Prevod ER modela v SQL

Iztok Savnik

FAMNIT

Literatura

Toby Teorey, Sam Lightstone, Tom Nadeau, H.V. Jagadish, Database Modeling and Design: Logical Design, 5th Edition, Morgan Kaufmann Pub., Inc. (Elsevier), 2011. (Chapter 5)

Uvod

- Osnovna pravila prevajanja ER diagramov v relacije so podana
  - Prevajanje entitet in diagramov
  - Nekatere integritetne omejitve so lahko uporabljene za implementacijo gradnikov ER modela
    - Deklaracije NULL, tuji ključi, integritetna omejitev UNIQUE, funkcijske odvisnosti
- Nekatere gradnike ER se ne da prevesti v SQL

Prevajanje entitet

- SQL tabela z isto informacijsko vsebino kot originalna entiteta iz katere je bila razvita
- Transformacija se uporablja za entitete z binarnim razmerjem
  - N-N
  - 1-N na „ena“ strani
  - 1-1 na obeh straneh
- Entitete, ki so povezane z binarno rekurzivno relacijo tipa N-N
- Entitete s ternarnim razmerjem ali z razmerjem višjega reda ali z generalizacijsko hierarhijo

Odvisne entitete

- SQL tabela z vgnezdenim tujim ključem entitete starša
- Transformacija se vedno naredi za entitete, ki imajo binarno razmerje
  - 1-N za entiteto na N (otrok) strani
  - 1-1 za eno od entitet
  - za vsako entiteto, ki ima binarno rekurzivno razmerje tipa 1-1 ali 1-N
- To je eden izmed dveh načinov na katere načrtovalska orodja obravnavajo razmerja

Pretvorba razmerij

- SQL tabela izpeljana iz razmerja
  - vsebuje tuje ključe vseh entitet v razmerju
- Transformacija se naredi za
  - binarna razmerja tipa N-N
  - razmerja, ki so rekurzivna in N-N
  - razmerja, ki so ternarna ali višjega reda
- To je drugi najbolj pogost način na katerega orodja obravnavajo razmerja v ER in UML
  - razmerje N-N se lahko definira samo s tabelo, ki vsebuje tuje ključe entitet v razmerju
  - ta tabela lahko vsebuje tudi atribute originalnega razmerja

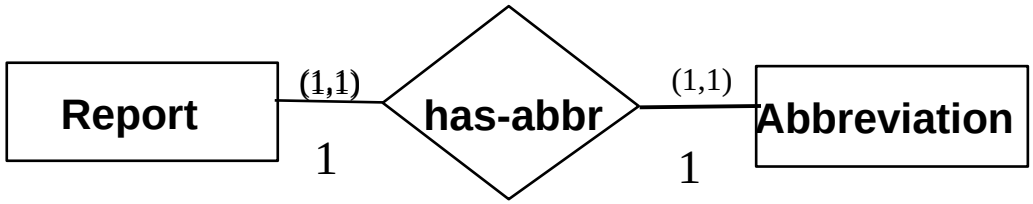
Uporaba null vrednosti

- Null vrednosti so dovoljene v SQL tabelah za tuje ključe, ki povezujejo opcijske entitete
- Null vrednosti niso dovoljene v SQL tabelah za tuje ključe, ki povezujejo obvezne entitete
- Null vrednosti niso dovoljene za vse ključe v SQL tabeli, ki je bila izpeljana iz razmerja tipa N-N, ker so smiselne samo celotne vrstice

Binarna razmerja

- Pogledali si bomo primere binarnih razmerij glede na števnost razmerij
- Kako prevedemo entitete? kako prevedemo binarno razmerje? kako prevedemo obvezno oz. opcijsko razmerje? katere SQL gradnike uporabljamo?
- Imamo tri tipe razmerij glede na strukturo: 1-1, 1-N in N-N
- Vsaka vloga v razmerju je lahko opcijska ali mandatorna

Razmerje 1-1 (2 x mandatorno)




```
create table report
   (report_no integer,
    report_name varchar(256),
    primary key(report_no);

create table abbreviation
    (abbr_no char(6),
     report_no integer not null unique,
     primary key (abbr_no),
     foreign key (report_no) references report
     on delete cascade on update cascade);
```

Razmerje 1-1 (1 x mandatorno)


```
create table employee
    (emp_id char(10),
     emp_name char(20),
     primary key (emp_id));
```

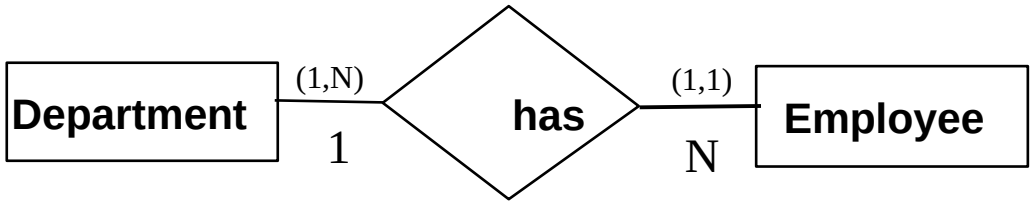
```
create table department
    (dept_no integer,
     dept_name char(20),
     mgr_id char(10) not null unique,
     primary key (dept_no),
     foreign key (mgr_id) references employee
          on delete set default on update cascade);
```

Razmerje 1-1 (2 x opcijsko)


```
create table engineer
   (emp_id char(10),
    desktop_no integer,
    primary key (emp_id));
```

```
create table desktop
   (desktop_no integer,
    emp_id char(10),
    primary key (desktop_no),
    foreign key (emp_id) references engineer
    on delete set null on update cascade);
```

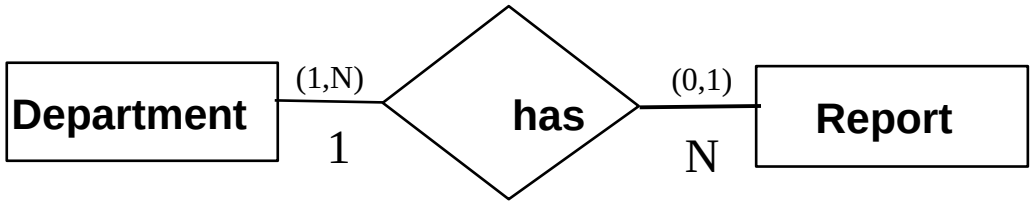
Razmerje 1-N (2 x mandatorno)




```
create table department
    (dept_no integer,
     dept_name char(20),
     primary key (dept_no));
```

```
create table employee
    (emp_id char(10),
     emp_name char(20),
     dept_no integer not null,
     primary key (emp_id),
     foreign key (dept_no) references department
        on delete set default on update cascade);
```

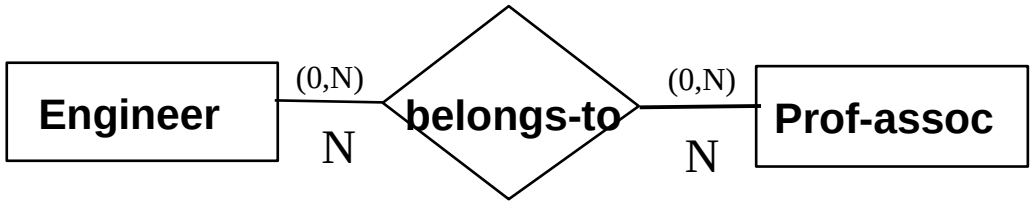
Razmerje 1-N (1 x mandatorno)




```
create table department
    (dept_no integer,
     dept_name char(20),
     primary key (dept_no));
```

```
create table report
    (report_no integer,
     dept_no integer,
     primary key (report_no),
     foreign key (dept_no) references department
        on delete set null on update cascade);
```

Razmerje N-N (2 x opcijsko)

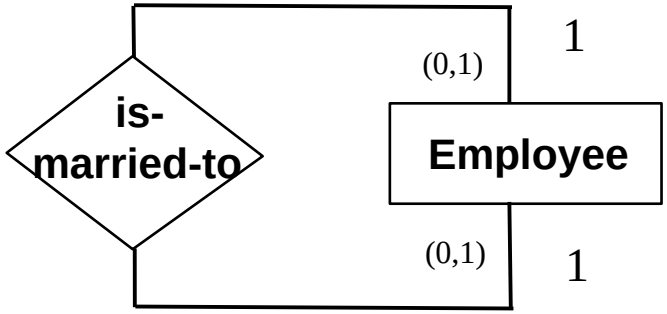



```
create table engineer
    (emp_id char(10),
     primary key (emp_id));

create table prof_assoc
    (assoc_name varchar(256),
     primary key (assoc_name));

create table belongs_to
    (emp_id char(10),
     assoc_name varchar(256),
     primary key (emp_id, assoc_name),
     foreign key (emp_id) references engineer
        on delete cascade on update cascade,
     foreign key (assoc_name) references prof_assoc
        on delete cascade on update cascade);
```

Rekurzivna razmerja

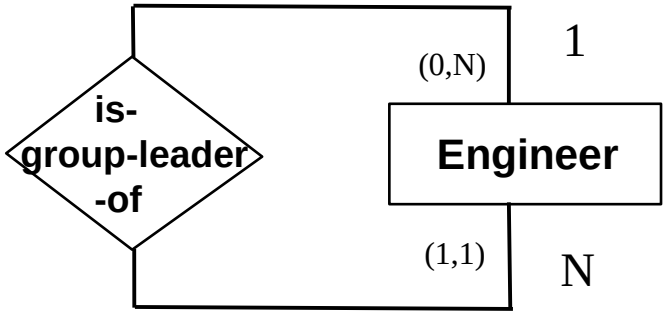
- Binarno razmerje na eni sami entiteti zahteva definicijo parov primerkov iste entitete
- Pari so lahko popolnoma opcijski ali na drugi strani obvezni
- Pari entitet so definirani s tujimi ključi v tabeli s katero je predstavljeno rekurzivno razmerje
- Pari so lahko definirani v svoji tabeli ali so znotraj tabele entitete odvisno od tipa razmerja

Rekurzivno razmerje 1-1 (2 x opcijsko)



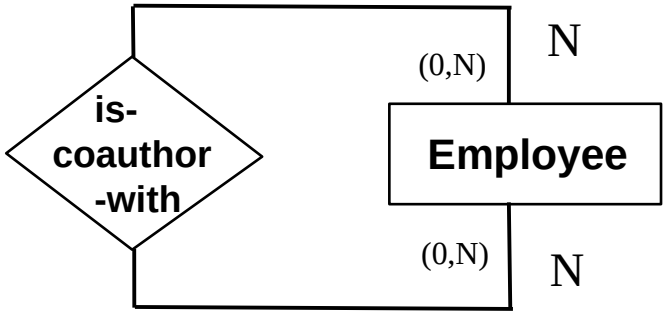

```
create table employee
    (emp_id char(10),
      emp_name char(20),
      spouse_id char(10),
      primary key (emp_id),
      foreign key (spouse_id) references employee
      on delete set null on update cascade);
```

Rekurzivno razmerje 1-N (1 x opcijsko)




```
create table engineer
   (emp_id char(10),
    leader_id char(10) not null,
    primary key (emp_id),
    foreign key (leader_id) references engineer
        on delete set default on update cascade);
```

Rekurzivno razmerje N-N (2 x opcijsko)

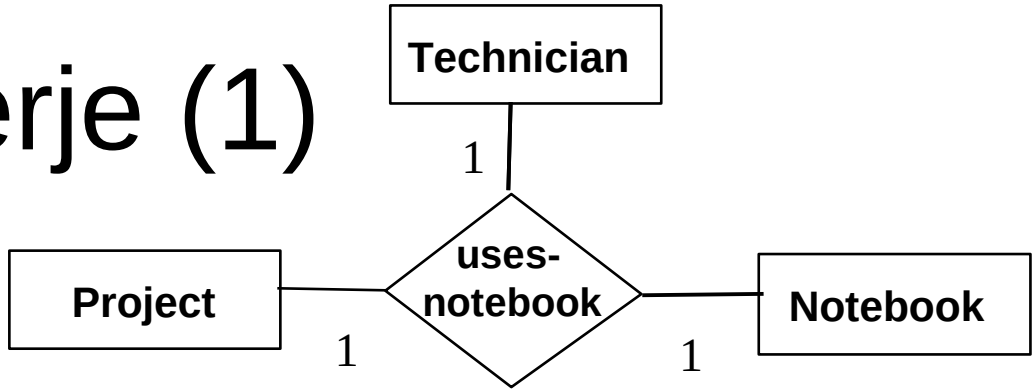



```
create table employee
   (emp_id char(10),
    emp_name char(20),
    primary key (emp_id));

create table coauthor
    (author_id char(10),
     coauthor_id char(10),
     primary key (author_id, coauthor_id),
     foreign key (author_id) references employee
        on delete cascade on update cascade,
     foreign key (coauthor_id) reference employee
        on delete cascade on update cascade);
```

Ternarna razmerja

- Ogledali si bomo nekatere primere
- Potrebno je poznati omejitve
- Kluči v ternarnem razmerju so vedno NOT NULL
- Ključi se morajo brisati kaskadno

Ternarno razmerje (1)



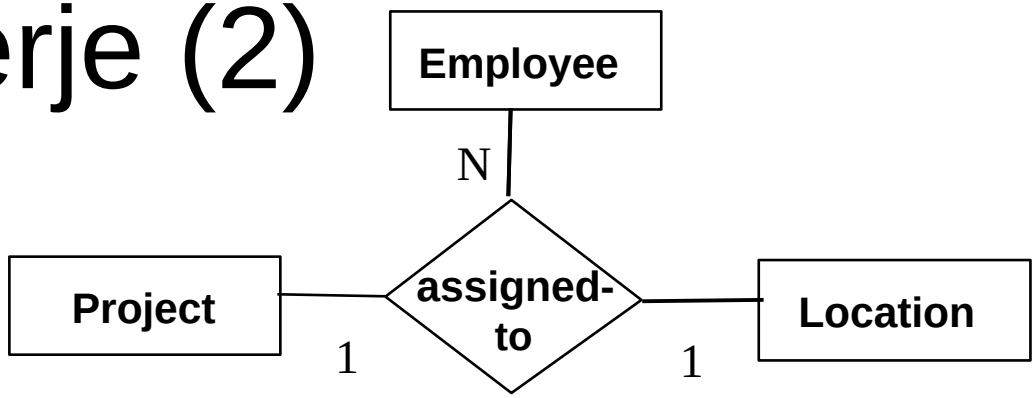

```
create table technician (
    emp_id char(10),
    primary key (emp_id));
create table project (
    project_name char(20),
    primary key (project_name));
create table notebook (
    notebook_no integer,
    primary key (notebook_no));
```

```
create table uses_notebook (
    emp_id char(10),
    project_name char(20),
    notebook_no integer not null,
    primary key (emp_id, project_name),
    foreign key (emp_id) references technician
      on delete cascade on update cascade,
    foreign key (project_name) references project
      on delete cascade on update cascade,
    foreign key (notebook_no) references notebook
      on delete cascade on update cascade,
    unique (emp_id, notebook_no),
    unique (project_name, notebook_no));
```

(1-1-1) – en projekt, en tehnik,
en notebook

emp_id, project_name → notebook_no
emp_id, notebook_no → project_name
project_name, notebook_no → emp_id

Ternarno razmerje (2)




```
create table employee (
   emp_id char(10),
   emp_name char(20),
   primary key (emp_id));
create table project (
   project_name char(20),
   primary key (project_name));
create table location (
   loc_name char(15),
   primary key (loc_name));
```

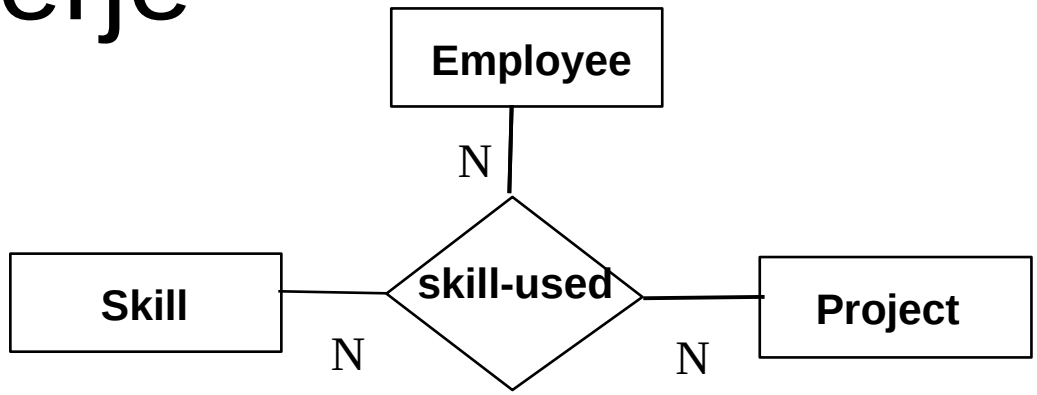
```
create table assigned_to (
   emp_id char(10),
   project_name char(20),
   loc_name char(15) not null,
   primary key (emp_id, project_name),
   foreign key (emp_id) references employee
      on delete cascade on update cascade,
   foreign key (project_name) references project
      on delete cascade on update cascade,
   foreign key (loc_name) references location
      on delete cascade on update cascade,
   unique (emp_id, loc_name));
```

(1-1-N) – en projekt, ena lokacija, več zaposlenih

emp_id, loc_name → project_name

emp_id, project_name → loc_name

Ternarno razmerje (N-N-N)




```
create table employee (
    emp_id char(10),
    emp_name char(20),
    primary key (emp_id));
create table skill (
    skill_type char(15),
    primary key (skill_type));
create table project (
    project_name char(20),
    primary key (project_name));
```

```
create table skill_used (
    emp_id char(10),
    skill_type char(15),
    project_name char(20),
    primary key (emp_id, skill_type, project_name),
    foreign key (emp_id) references employee
        on delete cascade on update cascade,
    foreign key (skill_type) references skill
        on delete cascade on update cascade,
    foreign key (project_name) references project
        on delete cascade on update cascade);
```

(N-N-N) – Zmožnosti so poljubno povezane z zaposlenimi in projekti

Generalizacija

- Prevod generalizacijske hierarhije sestavljene iz supertipa in podtipov lahko vodi do več SQL tabel
- Tabela izpeljana iz supertipa vsebuje ključ entitete supertipa in vse skupne atribute hierarhije
- Vsaka tabela izpeljana iz entitet podtipa vsebuje ključ entitete supertipa in samo atribute specifične za podtip
- Integriteto pri popravljanju je potrebno ohranjati tako, da vsi popravki, vstavljanja in brisanja spreminjajo tako tabelo super tipa kot tudi pripadajočo tabelo podtipa
  - uporabiti je potrebno omejitev cascade za tuje ključe
  - popravek ključa supertipa zahteva popravek v hierarhiji
  - popravek opisnega atributa zahteva popravek samo enega zapisa

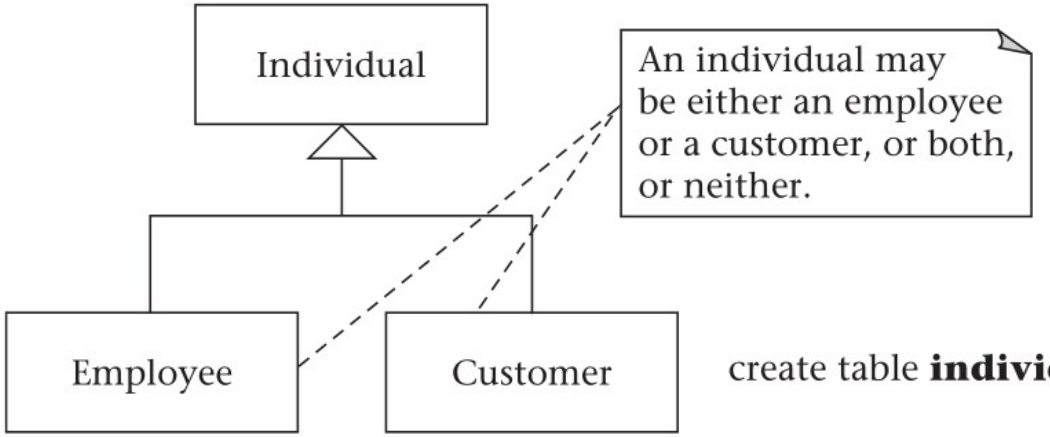
Generalizacija (2)

- Transformacijska pravila so ista za primer prekrivanja kot tudi za disjunktne podtipe
- Drugi pristop je definicija ene same tabele, ki vsebuje vse atribute tako podtipa kot tudi nadtipa
  - celotna hierarhija v eni tabeli
  - po potrebi uporabimo null vrednosti
- Tretji pristop je ena tabela za en podtip vendar porinemo skupne atribute navzdol k tabelam podtipov
- Imamo prednosti in slabosti teh treh pristopov
- Programska orodja za načrtovanje navadno uporabljajo vse tri možnosti

Generalizacija (3)

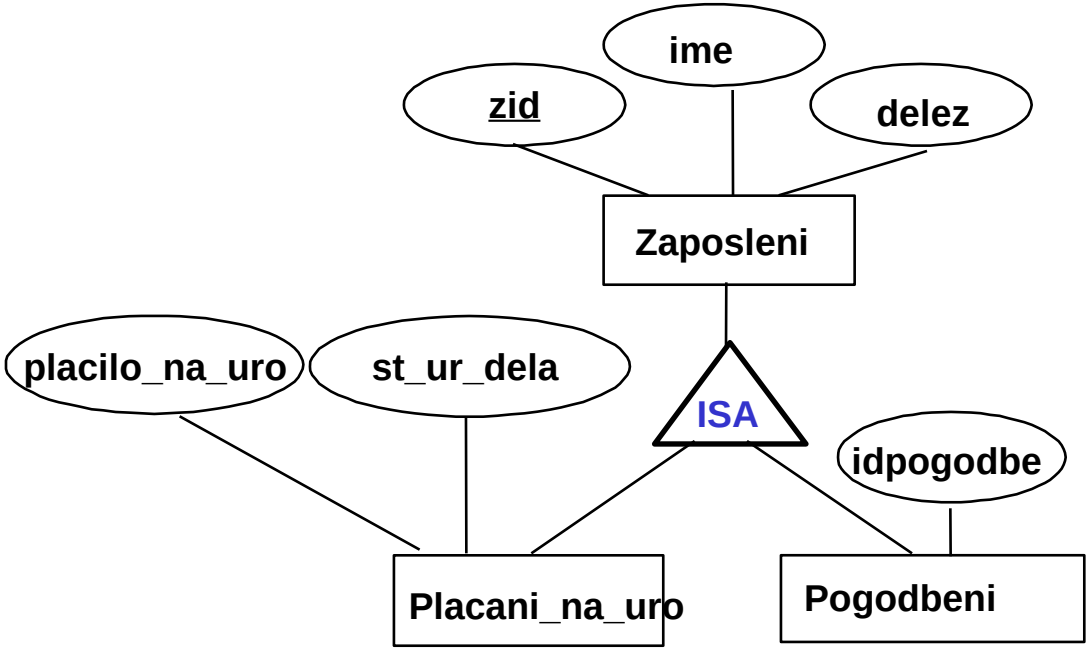
- Praktični sistemi velikokrat uporabijo klasifikacijski atribut v nadtipu s katerim lahko ločimo med instancami podtipov
- Če imamo več-nivojsko hierarhijo potem lahko uporabimo več klasifikacijskih atributov
- Pristop kombinira sistem tipov s klasifikacijsko hierarhijo definirano glede na vrednost izbranih atributov

Primer: Generalizacijska hierarhija




```
create table individual (indiv_id char(10),
                         indiv_name char(20),
                         indiv_addr char(20),
                         primary key (indiv_id));
create table employee (emp_id char(10),
                       job_title char(15),
                       primary key (emp_id),
                       foreign key (emp_id) references individual
                          on delete cascade on update cascade);
create table customer (cust_no char(10),
                       cust_credit char(12),
                       primary key (cust_no),
                       foreign key (cust_no) references individual
                          on delete cascade on update cascade);
```

Primer



- *Dedovanje !*
  - *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 generalizacije v relacije

A 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.

B Alternativa: Samo tabeli Placani_na_uro in Pogodbeni.

- *Placani_na_uro: <u>zid</u>, ime, delez, placilo_na_uro, st_ur_dela.*
- Vsak zaposleni mora biti v enem izmed podrazredov sicer se podvajajo podatki.