Relacijski podatkovni model

Predavanje 3/A

Zakaj študij relacijskega modela?

- Najširše uporabljan model.
  - Prodajalci: IBM, Informix, Microsoft, Oracle, Sybase, itd.
- "Legacy systems" v starih modelih.
  - Npr., IBM IMS
- Konkurenca: objektno-orientiran model
  - ObjectStore, Versant, Ontos
  - Združevanje: *objektno-relacijski model*
    - Informix Universal Server, UniSQL, O2, Oracle, DB2

Relacijska podatkovna baza

- *Relacijska PB:* *množica* *relacij.*
- *Relacija:* dva dela:
  - *Instanca* : *tabela*, ki ima vrstice in stolpce. #vrstic = *kardinalnost*, #stolpcev = *stopnja.*
  - *Shema* : določa ime relacije ter imena in tipe vseh stolpcev.
    - Studenti(*sid*: string, ime: string, *login*: string,

      starost: integer, po: real).
- Relacijo si lahko predstavljamo kot množico n-teric ali vrstic.

Primer relacije *študent*

sid	ime	login	star.	po
53666	Novak	novak@pef	18	7.2
53688	Kranjc	kranjc@ma	18	6.8
53650	Kranjc	kranjc@fi	19	9.3

- Kardinalnost = 3, stopnja = 5, vse vrstice različne
- Ali morajo biti vse vrstice v tabeli različne?

Relacijski povpraševalni jeziki

- Največja moč relacijskega modela:

  podpira enostaven povpraševalni jezik z veliko izrazno močjo.
- Vprašanja se lahko pišejo intuitivno, SUPB je odgovorna za učinkovito evaluacijo.
  - Razlog: natančna semantika relacijskih poizvedb.
  - Optimizator običajno temeljito preuredi poizvedbo in zagotavlja, da se odgovor ne spremni.

Povpraševalni jezik SQL

- Razvit na IBM (system R) leta 1970
- Potreba po standardu, ker ga uporablja veliko število proizvajalcev.
- Standardi:
  - SQL-86
  - SQL-89 (majhne spremembe)
  - SQL-92 (večje spremembe)
  - SQL-99 (večje spremembe, trenutni standard)

Povpraševalni jezik SQL

- Vprašanje*: Poišči vse študente, ki so stari 18 let.*

```
SELECT *
FROM Studenti S
WHERE S.starost=18
```

sid	ime	login	star	po
53666	Novak	novak@pef	18	3.4
53688	Kranjc	kranjc@ma	18	3.2

- Vprašanje: *poišči imena in login študentov*.

```
SELECT S.ime,
S.login
FROM Studenti S
```

Povpraševanje nad večimi relacijami

- Kaj izračuna naslednje vprašanje?

```
SELECT  S.ime, V.pid
FROM    Studenti S, Vpis V
WHERE   S.sid=V.sid AND V.ocena="10
```

Relaciji Studenti in Vpis:

sid	name	login	Star	po
53666	Novak	novak@pef	18	7.2
53688	Kranjc	kranjc@ma	18	6.8
53650	Kranjc	kranjc@fi	19	9.3

sid	pid	ocena
53831	Baze	5
53831	Matematika	7
53650	Geometrija	10
53666	Zgodovina	7

Dobimo:

S.ime	V.pid
Kranjc	Geometrija

Kreiranje relacij v SQL

- Kreiranje relacije *Studenti*. Tip se specificira za vsako polje. Pri vpisu podatkov v bazo, SPUB zahteva definiran tip podatka.
- Tabela *Vpis* vsebuje podatke o predmetih, ki so jih vpisali študenti.

```
CREATE TABLE Studenti
        (sid: CHAR(20),
         ime: CHAR(20),
         login: CHAR(10),
         star: INTEGER,
         po: REAL)
```

```
CREATE TABLE Vpis
        (sid: CHAR(20),
         pid: CHAR(20),
         ocena: CHAR(2))
```

Brisanje in spreminjanje relacij

```
DROP TABLE Studenti
```

- Zgornji ukaz izbriše relacijo Studenti -- shemo in zapise.

```
ALTER TABLE Studenti
        ADD COLUMN kraj: varchar(25)
```

- Shemo študenti spremenimo tako, da dodamo nov atribut *kraj* s katerim opišemo kraj stanovanja študenta.

Dodajanje in brisanje zapisov

- Vstavljanje enega zapisa:

```
INSERT INTO Studenti(sid, ime, login, star, po)
VALUES  (53688, ‘Novak’, ‘novak@pef’, 18, 3.2)
```

- Brišemo vse zapise, ki zadoščajo določenem pogoju, npr., ime=“Novak”:

```
DELETE
FROM Studenti S
WHERE S.ime = ‘Novak’
```

- *Možne so variante ukazov z večjo izrazno močjo; več kasne*

Integritetne omejitve (IC)

- IC: pogoj, ki mora biti izpolnjen za vsako n-terico relacije.
  - IC so določene pri definiciji sheme.
  - IC se preverjajo ob spreminjanju relacij.
- Legalna instanca relacije je takšna, ki zadovoljuje vse specificirane IC.
  - SPUB ne bi smel dopuščati nelegalnih instanc.
- Če SPUB preverja IC, potem so shranjeni podatki bližje pomenu v realnem svetu.
  - Izogibanje napakam pri vnosu!

Primarni ključ

- Množica atributov je ključ relacije če:
  1. Ne obstajata dva enaka zapisa z isto vrednostjo ključa.
  2. To ne velja za nobeno podmnožico ključa.
  - Če obstaja potem temu pravimo superključ?
  - Če obstaja več kot en ključ za relacijo potem izberemo enega, ki ga *imenujemo* primarni ključ.
- Npr. *sid je* ključ za relacijo *Studenti.* (*ime*?)
  Množica {*sid, po*} je superključ.

Primarni in kandidatni ključ v SQL

- Običajno je na voljo več kandidatnih ključev (definiramo kot UNIQUE) izmed katerih izberemo en primarni ključ.

- Pravilno: "Za danega študenta in predmet imamo eno samo oceno."
- Nepravilno: "Študenti se lahko vpišejo na samo en predmet in dobijo eno samo oceno za ta predmet; ne smeta obstajati dva študenta, ki imata isto oceno."
- Če uporabimo IC nepravilno lahko onemogočimo vnos dejanskih podatkov iz realnega sveta!

```
CREATE TABLE Vpis
  (sid CHAR(20)
   pid  CHAR(20),
   ocena CHAR(2),
   PRIMARY KEY  (sid,pid) )
CREATE TABLE Vpis
  (sid CHAR(20)
   pid  CHAR(20),
   ocena CHAR(2),
   PRIMARY KEY  (sid),
   UNIQUE (pid, ocena) )
```

Tuji ključi in referenčna integriteta

- *Tuj ključ* : Množica atributov neke relacije, ki referencira zapise druge relacije. Izbrana množica atributov mora ustrezati primarnem ključu druge relacije. Neke vrste "logični kazalec".
- Primer: *sid* je tuj ključ (Studenti):
  - Vpis(*sid*: string, *pid*: string, *ocena*: string)
  - Če je integritetna omejitv tujih ključev upoštevana v PB potem pravimo, da dosežemo referenčno integriteto; ni "visečih referenc".
  - Naštej imena podatkovnih modelov z / brez referenčne integritete?
    - Link & HTML!

Tuji ključi v SQL

- Samo študenti, ki so v tabeli *Studenti* lahko nastopajo v relaciji *Vpis*.

```
CREATE TABLE Vpis
   (sid CHAR(20),  pid CHAR(20),  ocena CHAR(2),
    PRIMARY KEY  (sid,pid),
    FOREIGN KEY (sid) REFERENCES Studenti )
```

Vpis

sid	pid	grade
53666	Baze	5
53666	Matematika	7
53650	Geometrija	10
53666	Zgodovina	7

Studenti

sid	ime	login	Star	po
53666	Novak	novak@cs	18	7.2
53688	Kranjc	kranjc@ma	18	6.8
53650	Kranjc	kranjc@fi	19	9.3

Zagotavljanje referenče integritete

- Poglejmo si tabeli Studenti in Vpis; *sid* v tabeli Vpis je tuj ključ, ki referencira tabelo Studenti.
- Kaj naj se zgodi, če poskušamo dodati zapis, ki vsebuje neobstoječ sid v tabelo Vpis? *Zavrnitev zapisa!*
- Kaj naj se zgodi, če izbrišemo zapis tabele Studenti?
  - Pobrišejo se tudi pripadajoči zapisi v Vpis.
  - Ne dovoli se brisanje zapisa Studenti na katerega se referencirajo zapisi iz tabele Vpis.
  - Vrednosti zbrisanih *sid* se v tabeli Vpis postavi na privzeto vrednost.
  - SQL: Postavi vrednosti izbrisanih *sid* v Vpis na null vrednost, ki pomeni *neznano* ali *nenavedeno*.
- Podobno v primeru, da se ključ sid v tabeli Studenti spremeni.

Referenčna integriteta v SQL

- SQL/92 in SQL:1999 podpira vse 4 možnosti pri brisanju in popravljanju.
  - Privzeto je NO ACTION (*delete/update je zavrnjen*)
  - CASCADE (zbriši vse zapise, ki se sklicujejo na zbrisani zapis)
  - SET NULL / SET DEFAULT (postavi tuj ključ v zapisih, ki se referencirajo na izbrisani zapis)

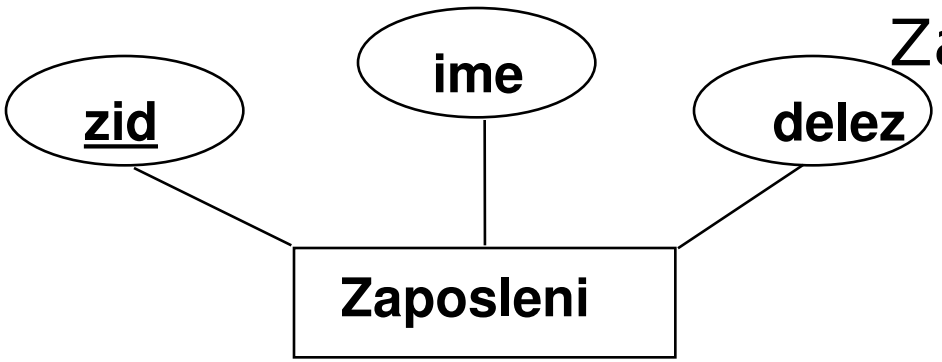
```
CREATE TABLE Vpis
   (sid CHAR(20),
    pid CHAR(20),
    ocena CHAR(2),
    PRIMARY KEY  (sid,pid),
    FOREIGN KEY (sid)
      REFERENCES Studenti
         ON DELETE CASCADE
         ON UPDATE SET DEFAULT )
```

Od kje so prišle IC omejitve?

- IC so osnovane na pomenu okolja, ki ga modeliramo z relacijskim modelom.
- Lahko preverimo podatkovno bazo ali je v skladu z integritetnimi omejitvami . Ne moremo pa NIKOLI sklepati, da je omejitev pravilna z opazovanjem modela (instance).
  - IC je izjava o vseh možnih instancah!
  - Iz primera vidimo, da ime ni ključ. Določitev, da sid je ključ je prepuščeno nam.
- Ključ in tuj ključ najbolj pogoste IC; bolj splošne IC tudi obstajajo.

Logično načrtovanje PB: prevod ER v relacije

- Entitetne množice v tabele:




```
CREATE TABLE
Zaposleni
        (zid CHAR(11),
        ime CHAR(20),
        delez INTEGER,
        PRIMARY KEY
(zid))
```

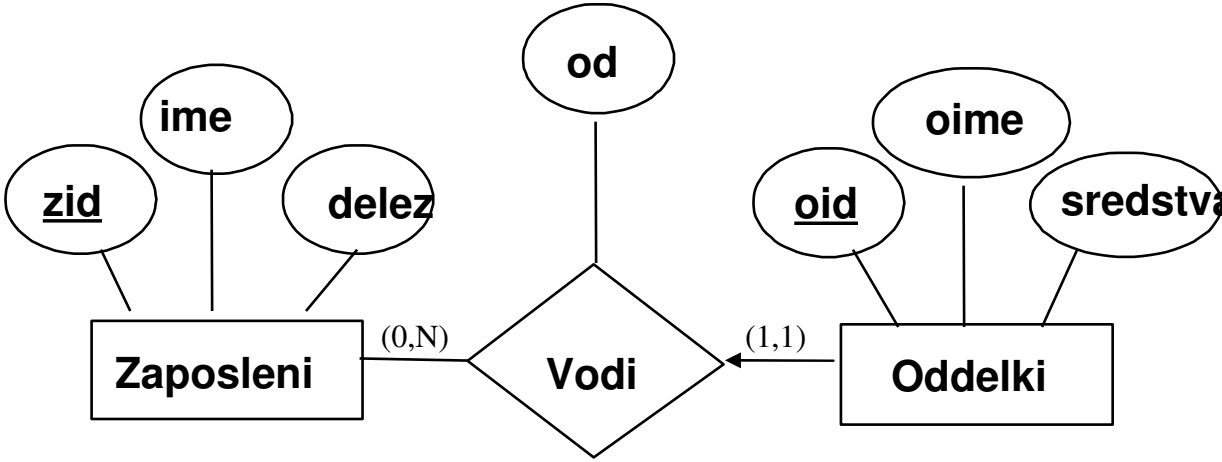
Množice razmerja v tabele

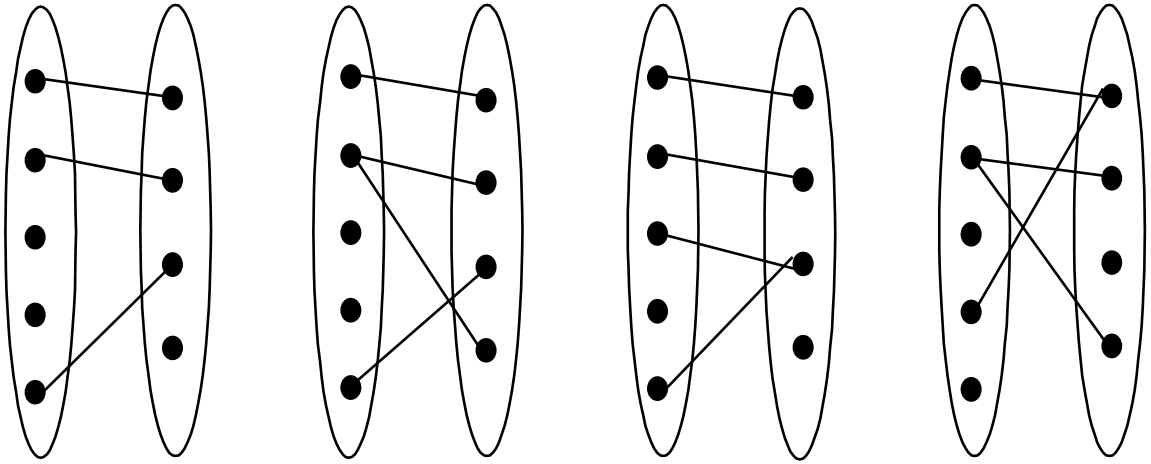
- Pri prevajanju razmerja v tabele mora množica atributov vsebovati:
  - Ključe za vsako sodelujočo entitetno množico (kot tuj ključ).
    - Ta množica atributov tvori superključ tabele.
  - Vse opisne atribute.

```
CREATE TABLE Dela_V(
  zid  CHAR(11),
  oid  INTEGER,
  od  DATE,
  PRIMARY KEY (zid, oid),
  FOREIGN KEY (zid)
      REFERENCES Zaposleni,
  FOREIGN KEY (oid)
      REFERENCES Oddelki)
```

Ponovitev: Ključ

- Vsak oddelek ima največ enega šefa; to je v skladu s ključem na razmerju Vodi.





1-1 1-Več Več-1 Več-Več

Prevod v relacijski model?

Prevajanje ER diagramov, ki imajo ključe

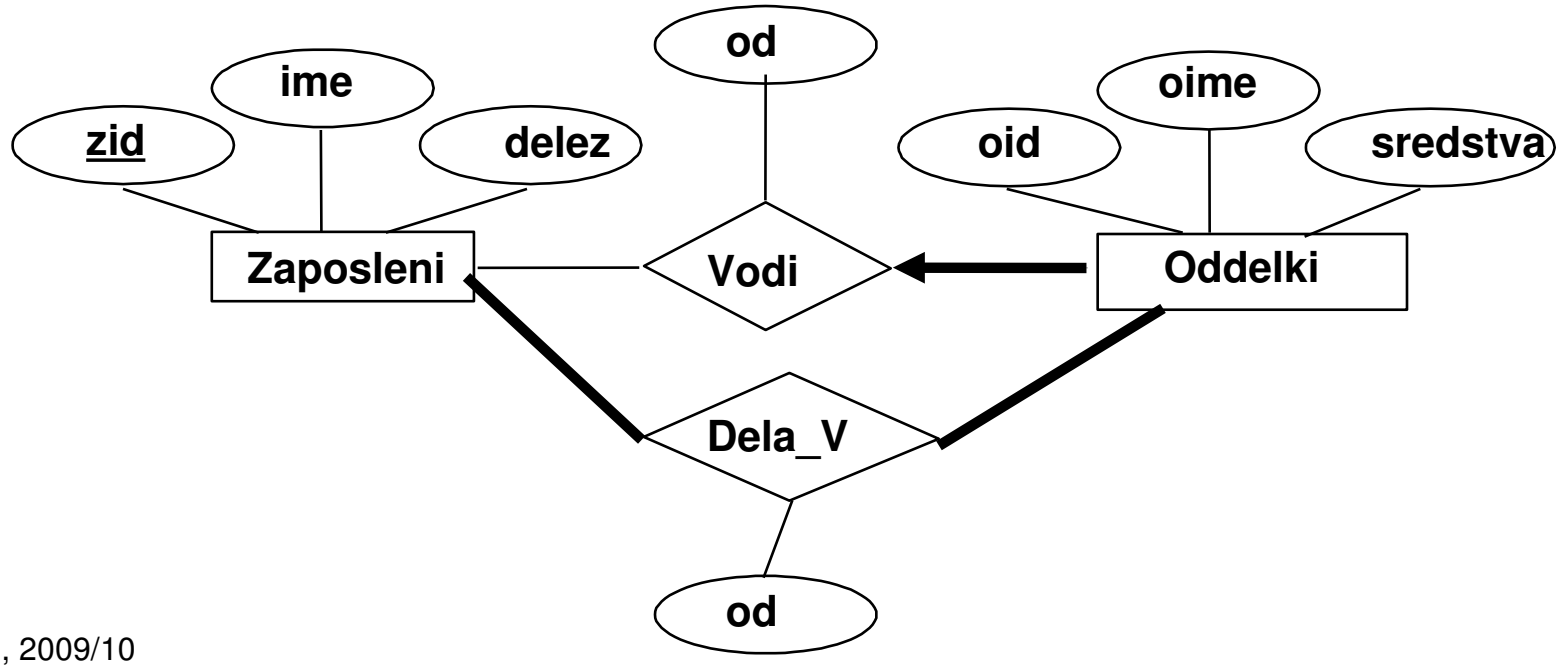
- Prevod razmerja v tabelo:
  - oid je ključ!
  - Različne tabele za Zaposleni in Oddelki.
- Ker ima vsak oddelek enega vodjo bi lahko združili Vodi in Oddelki.

```
CREATE TABLE Vodi(
  zid CHAR(11),
  oid INTEGER,
  od DATE,
  PRIMARY KEY (oid),
  FOREIGN KEY (zid) REFERENCES Zaposleni,
  FOREIGN KEY (oid) REFERENCES Oddelki)
```

```
CREATE TABLE OddelkiVodi(
  oid INTEGER,
  oime CHAR(20),
  sredstva REAL,
  zid CHAR(11),
  od DATE,
  PRIMARY KEY (oid),
  FOREIGN KEY (zid) REFERENCES Zaposleni)
```

Ponovitev: Obveznost članstva

- Ima vsak oddelek vodjo?
  - To smo imenovali obveznost članstva: članstvo entitet Oddelki v razmerju Vodi je totalno (vs. delno).
    - Vsaka vrednost *oid* v Oddelkih se mora pojaviti v eni vrstici tabele Vodi (z vrednostjo zid, ki ni *null*)



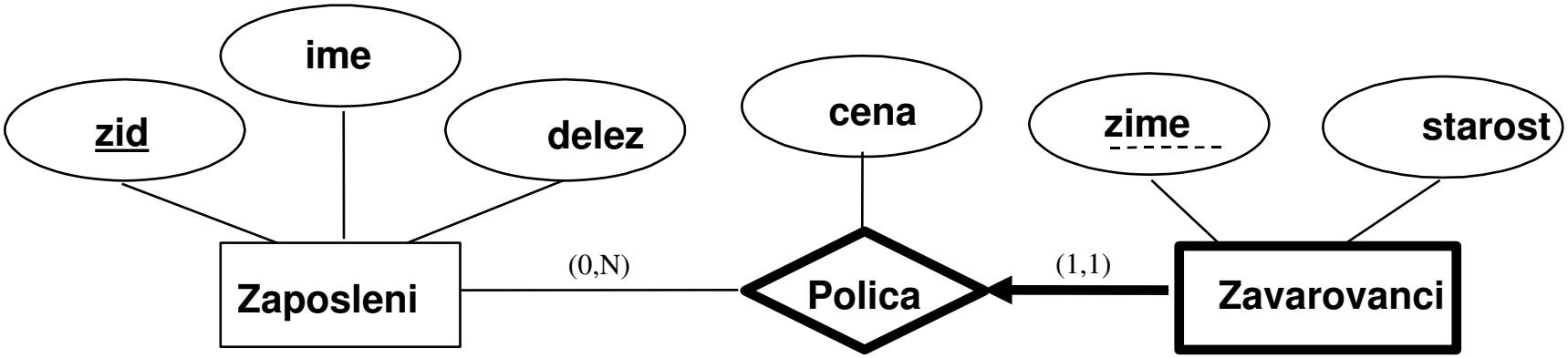
Obveznost in SQL

- Obveznost lahko izrazimo v primeru binarnih razmerij.
- Več z uporabo CHECK integritetnih omejitev.

```
CREATE TABLE  OddelkiVodi(
   oid  INTEGER,
   oime  CHAR(20),
   sredstva  REAL,
   zid  CHAR(11) NOT NULL,
   od  DATE,
   PRIMARY KEY  (oid),
   FOREIGN KEY  (zid) REFERENCES Zaposleni,
      ON DELETE NO ACTION)
```

Ponovitev: Šibke entitete

- Šibko entiteto lahko enolično identificiramo le z uporabo ključa močne entitete.
  - Entitetna množica lastnika in entitetna množica šibke entitete morajo sodelovati v razmerju tipa 1-več (en lastnik več šibkih entitet).
  - Šibka entitetna množica mora imeti totalno udeležbo v razmerju, ki je ključ razmerja.

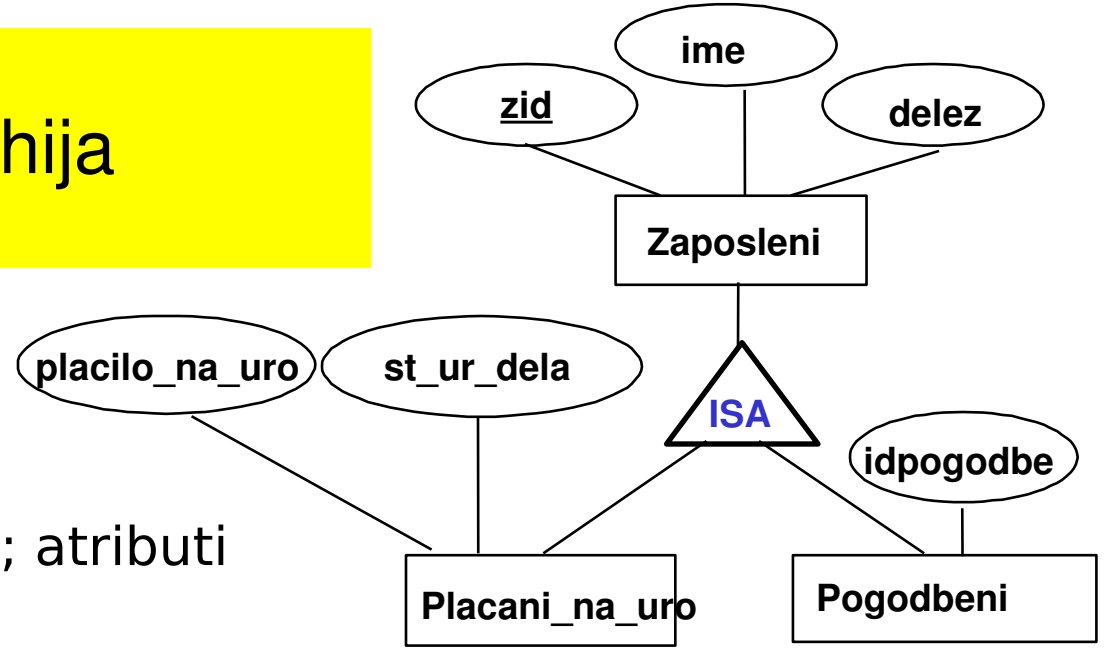


Prevajanje šibkih entitetnih množic

- Šibke entitetne množice in razmerje, ki jo identificira so prevedene v eno samo tabelo.
  - Ko se izbriše lastnik je potrebno pobrisati še vse šibke entitete.

```
CREATE TABLE Polica (
   zime CHAR(20),
   starost INTEGER,
   cena REAL,
   zid CHAR(11) NOT NULL,
   PRIMARY KEY (zime, zid),
   FOREIGN KEY (zid) REFERENCES Zaposleni,
      ON DELETE CASCADE)
```

Ponovitev: ISA Hierarhija



- Kot v C++, ali ostalih PL; atributi se dedujejo.
- Če deklariramo, da A **ISA** B, potem je vsaka A entiteta tudi B entiteta.

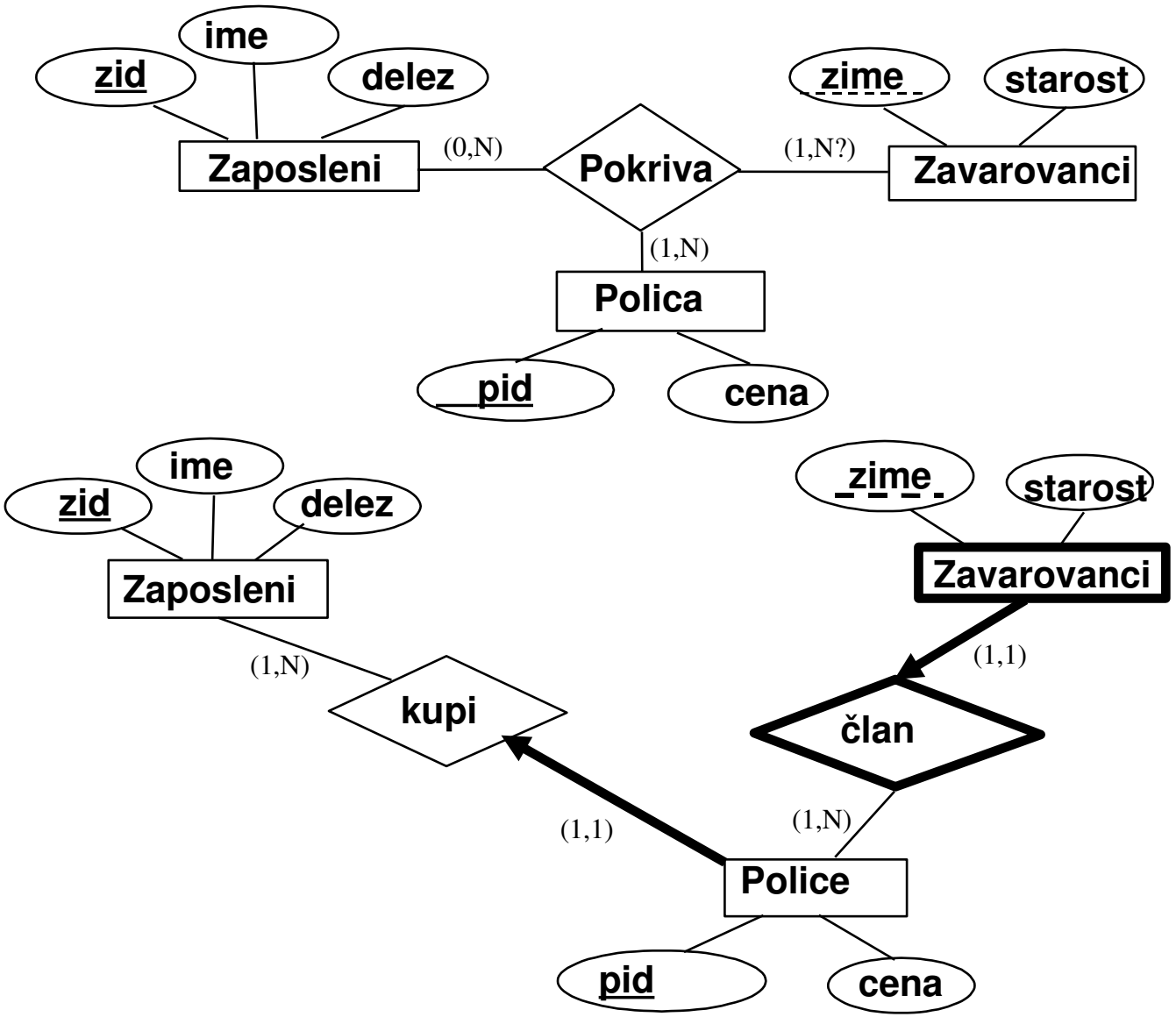
- *Prekrivanje pod-entitetnih množic*: Je lahko Tone v množici Placan_na_uro kot tudi v množici Pogodbeni? (*Da/ne*)
- *Pokrivanje nad-entitetne množice*: Mora vsak zaposleni nujno biti član tudi ene izmed podrejenih entitet? *(Da/ne)*

Prevajanje ISA hierarhije v relacije

- Splošen pristop:
  - 3 relacije: Zaposleni, Placani_na_uro, Pogodbeni.
    - *Placani_na_uro*: Vsak zaposleni je zapisan v tabeli Zaposleni. Placani na uro so zapisani tudi v tabeli Placani_na_uro. (*zid,placilo_na_uro,st_ur_dela)*; Pozor: pri brisanju se morata pobrisati oba zapisa (ON DELETE CASCADE).
    - Vprašanja, ki se dotikajo vseh zaposlenih so enostavna. Tista vprašanja, ki se dotikajo samo plačane na uro, zahtevajo stik dveh tabel.
- Alternativa: Samo tabeli Placani_na_uro in Pogodbeni.
  - *Placani_na_uro: zid, ime, delez, placilo_na_uro, st_ur_dela.*
  - Vsak zaposleni mora biti v enem izmed podrazredov sicer se podvajajo podatki.

Ponovitev: Binarno vs. ternarno razmerje

- Kaj so dodatne omejitve v drugem diagramu?



Binarno vs. ternarno razmerje

- Ključ razmerja nam dovoljuje, da združimo *kupi* z *Police* ter *clani* z *Zavarovanci*.

- Obveznost članstva vodi do NOT NULL omejitev.

- Kaj če bi bile Police šibka entiteta?

```
CREATE TABLE  Polica (
   idpolice  INTEGER,
   cena  REAL,
   zid  CHAR(11)  NOT NULL,
   PRIMARY KEY (idpolice).
   FOREIGN KEY (zid) REFERENCES Zaposleni,
      ON DELETE CASCADE )
```

```
CREATE TABLE Zavarovanci (
   zime  CHAR(20),
   starost  INTEGER,
   idpolice  INTEGER,
   PRIMARY KEY (zime, idpolice).
   FOREIGN KEY (idpolice) REFERENCES Police,
      ON DELETE CASCADE)
```

Pogledi

- Pogled (angl. VIEW) je relacija, toda shranimo definicijo in ne zapisov.

```
CREATE VIEW MladiAktivniStudenti(ime, ocena)
        AS SELECT S.ime, V.ocena
        FROM Studenti S, Vpis V
        WHERE S.sid = V.sid and S.starost<21
```

- Pogled se lahko izbriše z ukazom DROP VIEW.
  - Kako narediti DROP TABLE če je na tabeli definiran pogled?
    - DROP TABLE ukaz ima opcijo, ki omogoča uporabniku specifikacijo akcije.

Relacijski model: pregled

- Tabularna predstavitev podatkov.
- Enostavno in intuitivno, trenutno najbolj pogosto uporabljan model.
- Integritetne omejitve lahko vnesemo v podatkovno bazo na osnovi lastnosti modeliranega podatkovnega okolja. SUPB preveri veljavnost omejitev.
  - Pomembni omejitvi: primarni in tuj ključ.
  - Vedno so definirane domene za atribute.
- Na voljo je močan in dokaj naraven povpraševalni jezik.
- Pravila za prevajanje ER v relacijski model.