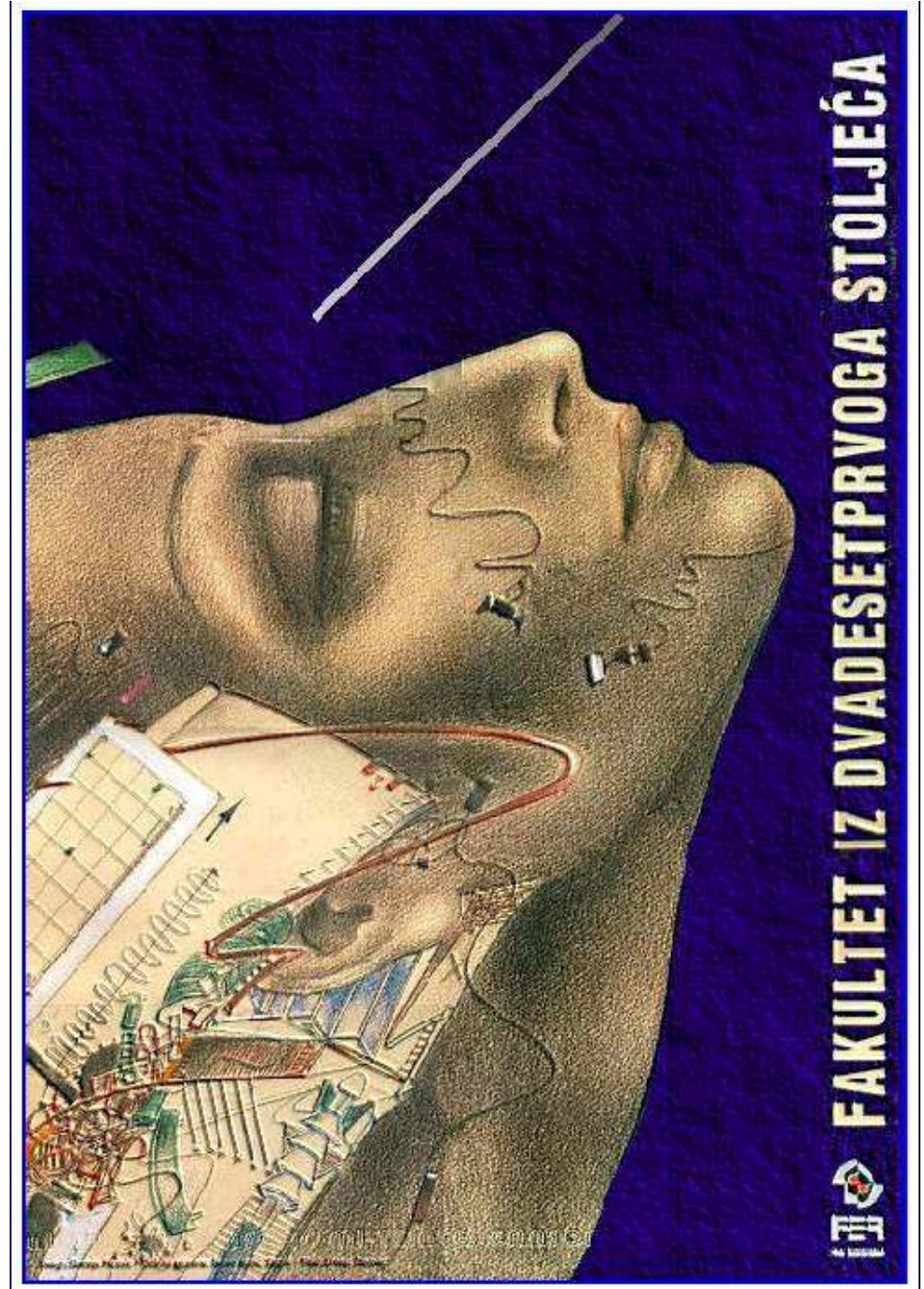


# Napredni modeli i baze podataka

Predavanja  
rujan 2008.

## UML i oblikovanje baze podataka



# Oblikovanje podataka (1)

---

**Data Modeling** - Performing analysis on the business processes and data to discover attributes of, and relationships between, data elements. An Entity Relationship Diagram (ERD) is the implementation of a data model. Another way to think of it is to discover the business rules of the data elements.



## Data modeling

In computer science, data modeling is the process of structuring and organizing data, typically using a database management system.

### Data modeling

The practice of analyzing an enterprise's data and identifying the relationships among the data.

## Data modeling

Data modeling is the task of formalizing the data requirements of the business process as a conceptual model.

## DATA MODELING

Data modeling is the process of defining what data you want to capture in your database and the relationships between data.

## Oblikovanje podataka (2)

---

- Oblikovanje podataka je izrada konceptualnog **modela podataka** na osnovu podataka o dijelu realnog svijeta koji se želi modelirati.



# Model podataka

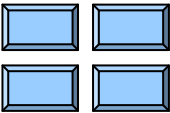

---

- Model podataka predstavlja objekte realnog svijeta i odnose među njima
- Svrha modela podataka:
  - sredstvo komunikacije
  - nacrt za kreiranje baze podataka

*Obje svrhe nije lako zadovoljiti jednim modelom → uvode se modeli na različitim razinama informacija.*

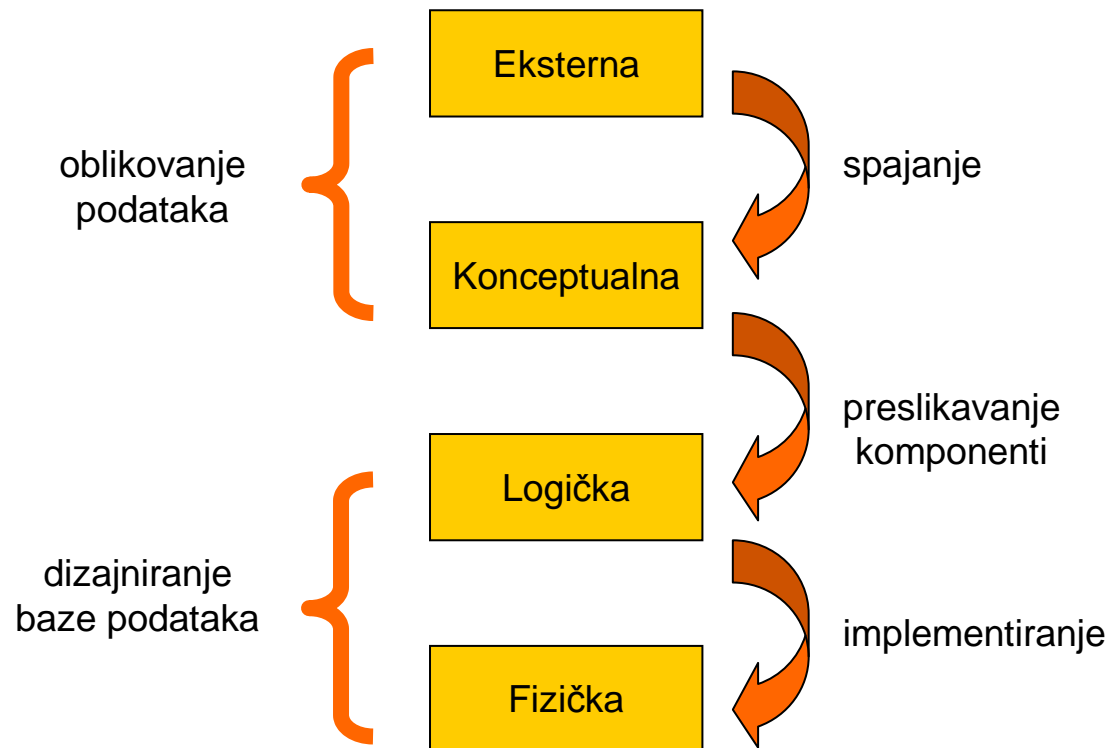
# Modeli podataka

---

razina informacija	modeli podataka	
Eksterna		Pogled na sustav od strane različitih grupa korisnika. Koristi se kao sredstvo komunikacije sa svakom od tih grupa.
Konceptualna		Zajednički pogled na sustav od strane svih korisnika. Koristi se kao sredstvo komunikacije sa cijelom grupom korisnika.
Logička		Pogled na podatke u skladu sa odabranom metodologijom za dizajniranje baze podataka.
Fizička		Pogled na fizički način pohrane podataka.

# Oblikovanje podataka i dizajniranje baze podataka

---



# Proces oblikovanja podataka – prikupljanje zahtjeva

---

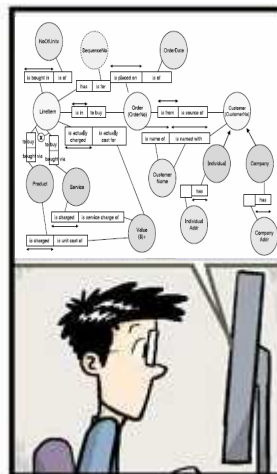
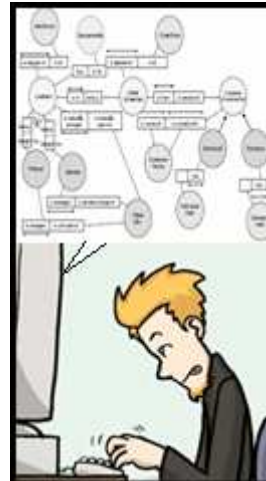
- Identifikacija poslovnih objekata
- Identifikacija direktnih veza
- Identifikacija atributa poslovnih objekata
- Identifikacija atributa koji jednoznačno označavaju poslovne objekte
- Identifikacija poslovnih pravila
- **Provjera valjanosti modela**
  - zbog ovog koraka važno je imati model koji razumiju korisnici, jer ga oni moraju i odobriti



# Proces oblikovanja podataka – izrada konceptualnog modela



prikupljeni podaci

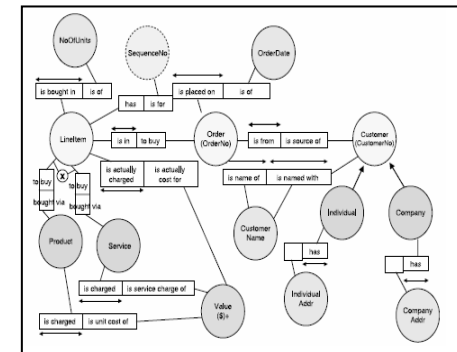


eksterni model



prikupljeni podaci

spajanje



konceptualan  
model

prikupljanje zahtjeva – obično  
intervjuiranjem

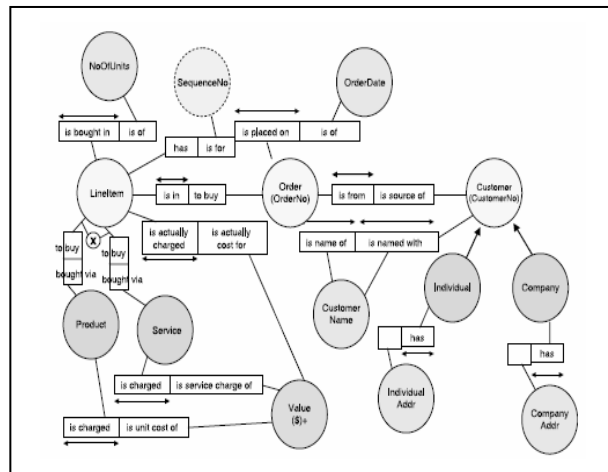


# Konceptualan model

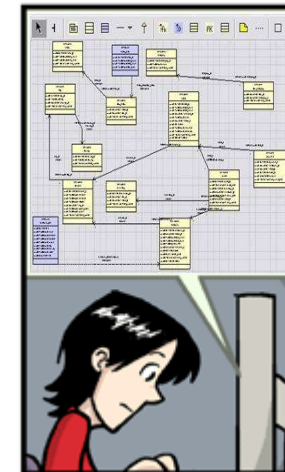
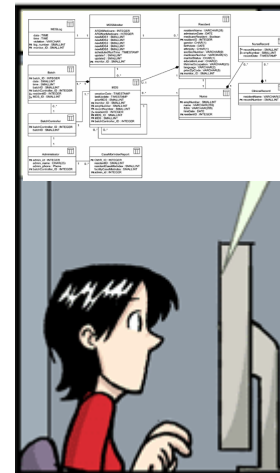
---

- Model na najvišoj razini
- Opći model
- Neovisan o samoj implementaciji (vrste sustava za upravljanje bazama podataka, programskog jezika, operacijskog sustava...)
- Prvenstveno se koristi kao sredstvo komunikacije o zahtjevima sustava

# Proces oblikovanja podataka – izrada logičkog modela



**konceptualan  
model**



**logički modeli**

# Logički model baze podataka

---

- U skladu sa odabranom metodologijom za izradu baze podataka
- Ovisi o vrsti DBMS-a
  - hijerarhijski
  - mrežni
  - relacijski
  - objektni

# Fizički model baze podataka

---

- Fizički model se kreira iz logičkog modela
- Odabrani SUBP direktno utječe na ovaj model
- Fokus prelazi na poboljšanje performansi
- Podrazumijeva samu implementaciju baze podataka
  - alociranje prostora
  - pohrana podataka i indeksa
  - načini kompresije podataka
  - konfiguracija hardvera

# Proces izrade modela podataka (1)

---

- Važnost konceptualnog modela
  - osigurava da su u bazi podataka pohranjeni svi podaci iz prikupljenih zahtjeva
  - kroz komunikaciju sa korisnicima osigurava se točnost podataka
  
- Ograničenja implementacijskih modela (logički/fizički)
  - ovi modeli prikazuju podatke karakteristično za vrstu SUBP-a (u slučaju relacijske baze to su dvodimenzionalne tablice), te kao takvi predstavljaju pogled sa stanovišta pohrane i upravljanja podacima – pokušaj stvaranja ovog modela kao polaznog može rezultirati nepotpunim i netočnim modelom podataka

# Proces izrade modela podataka (2)

---

- Potreba za općim modelom
  - opći model koristi komponente neovisne o vrsti SUBP-a koji će se koristiti pri implementaciji, te se tako osobe zadužene za oblikovanje mogu potpuno koncentrirati na obuhvaćanje pravog značenja informacija iz domene koja se oblikuje
- Jednostavan i jednoznačan proces transformacije modela
  - izgradnja logičkog modela iz konceptualnog modela slijedi propisane korake (npr. ER u relacijski model)
- Preslikavanje komponenti
  - model se sastoji od skupa različitih komponenti
  - dovoljno je poznavati proces transformacije svake od tih komponenti

# Kada u potpunosti slijediti proces izrade modela?

---

- Kada se pretpostavlja da će model podataka rezultirati velikim brojem komponenti
- Kada se radi o kompliciranom poslovnom procesu
- Veliki projekti na kojima je potrebno uključiti više osoba u proces oblikovanja podataka



# Tehnike oblikovanja podataka

---

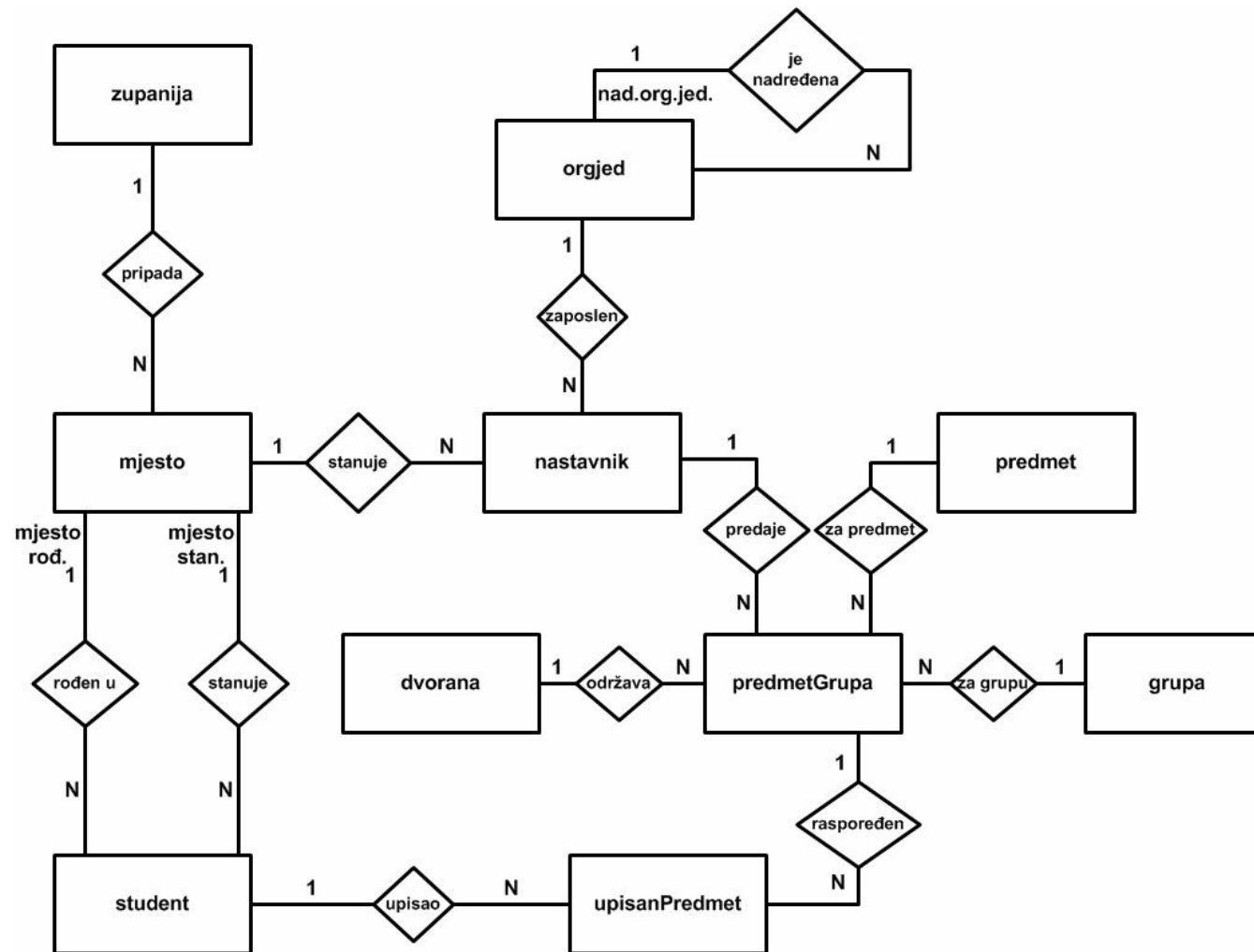
- E-R (entitet-veze)
- ORM (Object role modeling) – činjenično orijentirano oblikovanje
- UML – objektno orijentirano oblikovanje
- XML (Extensible Markup Language) – definira strukturu podataka

# ER model

---

- Model entiteti-veze
- Omogućuje eksplicitni prikaz veza koje u sebi sadrže važne semantičke informacije
- Postoji više tehnika ER modeliranja
  - Peter Chen-ov model
  - IE – Information Engineering
  - IDEF1x – Integration Definition for Information Modeling
  - Richard Barker-ov model

# ER model - primjer



# ORM model

---

- Često se naziva i NIAM (Natural language Information Analysis Method).
- Pojednostavljuje proces oblikovanja korištenjem prirodnog jezika
- U fokusu modeliranja je veza; definiraju se činjenice koje daju značaj vezi između objekata

# ORM model - primjer

**Fact Editor -- Create a New Fact**

Fact | Object | Examples | Constraints | Advanced

Object Name: Kupac Relationship kupuje Object name: Proizvod

Inverse relationship: Proizvod je kupovan od strane Kupac

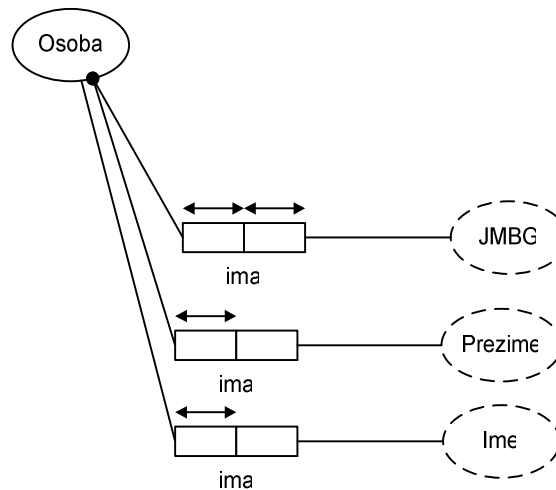
Input style

☐ Freeform: Capitalized

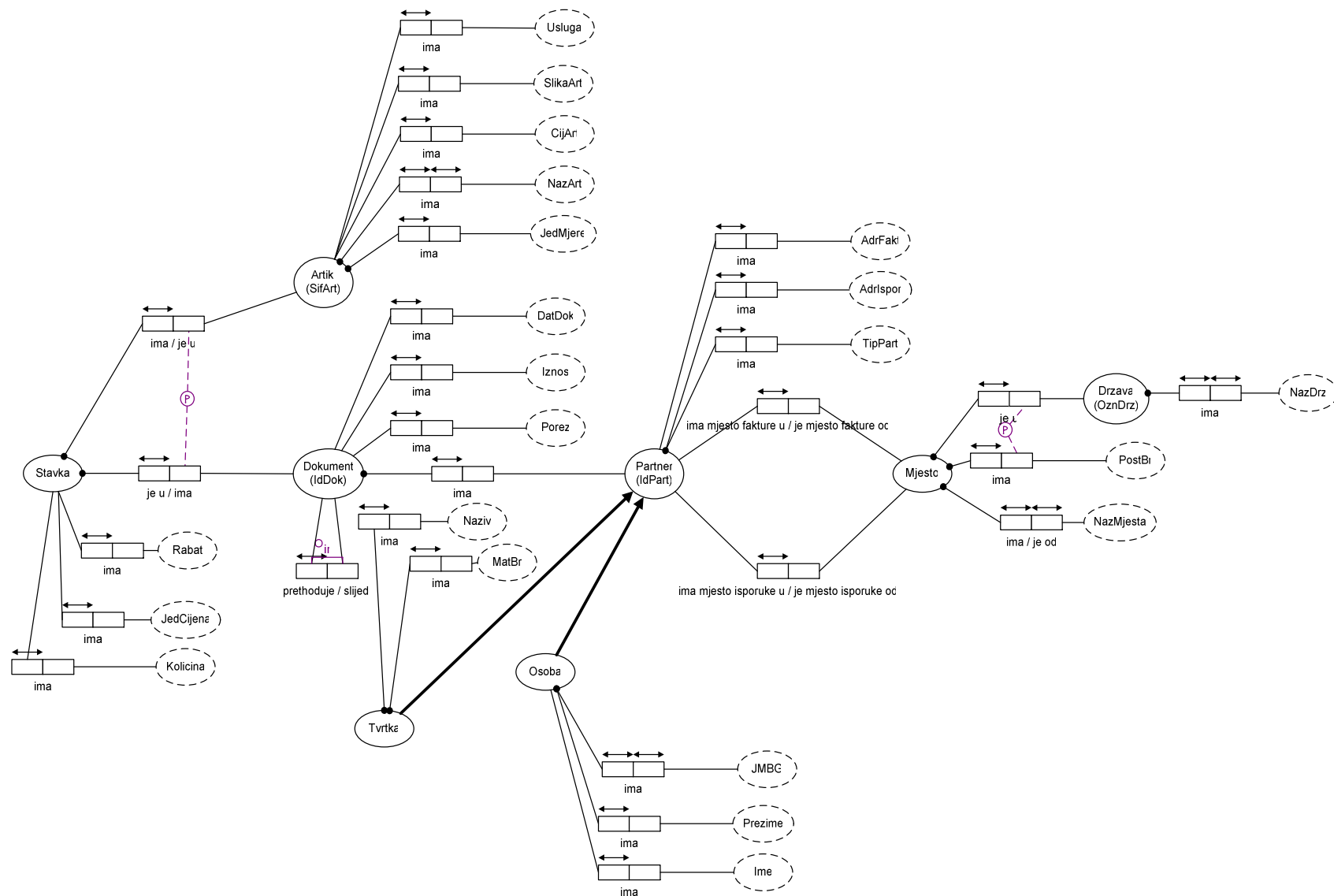
☒ Guided: Binary

[Kupac] kupuje [Proizvod] / [Proizvod] je kupovan od strane [Kupac]

Clear Apply OK Cancel



# ORM model - primjer



# XML model

---

- Nije prava tehnika oblikovanja, ali je način prezentacije podataka
- Predstavlja strukture podataka u tekstualnom obliku
- DTD, DSD ili XSD datoteke čine model podataka, dok sam XML dokument sadrži podatke



# XML model - primjer

---

```
<?xml version="1.0" encoding="UTF-8"?>
<Osoba xmlns:xsi="http://www.w3.org/
  2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Person.
xsd">
    <Ime>Trace</Ime>
    <Prezime>Galloway</Prezime>
    <Dob>32</Dob>
</Osoba>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/
   /XMLSchema">
  <xs:element name="Osoba">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Ime"
          type="xs:string"/>
        <xs:element name="Prezime"
          type="xs:string"/>
        <xs:element name="Dob"
          type="xs:integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

# UML model

---

- UML je prvenstveno namijenjen dizajniranju programske potpore, ali se može prilagoditi dizajniranju baze podataka
- Uveden zbog postrelacijskih baza podataka

# UML

---

- UML = Unified Modeling Language
  - ujedineni jezik za modeliranje
  - grafički jezik za vizualizaciju, specifikaciju, konstrukciju i dokumentiranje informacijskih sustava
  - pruža skup alata za sve uključene u informacijsku tehnologiju; skup alata podrazumijeva velik skup grafičkih elemenata, uključujući notaciju za klase, komponente, čvorove, aktivnosti, dijagrame tijeka, slučajeve uporabe, objekte, stanja i veze među svim navedenim elementima

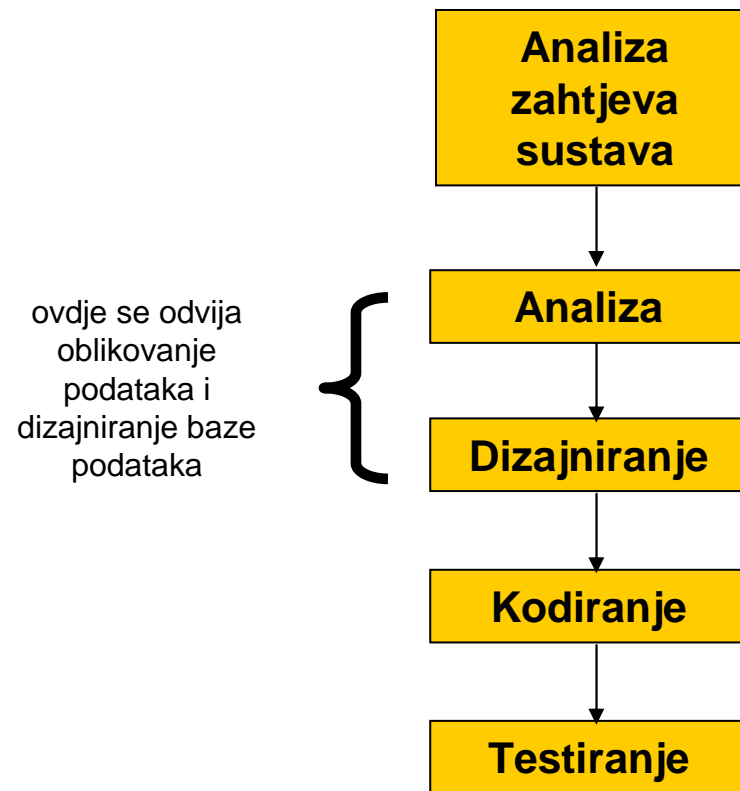
# Dijagrami

---

- Dijagrami strukture, Strukturni dijagrami
  - Class Diagram – dijagram razreda
  - Object Diagram – dijagram objekata
  - Component Diagram – dijagram komponenti
  - Deployment Diagram – dijagram ugradnje
- Dijagrami ponašanja
  - Use Case Diagram – dijagram slučajeva korištenja
  - Sequence Diagram – slijedni dijagrami
  - Collaboration Diagram – dijagrami kolaboracije
  - Statechart Diagram – dijagrami stanja
  - Activity Diagram – dijagrami aktivnosti

# Potpuni razvojni proces informacijskog sustava

---



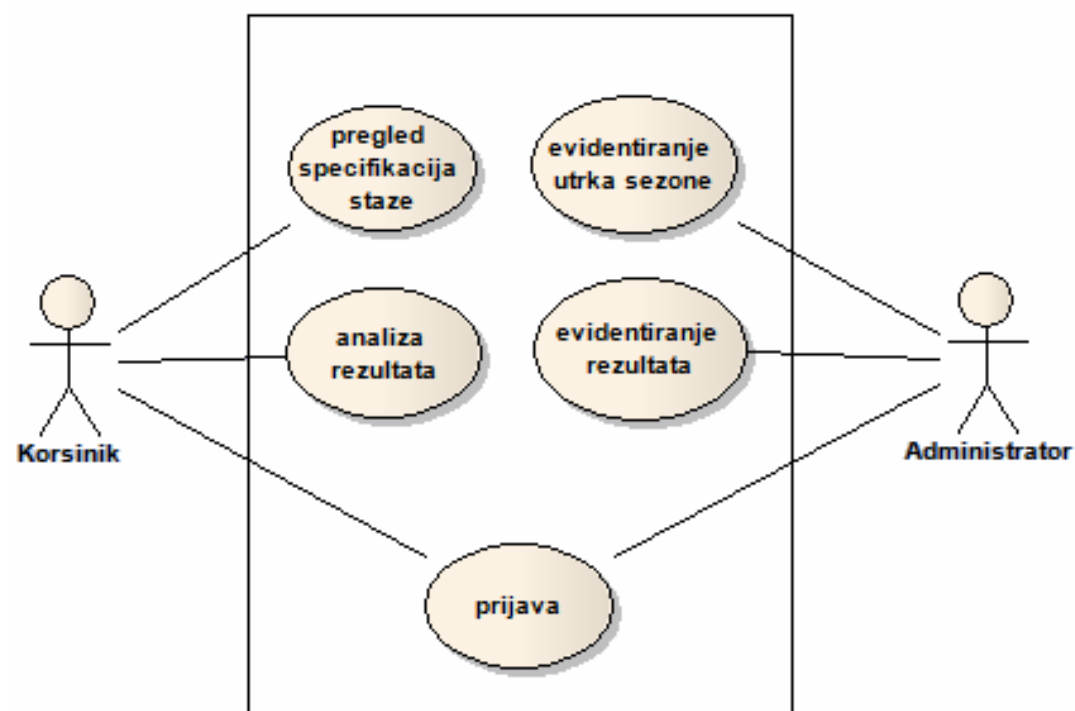
# Dijagram slučajeve korištenja

---

- Glumac (actor)
  - uloga koju korisnik ima u odnosu na sustav
  - proučava se da bi se odredile njegove potrebe
- Slučaj ponašanja (use case)
  - primjer, uzorak, obrazac ponašanja
  - slijed povezanih interakcija između sudionika i sustava
- Veze između glumaca i slučajeva korištenja
  - uključivanje («include») – povezuje zajedničke dijelove ponašanja
  - proširenje («extend») – varijacije normalnog ponašanja uz dodatna pravila
- Utjecaj na dizajn baze podataka
  - preslikavanjem slučaja korištenja dobije se uvid da određeni glumci zahtijevaju pristup određenim podacima na određen način

# Dijagram slučajeja korištenja - primjer

---

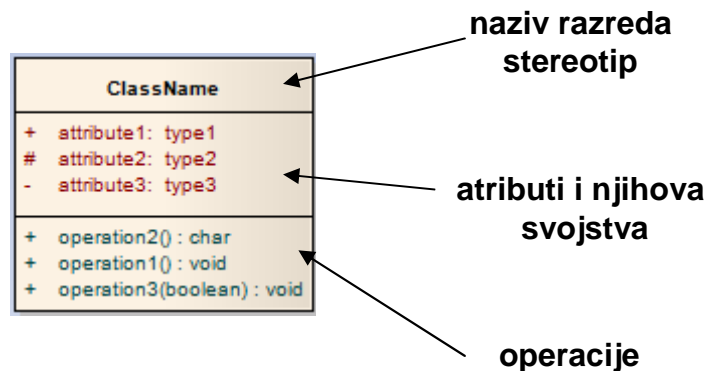




# Dijagram razreda (1)

---

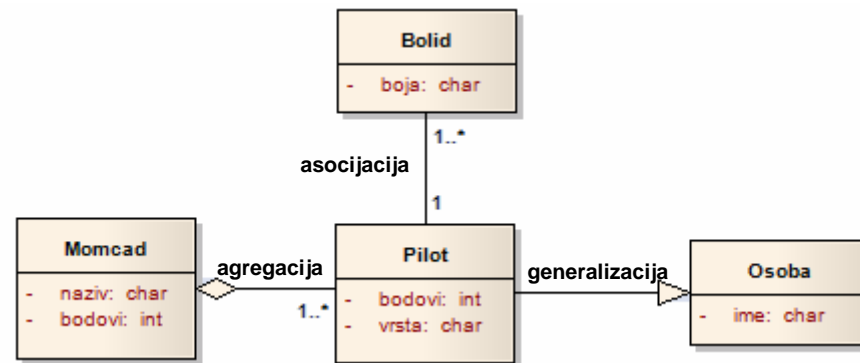
- Statički dijagram
- Opisuje strukturu razreda i veze među njima
- Trajnim razredima definira se stereotip <<*Persistent*>>
- Sličan ER dijagramu
- Elementi dijagrama
  - razredi, njihova struktura i ponašanje (atributi i operacije)
  - mogućnost pristupa (vidljivost)



# Dijagram razreda (2)

## Statički odnosi

- obične veze ili asocijacije (associations)
  - npr. "pilot vozi bolid"
- generalizacija - Zavisnost (dependency) i nasljeđivanje (inheritance)
  - npr. "pilot je osoba"
- gomilanje ili udruživanje, agregacija (aggregation),
  - npr. "pilot je član momčadi"
- Dodatne oznake
  - indikatori množine, mnogostrukosti (multiplicity) - kardinalnost
  - indikatori upravljivosti, navigabilnosti (navigability) – smjer povezivanja
  - uloge (role names)



# Dijagram razreda – posebne opcije

---

- *Stereotip (Stereotype) i označena vrijednost (tagged value)* proširuju rječnik UML-a
  - Stereotipovi se prikazuju korištenjem <<>>
  - Označene vrijednosti se prikazuju korištenjem {}
- Stereotip se koristi da bi klasificirao i proširio asocijacije, razrede i komponente (<<signal>>, <<persistent>>)
- Označene vrijednosti daju dodatne informacije za svaki element modela

# Dizajniranje relacijske baze

---

- Trenutno je najčešća kombinacija u razvoju programske potpore relacijska baza podataka kao podatkovni sloj na koji se naslanja objektno orijentiran model, tako da je objektni model dizajniran UML-om potrebno preslikati u relacijski model
- Problem se svodi na preslikavanje dijagrama razreda u relacijski model baze podataka
  - statička struktura razreda se lako preslika (atributi)
  - dinamičku strukturu razreda je teže preslikati budući da entitet u relacijskom modelu nema svoje pripadajuće operacije, tako da se koriste okidači, ograničenja i indeksi
- Budući da se radi preslikavanje dvije različite arhitekture, praksa je pokazala da je bolje nastojati da relacijski model nalikuje OO modelu, nego prilagođavati OO model relacijskom

# Razredi

---

## Pretvorba:

- svaki trajni (persistenti) razred postaje tablica u relacijskoj bazi
- svaki atribut postaje stupac te tablice
- svaka instanca tog razreda je jedna n-torka takve tablice

## ■ Problemi:

- pretvaranje tipova podataka atributa u SQL tipove podataka
- problem postavljanja primarnih ključeva
- preslikavanje operacija

# Razredi - primjer

---

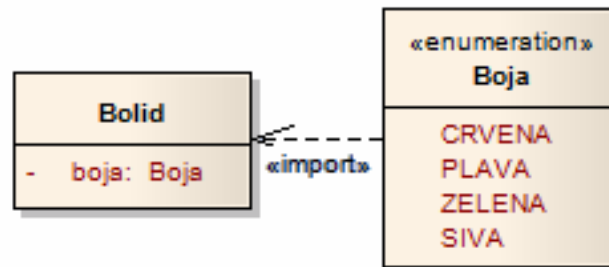
«Persistent» Osoba
+ OsobaID: int + Ime: char + Prezime: char + Spol: Spol + DatumRodj: date + Visina: float + BracniSatus: BracniStatus
+ getIme(): char + setIme(char): void

```
CREATE TABLE osoba(  
    OsobaID INTEGER PRIMARY KEY,  
    Ime VARCHAR (35),  
    Prezime Varchar (35),  
    Spol CHARACTER(2) NOT NULL CHECK (spol In ('M', 'Ž')),  
    DatumRodj DATE NOT NULL,  
    Visina FLOAT,  
    BracniStatus CHARACTER(1) NULL  
        CHECK (BracniStatus IN ('S', 'B', 'R', 'U')))
```

# Atributi i tipovi podataka (1)

---

- Svaki atribut UML razreda postaje stupac tablice u relacijskoj bazi podataka
  - treba voditi računa o duljini naziva (ovisno o vrsti SUBP-a)
  - vidljivost atributa u relacijskoj bazi je uvijek *public*
  - treba voditi računa o tipovima podataka
    - definira se tablica preslikavanja tipova (npr. String → VARCHAR(254))
    - problem predstavljaju pobrojani tipovi (enumerated type)

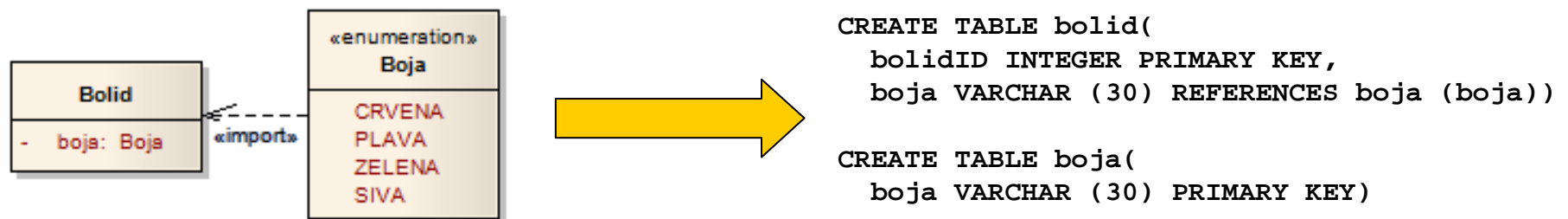




# Atributi i tipovi podataka (2)

- Pobrojani tipovi (Enumerated types)
  - ako se može definirati domena, kreira se novi tip  

```
CREATE DOMAIN Boja AS CHAR(20) NOT NULL  
CONSTRAINT Boja_Constraint CHECK (Boja IN ('CRVENA', 'ZELENA',  
'PLAVA', 'SIVA'))
```
  - ukoliko SUBP ne podržava domene, CHECK se mora ugraditi uz svaki atribut
    - nedostatak: mora se mijenjati svaki CHECK ukoliko se broj mogućih vrijednosti pobrojanog tipa poveća
  - kreira se tablica sa vrijednostima koje pobrojani tip može poprimiti i kreira se pripadajući strani ključ



# Ograničenja i ključevi

---

## ■ Ograničenja

- eksplicitno se postavi oznaka (označena vrijednost ili stereotip) za atribut koji može poprimiti NULL vrijednost
- ukoliko u UML modelu nema takve oznake znači da je taj atribut NOT NULL u relacijskom modelu
- sva ostala ograničenja zadana preko oznaka preslikavaju se CHECK naredbom kada je to moguće

## ■ Ključevi

- eksplicitno se postave oznake (označene vrijednosti ili stereotipovi) za attribute koji čine primarni ili alternativni ključ
- problem je postavljenje implicitnih identifikatora što se uglavnom svodi na kreiranje atributa tipa *SERIAL*

# Ključevi

---

- Relacijske baze se temelje na eksplicitnim identifikatorima
- Prema pravilima relacijske teorije:
  - ne može postojati nikakav identifikator koji nije ujedno i vrijednost atributa ili skupa atributa n-torke relacije
- Iako mnogi SUBPovi podržavaju skriveni identifikator kao što je *ROWID*, on je samo fizički identifikator, a ne i logički

# Dinamičko ponašanje

---

- Operacije i funkcije
  - jedino mjesto gdje se može opisati dinamičko ponašanje u relacijskim bazama podataka su funkcije ili procedure
  - treba razlučiti koja će se logika preslikati u funkcije na strani servera, koja u aplikacijski kod, a koja na klijentu
  - pravo nadjačavanje, preopterećivanje i dinamičko povezivanje ne postoje u relacijskim bazama, iako neki SUBP-ovi podržavaju višeobličje (polymorphism) takve stvari je bolje zaobići u relacijskom modelu
  - kada se u UML modelu koristi stereotip *Signal* znači da se operacija izvodi na neki događaj što odgovara okidačima u relacijskim bazama podataka

# Sučelja

---

- Sučelje (interface) - specifikacija članova razreda bez implementacije
  - sadrži samo deklaracije operacija
  - razred koji nasljeđuje sučelje, mora implementirati sve operacije
- Trajno (persistent) sučelje ne postoji u kontekstu relacijske baze
- Budući da sučelje nema attribute kod izrade relacijskog modela potrebno je implementirati operacije sučelja za svaki razred koji ga implementira

# Veze

---

- U relacijskoj bazi podataka ne postoje veze kao takve
- Velika je sličnost između modela veza u ER modelu i UML modelu
- Ključne razlike su u sljedećim slučajevima:
  - Nasljeđivanje
  - Agregacija
  - Asocijacija opisana kvalifikatorom (Qualified association)
- “Asocijacije opisane kvalifikatorom” i “poredane asocijacije” (ordered association) se rijetko koriste, pa se neće razmatrati

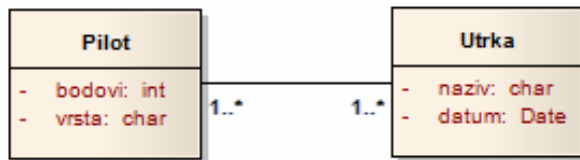
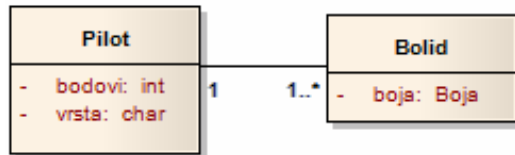
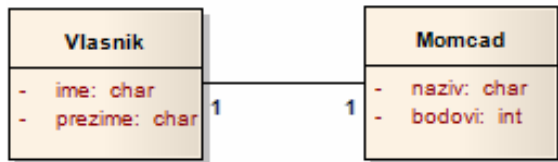
# Asocijacija (1)

---

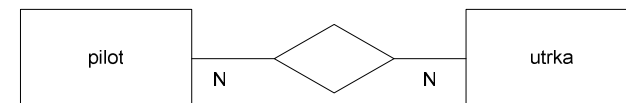
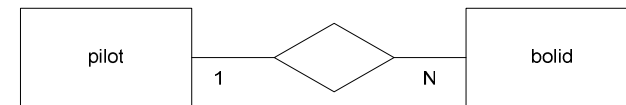
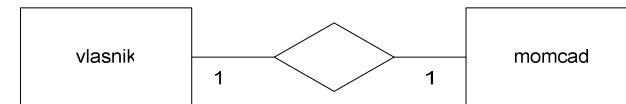
- Binarne asocijacije
  - svaka klasa koja pripada asocijaciji ima svoju ulogu i kardinalnost
  - svaka takva veza se tretira kao veza ER modela, te će relacijski model biti u skladu sa transformiranjem takve ER veze u relacijski model
- Ternarne, kvartarne, ... veze
  - u slučaju da je veza ternarna ili veće kardinalnosti, asocijacija se pretvara u tablicu u relacijskom modelu

# Asocijacija (2)

## UML



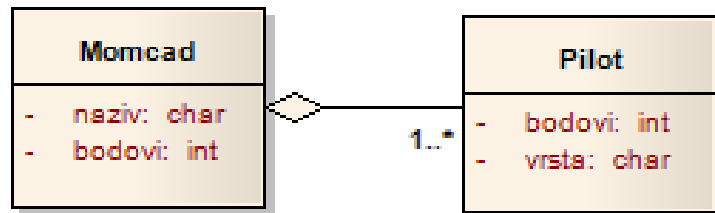
## ER





# Agregacija

- Agregacija (aggregation)
  - Poseban oblik asocijacije: veza "cjelina-dio"
  - Komponente su dijelovi objekta-agregata, takvi da
    - dio može nastati i postojati nezavisno od cjeline
    - uništenje "cjeline" ne uništava njene dijelove
- Preslikavanje u relacijski model je isto kao za asocijacije

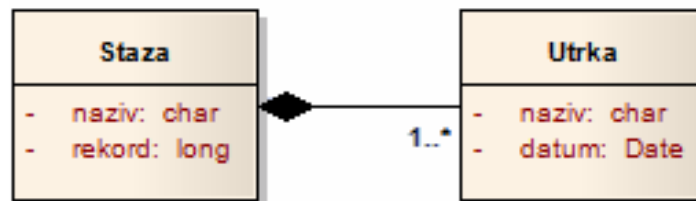


```
CREATE TABLE momcad(  
    momcadID INTEGER PRIMARY KEY,  
    naziv VARCHAR (30),  
    bodovi INTEGER)  
  
CREATE TABLE pilot(  
    pilotID INTEGER PRIMARY KEY  
    bodovi INTEGER,  
    vrsta CHAR(1),  
    momcadID REFERENCES momcad(momcadID))
```

# Kompozicija (1)

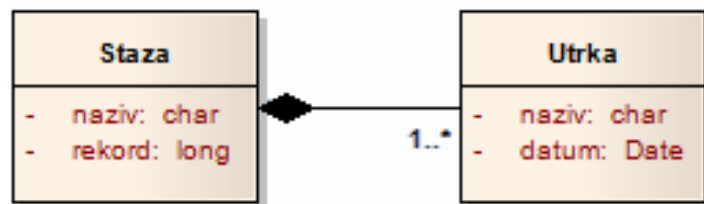
---

- Kompozicija (Composition)
  - "stroža" varijanta agregacije
  - objekt može pripadati samo jednoj cjelini
  - elementi postoje tako dugo dok postoji cjelina
    - kaskadno brisanje, tj. uništenje sastavnih objekata



## Kompozicija (2)

- Kompozicija odgovara vezi između entiteta vlasnika i slabog entiteta u relacijskom modelu
- Kreira se strani ključ sa kaskadnim brisanjem budući da ne može postojati *dio(slabi entitet)* ukoliko ne postoji *cjelina(entitet vlasnik)*

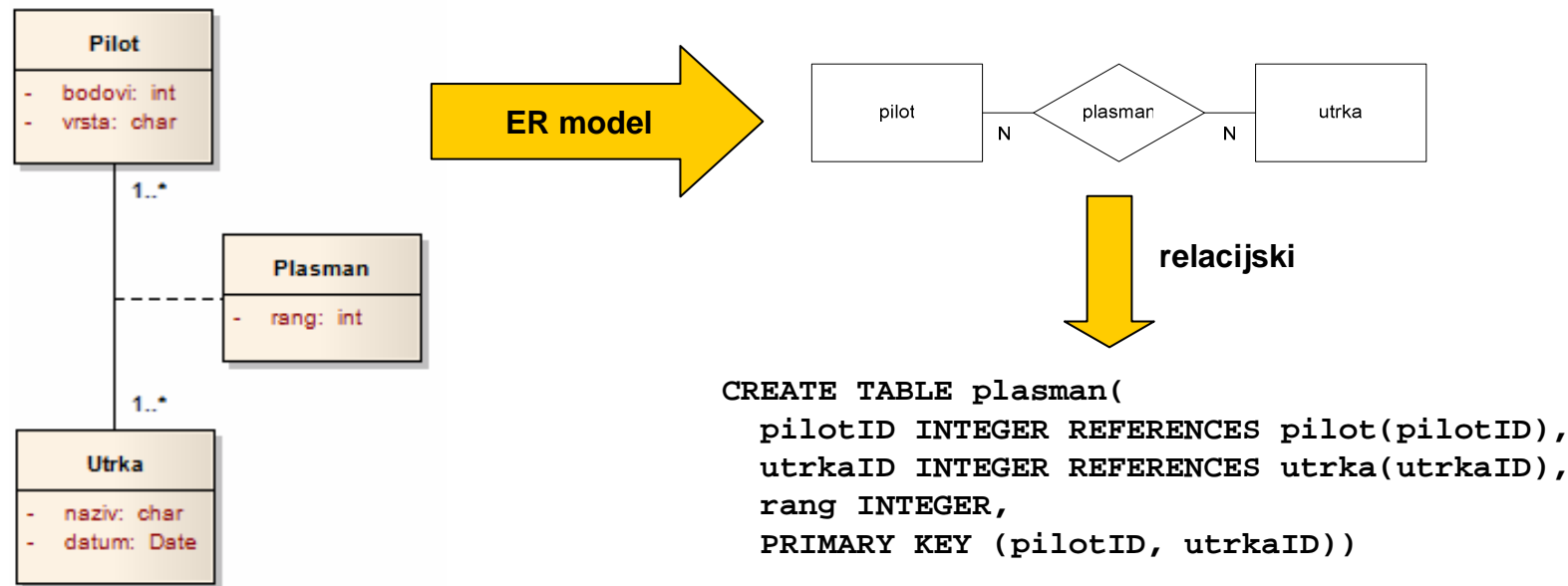


```
CREATE TABLE utrka(  
    stazaID INTEGER REFERENCES staza(stazaID)  
        ON DELETE CASCADE,  
    datum DATE,  
    naziv VARCHAR(30),  
    PRIMARY KEY (stazaID, datum))
```

```
CREATE TABLE staza(  
    stazaID INTEGER PRIMARY KEY  
    naziv VARCHAR(30),  
    rekord DECIMAL(7,4))
```

# Razred asocijacije

- Razred pridružen asocijaciji, koji pruža dodatne informacije o vezi
- Odgovara vezi koja ima vlastite attribute iz ER modela
- Takav razred pretvara se u zasebnu tablicu u relacijskom modelu, koja onda u procesu normalizacije može i nestati iz modela (u slučaju 1:N i 1:1 veza)



# Nasljeđivanje

---

Mogući scenariji:

- 1) Kreira se jedna tablica sa unijom svih atributa - ovo dovodi do nenormaliziranosti baze podataka
  - 2) Kreira se tablica za svaki razred sa atributima koji su vidljivi tom razredu (naslijeđeni atributi) – potrebni su strani ključevi
  - 3) Kreira se tablica za svaku razinu nasljeđivanja sa unijom atributa razreda na toj razini – potrebni su strani ključevi
- Višestruko nasljeđivanje
    - Preporuka je pokušati redizajnirati model podataka i izbjeći preslikavanje u relacijski model; kao posljedicu može uzrokovati nenormaliziranost baze podataka

# Koraci pretvorbe (1)

---

- UML razred postaje tablica
- UML atribut postaje stupac tablice
- UML tip podatka atributa se preslika u odgovarajući tip podatka relacijske baze
- Ukoliko atribut nema postavljenu oznaku *nullable*, postavlja mu se ograničenje NOT NULL
- Ukoliko UML atribut ima inicijalizator, uz atribut u relacijskoj bazi se postavlja DEFAULT vrijednost
- Za razrede koji nemaju generalizaciju, a imaju implicitan identifikator, kreirati primaran ključ (serial); za eksplicitno definirane identifikatore kreirati primaran ključ nad tim atributima
- Za podrazrede, dodati ključ roditelja u ograničenja primarnog ključa i stranog ključa

## Koraci pretvorbe (2)

---

- Za razrede asocijacija dodati primaran ključ iz svake tablice koja sudjeluje u asocijaciji u ograničenje primarnog i stranog ključa
- Kreirati UNIQUE ograničenja, koja su eksplicitno postavljena
- Za svako eksplicitno ograničenje dodati CHECK
- Kreirati strane ključeve na svaku 0..1, 1..1 ulogu u asocijaciji
- Kreirati primaran ključ tablice koja predstavlja dio iz kompozicije sa stranim ključem na tablicu koja predstavlja cjelinu; kreirati ON DELETE CASCADE
- Normalizirati binarne asocijacije tako da se udruže tablice sa istim primarnim ključevima
- Kreirati tablice za n-n i ternarne asocijacije, koje nemaju razrede asocijacija
- Kreirati primarne i strane ključeve tablicama koje nastanu iz n-n i ternarnih asocijacija

# Dizajniranje relacijske baze iz UML modela

---

- Potrebno je poznavanje tehnike pretvorbe ER modela u relacijski model
- Potrebno je naučiti dodatna pravila zbog specifičnosti UML modela
- Potrebno je znanje normalizacije





# Dizajniranje objektna baze (1)

---

- pretvaranje UML modela u objektni model baze podataka je prilično jednoznačan
- problemi se javljaju zbog mana OOSUBP-a; nepridržavanje dogovorenog standardu (ODMG)
- Koraci pretvorbe:
  - svaki UML trajni (*persistent*) razred postaje OOSUBP razred
  - UML sučelje koje trajni razred koristi postaje OOSUBP sučelje (ako objektni jezik OOSUBP-a podržava sučelja)
  - UML atributi razreda postaju atributi OOSUBP razreda (obično je potrebno provesti transformaciju tipova podataka i ograničenja)
  - većina OOSUBP-a ne podržava NULL vrijednost pa nema načina za preslikati UML *nullable* oznaku
  - ukoliko UML model ima inicijalizator, kod se dodaje u konstruktor

# Dizajniranje objektne baze (2)

---

- Koraci pretvorbe (nastavak):
  - svaki podrazred uključuje specificiranje svog nadređenog razreda
  - za razrede asocijacija kreirati odgovarajuće razrede
  - za eksplicitno definirane identifikatore treba kreirati odgovarajuće funkcije koje će provjeravati jedinstvenost u skupu objekata; ne postoje ključevi u OO jezicima
  - za svako eksplicitno ograničenje kreirati člansku funkciju i osigurati da se pravovremeno poziva
  - kreirati veze za svaku binarnu asocijaciju i za svaku vezu od razreda asocijacije do drugih klasa
  - kada se koristi agregacija kreirati kod ili koristiti OOSUBP pogodnosti da bi se osiguralo kaskadno brisanje
  - za ternarne asocijacije kreirati razrede asocijacija i veze od tog razreda predam svim razredima koji sudjeluju u vezi (ODMG ne podržava n-arne asocijacije, pa se moraju raditi preinake modela)