarajućim slabim entitetskim tipom i identifikujućim veznim tipovima sa odnosnim entitetskim tipovima (b)



(a)



(b)

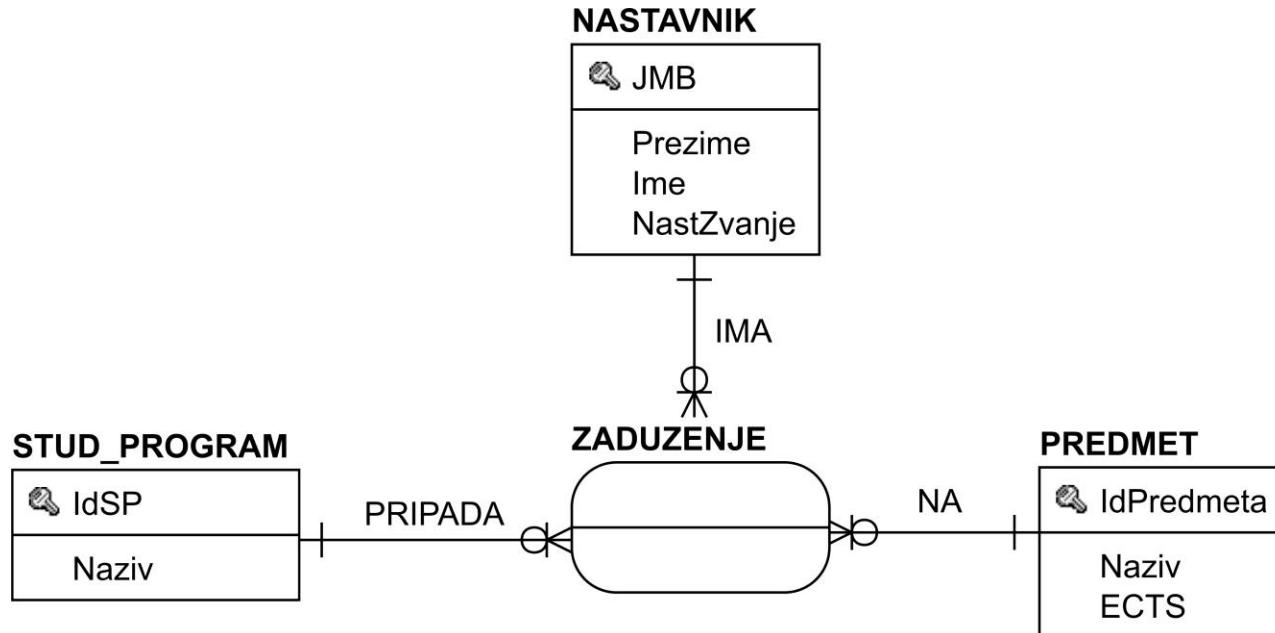
# IE notacija

---

- ▶ Ako u toku projektovanja identifikujemo atribute veznog tipa sa kardinalnošću mapiranja 1:M, tada:
  - ▶ Ako je participacija entiteta na strani M totalna, tada atribute veznog tipa dodajemo atributima entitetskog tipa na strani M
  - ▶ U suprotnom, za parcijalnu participaciju entiteta na strani M, postoje dve opcije:
    - ▶ U prvom slučaju, atribute veznog tipa možemo dodati atributima entitet-skog tipa na strani M, ali za njih ne sme biti definisano *not null* ograničenje (neki entiteti datog tipa ne učestvuju u vezi pa će ti entiteti na dodatim atributima imati vrednost *null*)
    - ▶ Druga opcija je da se dati vezni tip reprezentuje slabim entitetskim tipom i identifikujućim veznim tipom sa odnosnim entitetskim tipom na strani M i neidentifikujućim veznim tipom sa odnosnim entitetskim tipom na strani 1

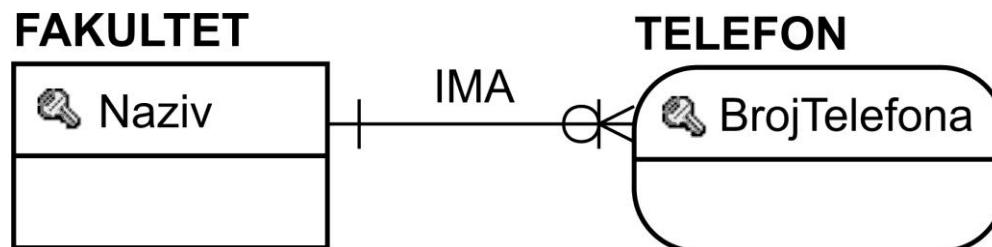
# IE notacija

- ▶ *n-arni* vezni tipovi reprezentuju se slabim entitetskim tipom i odgovarajućim identificujućim veznim tipovima, jer IE notacija ne podržava *n-arne* tipove veza



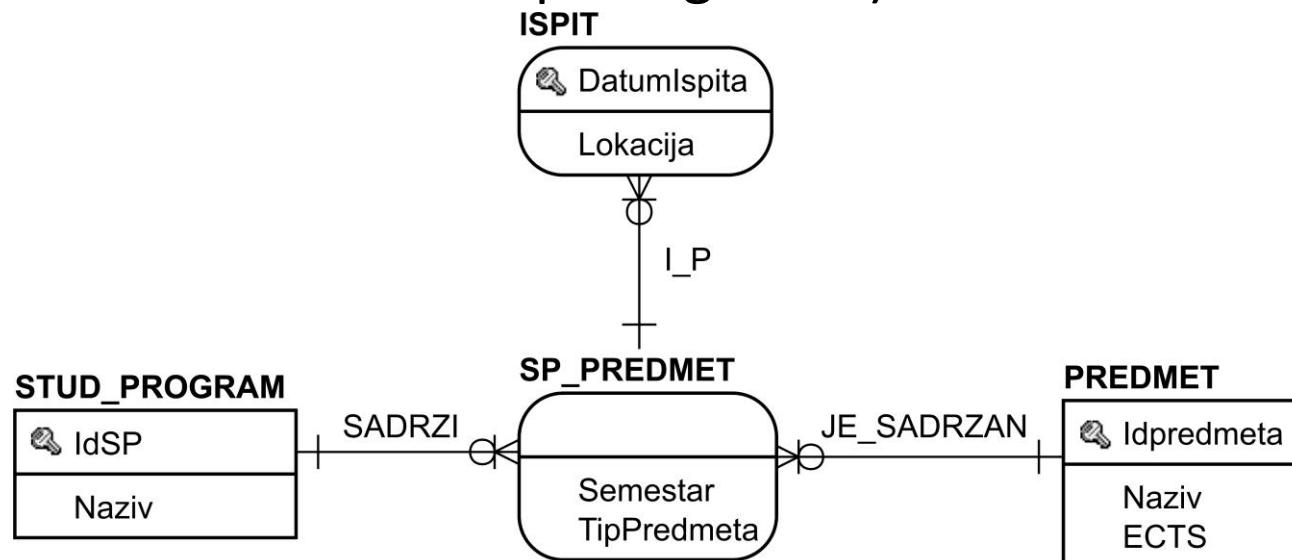
# IE notacija

- ▶ IE notacija ne omogućava ni modelovanje **višeznačnih atributa**, pa višeznačne attribute, koji se identifikuju tokom konceptualnog projektovanja, treba reprezentovati slabim entitetskim tipom i odgovarajućim identifikujućim veznim tipom sa entitetskim tipom kojem pripada identifikovani višeznačni atribut
- ▶ Osim diskriminatora koji reprezentuje višeznačni atribut, odnosni slabi entitetski tip može da ima i druge attribute



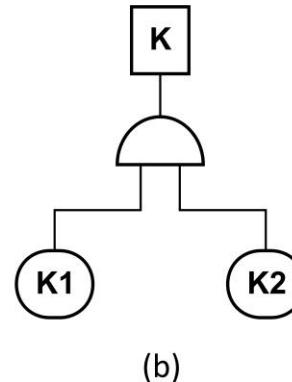
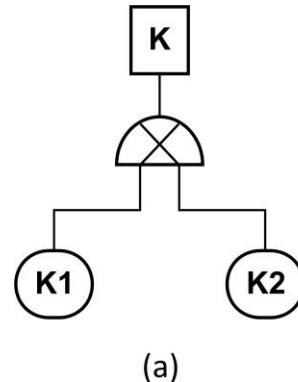
# IE notacija

- ▶ Koncept **agregacije** nije podržan u IE notaciji
- ▶ S obzirom na to da se u IE zasnovanom modelu, vezni tipovi koji imaju vlastita svojstva (a uspostavljanje dodatnih veza je takođe svojevrsno svojstvo) reprezentuju kao entitetski tipovi, modelovanje agregacije u IE notaciji je potpuno u skladu sa definicijom agregacije (agregacija podrazumeva da se vezni tip posmatra kao entitetski tip višeg nivoa)



# IE notacija

- ▶ IE notacija omogućava modelovanje veza **specijalizacije/generalizacije**
- ▶ Podržano je modelovanje:
  - ▶ *disjunktne* ili *ekskluzivne* specijalizacije (a)
  - ▶ *preklapajuće* ili *inkluzivne* specijalizacije (b)
- ▶ Nije podržano modelovanje ograničenja učešća (totalno/parcijalno) entiteta natklase
- ▶ Potklase se u IE dijagramu reprezentuju slabim entitetskim tipovima



# Proces projektovanja konceptualnog modela

---

- ▶ Ne postoji metodologija koja propisuje precizne korake koji bi uvek vodili ka jedinstvenom i jednoznačnom konceptualnom modelu nekog realnog sistema
- ▶ U procesu projektovanja projektanti se susreću sa različitim mogućnostima i dilemama
- ▶ Specifične percepcije realnog sistema mogu rezultovati **različitim zaključcima i različitim konceptualnim modelima za isti realni sistem**
- ▶ **Proces izgradnje konceptualnog modela baze podataka je gotovo uvek iterativan**, pri čemu se prvo kreira početni model, a zatim se u iterativnom procesu u svakoj iteraciji vrši analiza, otklanjaju nedostaci, rešavaju projektne dileme i poboljšava model

# Proces projektovanja konceptualnog modela

---

- ▶ Najopštiji pristup jeste da se **inicijalna identifikacija entitetskih skupova (tipova) i veza vrši na osnovu specifikacije zahteva**, odnosno **informacionih potreba sistema** za koji se projektuje baza podataka, pri čemu se (posebno u alatima za automatsko projektovanje) koriste heuristička, odnosno neformalna pravila za identifikaciju karakterističnih koncepata
- ▶ **Subjekti** u rečenicama najčešće reprezentuju entitete koji treba da se modeluju odgovarajućim entitetskim tipom
- ▶ **Predikati** (glagoli i glagolske fraze) koji u rečenicama povezuju subjekte i objekte (već reprezentovane entitetskim tipovima) reprezentuju se veznim tipovima

# Proces projektovanja konceptualnog modela

---

- ▶ **NLP (Natural Language Processing)** – obrada teksta u prirodnom jeziku
- ▶ Identifikacija osnovnih elemenata konceptualnog modela baze podataka na osnovu **tekstualne specifikacije informacionih potreba** nekog sistema
- ▶ Primeri *online* NLP alata:
  - ▶ CoreNLP
  - ▶ NLTK
  - ▶ TextBlob
  - ▶ Gensim
  - ▶ SpaCy
  - ▶ OpenNLP
- ▶ Primeri *online* NLP servisa:
  - ▶ TextRazor
  - ▶ Bitext
  - ▶ Aylien Text Analysis

# Proces projektovanja konceptualnog modela

- ▶ TextRazor (<https://www.textrazor.com/demo>):

 TextRazor.

Demo Technology Documentation ▾ Pricing | Login [Sign up](#)

[Edit Text](#) Language: eng Processed in: 0.5921 seconds

**Barclays** misled **shareholders** and the public about one of the biggest **investments** in the **bank's** history, a **BBC Panorama** investigation has found.

Words Phrases Relations Entities Meaning Dependency Parse

Entity	Confidence Score	Relevance Score	DBpedia Type	Freebase Type
Barclays (/m/05t8c5) (Q245343)	8.776	0.8425	Agent Organisation Company	/business/customer /business/issuer /organization/organization /business/business_operation /business/employer /exhibitions/exhibition_sponsor /award/ranked_item
Shareholder (/m/012l_n) (Q381136)	8.498	0.1065		/organization/role
Investment (/m/0g_f)	7.837	0.342		/education/field_of_study /broadcast/genre /internet/website_category

CATEGORIES

- 0.94 economy, business and finance>economy>macro economics>investments
- 0.73 economy, business and finance>business information>business finance>shareholder
- 0.70 economy, business and finance>economy
- 0.64 economy, business and finance>market and exchange>securities
- 0.49 economy, business and finance
- 0.48 crime, law and justice>law
- 0.48 economy, business and

```
source = datasource.getConnection();
connection.createStatement();
SQL = "SELECT * FROM student";
statement.executeUpdate();
ResultSet set = statement.executeQuery();
set.next();
```

# BAZE PODATAKA

SQL – DDL

# Uvod

---

- ▶ **SQL** (*Structured Query Language*) je upitni jezik visokog nivoa za rad sa relacionim bazama podataka, a baziran je na relacionom računu i relacionoj algebri
- ▶ Nastao je 70-tih godina prošlog veka u IBM-ovim istraživačkim laboratorijama u San Hozeu, u okviru *System R* projekta koji je trebao da istraži i demonstrira mogućnosti relationalnih DBMS
- ▶ Sastoje se od dve komponente:
  - ▶ **DDL** (*Data Definition Language*) komponenta – omogućava definisanje objekata u bazi podataka
  - ▶ **DML** (*Data Manipulation Language*) komponenta – omogućava manipulaciju podacima u bazi podataka

# Tipovi podataka (MySQL)

---

- ▶ Numerički tipovi:
  - ▶ Egzaktni numerički tipovi
    - ▶ Celobrojni
      - **tinyint** – jednobajtni celobrojni tip
      - **smallint** – dvobajtni celobrojni tip
      - **mediumint** – trobajtni celobrojni tip
      - **int** ili **integer** – četvorobajtni celobrojni tip
      - **bigint** – osmabajtni celobrojni tip
    - ▶ Racionalni
      - **decimal(*p,s*)**, **dec(*p,s*)**, **numeric(*p,s*)** ili **fixed(*p,s*)** – numerički tip podataka za egzaktnu predstavu decimalnih brojeva, gde broj *p* označava ukupan broj cifara broja, dok broj *s* označava broj cifara broja iza decimalne tačke

# Tipovi podataka (MySQL)

---

- ▶ Numerički tipovi:
  - ▶ Aproksimativni numerički tipovi
    - ▶ **real**, **double** ili **double precision** – osmabajtni aproksimativni numerički tip (dvostruka preciznost)
    - ▶ **float(*n*)** – četvorobajtni aproksimativni numerički tip (obična preciznost)
      - Ukoliko je vrednost preciznosti (*n*) u opsegu 0-23 – kreirana kolona je četvorobajtna **float** kolona (obična preciznost)
      - Ukoliko je vrednost preciznosti (*n*) u opsegu 24-53 – kreirana kolona je osmabajtna **double** kolona (dvostruka preciznost)
- ▶ Nizovi znakova:
  - ▶ **char(*n*)** – koristi se za specifikaciju tipa niza znakova fiksne dužine od *n* znakova, *n* > 0 (podrazumevana vrednost za *n* je 1)
  - ▶ **varchar(*n*)** – koristi se za specifikaciju tipa niza znakova promenljive dužine, koji sadrži minimalno 0, a maksimalno *n* znakova
  - ▶ **text** – koristi se za specifikaciju tipa niza znakova maksimalne dužine 65535 znakova

# Tipovi podataka (MySQL)

---

- ▶ Binarni tipovi (*Binary Large OBject*):
  - ▶ **tinyblob** – BLOB kolona maksimalne dužine 255 bajtova
  - ▶ **blob** – BLOB kolona maksimalne dužine 65535 bajtova
  - ▶ **mediumblob** – BLOB kolona maksimalne dužine 16777215 bajtova
  - ▶ **longblob** – BLOB kolona maksimalne dužine 4294967295 bajtova (4 GB)

# Tipovi podataka (MySQL)

---

- ▶ Datumski i vremenski tipovi:
  - ▶ **date**
  - ▶ **datetime**
  - ▶ **timestamp**
  - ▶ **time**
  - ▶ **year**
- ▶ Logički tip:
  - ▶ **bool** ili **boolean** – sinonim za **tinyint(1)**
- ▶ Ostali tipovi:
  - ▶ **enum** – enumeracija
  - ▶ **set** – skup
  - ▶ **json** – JSON tip
  - ▶ ...

# Kreiranje šeme i objekata šeme

---

- ▶ Opšti oblik SQL iskaza za kreiranje šeme baze podataka:

`create schema naziv_šeme;`

- ▶ Prilikom kreiranje šeme moguće je specifikovati podrazumevani skup znakova (**character set**), kao i podrazumevani skup pravila za poređenje znakova iz datog skupa znakova (**collate**)
- ▶ Nakon kreiranja šeme baze podataka, moguće je kreirati i pripadajuće objekte date šeme
- ▶ Ako se objekat koji se kreira želi dodati u određenu šemu, onda se nazivu objekta koji se kreira, dodaje kao prefiks naziv šeme, pri čemu su naziv šeme i naziv objekta razdvojeni tačkom kao separatorom (*naziv\_šeme.naziv\_objekta*)
- ▶ Ukoliko naziv šeme (prefiks) nije specifikovan, onda se objekat dodaje u tekuću, odnosno selektovanu šemu
- ▶ U MySQL-u, šema se selektuje **use** iskazom, čiji je opšti oblik:

`use naziv_šeme;`

# Kreiranje šeme i objekata šeme

---

- ▶ Tabele se kreiraju **create table** iskazom, koji uključuje specifikaciju imena tabele i kolona (naziva i domena kolona), pri čemu se može uključiti i specifikacija ograničenja, kao što su primarni ključ, strani ključ, provera uslova i sl.
- ▶ Opšti oblik **create table** iskaza:

```
create table naziv_tabele
(
    naziv_kolone1 tip_podatka [ograničenje_za_kolonu1],
    naziv_kolone2 tip_podatka [ograničenje_za_kolonu2],
    ...
    naziv_kolonen tip_podatka [ograničenje_za_kolonun]
    [, integritetsko_ograničenje1]
    [, integritetsko_ograničenje2]
    ...
    [, integritetsko_ograničenjem]
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Tipična ograničenja za kolonu:
  - ▶ **not null**
  - ▶ **default vrednost**
- ▶ Tipična integritetska ograničenja:
  - ▶ specifikacija primarnog ključa
    - primary key (*naziv\_kolone<sub>1</sub>*, *naziv\_kolone<sub>2</sub>*, ... , *naziv\_kolone<sub>n</sub>*)**
  - ▶ specifikacija stranog ključa
    - foreign key (*naziv\_kolone<sub>j1</sub>*, *naziv\_kolone<sub>j2</sub>*, ... , *naziv\_kolone<sub>in</sub>*)**
    - references *naziv\_referencirane\_tabele* (*naziv\_kolone<sub>i1</sub>*, *naziv\_kolone<sub>i2</sub>*, ... , *naziv\_kolone<sub>in</sub>*)**
  - ▶ ograničenje provere uslova
    - check (*P*)**

# Kreiranje šeme i objekata šeme

---

- ▶ Primer: SQL izraz kojim se kreira tabela *fakultet*

```
create table fakultet
(
    NazivFakulteta varchar(50),
    Adresa varchar(50) not null,
    primary key (NazivFakulteta)
);
```

- ▶ Za kolone koje su deo primarnog ključa podrazumeva se **not null** ograničenje
- ▶ Ukoliko samo jedna kolona čini primarni ključ tabele, tada je odgovarajuće ograničenje moguće navesti pri definiciji kolone, kao npr.:

```
create table fakultet
(
    NazivFakulteta varchar(50) primary key,
    Adresa varchar(50) not null
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Kako se u praksi često koriste *surogat/veštački* ključevi kao primarni ključevi, tako je pogodno da se za takve kolone vrednost ne specifikuje eksplicitno prilikom dodavanja novih podataka u tabelu, nego da DBMS automatski generiše vrednost za takve kolone
- ▶ Tipično, DBMS-ovi imaju podršku za automatsko generisanje vrednosti za (surogat) primarni ključ, ali različiti DBMS-ovi koriste različite ključne reči za specifikaciju takvih kolona
- ▶ Na primer, u MySQL-u se za specifikaciju takvih kolona koristi ključna reč **auto\_increment**:

```
create table fakultet
(
    IdFakulteta int auto_increment primary key,
    NazivFakulteta varchar(50) not null,
    Adresa varchar(50) not null
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Primer: SQL iskaz kojim se kreira tabela *predmet*

```
create table predmet
(
    IdPredmeta int,
    NazivPredmeta varchar(50) not null,
    ECTS tinyint not null,
    NazivFakulteta varchar(50) not null,
    primary key (IdPredmeta),
    foreign key (NazivFakulteta)
        references fakultet (NazivFakulteta)
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Pri definiciji stranog ključa moguće je navesti i naziv ograničenja koje specifikuje strani ključ, kao npr.:

```
create table predmet
(
    IdPredmeta int,
    NazivPredmeta varchar(50) not null,
    ECTS tinyint not null,
    NazivFakulteta varchar(50) not null,
    primary key (IdPredmeta),
    constraint FK_predmet_fakultet
        foreign key (NazivFakulteta)
        references fakultet (NazivFakulteta)
);
```

- ▶ Naziv ograničenja mora biti jedinstven na nivou šeme baze podataka

# Kreiranje šeme i objekata šeme

---

- ▶ Pri definiciji stranog ključa, moguće je navesti i **ograničenja referencijalnog integriteta** za akcije brisanja (**on delete**) i ažuriranja (**on update**)
- ▶ Moguće vrednosti su:
  - ▶ **cascade**
  - ▶ **set null**
  - ▶ **set default** (nije podržana u MySQL-u)
  - ▶ **restrict** (podrazumevana u MySQL-u)
  - ▶ **no action**

# Kreiranje šeme i objekata šeme

---

- ▶ Primer: SQL izraz kojim se kreira tabela *predmet*, pri čemu su pri definiciji stranog ključa navedena i ograničenja referencijalnog integriteta za akcije brisanja i ažuriranja

```
create table predmet
(
    IdPredmeta int,
    NazivPredmeta varchar(50) not null,
    ECTS tinyint not null,
    NazivFakulteta varchar(50) not null,
    primary key (IdPredmeta),
    constraint FK_predmet_fakultet
        foreign key (NazivFakulteta)
            references fakultet (NazivFakulteta)
            on update cascade on delete restrict
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Primer: SQL izraz kojim se kreira tabela *predmet*, pri čemu je specifikovano ograničenje koje specifičuje da vrednost kolone *ECTS* mora da bude veća od nule

```
create table predmet
(
    IdPredmeta int,
    NazivPredmeta varchar(50) not null,
    ECTS tinyint not null,
    NazivFakulteta varchar(50) not null,
    primary key (IdPredmeta),
    constraint FK_predmet_fakultet
        foreign key (NazivFakulteta)
            references fakultet (NazivFakulteta)
            on update cascade on delete restrict,
    constraint CHK_predmet_ects
        check (ECTS>0)
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Pored kolona koje čine primarni ključ tabele, često postoje i druge kolone nad kojima treba obezbediti jedinstvenost vrednosti
- ▶ Da bi se obezbedila jedinstvenost vrednosti kolona koje ne čine primarni ključ, koristi se **unique** specifikacija
- ▶ Kolone koje čine deo **unique** specifikacije mogu imati **null** vrednosti
- ▶ **Unique** specifikacija se može navesti pri definiciji kolone (ukoliko samo jednu kolonu treba uključiti u specifikaciju) ili kao integritetsko ograničenje (koje može biti imenovano)

# Kreiranje šeme i objekata šeme

---

- ▶ Primer: SQL izraz kojim se kreira tabela *student*, pri čemu je navedena **unique** specifikacija nad kolonom *BrojIndeksa*

```
create table student
(
    JMB char(13),
    BrojIndeksa char(8) not null,
    Prezime varchar(20) not null,
    Ime varchar(20) not null,
    primary key (JMB),
    unique (BrojIndeksa)
);
```

# Kreiranje šeme i objekata šeme

---

- ▶ Indeksi nad tabelom se kreiraju **create index** izrazom čiji je opšti oblik:

```
create [unique] index naziv_indexa on naziv_tabele
(
    naziv_kolone1 [asc | desc]
    [, naziv_kolone2 [asc | desc]]
    ...
    [, naziv_kolonen [asc | desc]]
);
```

- ▶ Izraz **create unique index** omogućava kreiranje indeksa sa jedinstvenim vrednostima
- ▶ Primer: SQL izraz kojim se kreira indeks *IX\_predmet\_nazivpredmeta* nad tabelom *predmet* po koloni *NazivPredmeta*

```
create index IX_predmet_nazivpredmeta on predmet
(
    NazivPredmeta
);
```

# Modifikacije šeme

---

- ▶ Šema se briše **drop schema** iskazom čiji je opšti oblik:  
`drop schema naziv_šeme {restrict | cascade};`
- ▶ U MySQL-u nije podržano specifikovanje opcija ponašanja **drop schema** iskaza
- ▶ Tabele se brišu **drop table** iskazom čiji je opšti oblik:  
`drop table naziv_tabele {restrict | cascade};`
- ▶ Navođenje opcija ponašanja **drop table** iskaza u MySQL-u je opciono
- ▶ Struktura tabele menja se odgovarajućim **alter table** iskazom
- ▶ Opšti oblik iskaza za dodavanje novih kolona:  
`alter table naziv_tabele add [column] naziv_kolone  
tip_podatka [ograničenje_za_kolonu];`
- ▶ Nove kolone se dodaju na kraj tabele
- ▶ U MySQL-u je omogućena specifikacija mesta za dodavanje kolone opcijama **first** (na početak) i **after** (posle navedene kolone)

# Modifikacije šeme

---

- ▶ Opšti oblik iskaza za postavljanje podrazumevane vrednosti:

```
alter table naziv_tabele alter [column] naziv_kolone  
set default vrednost;
```

- ▶ Opšti oblik iskaza za ukidanje podrazumevane vrednosti:

```
alter table naziv_tabele alter [column] naziv_kolone  
drop default;
```

- ▶ U MySQL-u postoje proširenja **alter table** iskaza kojima se može vršiti modifikacija kolone

- ▶ Opšti oblik iskaza za modifikaciju kolone (**modify**):

```
alter table naziv_tabele modify [column] naziv_kolone  
novi_tip_podatka [novo_ograničenje_za_kolonu];
```

- ▶ Opšti oblik iskaza za modifikaciju kolone (**change**):

```
alter table naziv_tabele change [column] naziv_kolone  
novi_naziv_kolone novi_tip_podatka  
[novo_ograničenje_za_kolonu];
```

# Modifikacije šeme

---

- ▶ Opšti oblik iskaza za brisanje kolone iz tabele:

```
alter table naziv_tabele drop [column] naziv_kolone  
{restrict | cascade};
```

- ▶ Opšti oblik iskaza za specifikaciju primarnog ključa:

```
alter table naziv_tabele add primary key (naziv_kolone1,  
naziv_kolone2, ... , naziv_kolonen);
```

- ▶ Opšti oblik iskaza za ukidanje primarnog ključa:

```
alter table naziv_tabele drop primary key;
```

- ▶ Po standardu, ključevi (primarni i strani) brišu se iskazom za brisanje ograničenja (**drop constraint**) koji nije podržan u MySQL-u

# Modifikacije šeme

---

- ▶ Opšti oblik iskaza za specifikaciju stranog ključa:

```
alter table naziv_tabele add [constraint naziv_ograničenja]
foreign key (naziv_kolonej1, naziv_kolonej2, ... ,
naziv_kolonein)
references naziv_referencirane_tabele (naziv_kolonei1,
naziv_kolonei2, ... , naziv_kolonein);
```

- ▶ Opšti oblik iskaza za brisanje stranog ključa:

```
alter table naziv_tabele drop foreign key naziv_ograničenja;
```

- ▶ Opšti oblik iskaza za brisanje indeksa:

```
drop index naziv_indeksa on naziv_tabele;
```

- ▶ U MySQL-u postoje proširenja **alter table** iskaza za manipulaciju indeksima

- ▶ Opšti oblik **alter table** iskaza za brisanje indeksa:

```
alter table naziv_tabele drop index naziv_indeksa;
```

# Modifikacije šeme

---

- ▶ Opšti oblik **alter table** iskaza za kreiranje indeksa:

```
alter table naziv_tabele add index naziv_indexa
(
    naziv_kolone1 [asc | desc]
    [, naziv_kolone2 [asc | desc]]
    ...
    [, naziv_kolonen [asc | desc]]
);
```

- ▶ Opšti oblik **alter table** iskaza za kreiranje **unique** indeksa:

```
alter table naziv_tabele add unique [index] naziv_indexa
(
    naziv_kolone1 [asc | desc]
    [, naziv_kolone2 [asc | desc]]
    ...
    [, naziv_kolonen [asc | desc]]
);
```

# Modifikacije šeme

---

- ▶ Primer: SQL iskazi kojima se kreiraju tabele *predmet* i *fakultet*, a zatim se u tabelu *predmet* (naknadno) dodaje odgovarajuća kolona, kao i strani ključ

```
create table predmet
(
    IdPredmeta int,
    NazivPredmeta varchar(50) not null,
    ECTS tinyint not null
    primary key (IdPredmeta)
);
create table fakultet
(
    NazivFakulteta varchar(50),
    Adresa varchar(50) not null,
    primary key (NazivFakulteta)
);
alter table predmet add column NazivFakulteta varchar(50) not null;
alter table predmet add constraint FK_predmet_fakultet
foreign key (NazivFakulteta)
references fakultet (NazivFakulteta)
on update cascade on delete restrict;
```

```
    connection = datasource.getConnection();
    connection.createStatement();
    SQL = "SELECT * FROM ";
    statement.executeUpdate();
    set.next());
```

# BAZE PODATAKA

Transformacija konceptualog modela u relacioni model i  
DDL skripta za generisanje šeme

# Uvod

---

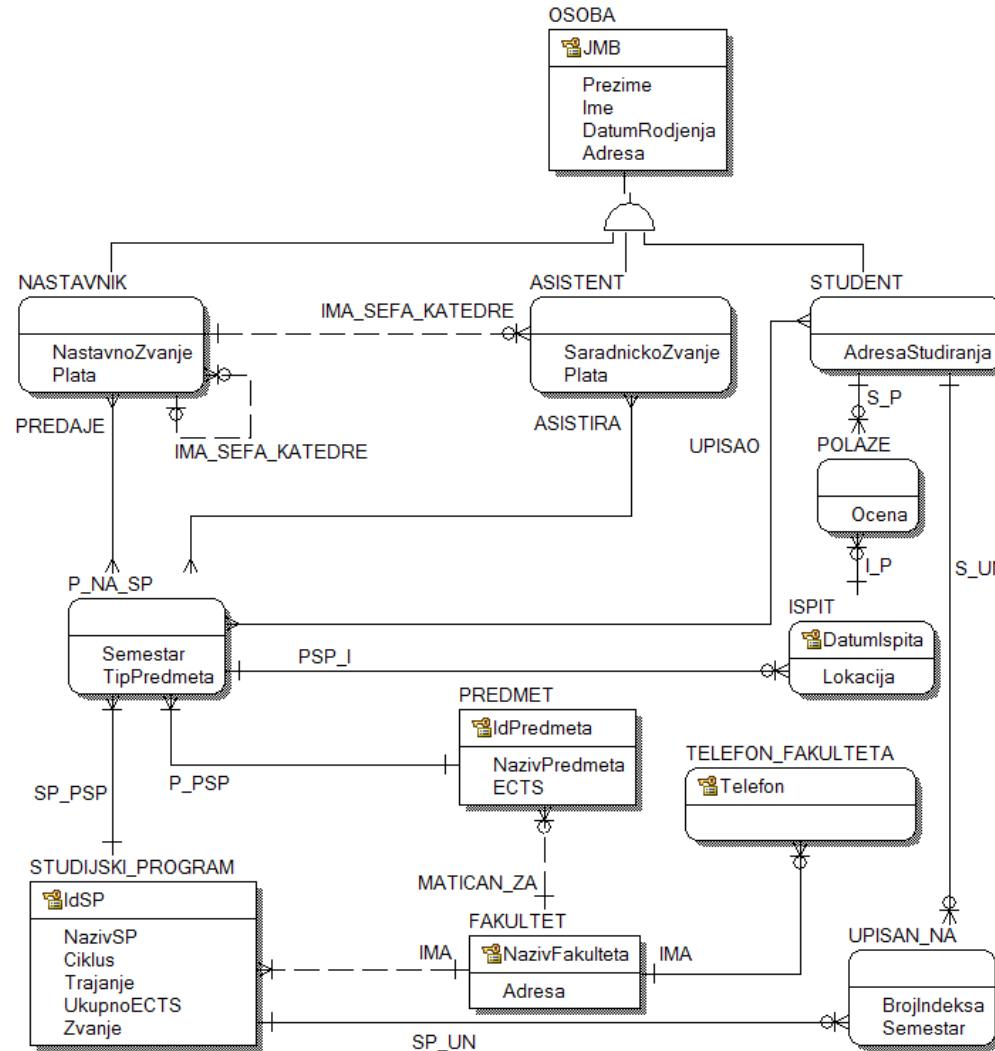
- ▶ **Relacioni model baze podataka** (šema relacione baze podataka) konkretnog sistema opisuje strukturu podataka, njihovo značenje, veze između podataka i ograničenja na implementacionom logičkom nivou
- ▶ **Šema relacione baze podataka** generiše se (najčešće) na osnovu konceptualne šeme, i predstavlja izlaz i rezultat faze logičkog projektovanja
- ▶ Šema relacione baze podataka može vizuelno da se predstavi **dijagramom**

# Uvod

---

- ▶ U nastavku su **specifikovana pravila** za generisanje šeme relacione baze podataka na osnovu konceptualnog modela
- ▶ Pri tome, **znakom \*** su označena pravila za koncepte koje podržava **IE** notacija
- ▶ Za ilustraciju pojedinih pravila korišten je **uprošteni IE konceptualni model baze podataka univerziteta**
- ▶ Specifikacija **DDL skripte** za generisanje šeme relacione baze podataka na osnovu uproštenog konceptualnog modela baze podataka univerziteta data je u pratećim materijalima

# Uprošteni IE konceptualni model baze podataka univerziteta



# Mapiranje jakih entitetskih tipova **(Pravilo 1\*)**

---

- ▶ Za svaki jaki entitetski tip  $X$  generiše se relacija šema koja uključuje sve proste (atomične) attribute entitetskog tipa  $X$  i proste komponente njegovih kompozitnih atributa
- ▶ Atributi primarnog ključa jakog entitetskog tipa čine primarni ključ relacione šeme
- ▶ Primer: Mapiranje jakog entitetskog tipa *FAKULTET*

```
create table fakultet
(
    NazivFakulteta varchar(50),
    Adresa varchar(50) not null,
    primary key (NazivFakulteta)
);
```

# Mapiranje slabih entitetskih tipova **(Pravilo 2\*)**

---

- ▶ Za svaki slabi entitetski tip  $X$ , egzistencijalno zavisan od identificujućeg entitetskog tipa  $Y$ , generiše se relaciona šema koja uključuje sve proste (atomične) attribute slabog entitetskog tipa  $X$  i proste komponente njegovih kompozitnih atributa, plus primarni ključ identificujućeg entitetskog tipa  $Y$
- ▶ Primarni ključ generisane relacione šeme je unija atributa primarnog ključa identificujućeg entitetskog tipa  $Y$  i atributa koji čine diskriminator slabog entitetskog tipa  $X$
- ▶ Atributi primarnog ključa identificujućeg entitetskog tipa  $Y$  u generisanoj relacionoj šemi čine strani ključ, koji referencira relacionu šemu u koju je mapiran identificujući entitetski tip  $Y$

# Mapiranje slabih entitetskih tipova **(Pravilo 2\*)**

---

- ▶ Primer: Mapiranje slabog entitetskog tipa

*TELEFON\_FAKULTETA*

```
create table telefon_fakulteta
(
    NazivFakulteta varchar(50),
    Telefon varchar(20),
    primary key (NazivFakulteta, Telefon),
    constraint FK_telefon_fakulteta_fakultet
        foreign key (NazivFakulteta)
            references fakultet (NazivFakulteta)
);
```

- ▶ Strani ključ može biti definisan prilikom kreiranja tabele koja odgovara slabom entitetskom tipu (ukoliko je kreirana tabela koja odgovara identifikujućem entitetskom tipu) ili naknadno **alter table** iskazom (nakon kreiranja tabele koja odgovara identifikujućem entitetskom tipu)

# Mapiranje specijalizacija **(Pravilo 3\*)**

---

- ▶ Koncept specijalizacije mapira se na jedan od sledećih načina:
  - A. **Vertikalni princip mapiranja (generisanje relacionih šema za superklasu i za potklase)**
    - ▶ Za entitetski tip  $X$ , koji reprezentuje superklasu, generiše se relaciona šema koja uključuje sve proste (atomične) attribute superklase  $X$ , kao i proste komponente kompozitnih attribute superklase
    - ▶ Primarni ključ generisane relacione šeme čine atributi primarnog ključa superklase
    - ▶ Za svaki entitetski tip nižeg nivoa/potklasu, generiše se relaciona šema koja uključuje attribute primarnog ključa superklase, plus sve proste (atomične) attribute potklase, kao i proste komponente kompozitnih attribute potklase
    - ▶ Primarni ključ generisane relacione šeme koja korespondira potklasi čine atributi primarnog ključa superklase, koji ujedno predstavljaju i strani ključ u generisanoj relacionoj šemi koji referencira primarni ključ u relacionoj šemi koja korespondira superklasi
    - ▶ **Ova opcija je primenljiva u svim varijantama specijalizacije (totalna ili parcijalna, disjunktna ili preklapajuća)**
    - ▶ **Alati tipično primenuju ovaj princip**

# Mapiranje specijalizacija **(Pravilo 3\*)**

---

## B. Horizontalni princip mapiranja (generisanje relacionih šema samo za potklase)

- ▶ Za svaku potklasu generiše se relaciona šema koja uključuje sve proste atribute superklase (uključujući i proste komponente kompozitnih atributa superklase) i sve proste atribute potklase (uključujući i proste komponente kompozitnih atributa potklase)
- ▶ Primarni ključ generisane relacione šeme čine atributi primarnog ključa superklase
- ▶ Ova opcija je prihvatljiva u slučajevima totalne i disjunktne specijalizacije

# Mapiranje specijalizacija **(Pravilo 3\*)**

---

## c. Generisanje jedne relacione šeme za superklasu i potklase sa diskriminatorom

- ▶ Ova varijanta mapiranja specijalizacije generiše jednu relacionu šemu za superklasu i sve potklase, koja uključuje uniju svih prostih atributa superklase i potklasa (uključujući i proste komponente kompozitnih atributa superklase i potklasa), plus dodatni diskriminirajući atribut na osnovu kojeg se identificuje pripadnost odgovarajućoj potklasi
- ▶ Pri tome, nije moguće specifikovati *not null* ograničenje na specifičnim atributima potklasa
- ▶ Ukoliko je specijalizacija definisana atributom, onda dodatni diskriminirajući atribut nije potreban
- ▶ Ukoliko je specijalizacija disjunktna, primarni ključ relacione šeme čine atributi primarnog ključa superklase
- ▶ U slučaju specijalizacije koja nije disjunktna, primarni ključ relacione šeme bila bi unija primarnog ključa superklase i diskriminatora
- ▶ Korespondentna relacija tipično sadrži veliki broj *null* vrednosti, pa ovo mapiranje može biti izbor ako su performanse pri pristupu podacima od kritičnog značaja, jer omogućava pristup traženim podacima bez spajanja relacija

# Mapiranje specijalizacija **(Pravilo 3\*)**

---

## D. Generisanje jedne relacione šeme za superklasu i potklase sa više indikatorskih atributa

- ▶ Ova varijanta mapiranja specijalizacije generiše jednu relacionu šemu za superklasu i sve potklase, koja uključuje uniju svih prostih atributa superklase i potklasa, uključujući proste komponente kompozitnih atributa superklase i potklasa, kao i dodatne indikatorske attribute  $F_1, \dots, F_m$  za potklase  $S_1, \dots, S_m$
- ▶ Pri tome, nije moguće specifikovati *not null* ograničenje na specifičnim atributima potklasa
- ▶ Primarni ključ generisane relacione šeme je primarni ključ superklase
- ▶ Ova varijanta mapiranja može biti prihvatljiva ako je specijalizacija preklapajuća, a kako korespondentna relacija može da sadrži veliki broj *null* vrednosti, ovo mapiranje može biti izbor ako su performanse pri pristupu podacima od kritičnog značaja

# Mapiranje specijalizacija **(Pravilo 3\*)**

---

## E. Mapiranje deljenih potklasa

- ▶ Ukoliko potklaša nasleđuje više superklasa, mogu se primeniti različite opcije 3A-3D za specijalizaciju po više putanja

# Mapiranje specijalizacija **(Pravilo 3\*)**

---

- ▶ Primer: Mapiranje natklase *OSOBA* i potklase *STUDENT* primenom principa vertikalnog mapiranja

```
create table osoba
(
    JMB char(13),
    Prezime varchar(20) not null,
    Ime varchar(20) not null,
    DatumRodjenja date not null,
    Adresa varchar(50) not null,
    primary key (JMB)
);
create table student
(
    JMB char(13),
    AdresaStudiranja varchar(50),
    primary key (JMB),
    constraint FK_student_osoba
    foreign key (JMB)
    references osoba (JMB)
);
```

# Mapiranje unija

## (**Pravilo 4**)

---

- ▶ Za svaku uniju se generiše posebna relaciona šema koja uključuje specifični **veštački (surogat) ključ VK** i sve specifične proste attribute unije (ako ih unija ima)
- ▶ Veštački ključ VK je primarni ključ generisane relateione šeme
- ▶ Dodatno, relateione šeme u koje se mapiraju superklase koje nasleđuje uniju potrebno je proširiti atributom veštačkog ključa VK unije
- ▶ Dodati atribut VK u relacionim šemama superklasa predstavlja strani ključ, koji referencira relacionu šemu koja se generiše za uniju
- ▶ Ako superklase unije imaju isti tip primarnog ključa  $K$  i disjunktne opsege vrednosti na primarnom ključu, onda se primarni ključ superklasa može koristiti umesto veštačkog ključa u relacionoj šemi koja korespondira uniji pa dodatni atribut u relacionim šemama superklasa nije potreban
- ▶ U tom slučaju ne možemo specifikovati  $K$  kao strani ključ niti u relacionim šemama koje korespondiraju superklasama niti u relacionoj šemi koja korespondira uniji

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja M:M (**Pravilo 5\***)

---

- ▶ Za svaki binarni M:M tip veze  $R$ , generiše se vezna relaciona šema koja uključuje atributе primarnih ključeva entitetskih tipova koji učestvuju u tipu veze  $R$ , plus sve proste atributе i proste komponente kompozitnih atributа tipa veze  $R$
- ▶ Primarni ključ generisane relacione šeme je unija atributa primarnih ključeva entitetskih tipova koji učestvuju u veznom tipu
- ▶ Generisana relaciona šema ima dva strana ključa: prvi strani ključ čine atributi primarnog ključa prvog entitetskog tipa koji učestvuje u veznom tipu, a drugi strani ključ čine atributi primarnog ključa drugog entitetskog tipa koji učestvuje u veznom tipu

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja M:M (**Pravilo 5\***)

---

- ▶ Primer: Mapiranje veznog tipa *UPISAO* (prepostavlja se da je mapiranje odnosnih entitetskih tipova već izvršeno)

```
create table upisao
(
    JMB char(13),
    IdPredmeta int,
    IdSP int,
    primary key (JMB, IdPredmeta, IdSP),
    constraint FK_upisao_student
        foreign key (JMB)
            references student (JMB),
    constraint FK_upisao_p_na_sp
        foreign key (IdPredmeta, IdSP)
            references p_na_sp (IdPredmeta, IdSP)
);
```

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja 1:M (**Pravilo 6\***)

---

- ▶ Ako je participacija entitetskog tipa  $Y$  sa M strane **veznog tipa  $R$  totalna**, onda se tip veze  $R$  ne mapira u novu relacionu šemu, već se u relacionu šemu  $T$ , generisanu za entitetski tip  $Y$ , dodaju atributi primarnog ključa entitetskog tipa  $X$  koji u vezi participira sa strane jedan (**priступ sa korištenjem stranog ključa**)
- ▶ Dodati atributi čine strani ključ u relacionoj šemi  $T$ , koji referencira relacionu šemu  $S$  u koju se mapira entitetski tip  $X$
- ▶ Ukoliko tip veze  $R$  ima proste opisne attribute, oni se takođe uključuju u relacionu šemu  $T$ , kao i proste komponente kompozitnih atributa tipa veze  $R$

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja 1:M (**Pravilo 6\***)

---

- ▶ Ako participacija entitetskog tipa  $Y$  sa strane  $M$  veze  $R$  nije totalna, onda postoje dve opcije:
  - A. Primena **pristupa sa korištenjem stranog ključa** pri čemu se za atribute dodate u relacionu šemu  $T$ , generisanu za entitetski tip  $Y$ , ne sme definisati *not null* ograničenje
  - B. Generisanje posebne vezne relacione šeme za tip veze  $R$ , **analogno** relacionoj šemi u koju se prevodi **vezni tip M:M**, pri čemu primarni ključ generisane relacione šeme čine atributi primarnog ključa entitetskog tipa  $Y$
- ▶ **Alati tipično primenjuju pristup sa korištenjem stranog ključa**

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja 1:M (**Pravilo 6\***)

---

- ▶ Primer: Mapiranje entitetskog tipa *PREDMET* i veznog tipa *MATICAN\_ZA*

```
create table predmet
(
    IdPredmeta int,
    NazivPredmeta varchar(50) not null,
    ECTS tinyint not null,
    NazivFakulteta varchar(50) not null,
    primary key (IdPredmeta),
    constraint FK_predmet_fakultet
        foreign key (NazivFakulteta)
        references fakultet (NazivFakulteta)
);
```

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja 1:1 (***Pravilo 7\****)

---

- ▶ Ako je participacija entitetskog tipa  $Y$  u tipu veze  $R$  sa kardinalnošću mapiranja **1:1 totalna**, onda se tip veze  $R$  u ne mapira u novu relacionu šemu, već treba primeniti **pristup sa korištenjem stranog ključa**, opisan u *pravilu 6*
- ▶ Ako je participacija oba entitetska tipa  $X$  i  $Y$  u **1:1 tipu veze  $R$  totalna**, onda postoje dve opcije:
  - ▶ **Opcija sa primenom stranog ključa**, pri čemu je izbor šeme koja korespondira jednom od entitetskih tipova  $X$  ili  $Y$ , kojoj se dodaju atributi stranog ključa i atributi veze, proizvoljan
  - ▶ **Opcija spajanja relacija**, gde se može kreirati jedna relaciona šema koja bi reprezentovala i entitetske tipove  $X$  i  $Y$ , i vezni tip  $R$ , a koja bi uključivala sve proste atribute entitetskih tipova  $X$  i  $Y$ , i proste opisne atribute veznog tipa (uključujući i proste komponente kompozitnih atributa i entitetskih tipova  $X$  i  $Y$  i veznog tipa  $R$ )

# Mapiranje binarnih tipova veze sa kardinalnošću mapiranja 1:1 (***Pravilo 7\****)

---

- ▶ Ako participacija nijednog entitetskog tipa koji učestvuju u 1:1 tipu veze *R* nije totalna, onda su moguća dva pristupa, kao i za slučaj veze sa mapiranjem 1:M, gde participacija entiteta entitetskog skupa sa strane M nije totalna:
  - ▶ Prvi pristup je pristup sa **korištenjem stranog ključa**, sa napomenama analognim u *pravilu 6A*
  - ▶ Drugi pristup je **formiranje vezne relacione šeme**, koja se formira po *pravilu 6B* i uz iste napomene
- ▶ **Alati tipično primenjuju pristup sa korištenjem stranog ključa, pri čemu je moguće da se za dodate kolone stranog ključa definiše i ograničenje jedinstvenosti vrednosti**

# Mapiranje $n$ -arnih tipova veze **(Pravilo 8)**

---

- ▶ Za svaki  $n$ -arni tip veze  $R$ , kreira se relacione šeme koja uključuje attribute primarnih ključeva entitetskih tipova koji učestvuju u veznom tipu i proste attribute veznog tipa (uključujući proste komponente kompozitnih atributa tipa veze)
- ▶ Atributi primarnog ključa generisane relacione šeme čine atributi primarnog ključa tipa veze
- ▶ Primarni ključ tipa veze je podskup primarnih ključeva entitetskih tipova koji učestvuju u veznom tipu i zavisan je od kardinalnosti mapiranja  $n$ -arnog tipa veze
- ▶ Atributi primarnih ključeva pojedinih entitetskih tipova u generisanoj šemi su strani ključevi koji referenciraju korespondentne relacione šeme odnosnih entitetskih tipova koji participiraju u veznom tipu

# Mapiranje agregacija **(Pravilo 9)**

---

- ▶ Agregacija se transformiše u relacione šeme sledeći prethodno definisana pravila
- ▶ Vezni tip koji se tretira kao agregacija, prvo se mapira u relacionu šemu po pravilima za mapiranje binarnog ili  $n$ -arnog tipa veze
- ▶ Pri povezivanju agregacije tipom veze  $R$  sa drugim entitetskim tipom/tipovima, agregacija se tretira kao entitetski tip višeg nivoa, a vezni tip mapira u relacionu šemu po pravilima za tip veze

# Mapiranje višeznačnih atributa **(Pravilo 10)**

---

- ▶ Za svaki višeznačni atribut  $M$  kreira se posebna relaciona šema
- ▶ Atribute korespondentne relacione šeme  $V$  čini višeznačni atribut  $M$  i atributi primarnog ključa  $K$  relacione šeme  $S$  koja reprezentuje entitetski tip ili tip veze kojem  $M$  pripada
- ▶ Ako je atribut  $M$  kompozitni, onda sve njegove proste komponente postaju atributi relacione šeme  $V$
- ▶ Atributi  $K$  u kreiranoj relacionoj šemi  $V$  čine strani ključ koji referencira šemu  $S$
- ▶ Primarni ključ relacione šeme  $V$  su svi atributi kreirane relacione šeme (unija atributa  $K$  i atributa koji potiču od višeznačnog atributa  $M$ )

```
source = database.  
getConnection()  
connection.createStatement()  
selectSQL = "SELECT * FROM"  
statement.execute()  
resultSet.next()  
resultSet.
```

# BAZE PODATAKA

SQL – DML

# Uvod

---

- ▶ **SQL** (*Structured Query Language*) je upitni jezik visokog nivoa za rad sa relacionim bazama podataka, a baziran je na relacionom računu i relacionoj algebri
- ▶ Nastao je 70-tih godina prošlog veka u IBM-ovim istraživačkim laboratorijama u San Hozeu, u okviru *System R* projekta koji je trebao da istraži i demonstrira mogućnosti relationalnih DBMS
- ▶ Sastoje se od dve komponente:
  - ▶ **DDL** (*Data Definition Language*) komponenta – omogućava definisanje objekata u bazi podataka
  - ▶ **DML** (*Data Manipulation Language*) komponenta – omogućava manipulaciju podacima u bazi podataka

# Selekcija i prikaz podataka

---

- ▶ Jedna od najznačajnijih korisničkih aktivnosti je ekstrakcija podataka iz baze i prikaz podataka korisniku u odgovarajućoj formi, što se realizuje **upitima**
- ▶ Upit se sastoji od, minimalno, dve klauzule:
  - ▶ **select** – koristi za selekciju kolona čije se vrednosti žele dobiti u rezultatu
  - ▶ **from** – koristi se za specifikaciju liste naziva tabela
- ▶ Klauzula **where** je opcionalna, a koristi se za specifikaciju uslova koji treba da zadovolje redovi koji će biti prikazani u rezultatu
- ▶ **Rezultat SQL upita je tabela**

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju svi podaci iz tabele *osoba*

```
select *  
from osoba;
```

JMB	Prezime	Ime	DatumRodjenja	Adresa
0308983126116	Mirković	Ana	1983-08-03	Ulica 3
0512983100067	Popović	Slavko	1983-12-05	Ulica 17
0607989100027	Stojanović	Danihel	1989-07-06	Ulica 19
0702964105027	Gavrić	Mirjana	1964-02-07	Ulica 74
1002952100005	Mitrović	Nikola	1952-02-10	Ulica 63
1006949100067	Soldat	Stanko	1949-06-10	Ulica 101
1010988101124	Babić	Dejan	1988-10-10	Ulica 5
1206986101234	Mirković	Marko	1986-06-12	Ulica 1
1210987100018	Janković	Petar	1987-10-12	Ulica 16
1312981163309	Filipović	Mirko	1981-12-13	Ulica 70
1503990125037	Vasković	Jelena	1990-03-15	Ulica 6
1804964163303	Savić	Nenad	1964-04-18	Ulica 26
1907951100012	Lazić	Zoran	1951-07-19	Ulica 30
2102979163201	Janković	Janko	1979-02-21	Ulica 2
2108968196769	Petković	Milena	1968-08-21	Ulica 84
2108988105034	Đukić	Milijsana	1988-08-21	Ulica 34
2208988105039	Miljević	Branka	1988-08-22	Ulica 80
2804950103891	Ninković	Miloš	1950-04-28	Ulica 4

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju prezimena svih osoba

```
select all Prezime from osoba;
```

Ime	Prezime	Ime	Datum	Rodjenja	Adresa	
Mirko	ić	16	Mirković	Ara	1983-8-03	Ulica 3
Đorđe	Popović	Slavko	1983-2-05	Ulica 17		
Stojan	ović	27	Stojanović	Danihel	1989-7-06	Ulica 19
Gavrilo	4105027	Gavrić	Marijana	1964-2-07	Ulica 74	
Mitro	ić	05	Mitrović	Nikola	1952-2-10	Ulica 63
Soldat	9100067	Soldat	Slavko	1949-6-10	Ulica 101	
Babić		24	Babić	Dejan	1988-0-10	Ulica 5
Mirko	6601234	Mirković	Mirko	1986-6-12	Ulica 1	
Janko	ić	18	Janković	Petar	1987-0-12	Ulica 16
Filip	7163309	Filipović	Mirko	1981-2-13	Ulica 70	
Vasko	ić	37	Vasković	Jelena	1990-3-15	Ulica 6
Savić	4163303	Savić	Nenad	1964-4-18	Ulica 26	
Lazić		12	Lazić	Zoran	1951-7-19	Ulica 30
Janko	9663201	Janković	Janko	1979-2-21	Ulica 2	
Petko	ić	69	Petković	Milena	1968-8-21	Ulica 84
Đukić	8105034	Đukić	Milijana	1988-8-21	Ulica 34	
Milje	ić	39	Miljević	Branka	1988-8-22	Ulica 80
Ninković	0603891	Ninković	Miloš	1950-4-28	Ulica 4	

- ▶ Kako se opcija **all** podrazumeva (duplicati se ne eliminišu iz rezultata), prethodni upit ekvivalentan je sledećem upitu:

```
select Prezime from osoba;
```

# Selekcija i prikaz podataka

---

- ▶ Primer: SQL upit kojim se prikazuju prezimena svih osoba (pri čemu se eliminišu duplikati)

```
select distinct Prezime  
from osoba;
```

Prezime
Mirković
Popović
Stojanović
Gavrić
Mitrović
Soldat
Babić
Janković
Filipović
Vasković
Savić
Lazić
Petković
Đukić
Miljević
Ninković

# Selekcija i prikaz podataka

---

- ▶ Redovi se selektuju navođenjem **where** klauzule i uslova koji mora biti ispunjen
- ▶ Simboli osnovnih operatora poređenja u SQL-u su: =, <>, <, <=, > i >=
- ▶ Logički operatori su: **and**, **or** i **not**
- ▶ Prioritet logičkih operatora, počevši od operatara sa najvišim prioritetom je:
  - ▶ **not**
  - ▶ **and**
  - ▶ **or**

# Selekcija i prikaz podataka

- Primer: SQL upit kojim se prikazuju identifikatori i nazivi predmeta koji imaju više od šest ECTS bodova

```
select IdPredmeta, NazivPredmeta  
from predmet  
where ECTS>6;
```

<b>IdPredmeta</b>	<b>NazivPredmeta</b>	<b>CTS</b>	<b>NazivFakulteta</b>
1111	Programski jezici 1	Elektrotehnički fakultet	
1112	Programski jezici 2	Elektrotehnički fakultet	
1113	Programski jezici 3	Elektrotehnički fakultet	
1214	Baze podataka računarskih sistema	Elektrotehnički fakultet	
1221	Digitalne sisteme	Elektrotehnički fakultet	
1222	Digitalna programiranje	Elektrotehnički fakultet	
1122	Analogni i digitalni filtri	Elektrotehnički fakultet	
1211	Elementarni principi informacije	Elektrotehnički fakultet	
1211	Internet tehnologije	Elektrotehnički fakultet	
1212	Sigurnost računarskih sistema	Elektrotehnički fakultet	
1213	Baze podataka (viši nivo)	Elektrotehnički fakultet	
1311	Internet programiranje	Elektrotehnički fakultet	
1312	Kriptografija	Elektrotehnički fakultet	
2111	Linearna algebrija	Matematički fakultet	
2112	Ravnina geometrija	Matematički fakultet	
2113	Analitička geometrija	Matematički fakultet	
2114	Fizika atoma i molekula	Prirodno-matematički fakultet	

# Selekcija i prikaz podataka

---

- ▶ Primer: SQL upit kojim se prikazuju nazivi studijskih programa sa prvog ciklusa studija na Elektrotehničkom fakultetu

```
select NazivSP  
from studijski_program  
where Ciklus=1 and NazivFakulteta='Elektrotehnički fakultet';
```

---

NazivSP
Računarstvo i informatika
Elektronika i telekomunikacije

# Selekcija i prikaz podataka

---

- ▶ Ukoliko se traži provera nekog intervala, uključujući granice, onda se umesto operatora poređenja može koristiti operator **between**
- ▶ Primer: SQL upit kojim se prikazuju identifikatori, nazivi i ECTS bodovi predmeta za koje je vrednost identifikatora između 1111 i 1121

```
select IdPredmeta, NazivPredmeta, ECTS  
from predmet  
where IdPredmeta between 1111 and 1121;
```

IdPredmeta	NazivPredmeta	ECTS
1111	Programski jezici 1	7
1112	Programski jezici 2	6
1113	Softversko inženjerstvo	6
1114	Baze podataka	8
1115	Informacioni sistemi	6
1121	Digitalne telekomunikacije	5

# Selekcija i prikaz podataka

---

- ▶ Ukoliko se traži provera pripadnosti vrednosti kolone nekoj listi (skupu) vrednosti, može se koristiti operator **in**
- ▶ Primer: SQL upit kojim se prikazuju identifikatori, nazivi i ECTS bodovi predmeta čiji identifikator ima vrednost 1111, 1114 ili 1311

```
select IdPredmeta, NazivPredmeta, ECTS  
from predmet  
where IdPredmeta in (1111, 1114, 1311);
```

IdPredmeta	NazivPredmeta	ECTS
1111	Programski jezici 1	7
1114	Baze podataka	8
1311	Internet programiranje	7

# Selekcija i prikaz podataka

---

- ▶ Poređenje uzorka sa stringovima (nizovima znakova) obavlja se korištenjem operatora **like**
- ▶ Specijalni znakovi koji se mogu koristiti pri zadavanju uzorka su:
  - ▶ % (postotak) – odgovara bilo kojem nizu (od nula ili više) znakova
  - ▶ \_ (donja crta) – odgovara tačno jednom, bilo kojem znaku
- ▶ Primer: SQL upit kojim se prikazuju prezimena i imena osoba čije prezime završava slovima 'ić', a ime slovom 'a'

```
select Prezime, Ime  
from osoba  
where Prezime like '%ić' and Ime like '%a';
```

Prezime	Ime
Mirković	Ana
Gavrić	Mirjana
Mitrović	Nikola
Vasković	Jelena
Petković	Milena
Đukić	Milijana
Miljević	Branka

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju prezimena i imena osoba čije ime sadrži slova 'an'

```
select Prezime, Ime  
from osoba  
where Ime like '%an%';
```

Prezime	Ime
Mirković	Ana
Stojanović	Danijel
Gavrić	Mirjana
Soldat	Stanko
Babić	Dejan
Lazić	Zoran
Janković	Janko
Đukić	Milijana
Miljević	Branka

- ▶ Primer: SQL upit kojim se prikazuju prezimena i imena osoba kod kojih je drugo slovo imena 'i', a treće 'r'

```
select Prezime, Ime  
from osoba  
where Ime like '_ir%';
```

Prezime	Ime
Gavrić	Mirjana
Filipović	Mirko

# Selekcija i prikaz podataka

---

- ▶ U SQL-u je omogućeno preimenovanje zaglavlja kolone, navođenjem ključne reči **as** (opciono) i alternativnog naziva (alijasa) nakon stvarnog naziva
- ▶ **Select** klauzula može da sadrži i izraze koji se mogu formirati od kolona sa numeričkim domenima, numeričkih konstanti, aritmetičkih operatora i različitih funkcija
- ▶ Osnovni aritmetički operatori koji se mogu koristiti pri sračunavanju vrednosti, ali i u **where** klauzuli, su: +, -, \* i /
- ▶ Osnovne funkcije (MySQL):
  - ▶ Algebarske funkcije
    - ▶ **abs(*b*)**
    - ▶ **mod(*b, d*)**
    - ▶ **pow(*b, n*)**
    - ▶ **sqrt(*b*)**
    - ▶ **round(*b, d*)**
    - ▶ **truncate(*b, d*)**

# Selekcija i prikaz podataka

---

- ▶ Osnovne funkcije (MySQL):
  - ▶ Funkcije za manipulaciju stringovima
    - ▶ `upper(s)`
    - ▶ `lower(s)`
    - ▶ `char_length(s)`
    - ▶ `concat(s1, s2, ...)`
    - ▶ `substr(s, p, d)`
    - ▶ `trim(s)`
  - ▶ Funkcije za manipulaciju datumima
    - ▶ `str_to_date(s, f)`
    - ▶ `date_format(d, f)`
    - ▶ `day(d)`
    - ▶ `month(d)`
    - ▶ `year(d)`
    - ▶ `hour(t)`
    - ▶ `minute(t)`
    - ▶ `second(t)`
    - ▶ `sysdate()`

# Selekcija i prikaz podataka

---

## ▶ Osnovi specifikatori formata datuma (MySQL):

- ▶ %c – mesec (0...12)
- ▶ %d – dan u mesecu (00...31)
- ▶ %e – dan u mesecu (0...31)
- ▶ %H – sat (00...23)
- ▶ %h – sat (01...12)
- ▶ %i – minuta (00...59)
- ▶ %k – sat (0...23)
- ▶ %l – sat (1...12)
- ▶ %m – mesec (00...12)
- ▶ %p – AM ili PM
- ▶ %S – sekunda (00...59)
- ▶ %s – sekunda (00...59)
- ▶ %Y – godina, 4 cifre
- ▶ %y – godina, 2 cifre

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi i iznosi plata, izraženi u evrima (1 EUR = 1.95 KM), svih nastavnika

```
select JMB, Plata/1.95 as Plata_EUR  
from nastavnik;
```

JMB	Plata_EUR
0702964105027	923.076923
1002952100005	1230.769231
1006949100067	1025.641026
1804964163303	1025.641026
1907951100012	1179.487179
2108968196769	923.076923
2804950103891	1230.769231

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi i iznosi plata, izraženi u evrima (1 EUR = 1.95 KM) i zaokruženi na dve decimale, svih nastavnika

```
select JMB,  
round(Plata/1.95, 2) as Plata_EUR  
from nastavnik;
```

JMB	Plata_EUR
0702964105027	923.08
1002952100005	1230.77
1006949100067	1025.64
1804964163303	1025.64
1907951100012	1179.49
2108968196769	923.08
2804950103891	1230.77

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena i datumi rođenja (formatirano) osoba čije ime ima barem sedam slova

```
select JMB, Prezime, Ime,  
date_format(DatumRodjenja, '%d.%m.%Y.') as DatumRodjenja  
from osoba  
where char_length(Ime)>=7;
```

JMB	Prezime	Ime	DatumRodjenja
0607989100027	Stojanović	Danijel	06.07.1989.
0702964105027	Gavrić	Mirjana	07.02.1964.
2108988105034	Đukić	Milijana	21.08.1988.

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi i inicijali osoba koje su rođene 1983. godine

```
select JMB,  
concat(substr(Ime, 1, 1), '.',  
substr(Prezime, 1, 1), '.') as Inicijali  
from osoba  
where year(DatumRodjenja)=1983;
```

JMB	Inicijali
0308983126116	A.M.
0512983100067	S.P.

# Selekcija i prikaz podataka

---

- ▶ Prilikom specifikacije upita moguće je koristiti **case** izraz, koji ima dve forme:

```
case
  when uslov1 then rezultat1
  [when uslov2 then rezultat2]
  ...
  [when uslovn then rezultatn]
  [else rezultat]
end
```

```
case izraz
  when izraz1 then rezultat1
  [when izraz2 then rezultat2]
  ...
  [when izrazn then rezultatn]
  [else rezultat]
end
```

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju identifikatori, nazivi i ciklusi (slovima) svih studijskih programa

```
select IdSP, NazivSP,  
       case Ciklus  
           when 1 then 'Prvi'  
           when 2 then 'Drugi'  
           when 3 then 'Treći'  
           else 'Nepoznato'  
       end as CiklusStudija  
  from studijski_program;
```

IdSP	NazivSP	CiklusStudija
111	Računarstvo i informatika	Prvi
112	Elektronika i telekomunikacije	Prvi
121	Računarstvo i informatika	Drugi
131	Informaciono-komunikacione tehnologije	Treći
211	Fizika	Prvi

# Selekcija i prikaz podataka

- ▶ Redovi rezultata se mogu sortirati korištenjem **order by** klauzule
- ▶ Podrazumeva se rastući redosled, koji se može eksplicitno specifikovati opcijom **asc**
- ▶ Opadajući redosled se može specifikovati opcijom **desc**
- ▶ Primer: SQL upit kojim se prikazuju godine rođenja, prezimena i imena svih osoba, sortirano opadajuće po godini rođenja, a rastuće po prezimenu i imenu

```
select year(DatumRodjenja) as  
GodinaRodjenja, Prezime, Ime  
from osoba  
order by GodinaRodjenja desc,  
Prezime asc, Ime asc;
```

GodinaRodjenja	Prezime	Ime
1990	Vasković	Jelena
1989	Stojanović	Danihel
1988	Babić	Dejan
1988	Miljević	Branka
1988	Đukić	Milijana
1987	Janković	Petar
1986	Mirković	Marko
1983	Mirković	Ana
1983	Popović	Slavko
1981	Filipović	Mirko
1979	Janković	Janko
1968	Petković	Milena
1964	Gavrić	Mirjana
1964	Savić	Nenad
1952	Mitrović	Nikola
1951	Lazić	Zoran
1950	Ninković	Miloš
1949	Soldat	Stanko

# Selekcija i prikaz podataka

---

- ▶ SQL podržava koncept *null* vrednosti
- ▶ Izvršavanje aritmetičke operacije koja uključuje kao operand *null* vrednost, daje *null* vrednost kao rezultat
- ▶ Operacije poređenja (=, <>, <, <=, >, >=), kada je jedan ili oba operanda *null* vrednost, daju kao rezultat logičku vrednost *null* (koja odgovara logičkoj vrednosti *unknown* koja se koristi u terminologiji relacione algebre)
- ▶ Specifikacija predikata u SQL-u najčešće uključuje više članova čije vrednosti mogu biti *istinito*, *lažno* i *null*, povezanih logičkim operatorima **and**, **or** i **not**
- ▶ Za proveru da li je neka vrednost *null*, koriste se ključne reči **is null**, odnosno **is not null**
- ▶ Način obrade *null* vrednosti može da se specifikuje funkcijom **coalesce**
- ▶ Funkcija **coalesce** prihvata promenljih broj argumenata, a vraća prvu vrednost koja nije *null* ili vraća *null* ako su sve vrednosti jednake *null*:  
**coalesce(vrednost<sub>1</sub>, vrednost<sub>2</sub>, ...)**

# Selekcija i prikaz podataka

- ▶ Rezultat izvršenja logičkih operacija, pri čemu jedan ili oba operanda mogu imati vrednost *null*:

p	not p
<i>true</i>	<i>false</i>
<i>false</i>	<i>true</i>
<i>null</i>	<i>null</i>

p	t	p and t	p or t
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>null</i>	<i>null</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>null</i>	<i>false</i>	<i>null</i>
<i>null</i>	<i>true</i>	<i>null</i>	<i>true</i>
<i>null</i>	<i>false</i>	<i>false</i>	<i>null</i>
<i>null</i>	<i>null</i>	<i>null</i>	<i>null</i>

# Selekcija i prikaz podataka

---

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi nastavnika i matični brojevi njihovih šefova katedri

```
select JMB, JMBSefaKatedre  
from nastavnik;
```

JMB	JMBSefaKatedre
0702964105027	NULL
1907951100012	NULL
2804950103891	NULL
1002952100005	1907951100012
1006949100067	1907951100012
2108968196769	1907951100012
1804964163303	2804950103891

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi nastavnika koji nemaju šefa katedre

```
select JMB  
from nastavnik  
where JMBSefaKatedre is null;
```

JMB
0702964105027
1907951100012
2804950103891

# Selekcija i prikaz podataka

---

- ▶ **Agregatne funkcije** imaju kao argument kolekciju vrednosti i vraćaju kao rezultat jednu, skalarnu agregatnu vrednost, koja reprezentuje neku zbirnu karakteristiku kolekcije
- ▶ *Ne mogu se pojaviti u **where** klauzuli*
- ▶ Osnovne agregatne funkcije:
  - ▶ **count(\*)**
  - ▶ **count(*n*) ili count(all *n*)**
  - ▶ **count(distinct *n*)**
  - ▶ **sum(*n*) ili sum(all *n*)**
  - ▶ **sum(distinct *n*)**
  - ▶ **avg(*n*) ili avg(all *n*)**
  - ▶ **avg(distinct *n*)**
  - ▶ **max(*n*)**
  - ▶ **min(*n*)**

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuje prosek plate nastavnika koji imaju nastavno zvanje 'redovni profesor'

```
select avg(Plata) as ProsekPlate  
from nastavnik  
where NastavnoZvanje='redovni profesor';
```

ProsekPlate
2366.666667

- ▶ Primer: SQL upit kojim se prikazuje najmanji i najveći iznos plate svih nastavnika

```
select min(Plata) as MinPlata,  
max(Plata) as MaxPlata  
from nastavnik;
```

MinPlata	MaxPlata
1800.00	2400.00

- ▶ Primer: SQL upit kojim se prikazuje ukupan broj nastavnika koji nemaju šefa katedre

```
select count(*) as Ukupno  
from nastavnik  
where JMBSefaKatedre is null;
```

Ukupno
3

# Selekcija i prikaz podataka

- ▶ Često postoji potreba da se dobiju **agregatne vrednosti za disjunktne grupe redova**
- ▶ Za specifikaciju grupisanja redova koristi se **group by** klauzula
- ▶ Primer: SQL upit kojim se prikazuju prosečni iznosi plata nastavnika (zaokruženi na dve decimale) po pojedinim nastavnim zvanjima

```
select NastavnoZvanje, round(avg(Plata), 2) as ProsekPlate  
from nastavnik  
group by NastavnoZvanje;
```

NastavnoZvanje	ProsekPlate
docent	1800.00
redovni profesor	2366.67
vanredni profesor	2000.00

- ▶ MySQL funkcija **group\_concat** omogućava konkatenaciju vrednosti grupe
- ▶ Primer: SQL upit kojim se prikazuje naziv fakulteta i brojevi telefona

```
select NazivFakulteta, group_concat(Telefon) as Telefon  
from telefon_fakulteta  
group by NazivFakulteta;
```

NazivFakulteta	Telefon
Elektrotehnički fakultet	+387 (0) 51 221 820, +387 (0) 51 221 855
Prirodno-matematički fakultet	+387 (0) 51 319 142

# Selekcija i prikaz podataka

---

- ▶ Kao što se u **where** klauzuli navode uslovi koje svaki red mora da zadovolji da bi pripao rezultatu, tako se u **having** klauzuli navode uslovi koje svaka grupa mora da zadovolji da bi pripala rezultatu
- ▶ Primer: SQL upit kojim se prikazuju prosečni iznosi plata nastavnika (zaokruženi na dve decimale) po pojedinim nastavnim zvanjima koja imaju barem tri nastavnika

```
select NastavnoZvanje, round(avg(Plata), 2) as ProsekPlate  
from nastavnik  
group by NastavnoZvanje  
having count(*)>=3;
```

NastavnoZvanje	ProsekPlate
redovni profesor	2366.67

# Selekcija i prikaz podataka

---

- ▶ Navođenjem više tabela u **from** klauzuli omogućava se spajanje informacija iz različitih tabela Dekartovim proizvodom
- ▶ Navođenjem uslova u **where** klauzuli, tako da u rezultat ulaze samo međusobno odgovarajuće/uparene informacije redova dobijenih Dekartovim proizvodom (koji eventualno treba da zadovolje i dodatne uslove za selekciju), pri čemu lista kolona u **select** klauzuli projektuje selektovane kombinacije redova na željeni skup vrednosti, dobija se forma upita koja se često naziva **select-project-join**
- ▶ Ako se u **where** klauzuli specifikuje spajanje tabela iz **from** klauzule po uslovu jednakosti na zajedničkim kolonama, a u **select** klauzuli specifikuje lista svih različitih kolona, dobija se upit koji odgovara **prirodnom spajanju tabela**

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena i iznosi plata nastavnika koji imaju nastavno zvanje 'docent'

```
select n.JMB, Prezime, Ime, Plata  
from nastavnik n, osoba o  
where n.JMB=o.JMB and NastavnoZvanje='docent';
```

JMB	Prezime	Ime	Plata
0702964105027	Gavrić	Mirjana	1800.00
2108968196769	Petković	Milena	1800.00

- ▶ Primer: SQL upit kojim se realizuje prirodno spajanje tabela *fakultet* i *telefon\_fakulteta*

```
select f.NazivFakulteta, Adresa, Telefon  
from fakultet f, telefon_fakulteta tf  
where f.NazivFakulteta=tf.NazivFakulteta;
```

NazivFakulteta	Adresa	Telefon
Elektrotehnički fakultet	Patre 5, Banja Luka	+387 (0) 51 221 820
Elektrotehnički fakultet	Patre 5, Banja Luka	+387 (0) 51 221 855
Prirodno-matematički fakultet	Mladena Stojanovića 2, Banja Luka	+387 (0) 51 319 142

# Selekcija i prikaz podataka

---

- ▶ Koncept **spojene tabele** ima za cilj poboljšanje preglednosti i razumljivosti upita objedinjavanjem specifikacije Dekartovog proizvoda i uslova spajanja u **from** klauzuli upita, tako da se u **where** klauzuli specifikuju samo uslovi selekcije
- ▶ Uslovi spajanja u **from** klauzuli se specifikuju **join** operatorom (u različitim varijantama), čime se kao rezultat procesiranja **from** klauzule dobija jedna, spojena tabela
- ▶ Vrste spajanja su:
  - ▶ **inner join**
  - ▶ **left outer join**
  - ▶ **right outer join**
  - ▶ **full outer join**
  - ▶ **cross join**
- ▶ Uslovi spajanja mogu biti:
  - ▶ **natural**
  - ▶ **on uslov**
  - ▶ **using( $k_1, k_2, \dots, k_n$ )**

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena i imena asistenata koji imaju saradničko zvanje 'viši asistent'

```
select a.JMB, Prezime, Ime  
from asistent a  
inner join osoba o on o.JMB=a.JMB  
where SaradnickoZvanje='viši asistent';
```

JMB	Prezime	Ime
0308983126116	Mirković	Ana
1312981163309	Filipović	Mirko
2102979163201	Prezime	Janko

- ▶ Primer: SQL upit kojim se prikazuju identifikatori svih predmeta koji se izvode na studijskom programu čiji je identifikator 211, semestri i tipovi predmeta, te matični brojevi asistenta koji asistiraju na datim predmetima

```
select psp.IdPredmeta, Semestar, TipPredmeta, JMB  
from p_na_sp psp  
left outer join asistira a on a.IdPredmeta=psp.IdPredmeta and  
a.IdSP=psp.IdSP  
where psp.IdSP=211;
```

IdPredmeta	Semestar	TipPredmeta	JMB
2112	3	I	1312981163309
2113	5	O	1312981163309
2114	6	O	NULL

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena i imena studenata, nazivi i ciklusi studijskih programa, te prosečne ocene studenata (po studijskim programima), sortirano u opadajućem redosledu po prosečnoj oceni

```
select o.JMB, Prezime, Ime, NazivSP, Ciklus,
round(avg(Ocena), 2) as ProsekOcena
from osoba o
inner join polaze p on p.JMB=o.JMB
inner join studijski_program sp on sp.IdSP=p.IdSP
where Ocena>5
group by o.JMB, p.IdSP
order by ProsekOcena desc;
```

JMB	Prezime	Ime	NazivSP	Ciklus	ProsekOcena
1206986101234	Mirković	Marko	Računarstvo i informatika	2	9.67
2208988105039	Miljević	Branka	Fizika	1	9.33
1206986101234	Mirković	Marko	Računarstvo i informatika	1	8.83
1210987100018	Janković	Petar	Računarstvo i informatika	1	7.80
2108988105034	Đukić	Milijana	Elektronika i telekomunikacije	1	7.33
1010988101124	Babić	Dejan	Računarstvo i informatika	1	6.67

# Selekcija i prikaz podataka

- ▶ Operatore spajanja možemo grupisati korištenjem **malih zagrada**
- ▶ Primer: SQL upit kojim se prikazuju identifikatori svih predmeta koji se izvode na studijskom programu čiji je identifikator 211, semestri i tipovi predmeta, te matični brojevi, prezime i imena asistenta koji asistiraju na datim predmetima

```
select psp.IdPredmeta, Semestar, TipPredmeta, a.JMB, Prezime, Ime
from p_na_sp psp
left outer join (
    asistira a inner join osoba o on o.JMB=a.JMB
) on a.IdPredmeta=psp.IdPredmeta and a.IdSP=psp.IdSP
where psp.IdSP=211;
```

IdPredmeta	Semestar	TipPredmeta	JMB	Prezime	Ime
2112	3	I	1312981163309	Filipović	Mirko
2113	5	O	1312981163309	Filipović	Mirko
2114	6	O	NULL	NULL	NULL

# Selekcija i prikaz podataka

---

- ▶ SQL upit se može specifikovati u okviru drugog, spoljašnjeg SQL upita, što se označava terminom **gnežđenje** (eng. *nesting*) **SQL upita**
- ▶ Podupit, kao i svaki drugi SQL upit, daje kao rezultat tabelu koja može da sadrži rezultantne redove ili da bude prazna tabela
- ▶ Rezultat podupita se najčešće koristi za specifikaciju uslova selekcije redova u rezultat (u **where** klauzuli spoljašnjeg upita)
- ▶ Kada je rezultat podupita tabela sa jednom kolonom i jednom vrednošću, odnosno jedna, skalarna vrednost, dozvoljeno je korištenje operatora poređenja ( $=$ ,  $<>$ ,  $<$ ,  $<=$ ,  $>$  i  $>=$ ) za poređenje vrednosti kolone i rezultata podupita
- ▶ Ukoliko je rezultat podupita kolekcija vrednosti onda je moguće poređenje vrednosti neke kolone sa članovima kolekcije (ili skupa), korištenjem operatora poređenja ( $=$ ,  $<>$ ,  $<$ ,  $<=$ ,  $>$  i  $>=$ ) u kombinaciji sa ključnom reči **any** (**some** je sinonim) ili **all**

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena i imena šefova katedri

```
select JMB, Prezime, Ime  
from osoba  
where JMB in (select JMBSefaKatedre from nastavnik);
```

JMB	Prezime	Ime
1907951100012	Lazić	Zoran
2804950103891	Ninković	Miloš

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena i iznosi plata nastavnika koji imaju platu veću od prosečne plate svih nastavnika

```
select n.JMB, Prezime, Ime, Plata  
from nastavnik n  
inner join osoba o on o.JMB=n.JMB  
where Plata > (select avg(Plata) from nastavnik);
```

JMB	Prezime	Ime	Plata
1002952100005	Mitrović	Nikola	2400.00
1907951100012	Lazić	Zoran	2300.00
2804950103891	Ninković	Miloš	2400.00

# Selekcija i prikaz podataka

---

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena i iznosi plata nastavnika koji nemaju nastavno zvanje 'docent', a imaju platu veću od barem jednog nastavnika koji ima nastavno zvanje 'docent'

```
select n.JMB, Prezime, Ime, Plata
from nastavnik n
inner join osoba o on o.JMB=n.JMB
where NastavnoZvanje<>'docent' and Plata>any
(select Plata from nastavnik where NastavnoZvanje='docent');
```

JMB	Prezime	Ime	Plata
1002952100005	Mitrović	Nikola	2400.00
1006949100067	Soldat	Stanko	2000.00
1804964163303	Savić	Nenad	2000.00
1907951100012	Lazić	Zoran	2300.00
2804950103891	Ninković	Miloš	2400.00

# Selekcija i prikaz podataka

- ▶ **Korelisani poduputi** su poduputi koji referenciraju kolone tabela spoljašnjeg upita za selekciju redova rezultata, tako da je resultantna tabela poduputa zavisna od redova tabela spoljašnjeg upita
- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena i imena studenata, te identifikatori predmeta koji se izvode na studijskom programu čiji je identifikator 111, a koje su studenti položili ocenom većom od prosečne ocene ostvarene na datom studijskom programu

```
select o.JMB, Prezime, Ime, IdPredmeta, Ocena
  from osoba o inner join polaze p on p.JMB=o.JMB
 where IdSP=111 and
Ocena>(select avg(p1.Ocena)
      from polaze p1
     where p1.JMB=p.JMB
       and p1.IdSP=111
       and p1.Ocena>5)
 order by o.JMB, Ocena,
IdPredmeta;
```

JMB	Prezime	Ime	IdPredmeta	Ocena
1010988101124	Babić	Dejan	1111	7
1010988101124	Babić	Dejan	1112	7
1206986101234	Mirković	Marko	1112	9
1206986101234	Mirković	Marko	1115	9
1206986101234	Mirković	Marko	1111	10
1206986101234	Mirković	Marko	1114	10
1210987100018	Janković	Petar	1111	8
1210987100018	Janković	Petar	1112	9
1210987100018	Janković	Petar	2111	9

# Selekcija i prikaz podataka

---

- ▶ Tabele se najčešće specifikuju referenciranjem imena stalnih tabela baze podataka, ali to mogu da budu i tabele generisane specifikacijom SQL upita u okviru **from** klauzule, koji daje kao rezultat privremenu **izvedenu tabelu** (eng. *derived table*)
- ▶ Primer: SQL upit kojim se prikazuje najveća prosečna ocena koju je neki student ostvario na nekom studijskom programu

```
select max(PO) as MaxPO
from
(select round(avg(Ocena), 2) as PO
from polaze
where Ocena>5
group by JMB, IdSP) as prosecne_ocene;
```

MaxPO
9.67

# Selekcija i prikaz podataka

---

- ▶ **With** klauzula omogućava izdvajanje podupita iz osnovnog SQL upita njihovom specifikacijom i imenovanjem u okviru **with** klauzule (*Common Table Expressions – CTE*), te referenciranje tako definisanih podupita na više mesta u osnovnom SQL upitu
- ▶ MySQL podržava **with** klauzulu od verzije 8.0
- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena i iznosi plata nastavnika koji imaju platu veću od prosečne plate svih nastavnika

```
with prosecna_plata(Iznos) as
(select avg(Plata) from nastavnik)
select o.JMB, Prezime, Ime, Plata
from osoba o, nastavnik n, prosecna_plata p
where o.JMB=n.JMB and Plata>Iznos;
```

JMB	Prezime	Ime	Plata
1002952100005	Mitrović	Nikola	2400.00
1907951100012	Lazić	Zoran	2300.00
2804950103891	Ninković	Miloš	2400.00

# Selekcija i prikaz podataka

---

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena i imena studenata upisani u osmi semestar studijskog programa čiji je identifikator 111

```
with upis as
(select JMB from upisan_na where IdSP=111 and Semestar=8)
select o.JMB, Prezime, Ime
from osoba o
inner join upis u on u.JMB=o.JMB;
```

JMB	Prezime	Ime
1206986101234	Mirković	Marko
1210987100018	Janković	Petar

# Selekcija i prikaz podataka

---

- ▶ SQL podržava setovske operacije unije (**union**), preseka (**intersect**) i razlike (**except**)
- ▶ SQL setovske operacije podrazumevano eliminiju duplikate iz rezultata
- ▶ Ukoliko ne treba eliminisati duplikate iz rezultata, onda je potrebno uz **union/intersect/except** dodati klauzulu **all**
- ▶ U MySQL-u, operacije preseka i razlike podržane su od verzije 8.0.31, dok je operacija unije podržana još od verzije 4.0.0

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena, zvanja (nastavnička ili saradnička) i iznosi plata svih nastavnika i asistenata

```
(select JMB, Prezime, Ime, NastavnoZvanje as Zvanje, Plata
from nastavnik
inner join osoba using(JMB))
union all
(select JMB, Prezime, Ime, SaradnickoZvanje, Plata
from asistent
inner join osoba using(JMB));
```

JMB	Prezime	Ime	Zvanje	Plata
0702964105027	Gavrić	Mirjana	docent	1800.00
1002952100005	Mitrović	Nikola	redovni profesor	2400.00
1006949100067	Soldat	Stanko	vanredni profesor	2000.00
1804964163303	Savić	Nenad	vanredni profesor	2000.00
1907951100012	Lazić	Zoran	redovni profesor	2300.00
2108968196769	Petković	Milena	docent	1800.00
2804950103891	Ninković	Miloš	redovni profesor	2400.00
0308983126116	Mirković	Ana	viši asistent	1300.00
0512983100067	Popović	Slavko	asistent	1200.00
1206986101234	Mirković	Marko	asistent	1200.00
1312981163309	Filipović	Mirko	viši asistent	1350.00
2102979163201	Janković	Janko	viši asistent	1300.00

# Selekcija i prikaz podataka

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi osoba koje su i studenti i asistenti

```
(select JMB from student)
intersect
(select JMB from asistent);
```

JMB
1206986101234

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi osoba koje nisu nastavnici

```
(select JMB from osoba)
except
(select JMB from nastavnik);
```

JMB
0308983126116
0512983100067
0607989100027
1010988101124
1206986101234
1210987100018
1312981163309
1503990125037
2102979163201
2108988105034
2208988105039

# Modifikacije baze

---

- ▶ Za brisanje redova iz tabele koristi se **delete** iskaz, čiji je opšti oblik:  
`delete from naziv_tabele [where uslov];`
- ▶ Primer: SQL iskaz kojim se brišu brojevi telefona Elektrotehničkog fakulteta  
`delete from telefon_fakulteta  
where NazivFakulteta='Elektrotehnički fakultet';`
- ▶ Primer: SQL iskaz kojim se brišu svi brojevi telefona  
`delete from telefon_fakulteta;`
- ▶ Za umetanje se koristi **insert** iskaz, koji omogućava umetanje jednog ili više redova u baznu tabelu relacione baze podataka
- ▶ Opšti oblik **insert** iskaza:  
`insert into naziv_tabele[(naziv_kolone1, ... , naziv_kolonen)]  
values (vrednost1, ... , vrednostn);`
- ▶ Primer: SQL iskaz kojim dodaje novi predmet ('Računarska grafika')  
`insert into predmet values  
(1116, 'Računarska grafika', 6, 'Elektrotehnički fakultet');`

# Modifikacije baze

---

- ▶ MySQL omogućava specifikovanje klauzule **on duplicate key update** prilikom specifikovanja **insert** iskaza
- ▶ Ako se specifiкуje **on duplicate key update** klauzula, a pri tome postoji duplikat na jedinstvenom ili primarnom ključu za red koji se dodaje, tada se umesto umetanja novog reda izvršava ažuriranje postojećeg reda
- ▶ Primer: SQL iskaz kojim se dodaje novi predmet ('Računarska grafika'), a kojim se, u slučaju da postoji predmet sa istim identifikatorom, ažurira broj ECTS bodova

```
insert into predmet values  
  (1116, 'Računarska grafika', 5, 'Elektrotehnički fakultet') as pr  
  on duplicate key update ECTS=pr.ECTS;
```

- ▶ Vrednosti reda koji se dodaje moguće je referencirati u **update** izrazu upotrebom alijasa (npr. *pr.ECTS*)

# Modifikacije baze

---

- ▶ Za ažiriranje podataka u SQL-u se koristi **update** izraz, koji omogućava promene vrednosti na pojedinim kolonama nekih redova tabele
- ▶ Opšti oblik **update** izraza:

```
update naziv_tabele  
set naziv_kolone1=vrednost1  
[, naziv_kolone2=vrednost2]  
...  
[, naziv_kolonen=vrednostn]  
[where uslov];
```

- ▶ Primer: SQL izraz kojim se nastavniku čiji je matični broj 0702964105027 plata povećava za 20%

```
update nastavnik set Plata=Plata*1.2 where JMB='0702964105027';
```

- ▶ Primer: SQL izraz kojim se asistentu čiji je matični broj 1206986101234 saradničko zvanje menja u 'viši asistent', a plata povećava za 100

```
update asistent  
set SaradnickoZvanje='viši asistent', Plata=Plata+100  
where JMB='1206986101234';
```

```
source = datasource.getConnection();
connection.createStatement();
SQL = "SELECT * FROM ";
statement.executeUpdate();
ResultSet set = statement.executeQuery();
set.next());
```

# BAZE PODATAKA

SQL – pogledi, potvrde, trigeri, uskladištene procedure

# Pogledi

---

- ▶ **Pogled** (eng. *view*) je tabela koja se generiše iz drugih tabela na osnovu specifikovanog izraza
- ▶ Izraz na osnovu kojeg se generiše pogled/tabela je upit, koji referencira bazne tabele i/ili druge poglede
- ▶ SQL standard definiše poglede kao virtuelne tabele koje ne egzistiraju trajno u bazi podataka, već se generišu kada se pogled referencira
- ▶ U rečniku baze podataka se čuva ime pogleda, definicija upita pridruženog pogledu, informacije o kreatoru itd.
- ▶ Kada se pogled referencira, onda se, na osnovu imena pogleda u rečniku podataka, pronađi upit pridružen pogledu, koji se nakon toga izvršava i generiše se privremena tabela
- ▶ Upiti mogu biti vrlo kompleksni sa dugim vremenom izvršenja, pa često referenciranje takvih pogleda može značajno da degradira performanse sistema
- ▶ Alternativna opcija, koja nije standardizovana SQL standardom, ali je uglavnom podržana u najznačajnijim SQL DBMS, je materijalizovani pogled, koji predstavlja fizičku tabelu koja se kreira i koja trajno egzistira u bazi podataka
- ▶ Sadržaj fizičke tabele – materijalizovanog pogleda mora u svakom trenutku biti ekvivalentan rezultatu upita kojim je pogled definisan

# Pogledi

---

- ▶ Opšti oblik definicije pogleda:

```
create view naziv_pogleda[(naziv_kolone1, ... , naziv_kolonen)]  
as upit  
[with [cascaded | local] check option];
```

- ▶ Pogledi se brišu **drop view** iskazom čiji je opšti oblik:

```
drop view naziv_pogleda {restrict | cascade};
```

- ▶ Primer: SQL iskaz kojim se kreira pogled koji sadrži informacije o nastavnicima (matični brojevi, prezimena, imena, nastavna zvanja i plate)

```
create view nastavnik_info as  
select n.JMB, Prezime, Ime, NastavnoZvanje, Plata  
from nastavnik n inner join osoba o on o.JMB=n.JMB;
```

- ▶ Primer: SQL upit kojim se prikazuju matični brojevi, prezimena, imena i iznosi plata nastavnika koji imaju nastavno zvanje 'docent'

```
select JMB, Prezime, Ime, Plata  
from nastavnik_info  
where NastavnoZvanje='docent';
```

JMB	Prezime	Ime	Plata
0702964105027	Gavrić	Mirjana	1800.00
2108968196769	Petković	Milena	1800.00

# Pogledi

- ▶ Primer: SQL izraz kojim se kreira pogled koji sadrži informacije o studentima (matični brojevi, prezimena, imena, nazivi i ciklusi studijskih programa, te prosečne ocene studenata po studijskim programima)

```
create view student_info(JMB, Prezime, Ime,
    NazivSP, Ciklus, ProsekOcena) as
select o.JMB, Prezime, Ime, NazivSP, Ciklus, round(avg(Ocena), 2)
from osoba o inner join polaze p on p.JMB=o.JMB
inner join studijski_program sp on sp.IdSP=p.IdSP
where Ocena>5
group by o.JMB, p.IdSP
order by 6 desc;
```

- ▶ Primer: SQL upit kojim se prikazuju informacije o studentima

```
select * from student_info;
```

JMB	Prezime	Ime	NazivSP	Ciklus	ProsekOcena
1206986101234	Mirković	Marko	Računarstvo i informatika	2	9.67
2208988105039	Miljević	Branka	Fizika	1	9.33
1206986101234	Mirković	Marko	Računarstvo i informatika	1	8.83
1210987100018	Janković	Petar	Računarstvo i informatika	1	7.80
2108988105034	Đukić	Milijana	Elektronika i telekomunikacije	1	7.33
1010988101124	Babić	Dejan	Računarstvo i informatika	1	6.67

# Modifikacija baze kroz poglede

---

- ▶ S obzirom na to da su pogledi tabele, pogledi se mogu referencirati na svim mestima u iskazima na kojima je dozvoljeno referenciranje tabela, pa tako i u iskazima za modifikaciju baze podataka (brisanje, umetanje, ažuriranje)
- ▶ Pošto su pogledi virtualne, privremene tabele koje se generišu iz baznih tabela, modifikacija pogleda mora rezultovati modifikacijom baznih tabela
- ▶ Na primer, ako definišemo pogled:

```
create view predmeti6ECTS as
select *
from predmet
where ECTS=6;
```

onda bi umetanje sledećeg reda kroz prethodni pogled:

```
insert into predmeti6ECTS
values (1116, 'Računarska grafika', 6, 'Elektrotehnički fakultet');
```

rezultovalo umetanjem navedenog reda u tabelu *predmet*, a taj red bi bio sadržan i u rezultatu izvršenja sledećeg upita:

```
select * from predmeti6ECTS;
```

# Modifikacija baze kroz poglede

---

- ▶ Zbog prirode pogleda, modifikacija kroz poglede može generisati različite anomalije
- ▶ Na primer, umetanje sledećeg reda kroz prethodni pogled:

```
insert into predmeti6ECTS values (1117, 'Elektronsko poslovanje',  
5, 'Elektrotehnički fakultet');
```

rezultovalo bi umetanjem navedenog reda u tabelu *predmet*, ali taj red ne bi bio sadržan u rezultatu izvršenja sledećeg upita:

```
select * from predmeti6ECTS;
```

jer umetnuti red ne zadovoljava uslov selekcije definisan za pogled

- ▶ Ova anomalija se može izbeći korištenjem **with check option** klauzule, pa bi prethodna definicija pogleda bila:

```
create view predmeti6ECTS as  
select *  
from predmet  
where ECTS=6  
with check option;
```

# Modifikacija baze kroz poglede

---

- ▶ Specifikacijom **with check option** klauzule može se izbeći mogućnost umetanja kroz pogled reda koji se ne može videti kroz taj isti pogled
- ▶ Opcije **with check option** klauzule, **cascaded** i **local**, odnose se na opseg provere u pomenutim situacijama kada se pogled definiše korištenjem drugog pogleda
- ▶ Opcija **local** ograničava proveru samo na pogled koji se definiše
- ▶ Opcija **cascaded** je podrazumevana, a definiše proveru kako pogleda koji se definiše tako i potpogleda
- ▶ Komercijalni DBMS uglavnom ograničavaju mogućnost direktne modifikacije baze samo kroz poglede koji su definisani nad jednom tabelom pri čemu nije korišteno sračunavanje vrednosti kolona

# Potvrde

---

- ▶ **Potvrda** (eng. *assertion*) je opšti tip ograničenja kojim se specifiкуje uslov koji uvek mora da bude ispunjen na bazi podataka
- ▶ Kada se kreira potvrda, sistem testira bazu podataka za validnost s obzirom na potvrdu – ako je potvrda validna, onda svaka naredna modifikacija baze podataka biće dozvoljena samo ako ne narušava validnost potvrde
- ▶ S obzirom da testiranje potvrde može da zahteva značajno vreme, potvrde treba koristiti *pažljivo i sa oprezom*
- ▶ DBMS-ovi uglavnom ne podržavaju potvrde, ali se ekvivalentna funkcionalnost može implementirati korištenjem trigera

# Potvrde

---

- ▶ Opšti oblik definicije potvrde:

```
create assertion naziv_potvrde  
check <uslov>;
```

- ▶ Potvrde se brišu **drop assertion** iskazom čiji je opšti oblik:

```
drop assertion naziv_potvrde;
```

- ▶ Primer: SQL iskaz kojim se kreira potvrda koja obezbeđuje da student ne može da sluša predmet sa studijskog programa kojeg nije upisao

```
create assertion provera_upis check (  
    not exists (  
        select *  
        from upisao  
        where (JMB, IdSP) not in (  
            select JMB, IdSP from upisan_na  
        )));
```

- ▶ Napomena: Kako u SQL-u nije moguće specifikovati konstrukciju "za svako  $X, P(X)$ ", gde je  $P$  uslov, koristi se ekvivalentna konstrukcija "ne postoji  $X$  takav da nije  $P(X)$ "

# Trigeri

---

- ▶ **Triger** (eng. *trigger*) predstavlja akcije, specifikovane jednim ili više SQL izkaza, koje se izvršavaju na pojavu određenih događaja (eng. *event*)
- ▶ Pojava događaja se generiše izvršavanjem određene operacije nad datim objektom (bazna tabela ili pogled) baze
- ▶ Operacije nad objektima baze koje mogu da generišu događaj su operacije modifikacije objekta (brisanje, umetanje ili ažuriranje redova)
- ▶ Triggerske akcije se mogu specifikovati za izvršenje neposredno pre (**before**) ili nakon (**after**) operacije koja generiše trigerski događaj
- ▶ Pored trigerskih događaja, u okviru specifikacije trigera može se specifikovati i dodatni uslov za izvršavanje trigerskih akcija

# Trigeri

---

- ▶ Opšti oblik definicije trigera:

```
create trigger naziv_trigera {before | after}
{insert | delete | update[ of lista_kolona]}
on naziv_tabele
[referencing lista_alijasa]
[for each {row | statement}]
[when (uslov)]
<iskaz>
```

gde *lista\_alijasa* može biti:

- ▶ **old [row] [as] naziv\_reda<sub>1</sub>**
  - ▶ **new [row] [as] naziv\_reda<sub>2</sub>**
  - ▶ **old table [as] naziv\_tabele<sub>1</sub>**
  - ▶ **new table [as] naziv\_tabele<sub>2</sub>**
- ▶ Trigeri se brišu **drop trigger naziv\_trigera**

```
drop trigger naziv_trigera;
```

# Trigeri

---

- ▶ MySQL omogućava kreiranje trigera samo na nivou reda (**for each row**)
- ▶ Alijasi tranzicionih promenljivih su predefinisani (**new** i **old**) i ne mogu se menjati
- ▶ Primer (pretpostavka je da je tabela *osoba* izmenjena tako da je dodana kolona *Pol*): SQL iskaz kojim se kreira triger koji prilikom umetanja novog reda u tabelu *osoba* postavlja vrednost kolone *Pol* na osnovu desete cifre matičnog broja (0-4 – muški, 5-9 – ženski)

```
create trigger postavi_pol before insert
on osoba
for each row
set new.Pol = if(substr(new.JMB, 10, 1) <= '4', 'muški', 'ženski');
```

- ▶ Iskazom **set** postavlja se vrednost kolone *Pol*, pri čemu se za proveru desete cifre matičnog broja koristi funkcija **if** čiji je prototip:

```
if(izraz1, izraz2, izraz3)
```

- ▶ Ako je *izraz<sub>1</sub>* jednak logičkoj vrednosti *istina*, funkcija vraća *izraz<sub>2</sub>* – inače vraća *izraz<sub>3</sub>*

# Trigeri

---

- ▶ Primer (prepostavka je da je tabela *ispit* izmenjena tako da su dodane kolone *Izaslo* i *Polozilo*): SQL iskaz kojim se kreira triger koji nakon dodavanja novog reda u tabelu *polaze* ažurira odgovarajuće vrednosti kolona *Izaslo* i *Polozilo* (tabele *ispit*)

```
create trigger novo_polaganje after insert
on polaze
for each row
update ispit
set IZASLO=IZASLO+1,
    POLOZILO=POLOZILO+(new.Ocena>5)
where DatumIspita=new.DatumIspita and
      IdPredmeta=new.IdPredmeta and IdSP=new.IdSP;
```

# Uskladištene procedure

---

- ▶ **Uskladištene procedure** (eng. *stored procedures*) su programske procedure koje se pohranjuju u okviru baze podataka, i koje se mogu pozivati iz različitih klijentskih programa i aplikacija
- ▶ Kao i standardne procedure, uskladištene procedure izvršavaju projektovani zadatak/funkciju generišući željeni rezultat na bazi vrednosti argumenata specifikovanih pri pozivu procedure (ako je procedura sa definisanim parametrima)
- ▶ Za pogodnu implementaciju željenih funkcija i poslovne logike, implementacioni jezik uskladištenih procedura mora da sadrži programske konstrukcije za uslovne iskaze, petlje, složene blok iskaze itd.
- ▶ Nedostatak uskladištenih procedura je što većina DBMS isporučioca ima specifične proceduralne jezike za programiranje uskladištenih procedura
- ▶ Primeri procedura koji slede definisani su u MySQL-u

# Uskladištene procedure

---

- ▶ Osnovni oblik definicije procedure u MySQL-u:

```
create procedure naziv_procedure (
    [[in | out | inout]] parametar1 tip_podatka[, ...])
<telo_procedure>
```

- ▶ Parametri procedure mogu biti ulazni (**in**), izlazni (**out**) i ulazno-izlazni (**inout**)
- ▶ Podrazumevano, parametri su ulazni
- ▶ Telo procedure može biti jedan iskaz (npr. **select**) ili složeni (blokovski) iskaz koji se sastoji od više iskaza
- ▶ Blokovski iskazi se pišu korištenjem **begin** i **end**
- ▶ Ukoliko je telo procedure složeni iskaz (**begin/end** blok), pojedini iskazi u složenom iskazu su razdvojeni delimiterom ;
- ▶ U tom slučaju je potrebno privremeno promeniti delimiter (npr. \$\$)
- ▶ Isto važi i za definiciju trigera u MySQL-u

# Uskladištene procedure

---

- ▶ Varijable
  - ▶ Lokalne varijable definišu se **declare** izrazom (**declare** izraz je dozvoljeno koristiti samo na početku **begin/end** bloka) čiji je opšti oblik:

```
declare varijabla1 [, varijabla2, ...] tip_podatka  
[default vrednost];
```
  - ▶ Osim lokalnih varijabli postoje i korisnički definisane varijable, koje su specifične za konekciju
  - ▶ Varijabla koju je definisao jedan korisnik nije vidljiva od strane drugih korisnika, a automatski se oslobađa kada korisnik zatvori konekciju
  - ▶ Korisnički definisane varijable se pišu u obliku `@varijabla`
- ▶ MySQL podržava sledeće izraze za kontrolu toka: **case, if, loop, while, repeat, iterate i leave**

# Uskladištene procedure

---

- ▶ Opšti oblici case iskaza:

```
case vrednost
    when vrednost1 then <iskaz>
    [when vrednost2 then <iskaz>]
    ...
    [when vrednostn then <iskaz>]
    [else <iskaz>]
end case;

case
    when uslov1 then <iskaz>
    [when uslov2 then <iskaz>]
    ...
    [when uslovn then <iskaz>]
    [else <iskaz>]
end case;
```

# Uskladištene procedure

---

- ▶ Opšti oblik **if** iskaza:

```
if uslov1 then <iskaz>
[elseif uslov2 then <iskaz>]
...
[elseif uslovn then <iskaz>]
[else <iskaz>]
end if;
```

- ▶ Opšti oblik **loop** iskaza:

```
[Labela:] loop
<iskaz>
end loop [Labela];
```

- ▶ Opšti oblik **while** iskaza:

```
[Labela:] while uslov do
<iskaz>
end while [Labela];
```

# Uskladištene procedure

---

- ▶ Opšti oblik **repeat** iskaza:

```
[Labela:] repeat  
  <iskaz>  
  until uslov  
  end repeat [Labela];
```

- ▶ Opšti oblik **iterate** iskaza:

```
iterate Labela;
```

- ▶ Iskaz **iterate** omogućava ponovno startovanje petlje, pa se zato može naći u okviru **loop**, **while** ili **repeat** iskaza

- ▶ Opšti oblik **leave** iskaza:

```
leave Labela;
```

- ▶ Iskaz **leave** omogućava prekid izvršavanja petlje, a u zavisnosti od specifikovane labele moguće je i prekidanje izvršavanja procedure

# Usklađene procedure

---

- ▶ Kursori (eng. *cursors*)

- ▶ Opšti oblik iskaza za deklaraciju kursora:

- ```
declare naziv_kursora cursor for <upit>;
```

- ▶ Opšti oblik iskaza za otvaranje (prethodno deklarisano) kursora:

- ```
open naziv_kursora;
```

- ▶ Opšti oblik iskaza za dohvatanje sledećeg reda upita koji je pridružen datom kursoru:

- ```
fetch naziv_kursora into varijabla1 [, varijabla2, ...];
```

- ▶ Ako red postoji, vrednosti kolona se smeštaju u navedene varijable
    - ▶ Ako red ne postoji, pojavljuje se stanje *No Data* (kod stanja je '02000'), koje se može detektovati postavljanjem rukovaoca za to stanje (ili za stanje **NOT FOUND**)

- ▶ Opšti oblik iskaza za zatvaranje kursora :

- ```
close naziv_kursora;
```

# Uskladištene procedure

---

- ▶ Kontrola stanja
  - ▶ Stanja se mogu imenovati iskazom čiji je opšti oblik:  
`declare naziv_stanja condition for vrednost_stanja;`
  - ▶ *vrednost\_stanja* može biti:
    - ▶ *kod\_greške*
    - ▶ `sqlstate [value] kod_stanja`
  - ▶ MySQL kod greške je broj, a kod stanja je string literal dužine 5 znakova

# Usklađene procedure

---

- ▶ Kontrola stanja
  - ▶ Rukovaoci za stanja se definišu izrazom čiji je opšti oblik:

```
declare akcija handler for vrednost_stanja1
[, vrednost_stanja2, ...]
<izraz>
```
  - ▶ akcija može biti:
    - ▶ continue – nastavlja se izvršavanje procedure
    - ▶ exit – prekida se izvršavanje begin/end bloka u kojem je rukovalac deklarisan
    - ▶ undo – nije implementirano
  - ▶ vrednost\_stanja može biti:
    - ▶ kod\_greške
    - ▶ sqlstate [value] kod\_stanja
    - ▶ naziv\_stanja
    - ▶ sqlwarning
    - ▶ not found
    - ▶ sqlexception

# Uskladištene procedure

---

## ▶ Kontrola stanja

- ▶ MySQL kod greške je broj, a kod stanja je string literal dužine 5 znakova
- ▶ **sqlwarning** je skraćenica za klasu kodova stanja koji počinju sa '01'
- ▶ **not found** je skraćenica za klasu kodova stanja koji počinju sa '02'. Ovo je relevantno za kursore i koristi se za slučajeve kada kurzor dođe do kraja. Ukoliko kurzor dođe do kraja (tj. nema više redova), pojavljuje se stanje *No Data* (kod stanja je '02000'), koje se može detektovati postavljanjem rukovaoca za to stanje (ili za stanje **not found**).
- ▶ **sqlexception** je skraćenica za klasu kodova stanja koji ne počinju sa '00', '01' ili '02'
- ▶ Ukoliko se pojavi neko stanje, a nije definisan rukovalac za dato stanje, akcija zavisi od klase stanja:
  - ▶ Za stanja klase **sqlexception**, prekida se izvršavanje procedure kod iskaza u kojem je podignuto stanje, kao da je definisan **exit** rukovalac
  - ▶ Za stanja klase **sqlwarning** i **not found**, izvršavanje procedure se nastavlja, kao da je definisan **continue** rukovalac

# Uskladištene procedure

---

- ▶ Uskladištene procedure se pozivaju korištenjem **call** iskaza čiji je opšti oblik:

**call naziv\_procedure [([parametar<sub>1</sub> [, parametar<sub>2</sub>, ...]])];**

- ▶ Opšti oblik iskaza kojim se briše procedura:

**drop procedure naziv\_procedure;**

# Uskladištene procedure

- ▶ Primer: SQL izraz kojim se kreira procedura koja računa prosečnu ocenu studenta na datom studijskom programu (ulazni parametri procedura su matični broj studenta i identifikator studijskog programa, a izlazni parametar je prosečna ocena)

```
delimiter $$  
create procedure prosek_ocena(in pJMB char(13), in pIdSP int,  
      out pProsekOcena double)  
begin  
    select round(avg(Ocena), 2) into pProsekOcena  
    from polaze  
    where ocena>5 and JMB=pJMB and IdSP=pIdSP;  
end$$  
delimiter ;
```

- ▶ Primer: SQL izraz kojim se poziva procedura (iz prethodnog primera) i SQL upit kojim se prikazuje rezultat

```
call prosek_ocena('1206986101234', 121, @prosek);  
select @prosek; -- U MySQL-u klauzula nije obavezna
```

@prosek  
9.67

# Uskladištene procedure

---

- ▶ Primer: SQL izraz kojim se kreira procedura kojom se evidentira novo polaganje ispita (student ne može polagati ispit iz predmeta kojeg nije slušao ili predmeta kojeg je već položio)

```
delimiter $$  
create procedure evidentiraj_rezultat_ispita(in pJMB char(13),  
    in pDatumIspita date, in pIdPredmeta int, in pIdSP int,  
    in pOcena tinyint)  
begin  
    declare vSlusao, vPoložio bool default false;  
  
    select count(*)>0 into vSlusao from upisao  
    where JMB=pJMB and IdPredmeta=pIdPredmeta and IdSP=pIdSP;  
  
    select count(*)>0 into vPoložio from polaze  
    where Ocena>5 and JMB=pJMB and IdPredmeta=pIdPredmeta and IdSP=pIdSP;  
  
    if vSlusao and not vPoložio and pOcena between 5 and 10 then  
        insert into polaze  
            values (pJMB, pDatumIspita, pIdPredmeta, pIdSP, pOcena);  
    end if;  
end$$  
delimiter ;
```

# Uskladištene procedure

---

- ▶ Primer: SQL iskaz kojim se kreira procedura kojom se prikazuju položeni predmeti studenta po upisanim studijskim programima

```
delimiter $$  
create procedure student_studij_detaljno(in pJMB char(13))  
begin  
    declare vIdSP int; declare vKraj bool default false;  
    declare cUpisaniSP cursor for select IdSP from upisan_na where JMB=pJMB;  
    declare continue handler for not found set vKraj=true;  
    open cUpisaniSP;  
  
    petlja: loop  
        fetch cUpisaniSP into vIdSP;  
        if vKraj then leave petlja; end if;  
        select p.IdSP, p.IdPredmeta, NazivPredmeta, Semestar, Ocena, p.DatumIspita  
        from polaze p  
        inner join p_na_sp psp on psp.IdPredmeta=p.IdPredmeta and psp.IdSP=p.IdSP  
        inner join predmet pr on pr.IdPredmeta=psp.IdPredmeta  
        where p.Ocena>5 and p.JMB=pJMB and p.IdSP=vIdSP  
        order by psp.Semestar;  
    end loop petlja;  
    close cUpisaniSP;  
end$$  
delimiter ;
```

# Usklađene procedure

- ▶ Primer: SQL izraz kojim se, za studenta čiji je matični broj 1206986101234, prikazuju položeni predmeti (po studijskim programima) pozivom procedure iz prethodnog primera

```
call student_studij_detaljno('1206986101234');
```

<b>IdSP</b>	<b>IdPredmeta</b>	<b>NazivPredmeta</b>	<b>ECTS</b>	<b>Semestar</b>	<b>Ocena</b>	<b>DatumIspita</b>
111	2111	Linearna algebra	6	1	7	2006-09-11
111	1111	Programski jezici 1	7	2	10	2006-07-03
111	1112	Programski jezici 2	6	3	9	2007-01-30
111	1113	Softversko inženjerstvo	6	4	8	2007-09-05
111	1114	Baze podataka	8	6	10	2008-07-10
111	1115	Informacioni sistemi	6	7	9	2009-02-06

<b>IdSP</b>	<b>IdPredmeta</b>	<b>NazivPredmeta</b>	<b>ECTS</b>	<b>Semestar</b>	<b>Ocena</b>	<b>DatumIspita</b>
121	1211	Internet tehnologije	6	1	9	2010-09-22
121	1212	Sigurnost računarskih sistema	8	1	10	2010-07-22
121	1213	Baze podataka (viši nivo)	8	1	10	2010-02-19

# Uskladištene procedure

---

- ▶ MySQL omogućava "vraćanje" greške ili upozorenja korištenjem **signal** iskaza čiji je opšti oblik:

```
signal vrednost_stanja  
[set informacija1=vrednost1 [, set informacija2=vrednost2, ...]];
```

- ▶ *informacija* može biti: **message\_text**, **mysql\_errno** itd.
- ▶ Kod stanja '45000' predstavlja generičko stanje koje označava "korisnički definisani izuzetak"
- ▶ Primer: SQL iskaz kojim se kreira procedura koja deli dva cela broja

```
delimiter $$  
create procedure podeli(in pA int, in pB int, out pR double)  
begin  
    if pB=0 then  
        signal sqlstate '45000' set message_text='Deljenje s nulom.';  
    end if;  
    set pR=pA/pB;  
end$$  
delimiter ;
```

```
source = datasource.getConnection();
connection.createStatement();
SQL = "SELECT * FROM ";
statement.executeUpdate();
ResultSet set = statement.executeQuery();
set.next());
```

# BAZE PODATAKA

## SQL – sigurnost i autorizacija

# Sigurnost

---

- ▶ Zaštita od malicioznih pokušaja krađe, brisanja i modifikovanja podataka te zaštita od neželjenog uništenja ili oštećenja podataka usled havarija, sistemskih otkaza, prirodnih katastrofa itd.
- ▶ Nivoi zaštite:
  - ▶ Nivo DBMS (**mehanizmi autentikacije i autorizacije** omogućavaju određenim korisnicima pristup samo potrebnim podacima)
  - ▶ Nivo operativnog sistema
  - ▶ Mrežni nivo
  - ▶ Fizički nivo
  - ▶ Nivo korisnika

# Autorizacija

---

- ▶ U sistemu sa bazom podataka različiti regularni korisnici (ljudi i aplikativni programi), sa stanovišta svoje pozicije, uloge i zadataka u poslovnom sistemu, najčešće imaju potrebu samo za nekim aktivnostima na određenim delovima baze podataka, pa je potrebno ograničiti mogućnost pristupa podacima i aktivnosti korisnika u skladu sa njihovom ulogom i zadacima
- ▶ Za realizaciju prethodnih ciljeva koristi se koncept dodele prava/privilegija, odnosno autorizacije regularnih korisnika za izvršenje određenih aktivnosti na delovima baze podataka
- ▶ Korisnik može dobiti autorizaciju za sledeće aktivnosti:
  - ▶ Aktivnosti koje se odnose na rad sa podacima u bazi:
    - ▶ **autorizacija za čitanje podataka** iz baze
    - ▶ **autorizacija za umetanje podataka** u bazu
    - ▶ **autorizacija za ažuriranje podataka**
    - ▶ **autorizacija za brisanje podataka** iz baze
  - ▶ Aktivnosti koje se odnose na definisanje i menjanje strukture baze:
    - ▶ **autorizacija za kreiranje objekata** baze podataka
    - ▶ **autorizacija za promenu objekata** baze podataka
    - ▶ **autorizacija za brisanje objekata** baze podataka

# Dodela i povlačenje privilegija

---

- ▶ U MySQL-u, nalozi se kreiraju **create user** iskazom čiji je opšti oblik:  
`create user korisnik [identified by [password] 'lozinka'];`
- ▶ Ukoliko se korisniku želi omogućiti prijavljivanje bez lozinke, potrebno je izostaviti **identified by** klauzulu
- ▶ Ukoliko se specifikuje opcija **password**, tada se navodi *hash* vrednost lozinke
- ▶ Opšti oblik iskaza **grant** namenjenog za dodelu prava korisnicima nad objektima šeme baze podataka:  
`grant <lista_privilegija> on <naziv_objekta>  
    to <lista_korisnika/lista_uloga>;`
- ▶ Opšti oblik iskaza **revoke** namenjenog za povlačenje prava korisnicima nad objektima šeme baze podataka:  
`revoke <lista_privilegija> on <naziv_objekta>  
    from <lista_korisnika/lista_uloga>;`

# Primeri

---

## ▶ Primeri:

```
-- kreiranje korisnickog naloga
create user 'student'@'localhost' identified by 'student';

-- dodela privilegija na nivou seme baze podataka
grant all privileges on test.* to 'student'@'localhost';

-- dodela privilegija na nivou tabele
grant select on uniis.osoba to 'student'@'localhost';
grant select, insert, update on uniis.fakultet
to 'student'@'localhost';
grant delete on uniis.telefon_fakulteta to 'student'@'localhost';

-- dodela privilegija na nivou kolone
grant select (JMB, NastavnoZvanje, JMBSefaKatedre) on
uniis.nastavnik to 'student'@'localhost';

-- dodela privilegije za izvršavanje procedure
grant execute on procedure uniis.prosek_ocena_sp
to 'student'@'localhost';

-- aktiviranje privilegija
flush privileges;
```

# Uloge

---

- ▶ MySQL podržava definisanje uloga (**role**) od verzije 8.0
- ▶ Primeri:

```
-- kreiranje uloga
```

```
create role 'uniis_dev'@'localhost',
'uniis_read'@'localhost', 'uniis_write'@'localhost';
```

```
-- dodela privilegija
```

```
grant all privileges on uniis.* to 'uniis_dev'@'localhost';
```

```
grant select on uniis.* to 'uniis_read'@'localhost';
```

```
grant insert, update, delete on uniis.* to
```

```
'uniis_write'@'localhost';
```

```
-- kreiranje korisnickih nalog
```

```
create user 'dev_janko'@'localhost' identified by 'janko';
```

```
create user 'referent_petra'@'localhost' identified by 'petra';
```

# Uloge

---

- ▶ Primeri (nastavak):

```
-- umesto individualne dodelje privilegija,  
-- korisnicima se mogu dodeliti uloge  
grant 'uniis_dev'@'localhost' to 'dev_janko'@'localhost';  
grant 'uniis_read'@'localhost', 'uniis_write'@'localhost' to  
'referent_petra'@'localhost';  
  
-- iskaz set default role se koristi za podešavanje aktivnih uloga  
-- prilikom logovanja  
set default role all to  
'dev_janko'@'localhost', 'referent_petra'@'localhost';
```

# Autorizacija i pogledi

---

- ▶ Korisnicima se mogu dodeliti privilegije nad pogledima, pri čemu tim korisnicima ne moraju biti dodeljene privilegije nad baznim tabelama na osnovu kojih je definisan pogled
- ▶ S obzirom na to da se pogledima mogu sakriti podaci, kombinovanjem sigurnosti na nivou tabela i sigurnosti na nivou pogleda korisnicima se može ograničiti pristup samo na podatke koji su im neophodni
- ▶ Na primer, pomoću odgovarajućeg pogleda, radnicima u računovodstvu na Elektrotehničkom fakultetu može da se ograniči pristup samo na podatke o nastavnicima sa Elektrotehničkog fakulteta

# Pravo prosleđivanja privilegija

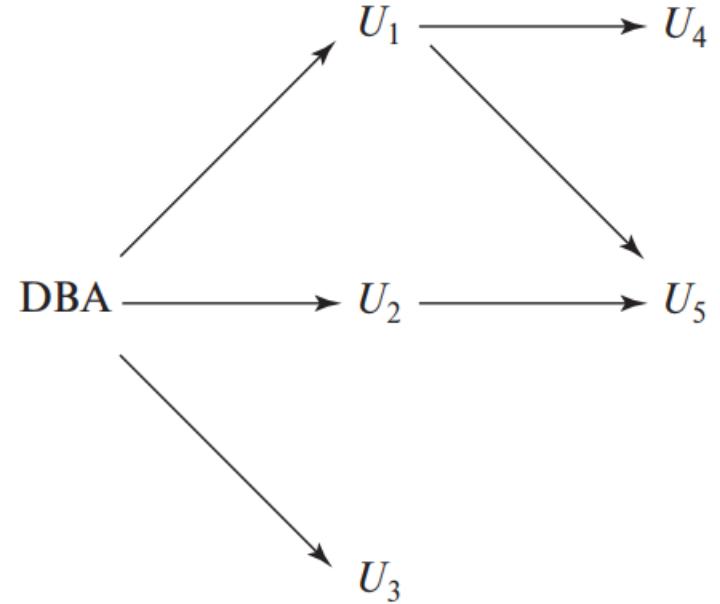
---

- ▶ Korisniku kojem se dodeljuje neka privilegija može da se dodeli i pravo prosleđivanja te privilegije drugim korisnicima
- ▶ To se postiže specifikovanje **with grant option** klauzule na kraju **grant** iskaza
- ▶ Primer:

```
grant select on uniis.osoba  
to 'student'@'localhost'  
with grant option;
```

# Graf autorizacije

- ▶ Prosleđivanje privilegije od jednog korisnika ka drugom može da se predstavi **grafom autorizacije**:
  - ▶ Čvorovi grafa autorizacije su korisnici
  - ▶ Koreni čvor grafa je administrator (DBA)
  - ▶ Grana  $U_i \rightarrow U_j$  postoji u grafu ako je korisnik  $U_i$  dodelio datu privilegiju korisniku  $U_j$
  - ▶ Sve grane grafa moraju biti deo putanje koja počinje od korenog čvora, tj. korisnik ima neku privilegiju *ako i samo ako* postoji putanja u grafu autorizacije od korenog čvora (administratora) do čvora koji predstavlja tog korisnika



# Graf autorizacije (povlačenje privilegija)

- ▶ Ako DBA povuče privilegiju od korisnika  $U_1$ :
  - ▶ Privilegija mora biti povučena i od korisnika  $U_4$ , jer korisnik  $U_1$  više nema privilegiju
  - ▶ Privilegija ne sme biti povučena od korisnika  $U_5$ , jer korisnik  $U_5$  ima drugu autorizacionu putanju od DBA preko  $U_2$
- ▶ Mora se sprečiti pojava ciklusa bez putanje od DBA:
  - ▶ DBA dodeljuje privilegiju korisniku  $U_7$
  - ▶ Korisnik  $U_7$  dodeljuje privilegiju korisniku  $U_8$
  - ▶ Korisnik  $U_8$  dodeljuje privilegiju korisniku  $U_7$
  - ▶ Ako DBA povuče privilegiju od  $U_7$ , tada se mora povući i privilegija od  $U_7$  ka  $U_8$ , kao i od  $U_8$  ka  $U_7$ , jer više ne postoji putanja od DBA ka  $U_7$  niti ka  $U_8$

