Увод у релационе базе података

9

Саша Малков
Универзитет у Београду
Математички факултет
2023/2024

[PM13]
Увод у РБП

Саша Малков

Тема 7
Логичко пројектовање

Логичко моделирање

- Концептуални модел је фокусиран на семантику и на односе
  - углавном не зависи од врсте база података која ће бити употребљавана у имплементацији
- Логички модел се спушта ближе изабраном имплементационом моделу
  - узима у обзир модел података који ће се употребљавати у имплементацији
- Апстрактни концепти концептуалног модела се преводе у конкретне логичке моделе
  - На пример, хијерархије класа из концептуалног модела се замењују одговарајућим скуповима релација, у случају релационог модела

Логички модел

- Предуслов за започињање логичког моделирања је одабирање врсте база података, тј. модела података
  - Логички модел узима у обзир изабрани модел података али то не значи да је логички модел исто што и физички модел
  - Логичка схема ће претрпети још додатних измена када се изабере конкретан СУБП и када се направи физички модел
- Улаз за поступак логичког моделирања је концептуални модел
- Излаз је детаљан логички (нпр. релациони) модел
  - логичка схема свих трајних објеката (релација)
  - спецификација свих услова и ограничења (*constraints*)
  - познати су сви кључеви и одговарајући сурогат-атрибути
  - као и начини имплементације свих односа

Улога логичког модела

- Логички модел узима у обзир
  - све специфичности изабраног модела података
  - додатне нефункционалне захтеве, који су у фази концептуалног пројектовања можда могли да остану занемарени
- Води се рачуна о
  - флексибилности
  - проширивости
  - али не на уштрб јасноће и чистоће модела
- Логички модел тачно одређује који се подаци чувају и како ће се њима руковати

Прескакање логичког модела

- Пројектанти су често у искушењу да после концептуалног модела одмах праве физички модел
  - Или се прескаче логички модел, или се заправо прескаче концептуални и одмах прави логички
  - Врло често је већ при прављењу концептуалног модела познато која имплементација СУБП ће се користити
- Такав приступ носи ризик да се изгубе неке од пожељних карактеристика базе података:
  - комплетност
  - интегритет
  - флексибилност
  - ефикасност
  - употребљивост

Прескакање логичког модела (2)

- Добар концептуални модел је изузетно важан први корак за достизање ових карактеристика
- Али није и довољан
- Ако се концептуални модел преведе непосредно у физички модел, обично се изгубе комплетност и интегритет, али и друге карактеристике

Прескакање логичког модела (3)

- Мање базе података могу да се добро испројектују и без логичког модела, али код већих база је проблем и при пројектовању и при одржавању
- За одговоре на нека питања је врло често потребно имати сва три модела
- На пример, при променама структуре базе података често се питамо зашто је нешто у физичком моделу направљено баш тако а не другачије? Да ли је то природан модел или оптимизација?
  - Одговор се често може добити само посматрањем сва три модела.

Кораци логичког пројектовања

- Логичко пројектовање има следеће кораке:
  - Превођење концептуалног модела у логички
  - Пречишћавање схеме

Итеративно моделирање

- Прављење логичког модела се изводи итеративно
- Прва итерација се прави на основу концептуалног модела
  - 1. корак: превођење логичког модела у логички
- Свака наредна итерација се прави мењањем претходне (енгл. *flexing*)
  - 2. корак: пречишћавање схеме
  - Критеријуми за мењање су жељене карактеристике:
    - Да ли модел испуњава функционалне захтеве?
    - Да ли модел испуњава нефункционалне захтеве?
    - Да ли је модел комплетан?
    - Да ли је у складу са изабраним моделом података (релационим)?
    - Да ли модел гарантује интегритет података?
    - Да ли модел пружа потребну флексибилност?
    - Да ли модел омогућава ефикасан рад?
    - Да ли је модел довољно употребљив?

Значајне одлуке

- Логичко моделирање може да има за резултат и неке значајне одлуке, нпр:
  - може да се не прави једна него више база података
  - структура базе података може да се прилагоди очекиваним променама (или очекиваној врсти промена)
  - могу да се доносе и неке одлуке у складу са специфичностима изабране имплементације СУБП
  - некада не могу да се остваре све тражене карактеристике, па је неопходно одлучити која је у конкретном случају важнија

Зависност од модела података

- Логички модел се прави на “језику” изабраног модела података
  - превођење концептуалног модела у логички има за циљ изабрани модел података
  - пречишћавање схеме узима у обзир карактеристике изабраног модела података
- Све надаље у вези са логичким пројектовањем ћемо радити за случај релационог модела података

[PM13]

Увод у РБП

Саша Малков

Тема 7.1

Логичко пројектовање
-
Превођење концептуалног модела у логички

Превођење конц. у логички модел

- Први корак (прва итерација) при прављењу логичког модела је превођење концептуалног модела (КМ) у логички модел
  - Оба модела могу да се изражавају на ЕР-у или УМЛ-у, па је "*превођење*" често "*трансформисање*" или "*допуњавање и надоградњивање*"
- Покушаћемо да превођење опишемо у форми скупа правила

Релациони модел као основ логичког модела

- У наставку ћемо логички модел да представљамо језиком релационог модела (као што је најчешће случај у пракси)
  - Ентитети се представљају као релације
  - Сложени односи се представљају као релације
  - Једноставни односи се представљају страним кључевима
- У случају да имамо неки други циљни модел података (нпр. скуп каталога кључ-вредност), већина наведеног и даље важи али у прилагођеном облику

Поступак превођења КМ у ЛМ

- Превести сваки ентитет у релацију са кључем и некључним атрибутима
- Превести сваки бинарни (или рекурзиван) однос *-* у релацију која садржи кључеве учесника у односу
- Превести сваки однос са 3 или више учесника у релацију

Превођење ентитета у релације

- Ако постоји однос 1-*, додати кључ ентитета са стране 1 у релацију на страни * као страни кључ
- Ако постоји однос 1-1, додати кључ једног ентитета у другу релацију као страни кључ
  - ако постоји природан однос родитељ-дете, додајемо кључ у дете
  - у супротном додајемо кључ у релацију са мање редова
- Сваки ентитет у хијерархији превести у релацију (или већ према примењеној стратегији)
- Свака релација мора да има примарни кључ
- Сувишни кључеви ће се елиминисати при пречишћавању схеме

Представљање односа страним кључем

- У релационом моделу основно средство представљања односа су страни кључеви
  - Страни кључ из “везане” релације Б према “базној” релацији А има врло ограничену семантику кардиналности:
    - кард. А је уобичајено 0..1, ако кључ може да садржи *NULL*
    - кард. А може да буде 1, ако кључ не сме да садржи *NULL*
    - кард. Б је уобичајено 0..*
    - кард. Б може да буде 0..1, ако се дода услов јединствености кључа уз допуштено понављање вредности *NULL*
- Све остале варијанте захтевају постављање додатних услова и ограничења
  - али и додавање нових релација

Превођење бинарних односа *-*

- Сваки однос се преводи у релацију која као стране кључеве садржи кључеве ентитета који учествују у односу
- Водити рачуна о усклађивању ограничења за примарне и стране кључеве са ф.зависностима

Превођење односа са 3 или више учесника

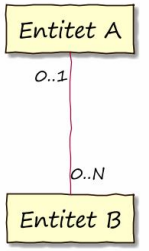
- Сваки однос се преводи у релацију која као стране кључеве садржи кључеве ентитета који учествују у односу
- Додати јединствене кључеве у складу са ф.зависностима
- Водити рачуна о усклађивању ограничења за примарне и стране кључеве са ф.зависностима

Превођење односа према кардиналности

- На наредним слајдовима ћемо видети како се различити бинарни односи преводе у релације на основу кардиналности
  - Полазни концептуални односи су представљени УМЛ-ом
  - У циљним релацијама препознајемо
    - потребне стране кључеве
    - потребне додатне услове интегритета
    - у неким примерима наводимо и схему табеле
- Нећемо све случајеве дискутовати на часу
  - потребно је да се обраде на основу слајдова и литературе

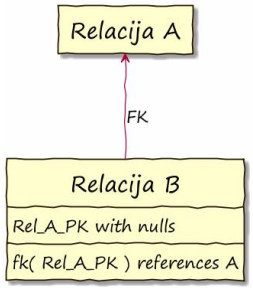
Бинарни односи 0..1 – 0..N

- Уобичајено за односе
  - *родитељ – потомак*
  - *целина – део*
  - ако *целина* **може** да буде без *делова*
  - ако *део* **може** да буде без *целине*
- Чест случај код агрегација



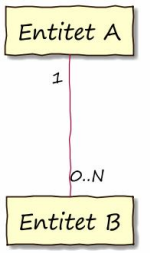
Бинарни односи 0..1 – 0..N

- Релацији која је *део*
  - додаје се страни кључ у односу на *целину*
  - страни кључ **може** да буде недефинисан



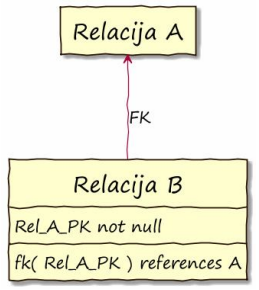
Бинарни односи 1 – 0..N

- Уобичајено за односе
  - *родитељ – потомак*
  - *целина – део*
  - ако *целина* **може** да буде без *делова*
  - ако *део* **не може** да буде без *целине*
- Чест случај код композиција



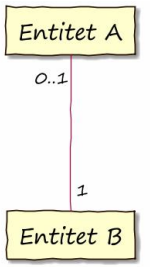
Бинарни односи 1 – 0..N

- Релацији која је *део*
  - додаје се страни кључ у односу на *целину*
  - страни кључ **не сме** да буде недефинисан



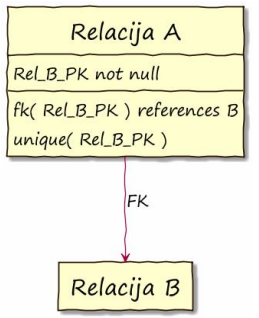
Бинарни односи 0..1 – 1

- Уобичајено за односе
  - *надређени – подређени*
  - *целина – опциони део*
- Пример:
  - Руководилац одељења А је службеник Б



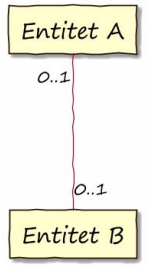
Бинарни односи 0..1 – 1

- Релацији која је *опциони део*
  - додаје се страни кључ у односу на *целину*
  - страни кључ **не сме** да буде недефинисан
  - вредност кључа мора да буде јединствена



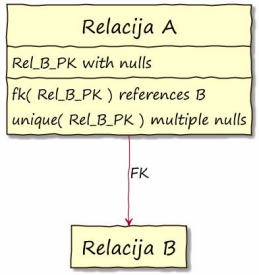
Бинарни односи 0..1 – 0..1

- Уобичајено за двосмерне опционе односе
- Пример:
  - Пројекат А може да има највише једног главног програмера Б
  - Програмер Б може да буде главни програмер на највише једном пројекту



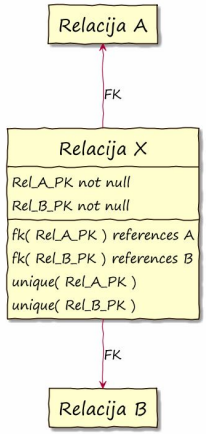
Бинарни односи 0..1 – 0..1

- Једној од релација
  - додаје се страни кључ у односу на другу
  - страни кључ **може** да буде недефинисан
  - вредност кључа мора да буде јединствена
    - недефинисане вредности смеју да се понављају
      - (то је проблем код неких система)



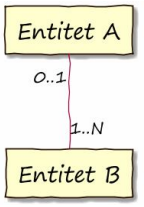
Бинарни односи 0..1 – 0..1

- ...алтернатива...
- Прави се нова *везна* релација
  - садржи само стране кључеве у односу на обе везане релације
  - кључеви **не смеју** да буду недефинисани
  - сви атрибути чине примарни кључ
  - вредност сваког од страних кључева мора да буде јединствена



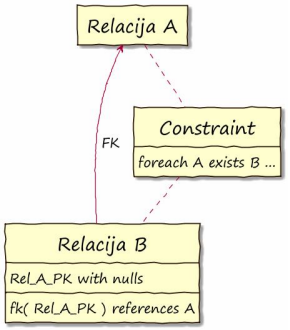
Бинарни односи 0..1 – 1..N

- Уобичајено за односе
  - агрегације са обавезним деловима
- Пример:
  - Свако одељење А има најмање једно додељено паркинг-место Б
  - Свако паркинг-место Б може да се додели највише једном одељењу А



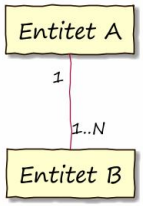
Бинарни односи 0..1 – 1..N

- Релацији која је *geo*
  - додаје се страни кључ у односу на *целин*
  - страни кључ **може** да буде недефинисан
  - додаје се услов базе података: “за сваки ентитет А мора да постоји бар један одговарајући ентитет Б”



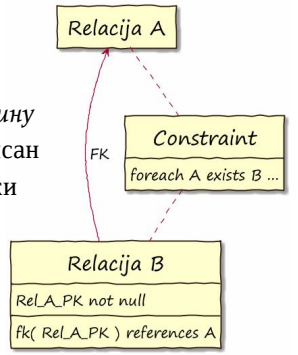
Бинарни односи 1 – 1..N

- Уобичајено за односе
  - композиције са обавезним деловима
- Пример:
  - Свако одељење А има најмање једно додељено паркинг-место Б
  - Свако паркинг-место Б припада једном одељењу А



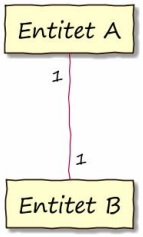
Бинарни односи 1 – 1..N

- Релацији која је *geo*
  - додаје се страни кључ у односу на *целину*
  - страни кључ **не сме** да буде недефинисан
  - додаје се услов базе података: “за сваки ентитет А мора да постоји бар један одговарајући ентитет Б”



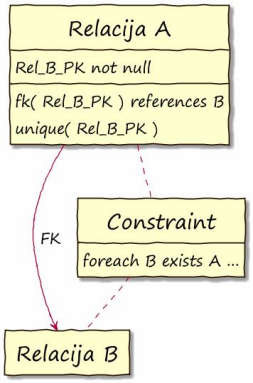
Бинарни односи 1 – 1

- Уобичајено за односе
  - обостраног ексклузивног придруживања
  - (релативно ретко, питање је да ли су то различити ентитети)



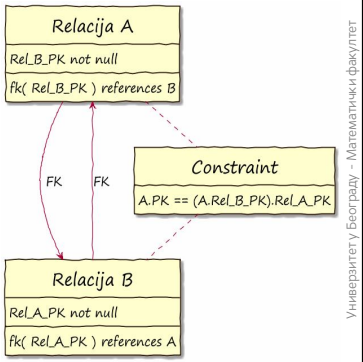
Бинарни односи 1 – 1

- Једној од релација
  - додаје се страни кључ у односу на другу
  - страни кључ **не сме** да буде недефинисан
  - вредност кључа мора да буде јединствена
  - додаје се услов базе података: “за сваки ентитет Б мора да постоји бар један одговарајући ентитет А”



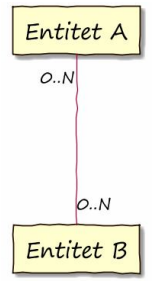
Бинарни односи 1 – 1

- ...алтернатива...
- У свакој од релација
  - додаје се страни кључ у односу на другу
  - атрибути кључа не смеју да буду недефинисани
  - додаје се услов којим се проверава узајамност везе



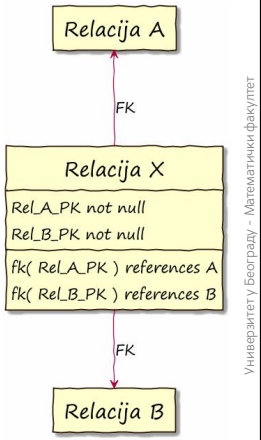
Бинарни односи 0..N – 0..N

- Уобичајено за односе
  - асоцијације без посебних ограничења



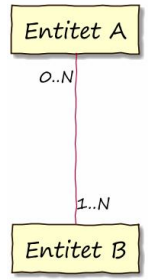
Бинарни односи 0..N – 0..N

- Прави се нова *везна* релација
  - садржи само стране кључеве у односу на обе везане релације
  - кључеви **не смеју** да буду недефинисани
  - сви атрибути чине примарни кључ



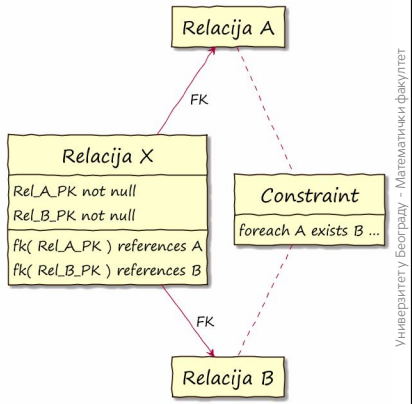
Бинарни односи 0..N – 1..N

- Уобичајено за односе
  - асоцијације слабих и јаких ентитета
  - агрегације код којих *geo* може да чини више целина, али не може да постоји самостално



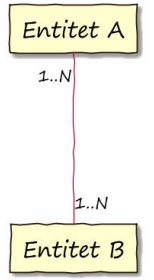
Бинарни односи 0..N – 1..N

- Прави се нова *везна* релација
  - садржи само стране кључеве у односу на обе везане релације
  - кључеви **не смеју** да буду недефинисани
  - сви атрибути чине примарни кључ
  - додаје се услов базе података: “за сваки ентитет А мора да постоји бар један одговарајући ентитет Б”



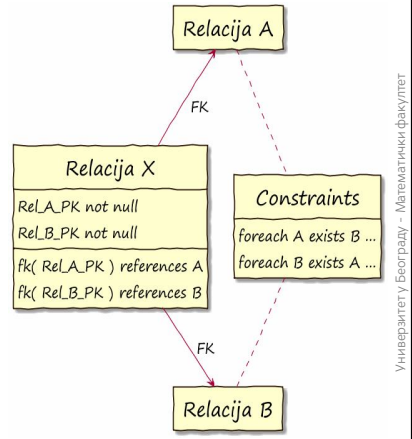
Бинарни односи 1..N – 1..N

- Уобичајено за односе
  - агрегације код којих
    - *део* може да чини више целина, али не може да постоји самостално
    - *целина* не може да постоји без *делова*
  - (релативно ретко)

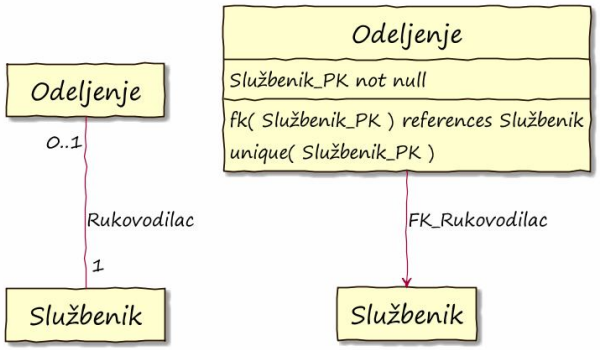


Бинарни односи 1..N – 1..N

- Прави се нова *везна* релација
  - садржи само стране кључеве у односу на обе везане релације
  - кључеви **не смеју** да буду недефинисани
  - сви атрибути чине примарни кључ
  - додаје се услов базе података: “за сваки ентитет А мора да постоји бар један одговарајући ентитет Б”
  - додаје се услов базе података: “за сваки ентитет Б мора да постоји бар један одговарајући ентитет А”



Пример односа 0..1 – 1

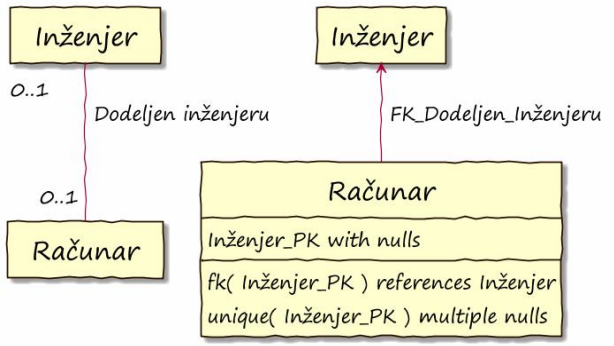


Пример односа 0..1 – 1

- Свако одељење има руководиоца
- Сваки запослени може да буде руководилац највише једног одељења

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


Пример односа 0..1 – 0..1

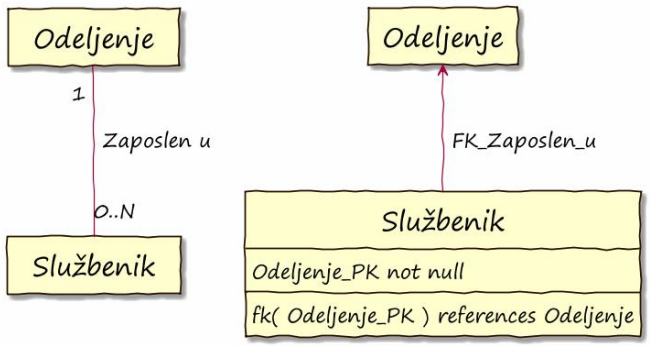


Пример односа 0..1 – 0..1

- Неки рачунари су додељени инжењерима
- Неким инжењерима су додељени рачунари

```
create table engineer (
     emp_id char(10),
     desktop_no integer,
     primary key( emp_id )
);
create table desktop (
     desktop_no integer,
     emp_id char(10) unique,      -- < -- потенцијалан проблем !!!
     primary key( desktop_no ),
     foreign key( emp_id )
             references engineer
             on delete set null
             on update set null
);
```


Пример односа 1 – 0..N

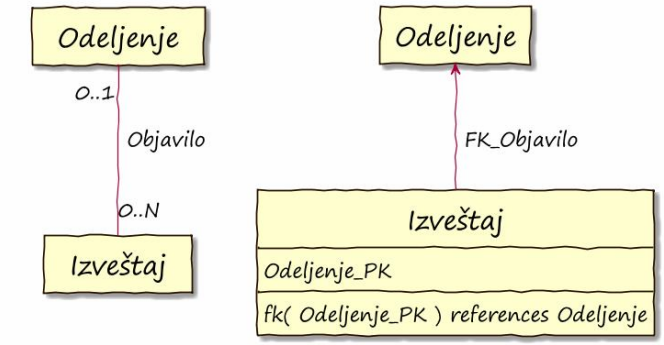


Пример односа 1 – 0..N

- Сваки службеник ради у тачно једном одељењу
- Свако одељење може да има 0 или више запослених

```
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 restrict
            on update restrict
);
```


Пример односа 0..1 – 0..N

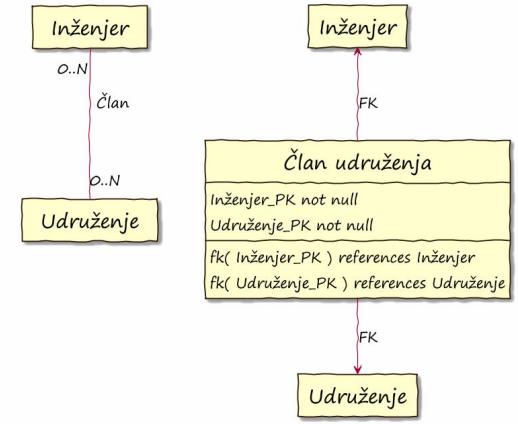


Пример односа 0..1 – 0..N

- Свако одељење објављује један или више извештаја
- Извештај може али не мора да буде објављен у неком одељењу

```
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 set null
);
```


Пример односа 0..N – 0..N



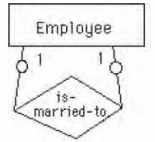
Пример односа 0..N – 0..N

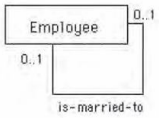
- Свако удружење може да има 0 или више инжењера
- Сваки инжењер може да буде члан 0 или више удружења

```
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
);
```


Бинарни циклични односи 1-1

- Било да су опциони или не
  - представљају се додатним атрибутима страног кључа
  - ако је опциони, сме да буде *NULL*

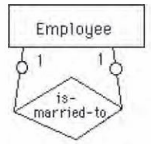


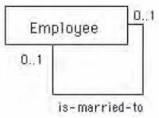


Бинарни циклични односи 1-1

- Сваки запослени може да буде у браку са другим запосленим у компанији

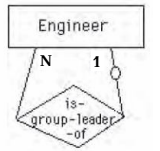
```
create table employee (
      emp_id char(10),
      emp_name char(20),
      spouse_id char(10),
      primary key( emp_id ),
      foreign key( spouse_id )
               references employee
               on delete set null
               on update set null
);
```

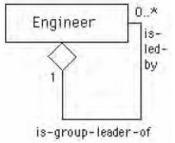




Бинарни циклични односи 1-*

- Ако је однос 1-*
  - страни кључ се уводи на страни "*"

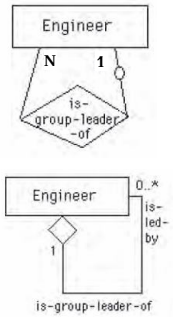




Бинарни циклични односи 1-*

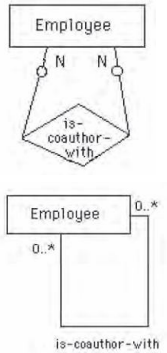
- Инжењери се групишу у тимове
- Сваки тим има тачно једног лидера

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



Бинарни циклични односи *-*

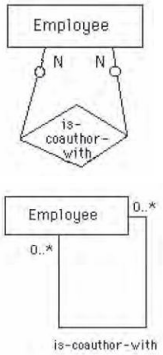
- Ако је однос *-*
  - представља се новом релацијом
  - као и обични бинарни односи *-*



Бинарни циклични односи *-*

- Сваки службеник може да пише извештај сам или са коаутором

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



Односи са више учесника

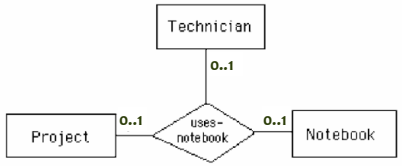
- Код односа са више учесника веома је важно да се води рачуна о нотацији кардиналности
- Кардиналности одређују зависности међу подацима

Према формалној и исправној нотацији кардиналности у дијаграмима ЕР, број "x" поред *Пројекта* означава да један пројекат учествује у "x" односа.

Односи са више учесника 1-1-1

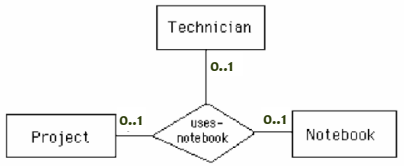
- Нова релација са страним кључевима
  - зависности се уређују допуштањем *NULL* и јединственим кључевима
- Пример:
  - Техничару је додељен највише један рачунар на неком од пројекта
  - Сваки рачунар припада највише једном пару (техничар, пројекат)
  - На сваком пројекту се додељује највише један рачунар
  - Зависности:
    - техничар -> пројекат, рачунар
    - рачунар -> техничар, пројекат
    - пројекат -> техничар, рачунар

Према формалној и исправној нотацији кардиналности у дијаграмима ЕР, број "x" поред *Пројекта* означава да један пројекат учествује у "x" односа.



Односи са више учесника 1-1-1 (2)

```
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 ),
      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( notebook_no ),
      unique( project_name )
);
```



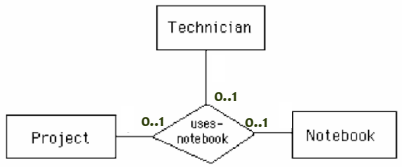
Односи са више учесника 1-1-1, алт.

- Ако се кардиналности тумаче као у дијаграму класа, онда је семантика потпуно измењена:
- Пример:
  - На једном пројекту, неки рачунар може да се додели највише једном техничару
  - Један рачунар може да се додели једном техничару на највише једном пројекту
  - На једном пројекту сваки техничар м највише један рачунар
  - Зависности:
    - техничар, пројекат -> рачунар
    - техничар, рачунар -> пројекат
    - пројекат, рачунар -> техничар

Коришћењем алтернативне нотације кардиналности, потпуно се мења смисао односа.

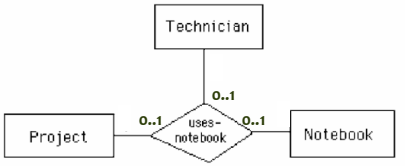
Према алтернативној нотацији, број "x" поред *Пројекта* означава да један пар (техничар, рачунар) учествује у "x" односа

Ако се користи такво означавање, то мора да се некако нагласи:
- или коментаром
- или навођењем броја ближе односу, уместо ближе ентитету



Односи са више учесника 1-1-1, алт. (2)

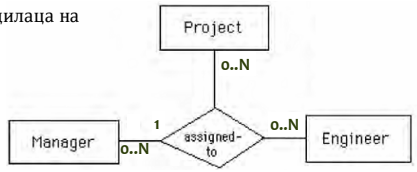
```
create table technician (
      emp_id char(10),
      primary key( emp_id ));
create table project (
      project_name char(20),
      primary key( project_name ));
create table notebook (
      notebook_no integer,
      primary key( notebook_no ));
create table uses_notebook (
      emp_id char(10),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



Односи са више учесника 1-*-*

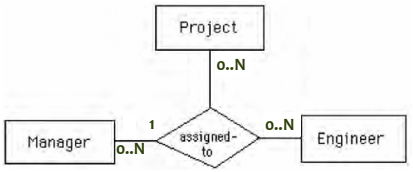
- Нова релација са страним кључевима
  - зависности се уређују избором примарног кључа
- Пример:
  - Сваки инжењер на сваком пројекту има тачно једног руководиоца
  - Пројекат може да има више руководилаца
  - Руководилац може да води више пројеката
  - Инжењер може да има више руководилаца на различитим пројектима
  - Зависности:
    - запослени, пројекат -> руководилац

Користимо комбиновану нотацију. Обратите пажњу на положај бројева који одређују кардиналности.



Односи са више учесника 1-*-* (2)

```
create table project (
      project_name char(20),
      primary key( project_name )
);
create table manager (
      mgr_id char(10),
      primary key( mgr_id )
);
create table engineer (
      emp_id char(10),
      primary key( emp_id )
);
create table manages (
      project_name char(20),
      mgr_id char(10),
      emp_id char(10),
      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
);
```



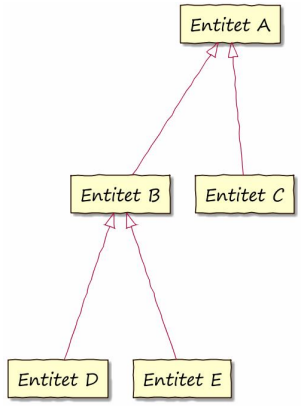
Остали случајеви

- Агрегација
  - своди се на обичне односе, обично 1-*
- Односи са више од 3 учесника
  - слично као односи са 3 учесника
  - мора више да се води рачуна о ф.зависностима
- Слаби ентитети
  - обично не захтева додатно поступање
  - превођење односа са јаким ент. решава проблем
- Хијерархије
  - ...

Превођење хијерархија ентитета

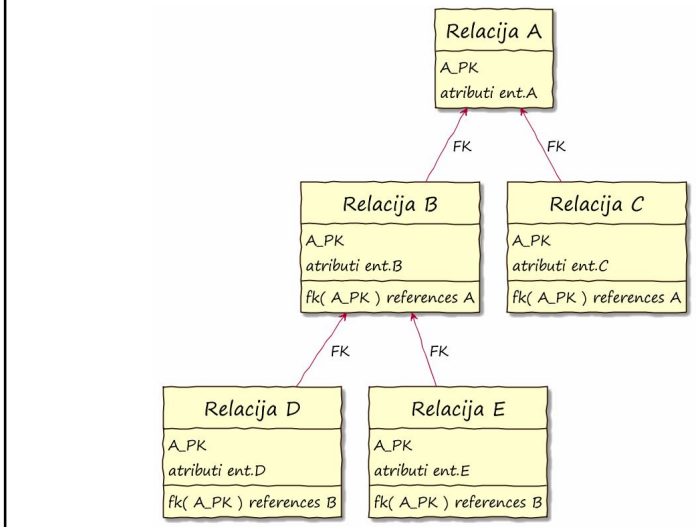
- Хијерархије ентитета се обично преводе на неки од три основна начина:
  - Сваки ентитет у посебну релацију
    - практично као да се не ради о односу специјализације него о неком другом односу
  - Сваки ентитет-лист у посебну релацију, али тако да укључи **све** наслеђене атрибуте
  - Цела хијерархија у једну релацију
- Могуће је и комбиновање метода, за различите делове хијерархије

Пример хијерархије



Превођење хијерархија – Сваки ентитет посебно

- За сваки ентитет се прави по релација, са страним кључем на непосредну базну релацију
  - свака релација обухвата само оне атрибуте који су за њу специфични
    - ...и још атрибуте кључа базне релације, који се користе као страни кључ
  - практично се специјализација третира као асоцијација или агрегација
- Овај приступ захтева честа спајања при читању и потенцијално је неефикасан
  - али је на нивоу логичког модела прихватљив

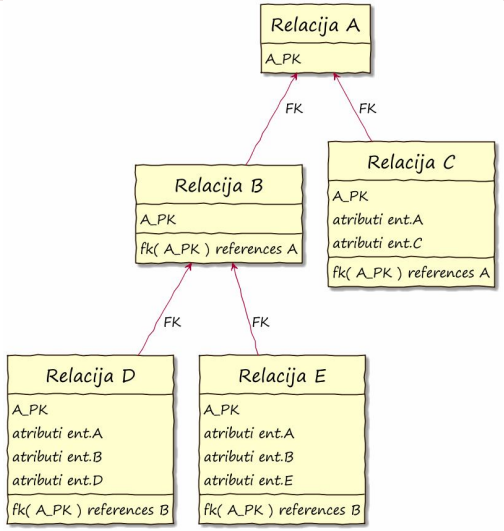


Превођење хијерархија – Сваки ентитет посебно (2)

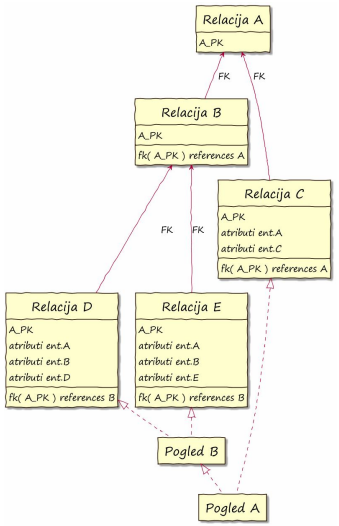
- Прекривање
  - ако је хијерархија *без прекривања*, све је у реду
  - ако је хијерархија *са прекривањем* онда је потребно да се обезбеди додатно средство за проверу покривености
    - т.ј. да сваки ред базног ентитета има одговарајући ред у неком листу
- Преклапање
  - Ако је хијерархија *без преклапања* онда је потребно да се обезбеди додатно средство за проверу да нема преклапања
    - т.ј. да се један ред базног ент. не реферише из више “изведених” ентитета
  - Ако је хијерархија *са преклапањем*, све је у реду

Превођење хијерархија – Сваки лист посебно

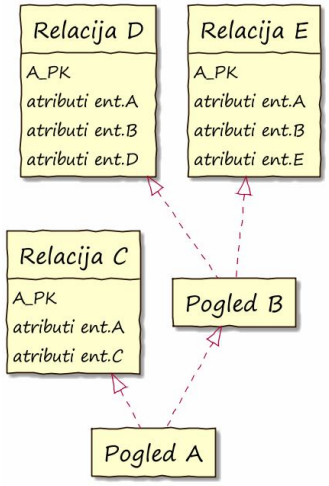
- За сваки ентитет-лист се прави по релација, са страним кључем на непосредну базну релацију
  - Свака релација обухвата атрибуте који су за њу специфични
    - …и **СВЕ** атрибуте **СВИХ** базних класа
    - … атрибути кључа базне релације се користе као страни кључ
- За ентитет који није лист
  - релација се изоставља или има само примарни кључ
  - прави се поглед који се рачуна као унија листова
- Употреба појединачних ентитета-листова је ефикасна
- Претраживање базног скупа ентитета (или било ког ентитета који није лист) је неефикасно (унија)



- У релацијама које одговарају листовима наводе се сви атрибути
- У релацијама које одговарају апстрактним класама наводе се само примарни кључеви



- Апстрактни ентитети се додатно моделирају погледима
  - сваки поглед је унија одговарајућих релација



- Штавише, за апстрактне ентитете ни не морају да се праве релације

Превођење хијерархија – Сваки лист посебно (2)

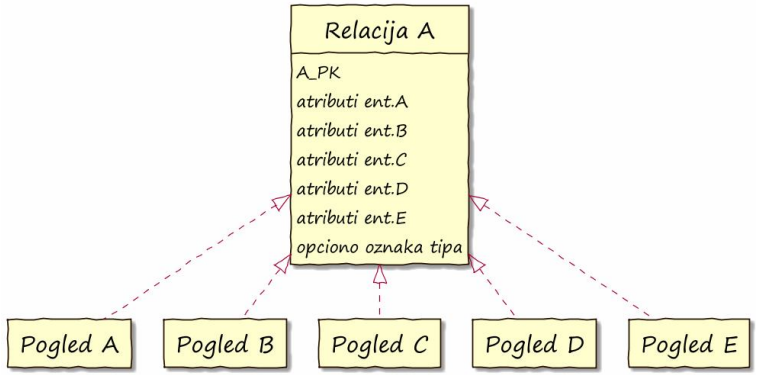
- Прекривање
  - ако је хијерархија *без преклапања* онда се практично своди на претходни случај, зато што свака класа може да буде лист
    - праве "унутрашње" класе су само оне које су апстрактне
  - ако је хијерархија *са преклапањем* онда је потребно да се обезбеди додатно средство за проверу покривености
    - т.ј. да сваки ред базног ентитета има одговарајући ред у неком листу
    - може да се избегне изостављањем релација апстрактних ентитета
- Преклапање
  - ако је хијерархија *без преклапања* онда је потребно да се обезбеди додатно средство за проверу да нема преклапања
    - т.ј. да се један ред базног ентитета не реферише из више различитих "изведених" ентитета
  - ако је хијерархија *са преклапањем* онда имамо редундантност
    - наслеђени делови ентитета могу да се редундантно понављају у више листова...

Превођење хијерархија – Све у једну релацију

- Једна релација моделира целу хијерархију
  - Обухвата све атрибуте који постоје у хијерархији
  - За сваки ентитет се користи само део атрибута...
  - ...остали имају недефинисане (празне) вредности
    - што може да се решава правилима интегритета
- Припадање врсте ентитету се проверава на основу садржаја атрибута
  - или на основу постојећих атрибута
  - или се додају сурогат-атрибути као ознаке типа
- Релативно ефикасна употреба
- Потенцијално неефикасно заузеће простора
  - новији СУБП то релативно добро решавају

Превођење хијерархија – Све у једну релацију (2)

- Припадање врсте ентитету се проверава на основу садржаја атрибута
  - или се додаје сурогат-атрибут као ознака ентитета (типа)
    - упити над изведеним ентитетима се своде на рестрикцију по ознаци типа
  - или на основу постојећих атрибута
    - упити над изведеним ентитетима се своде на рестрикцију по специфичним атрибутима ентитета, тј. проверава се да нису *NULL*



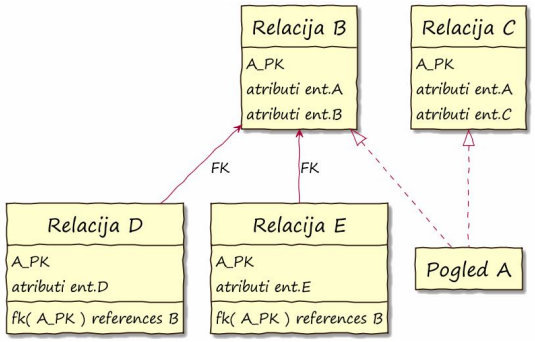
Превођење хијерархија – Све у једну релацију (3)

- Прекривање
  - ако је хијерархија *без прекривања* онда је све у реду
  - ако је хијерархија *са прекривањем* онда се додају провере
    - или се забрани да ознака типа одговара апстрактној класи
    - или се захтева да су атрибути неког листа дефинисани
- Преклапање
  - ако је хијерархија *без преклапања*...
  - ако је хијерархија *са преклапањем* онда је отежано проверавање припадности и интегритета, усложњава се употреба атрибута за означавање ентитета реда
    - један ред потенцијално одговара већем броју ентитета
    - може да се уведе по бинарни атрибут припадности за сваки ентитет
      - нпр. ако се користи 32-битни број, сваки бит може да одговара једном ентитету
    - провера не може да буде "ком ентитету припада ред?" него само "да ли ред припада датом ентитету?"

	Сваки ентитет посебно	**Сваки лист посебно**	**Само једна релација**
Без прекривања	+	Нема смисла, исто као претходно	+
Са прекривањем	Додатне провере	+ (ако су сви ун.чворови апс.) Иначе, додатне провере	Додатне провере (може локално, забрани се апс.тип)
Без преклапања	Додатне провере	Додатне провере	+ (локалне провере, на нивоу реда)
Са преклапањем	+	Редундатност	+ (али уз сложеније установљавање ентитета)
Читање	Спајање са базним релацијама	Листови ефикасно Остало помоћу уније	Ефикасно, рестрикција
Ажурирање	Ажурирање и базних релација	Само једна релација	Само једна релација

Превођење хијерархија – Комбиновано

- Различити методи могу да се комбинују у различитим сегментима исте хијерархије
- На пример:
  - један део хијерархије се моделира само једном релацијом а остатак додатним релацијама
  - или се комбинује приступ са релацијама за све ентитете и само за листове
  - ...



- Класе *A*, *B* и *C* су моделиране тако што су направљене релације само за листове хијерархије
- Класе *D* и *E* су имплементиране као додатне релације
- Овакав модел је добар ако је само *A* апстрактна класа
- Има предности ако може да буде преклапања између *D* и *E*, као и између *B* и *C*

Резиме

1. Користити алате за пројектовање, могу много да олакшају посао
2. Ентитети постају релације
3. Прости атрибути постају атрибути релација
4. Сложени атрибути се преводе у релације са страним кључем према родитељу
5. Односи 1-1 или 1-* морају да се повежу паровима страних и примарних кључева
6. Односи *-* се преводе у *везне* релације које имају два односа 1-* према полазним релацијама
7. Односи са више од 2 учесника се преводе у *везне* релације које имају одговарајуће односе према свим полазним релацијама
8. Хијерархије се преводе на неки од раније описаних начина
9. На крају обавезно пречистити схему (нормализовати)

Литература за тему

- Teorey, Lightstone, Nadeau, Jagadish, **Database Modeling and Design**, *5.ed, Elsevier, 2011.*
- Watt, Eng, **Database Design**, *2.ed, Open Edition, 2014.*
- Гордана Павловић-Лажетић, **Увод у релационе базе података,** *2.изд. Математички факултет, 1999.*
  - доступно онлајн: *http://poincare.matf.bg.ac.rs/~gordana//urbp-2016.htm*
- Ramakrishnan, Gehrke, **Database Management Systems**, *2.ed, 2000.*