Projektovanje baza podataka

Logičko modeliranje

19. maj 2026.

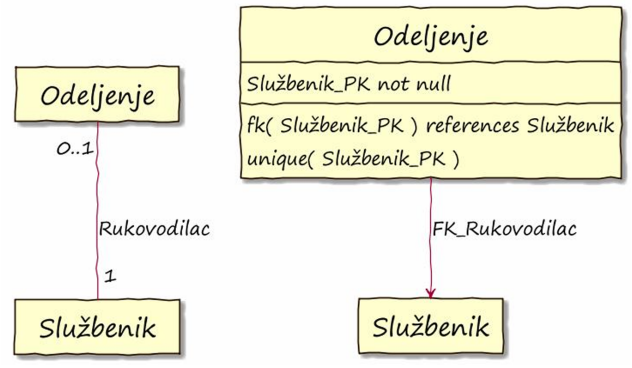
Sadržaj

Napomena

Slajdovi su nastali obradom slajdova prof. dr Saše Malkova za predmet Projektovanje baza podataka

Sadržaj

Primer odnosa 0..1 – 1

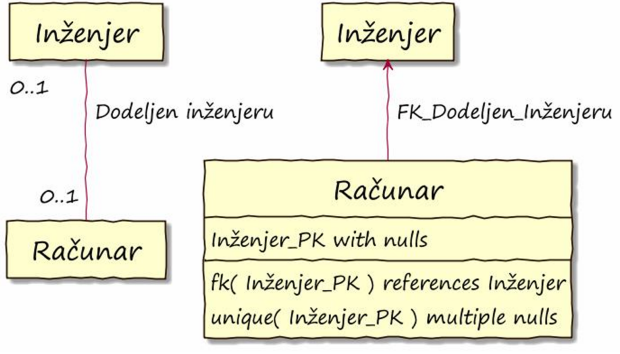


Primer odnosa 0..1 – 1 (2)

- Svako odeljenje ima rukovodioca
- Svaki zaposleni može da bude rukovodilac najviše jednog odeljenja

```
create table department (
    dept_no integer not null,
    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
);

create table employee (
    emp_id char(10) not null,
    emp_name char(20),
    primary key( emp_id )
);
```

Primer odnosa 0..1 – 0..1



Primer odnosa 0..1 – 0..1

- Neki računari su dodeljeni inženjerima
- Nekim inženjerima su dodeljeni računari

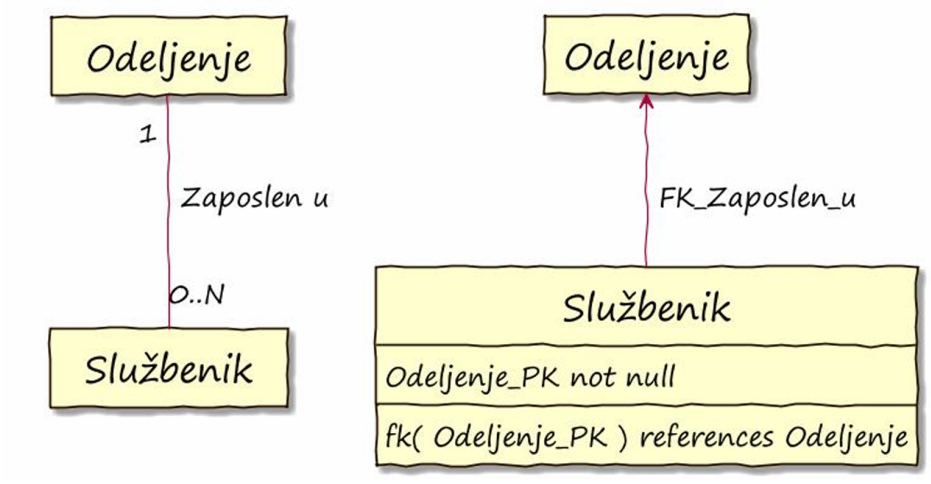
```
create table engineer (
    emp_id char(10) not null,
    ...
    primary key( emp_id )
);

create table desktop (
    desktop_no integer not null,
    emp_id char(10) unique, - potencijalan problem !!!
    primary key( desktop_no ),
    foreign key( emp_id )
        references engineer
        on delete set null
        on update cascade
);
```

Primer odnosa 0..1 – 0..1

- Rešenje za jedinstvenost atributa sa više dopuštenih nedefinisanih vrednosti:

```
create table engineer ( ... );

create table desktop (
    ...
    emp_id char(10),
    ...
);

create unique index desktop_emp
on desktop( emp_id )
exclude null keys};
```

Primer odnosa 1 – 0..N

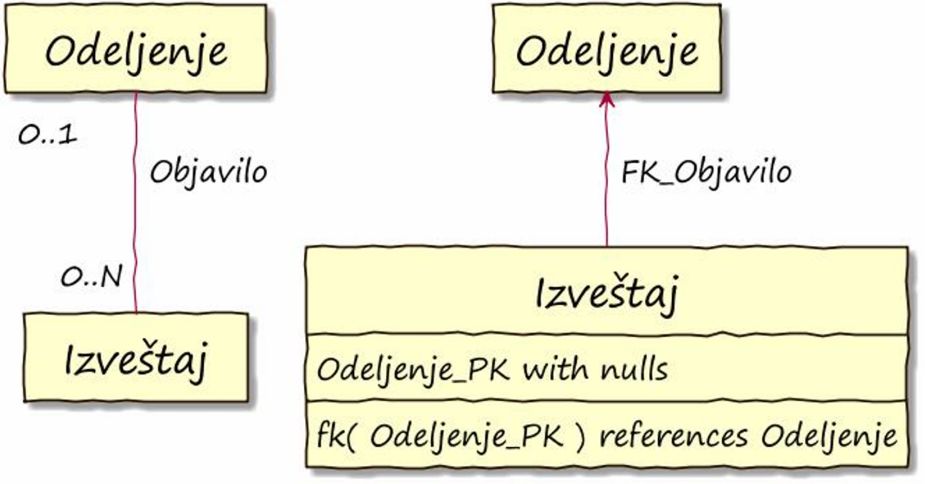


Primer odnosa 1 – 0..N

- Svaki službenik radi u tačno jednom odeljenju
- Svako odeljenje ima bar jednog zaposlenog

```
create table department (
    dept_no integer not null,
    dept_name char(20) not null,
    primary key( dept_no )
);

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

Primer odnosa 0..1 – 0..N

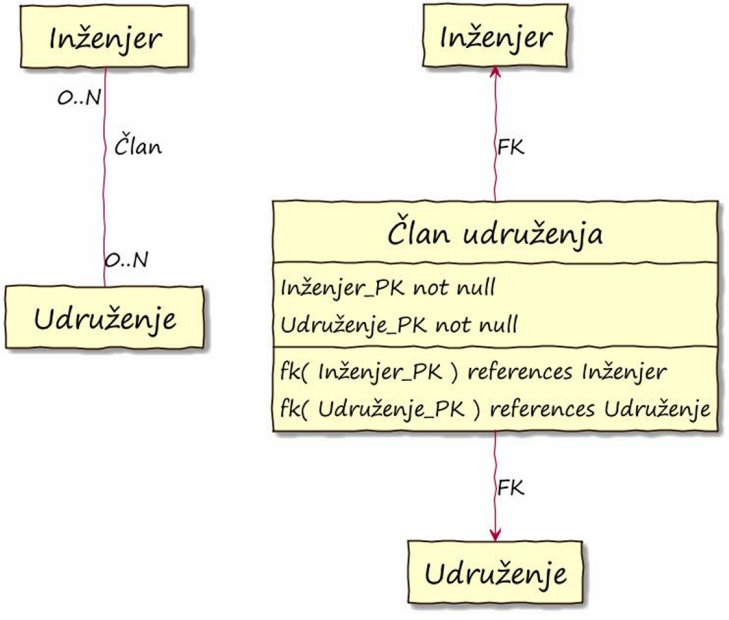


Primer odnosa 0..1 – 0..N

- Svako odeljenje objavljuje jedan ili više izveštaja
- Izveštaj može ali ne mora da bude objavljen u nekom odeljenju

```
create table department (
    dept_no integer not null,
    dept_name char(20) not null,
    primary key( dept_no )
);

create table report (
    report_no integer not null,
    dept_no integer,
    primary key( report_no ),
    foreign key( dept_no )
        references department}
        on delete set null
        on update cascade
);
```

Primer odnosa 0..N – 0..N



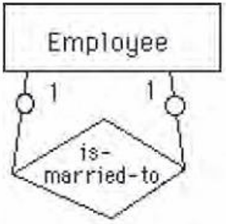
Primer odnosa 0..N – 0..N

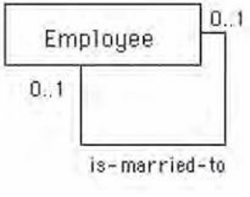
- Svako udruženje može da ima 0 ili više inženjera
- Svaki inženjer može da bude član 0 ili više udruženja

```
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) not null,
    assoc_name varchar(256) not null,
    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
);
```

Binarni ciklični odnosi 1-1 (0..1-1, 0..1-0..1)

- Bilo da su opcioni ili ne
  - predstavljaju se dodatnim atributima stranog ključa
  - ako je opcioni, sme da bude NULL

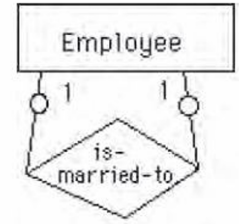


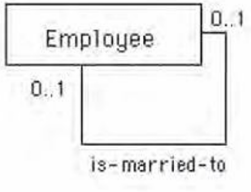


Binarni ciklični odnosi 1-1

- Svaki zaposleni može da bude u braku sa drugim zaposlenim u kompaniji (ali najviše sa jednim)

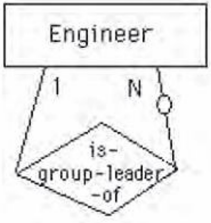
```
create table employee (
    emp_id char(10) not null,
    emp_name char(20),
   spouse_id char(10),
    primary key( emp_id ),
    foreign key( spouse_id )
       references employee
       on delete set null
       on update cascade
);
```

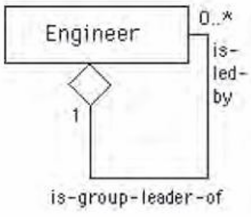




Binarni ciklični odnosi 1-*

- Ako je odnos 1-*
  - strani ključ se uvodi na strani "*"

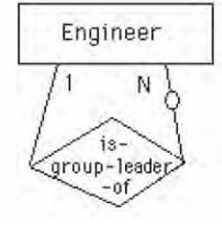




Binarni ciklični odnosi 1-*

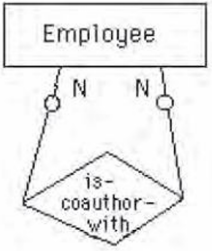
- Inženjeri se grupišu u timove
- Svaki tim ima tačno jednog lidera

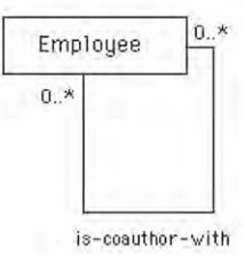
```
create table engineer (
    emp_id char(10) not null,
    leader_id char(10) not null,
    primary key( emp_id ),
    foreign key( leader_id )
       references engineer
       on delete set default
       on update cascade
);
```



Binarni ciklični odnosi *-*

- Ako je odnos *-*
  - predstavlja se novom relacijom
  - kao i obični binarni odnosi *-*

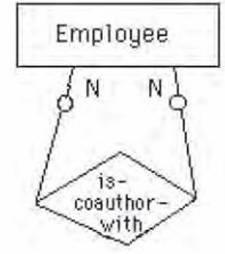




Binarni ciklični odnosi *-*

- Svaki službenik može da piše izveštaj sam ili sa koautorom

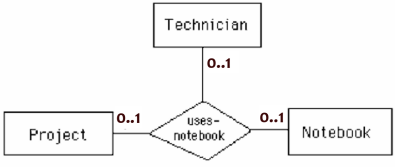
```
create table employee (
    emp_id char(10) not null,
    emp_name char(20) not null,
    primary key( emp_id )
);

create table coauthor (
    author_id char(10) not null,
    coauthor_id char(10) not null,
    primary key( author_id, coauthor_id ),
    foreign key( author_id )
        references employee
        on delete cascade
        on update cascade,

    foreign key( coauthor_id )
        references employee
        on delete cascade
        on update cascade
);
```



Odnosi sa više učesnika 1-1-1

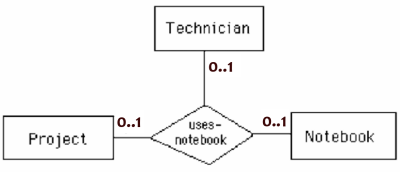
- Nova relacija sa stranim ključevima
  - zavisnosti se uređuju dopuštanjem NULL i jedinstvenim ključevima
- Primer:
  - Tehničaru je dodeljen najviše jedan računar na nekom od projekta
  - Svaki računar pripada najviše jednom paru (tehničar, projekat)
  - Na svakom projektu se dodeljuje najviše jedan računar
  - Zavisnosti:
    - tehničar $\rightarrow$ projekat, računar
    - računar $\rightarrow$ tehničar, projekat
    - projekat $\rightarrow$ tehničar, računar

Prema formalnoj i ispravnoj notaciji kardinalnosti u dijagramima ER, broj "x" pored Projekta označava da jedan projekat učestvuje u "x" odnosa.



Odnosi sa više učesnika 1-1-1

```
create table project (
    project_name char(20) not null,
    primary key( project_name )
);

create table notebook (
    notebook_no integer not null,
    primary key( notebook_no )
);

create table uses_notebook (
    emp_id char(10) not null,
    project_name char(20) not null,
    notebook_no integer not null,
    primary key( emp_id ),
    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( project_name ),
    unique( notebook_no )
);
```



Odnosi sa više učesnika 1-1-1, alt.

- Ako se kardinalnosti tumače kao u dijagramu klasa, onda je semantika potpuno izmenjena:

Primer:

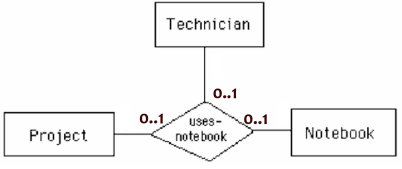
- Na jednom projektu, neki računar može da se dodeli najviše jednom tehničaru
- Jedan računar može da se dodeli jednom tehničaru na najviše jednom projektu
- Na jednom projektu svaki tehničar može da koristi najviše jedan računar

Zavisnosti:

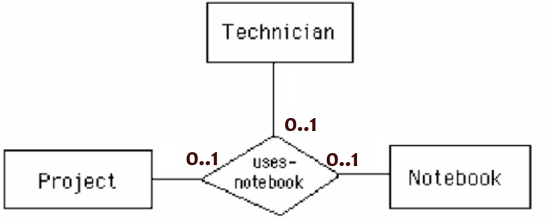
- tehničar, projekat $\rightarrow$ računar
- tehničar, računar $\rightarrow$ projekat
- projekat, računar $\rightarrow$ tehničar

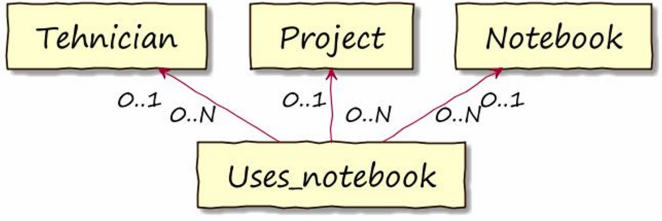
Korišćenjem alternativne notacije kardinalnosti, potpuno se menja smisao odnosa.
Prema alternativnoj notaciji, broj "x" pored Projekta označava da jedan par (tehničar, računar) učestvuje u "x" odnosa.
Ako se koristi takvo označavanje, to mora da se nekako naglasi:

- ili komentarom
- ili navođenjem broja uz odnos



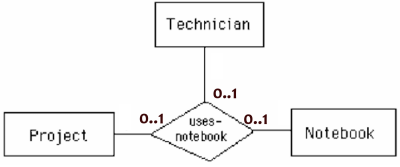
Poređenje sa notacijom dijagrama klasa





Odnosi sa više učesnika 1-1-1, alt.

```
create table project (
    project_name char(20) not null,
    primary key( project_name )
);

create table notebook (
    notebook_no integer not null,
    primary key( notebook_no )
);

create table uses_notebook (
    emp_id char(10) not null,
    project_name char(20) not null,
    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 )
);
```

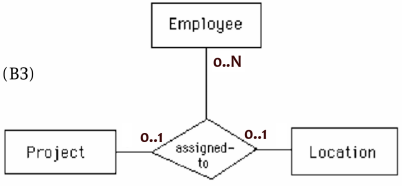


Odnosi sa više učesnika 1-1-*

- Nova relacija sa stranim ključevima
  - zavisnosti se uređuju dopuštanjem NULL i jedinstvenim ključevima
- Primer:
  - Svaki službenik može da radi na više projekata i lokacija
  - Na svakoj lokaciji službenik radi na najviše jednom projektu
  - Na svakom projektu službenik radi na najviše jednoj lokaciji
- Zavisnosti:
  - zaposleni, lokacija $\rightarrow$ projekat
  - zaposleni, projekat $\rightarrow$ lokacija
  - projekat, lokacija $\rightarrow$ zaposleni (VZ)

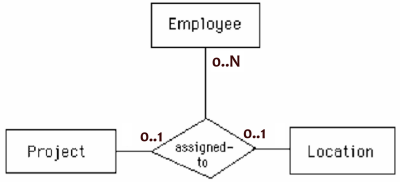
Koristimo kombinovanu notaciju.

Obratite pažnju na položaj brojeva koji određuju kardinalnosti.



Odnosi sa više učesnika 1-1-*

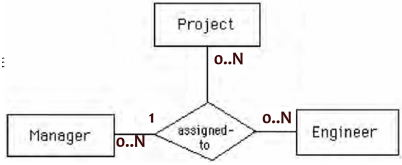
```
create table employee (
    emp_id char(10) not null,
    emp_name char(20),
    primary key( emp_id )
);
create table project (
    project_name char(20) not null,
    primary key( project_name )
);
create table location (
    loc_name char(15) not null,
    primary key( loc_name )
);
create table assigned_to (
    emp_id char(10) not null,
    project_name char(20) not null,
    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 )
);
```



Odnosi sa više učesnika 1-*-*

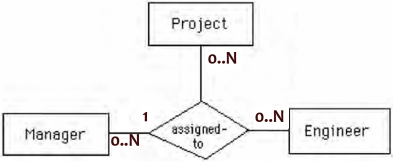
- Nova relacija sa stranim ključevima
  - zavisnosti se uređuju izborom primarnog ključa
- Primer:
  - Svaki inženjer na svakom projektu ima tačno jednog rukovodioca
  - Projekat može da ima više rukovodilaca
  - Rukovodilac može da vodi više projekata
  - Inženjer može da ima više rukovodilaca na različitim projektima
- Zavisnosti:
  - zaposleni, projekat $\rightarrow$ rukovodilac

Koristimo kombinovanu notaciju. Obratite pažnju na položaj brojeva koji određuju kardinalnosti.



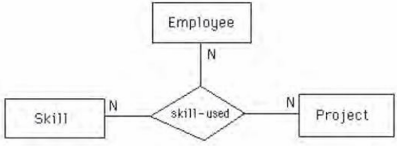
Odnosi sa više učesnika 1-*-*

```
create table project (
    project_name char(20) not null,
    primary key( project_name )
);
create table manager (
    mgr_id char(10) not null,
    primary key( mgr_id )
);
create table engineer (
    emp_id char(10) not null,
    primary key( emp_id )
);

create table manages (
    project_name char(20) not null,
    mgr_id char(10) not null,
    emp_id char(10) not null,
    primary key( project_name, emp_id ),
    foreign key( project_name )
        references project
        on delete cascade
        on update cascade,
    foreign key( mgr_id )
        references manager
        on delete cascade
        on update cascade,
    foreign key( emp_id )
        references engineer
        on delete cascade
        on update cascade
);
```



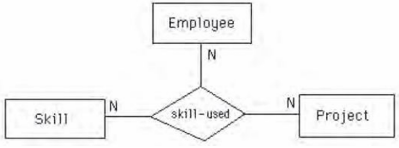
Odnosi sa više učesnika *-*-*

- Nova relacija sa stranim ključevima
- Primer:
  - Službenik može da koristi različite veštine na svakom od projekata
  - Svaki projekat može da ima više zaposlenih sa više različitih ili ponovljenih veština
- Zavisnosti:
  - samo višeznačne



Odnosi sa više učesnika *-*-*

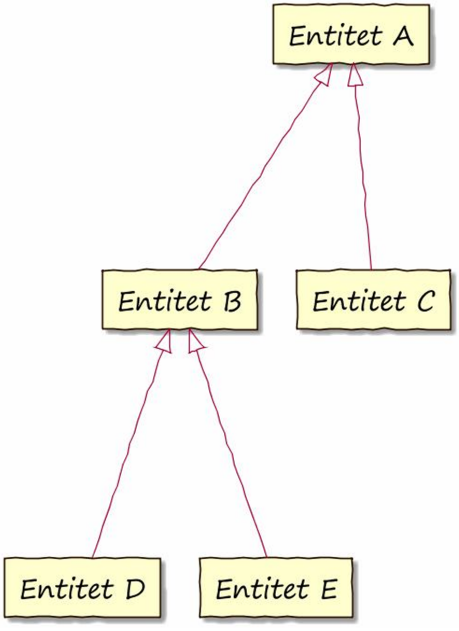
```
create table employee (
    emp_id char(10) not null,
    emp_name char(20),
    primary key( emp_id )
);
create table skill (
    skill_type char(15) not null,
    primary key( skill_type )
);
create table project (
    project_name char(20) not null,
    primary key( project_name )
);
create table skill_used (
    emp_id char(10) not null,
    skill_type char(15) not null,
    project_name char(20) not null,
    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
);
```



Prevođenje hijerarhija entiteta

- Hijerarhije entiteta se obično prevode na neki od tri osnovna načina:
  - Svaki entitet u posebnu relaciju
    - praktično kao da se ne radi o odnosu specijalizacije nego o nekom drugom odnosu
  - Svaki entitet-list u posebnu relaciju, ali tako da uključi **sve** nasleđene atribute
  - Cela hijerarhija u jednu relaciju
- Moguće je i kombinovanje metoda, za različite delove hijerarhije

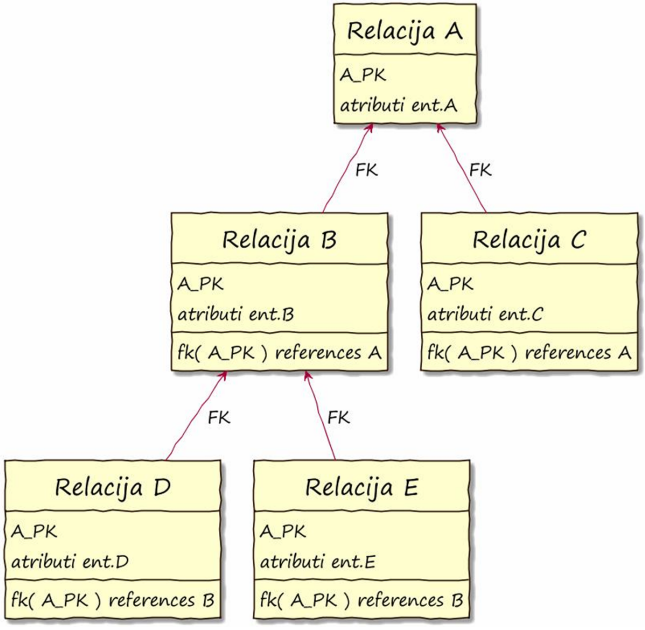
Primer hijerarhije



Prevođenje hijerarhija – Svaki entitet posebno

- Za svaki entitet se pravi po jedna relacija, sa stranim ključem na neposrednu baznu relaciju
  - svaka relacija obuhvata samo one atribute koji su za nju specifični
    - ...i još atribute ključa bazne relacije, koji se koriste kao strani ključ
  - praktično se specijalizacija tretira kao asocijacija ili agregacija
- Ovaj pristup zahteva česta spajanja pri čitanju i potencijalno je neefikasan
  - ali je na nivou logičkog modela prihvatljiv

Prevođenje hijerarhija – Svaki entitet posebno



Prevođenje hijerarhija – Svaki entitet posebno

Prekrivanje

- ako je hijerarhija **bez prekrivanja**, sve je u redu
- ako je hijerarhija **sa prekrivanjem** onda je potrebno da se obezbedi dodatno sredstvo za proveru pokrivenosti
  - tj. da svaki red baznog entiteta ima odgovarajući red u nekom listu

Preklapanje

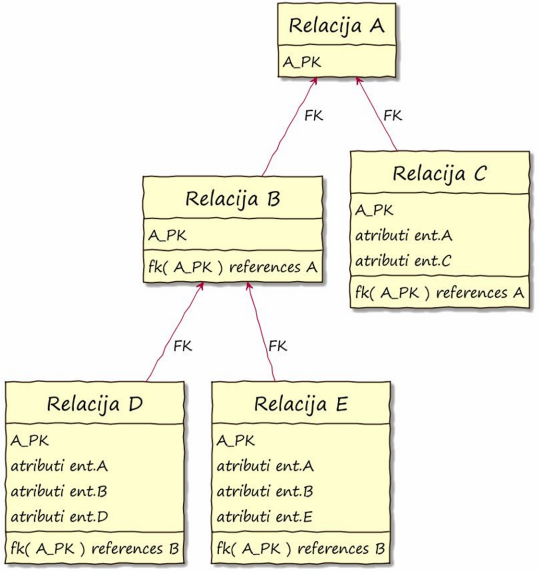
- Ako je hijerarhija **bez preklapanja** onda je potrebno da se obezbedi dodatno sredstvo za proveru da nema preklapanja
  - tj. da se jedan red baznog entiteta ne referiše iz više "izvedenih" entiteta
- Ako je hijerarhija **sa preklapanjem**, sve je u redu

Prevođenje hijerarhija – Svaki list posebno

- Za svaki entitet-list se pravi po jedna relacija, sa stranim ključem na neposrednu baznu relaciju
  - Svaka relacija obuhvata atribute koji su za nju specifični
    - ...i **SVE** atribute **SVIH** baznih klasa
    - ... atributi ključa bazne relacije se koriste kao strani ključ
- Za entitet koji nije list:
  - relacija se izostavlja ili ima samo primarni ključ
  - pravi se pogled koji se računa kao unija listova
- Upotreba pojedinačnih entiteta-listova je efikasna
- Pretraživanje baznog skupa entiteta (ili bilo kog entiteta koji nije list) je neefikasno (unija)

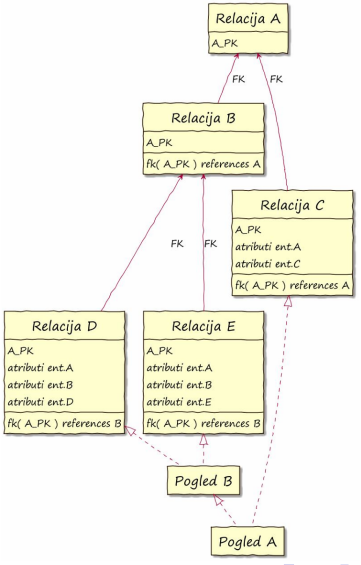
Prevođenje hijerarhija – Svaki list posebno

- U relacijama koje odgovaraju listovima navode se svi atributi
- U relacijama koje odgovaraju apstraktnim klasama navode se samo primarni ključevi



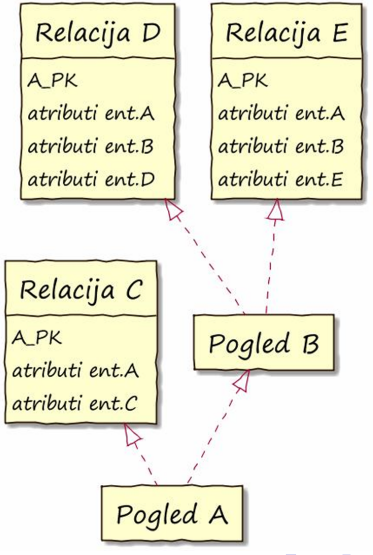
Prevođenje hijerarhija – Svaki list posebno

- Apstraktni entiteti se dodatno modeliraju pogledima
  - svaki pogled je unija odgovarajućih relacija



Prevođenje hijerarhija – Svaki list posebno

- Tehnički gledano, za apstraktne entitete ni ne moraju da se prave relacije
- U praksi su ipak potrebne radi staranja o integritetu, posebno u slučajevima bez preklapanja
- ako ne postoje dodatne relacije i strani ključevi, onda je otežana provera da li postoje redovi sa istim ID u različitim tabelama



Prevođenje hijerarhija – Svaki list posebno

Prekrivanje

- ako je hijerarhija **bez prekrivanja** onda se praktično svodi na prethodni slučaj, zato što svaka klasa može da bude list
  - prave "unutrašnje" klase su samo one koje su apstraktne
- ako je hijerarhija **sa prekrivanjem** onda je potrebno da se obezbedi dodatno sredstvo za proveru pokrivenosti
  - tj. da svaki red baznog entiteta ima odgovarajući red u nekom listu
  - može da se izbegne izostavljanjem relacija apstraktnih entiteta

Preklapanje

- ako je hijerarhija **bez preklapanja** onda je potrebno da se obezbedi dodatno sredstvo za proveru da nema preklapanja
  - tj. da se jedan red baznog entiteta ne referiše iz više različitih "izvedenih" entiteta
- ako je hijerarhija **sa preklapanjem** onda imamo redundantnost
  - nasleđeni delovi entiteta mogu da se redundantno ponavljaju u više listova...

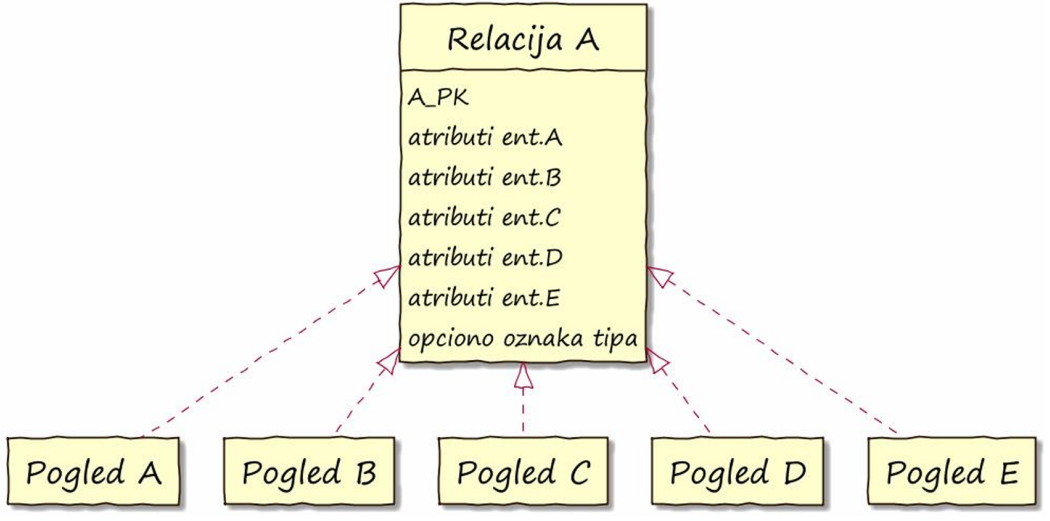
Prevođenje hijerarhija – Sve u jednu relaciju

- Jedna relacija modelira celu hijerarhiju
  - Obuhvata sve atribute koji postoje u hijerarhiji
  - Za svaki entitet se koristi samo deo atributa...
  - ...ostali imaju nedefinisane (prazne) vrednosti
    - što može da se rešava pravilima integriteta
- Pripadanje vrste entitetu se proverava na osnovu sadržaja atributa
  - ili na osnovu postojećih atributa
  - ili se dodaju surogat-atributi kao oznake tipa
- Relativno efikasna upotreba
- Potencijalno neefikasno zauzeće prostora
  - noviji SUBP to relativno dobro rešavaju

Prevođenje hijerarhija – Sve u jednu relaciju

- Pripadanje vrste entitetu se proverava na osnovu sadržaja atributa
  - ili se dodaje surogat-atribut kao oznaka entiteta (tipa)
    - upiti nad izvedenim entitetima se svode na restrikciju po oznaci tipa
  - ili na osnovu postojećih atributa
    - upiti nad izvedenim entitetima se svode na restrikciju po specifičnim atributima entiteta, tj. proverava se da nisu NULL

Prevođenje hijerarhija – Sve u jednu relaciju



Prevođenje hijerarhija – Sve u jednu relaciju

Prekrivanje

- ako je hijerarhija **bez prekrivanja** onda je sve u redu
- ako je hijerarhija **sa prekrivanjem** onda se dodaju provere
  - ili se zabrani da oznaka tipa odgovara apstraktnoj klasi
  - ili se zahteva da su atributi nekog lista definisani

Preklapanje

- ako je hijerarhija **bez preklapanja...**
- ako je hijerarhija **sa preklapanjem** onda je otežano proveravanje pripadnosti i integriteta, usložnjava se upotreba atributa za označavanje entiteta reda
  - jedan red potencijalno odgovara većem broju entiteta
  - može da se uvede po jedan binarni atribut pripadnosti za svaki entitet
  - provera ne može da bude: „kom entitetu pripada red?"
  - nego samo: „da li red pripada datom entitetu?"

Prevođenje hijerarhija – Rezime varijanti

Karakteristika	**Samo jedna relacija**	**Svaki list posebno**
Ažuriranje	Samo jedna relacija	Ažuriranje i baznih relacija
Čitanje	Efikasno, restrikcija	Listovi efikasno, ostalo pomoću unije
Sa preklapanjem	+ (ali uz složenije ustanovljavanje entiteta)	Redundantnost
Bez preklapanja	Dodatne provere	Dodatne provere
Sa prekrivanjem	Dodatne provere	+ (ako su svi unutrašnji čvorovi apstraktni)
Bez prekrivanja	+	Nema smisla, isto kao prethodno

Poređenje različitih pristupa prevođenju hijerarhija.

Prevođenje hijerarhija – Kombinovano

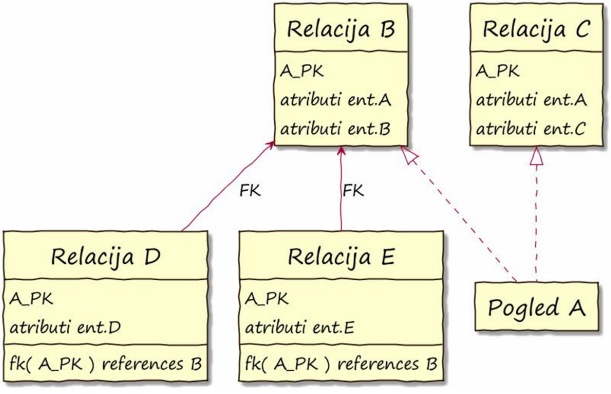
- Različiti metodi mogu da se kombinuju u različitim segmentima iste hijerarhije

Na primer:

- jedan deo hijerarhije se modelira samo jednom relacijom, a ostatak dodatnim relacijama
- ili se kombinuje pristup sa relacijama za sve entitete i samo za listove

Prevođenje hijerarhija – Primer kombinovane implementacije

- Klase A, B i C su modelirane tako što su napravljene relacije samo za listove hijerarhije
- Klase D i E su implementirane kao dodatne relacije
- Ovakav model je dobar ako je samo A apstraktna klasa
- Ima prednosti ako može da bude preklapanja između D i E, kao i između B i C

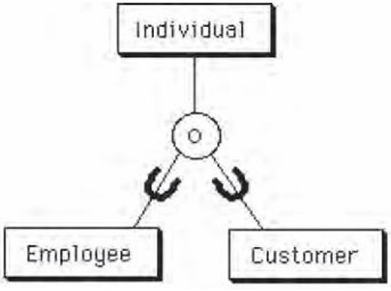


Hijerarhija – primer

- Već smo videli da može na različite načine:
  - Cela hijerarhija u jednu relaciju
  - Svaki entitet u posebnu relaciju, tj. kao da se radi o "običnom" odnosu entiteta
  - Svaki entitet-list u posebnu relaciju, ali tako da uključi sve nasleđene atribute

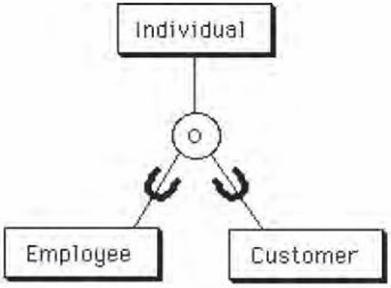
Primer:

- Osoba može da bude službenik ili mušterija ili oboje ili ništa od toga



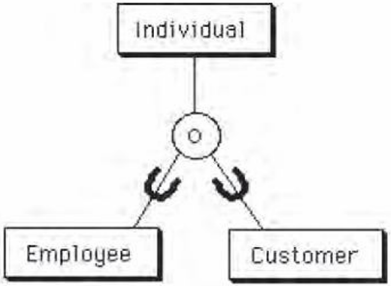
Hijerarhija – svaki entitet u relaciju

```
create table individual (
    indiv_id char(10) not null,
    indiv_name char(20),
    indiv_addr char(20),

    primary key( indiv_id )
);

create table employee (
    emp_id char(10) not null,
    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) not null,
    cust_credit char(12),

    primary key( cust_no ),

     foreign key( cust_no )
         references individual
        on delete cascade
        on update cascade
);
```



Hijerarhija – listovi u relacije

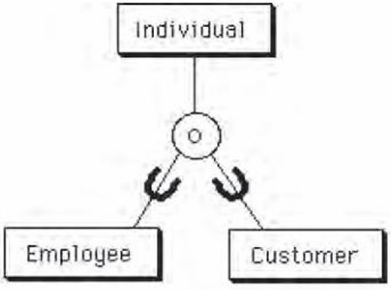
```
create table individual (
    indiv_id char(10) not null,

    primary key( indiv_id )
);

create table employee (
    emp_id char(10) not null,
     indiv_name char(20),
     indiv_addr char(20),
    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) not null,
     indiv_name char(20),
     indiv_addr char(20),
    cust_credit char(12),

    primary key( cust_no ),

    foreign key( cust_no )
        references individual
        on delete cascade
        on update cascade
);
```



Hijerarhija – listovi u relacije (2)

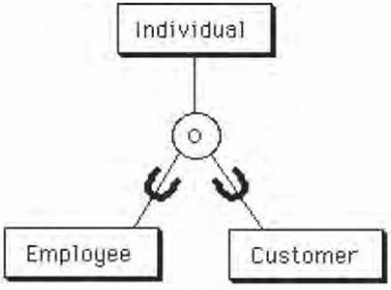
može i bez tabele individual

```
create table employee (
    emp_id char(10) not null,
    indiv_name char(20),
    indiv_addr char(20),
    job_title char(15),

    primary key( emp_id )
);

create table customer (
    cust_no char(10) not null,
    indiv_name char(20),
    indiv_addr char(20),
    cust_credit char(12),

    primary key( cust_no )
);
```



Hijerarhija – jedna tabela

```
create table individual (
    indiv_id char(10) not null,
    entity_type char(10) not null,

    indiv_name char(20),
    indiv_addr char(20),

    job_title char(15),
    cust_credit char(12),

    primary key( indiv_id )
);

create view employee ...;

create view customer ...;
```



Ostali slučajevi odnosa

Agregacija

- obično se svodi na obične odnose, obično 1-*

Odnosi sa više od 3 učesnika

- slično kao odnosi sa 3 učesnika
- mora više da se vodi računa o funkcionalnim zavisnostima

Slabi entiteti

- obično ne zahtevaju dodatno postupanje
- odnos sa jakim entitetom se obično prevodi u strani ključ

Rezime

1. Entiteti postaju relacije
2. Prosti atributi postaju atributi relacija
3. Složeni atributi se prevode u relacije sa stranim ključem prema roditelju
4. Binarni odnosi se prevode na opisane načine, zavisno od kardinalnosti
5. Odnosi sa više od 2 učesnika se prevode u **vezne** relacije koje imaju odgovarajuće odnose prema svim polaznim relacijama
6. Hijerarhije se prevode na neki od opisanih načina
7. Na kraju obavezno prečistiti šemu (normalizovati)

Literatura za temu

- Teorey, Lightstone, Nadeau, Jagadish, *Database Modeling and Design*, 5th ed., Elsevier, 2011.
- Watt, Eng, *Database Design*, 2nd ed., Open Edition, 2014.