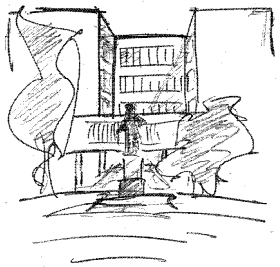
[P273]
Пројектовање база података

6

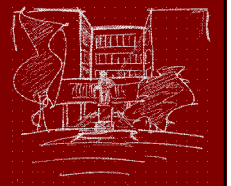


Саша Малков

Универзитет у Београду
Математички факултет

2023/2024

[P273]
Пројектовање база података

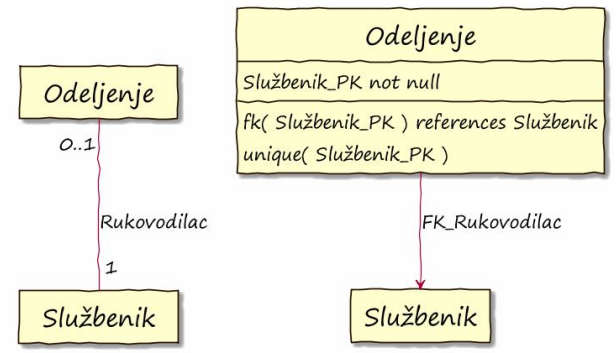


Саша Малков

Тема 8.2

Логичко моделирање
-
Превођење КМ у логички (наставак)

Пример односа 0..1 – 1

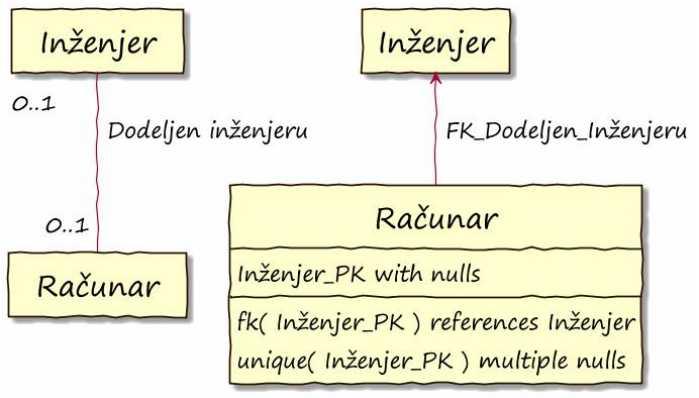


Пример односа 0..1 – 1 (2)

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

```
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 )
);
```

Пример односа 0..1 – 0..1



Пример односа 0..1 – 0..1 (2)

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

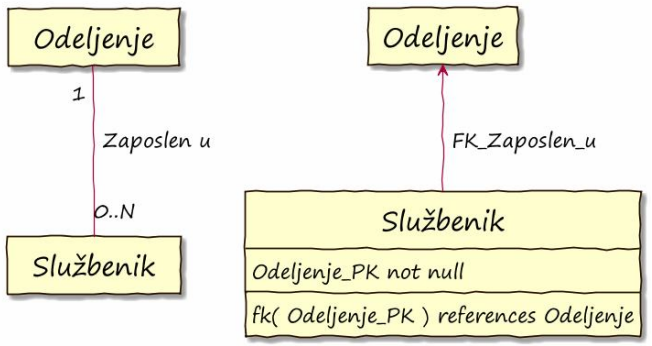
```
create table engineer (
     emp_id char(10) not null,
     ...
     primary key( emp_id )
);
create table desktop (
     desktop_no integer not null,
     emp_id char(10) unique,      -- < -- потенцијалан проблем !!!
     primary key( desktop_no ),
     foreign key( emp_id )
             references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


Пример односа 0..1 – 0..1 (3)

- Решење за јединственост атрибута са више допуштених недефинисаних вредности:

```
create table engineer ( ... );
create table desktop ( ...
       emp_id char(10),
       ... );
create unique index desktop_emp
       on desktop( emp_id )
       exclude null keys;
```


Пример односа 1 – 0..N

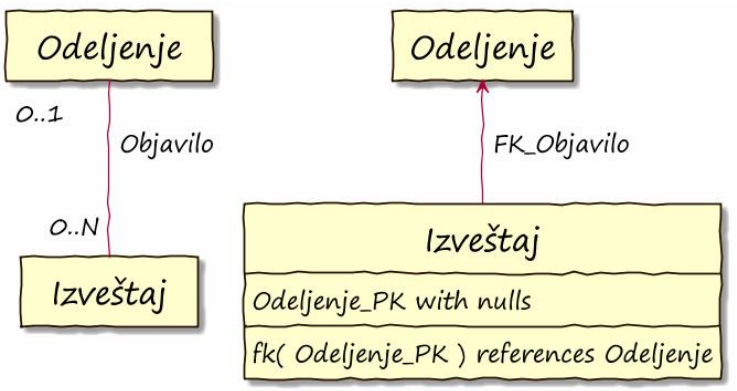


Пример односа 1 – 0..N (2)

- Сваки службеник ради у тачно једном одељењу
- Свако одељење има бар једног запосленог

```
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
);
```


Пример односа 0..1 – 0..N

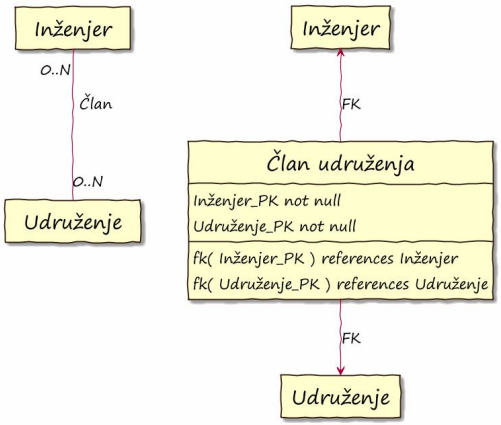


Пример односа 0..1 – 0..N (2)

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

```
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
);
```


Пример односа 0..N – 0..N



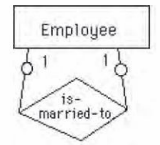
Пример односа 0..N – 0..N (2)

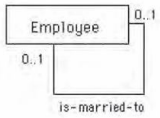
- Свако удружење може да има 0 или више инжењера
- Сваки инжењер може да буде члан 0 или више удружења

```
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 (0..1-1, 0..1-0..1)

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

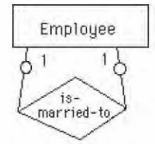


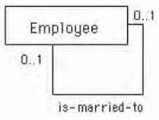


Бинарни циклични односи 1-1 (2)

- Сваки запослени може да буде у браку са другим запосленим у компанији (али највише са једним)

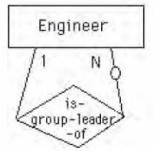
```
create table employee (
     emp_id char(10) not null,
     emp_name char(20),
     spouse_id char(10),
     primary key( emp_id ),
     foreign key( spouse_id )
              references employee
              on delete set null
              on update cascade
);
```

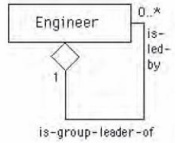




Бинарни циклични односи 1-*

- Ако је однос 1-*
  - страни кључ се уводи на страни “*”

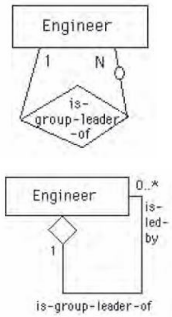




Бинарни циклични односи 1-* (2)

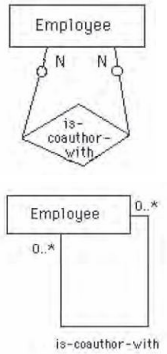
- Инжењери се групишу у тимове
- Сваки тим има тачно једног лидера

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



Бинарни циклични односи *-*

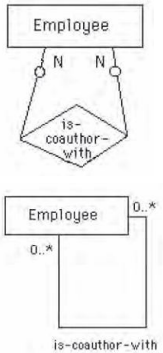
- Ако је однос *-*
  - представља се новом релацијом
  - као и обични бинарни односи *-*



Бинарни циклични односи *-* (2)

- Сваки службеник може да пише извештај сам или са коаутором

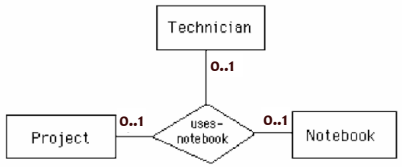
```
create table employee (
      emp_id char(10) not null,
      emp_name char(20) not null,
      primary key( emp_id )
);
create table coauthor (
      author_id char(10) not null,
      coauthor_id char(10) not null,
      primary key( author_id, coauthor_id ),
      foreign key( author_id )
   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



Односи са више учесника 1-1-1

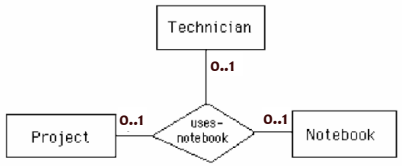
- Нова релација са страним кључевима
  - зависности се уређују допуштањем *NULL* и јединственим кључевима
- Пример:
  - Техничару је додељен највише један рачунар на неком од пројекта
  - Сваки рачунар припада највише једном пару (техничар, пројекат)
  - На сваком пројекту се додељује највише један рачунар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

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



Односи са више учесника 1-1-1 (2)

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



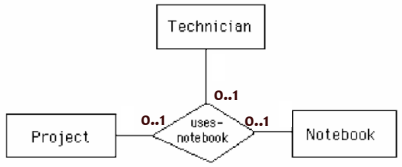
Односи са више учесника 1-1-1, алт.

- 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 може да користи највише један рачунар
- Зависности:
  - техничар, пројекат -> рачунар
  - техничар, рачунар -> пројекат
  - пројекат, рачунар -> техничар

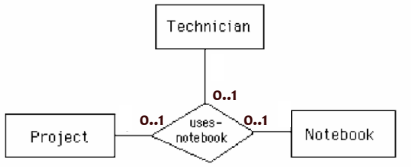
Коришћењем алтернативне нотације кардиналности, потпуно се мења смисао односа.

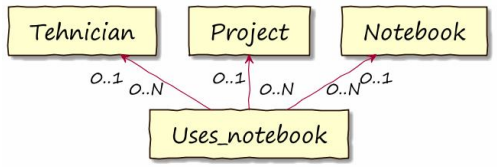
Према алтернативној нотацији, број "x" поред *Пројекта* означава да један пар (техничар, рачунар) учествује у "x" односа

Ако се користи такво означавање, то мора да се некако нагласи:
- или коментаром
- или навођењем броја уз однос



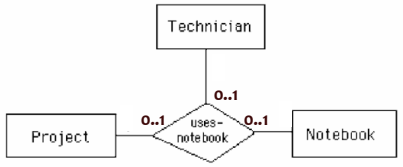
Поређење са нотацијом дијаграма класа





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

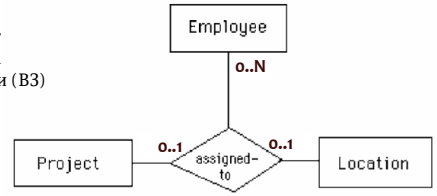
```
create table notebook (
    notebook_no integer not null,
    primary key( notebook_no ));
create table uses_notebook (
    emp_id char(10) not null,
    project_name char(20) n
    notebook_no integer not
    primary key( emp_id,
project_name ),
    foreign key( emp_id ) references
technician
 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



Односи са више учесника 1-1-*

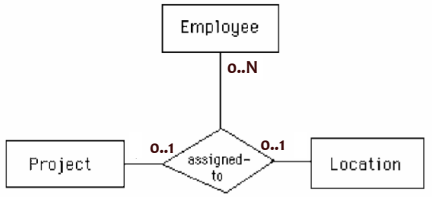
- Нова релација са страним кључевима
  - зависности се уређују допуштањем *NULL* и јединственим кључевима
- Пример:
  - Сваки службеник може да ради на више пројеката и локација
  - На свакој локацији службеник ради на највише једном пројекту
  - На сваком пројекту службеник ради на највише једној локацији
  - Зависности:
    - запослени, локација -> пројекат
    - запослени, пројекат -> локација
    - пројекат, локација --> запослени (ВЗ)

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



Односи са више учесника 1-1-* (2)

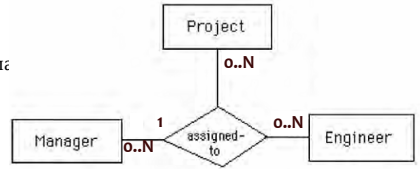
```
create table employee (
      emp_id char(10) not null,
      emp_name char(20),
      primary key( emp_id ));
create table project (
      project_name char(20) not null,
      primary key( project_name ));
create table location (
      loc_name char(15) not null,
      primary key( loc_name ));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



Односи са више учесника 1-*-*

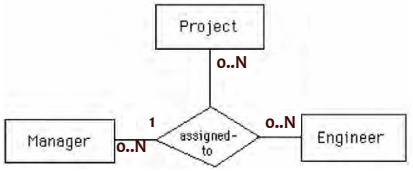
- Нова релација са страним кључевима
  - зависности се уређују избором примарног кључа
- Пример:
  - Сваки инжењер на сваком пројекту има тачно једног руководиоца
  - Пројекат може да има више руководилаца
  - Руководилац може да води више пројеката
  - Инжењер може да има више руководилаца на различитим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

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



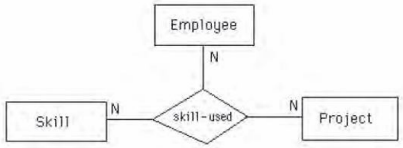
Односи са више учесника 1-*-* (2)

```
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
);
```



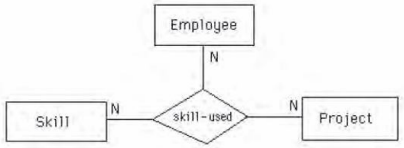
Односи са више учесника *-*-*

- Нова релација са страним кључевима
- Пример:
  - Службеник може да користи различите вештине на сваком од пројеката
  - Сваки пројекат може да има више запослених са више различитих или поновљених вештина
  - Зависности:
    - само вишезначне



Односи са више учесника *-*-* (2)

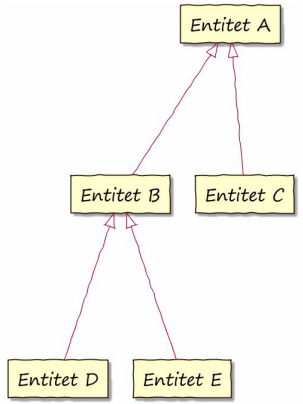
```
create table employee (
      emp_id char(10) not null,
      emp_name char(20),
      primary key( emp_id ));
create table skill (
      skill_type char(15) not null,
      primary key( skill_type ));
create table project (
      project_name char(20) not null,
      primary key( project_name )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



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

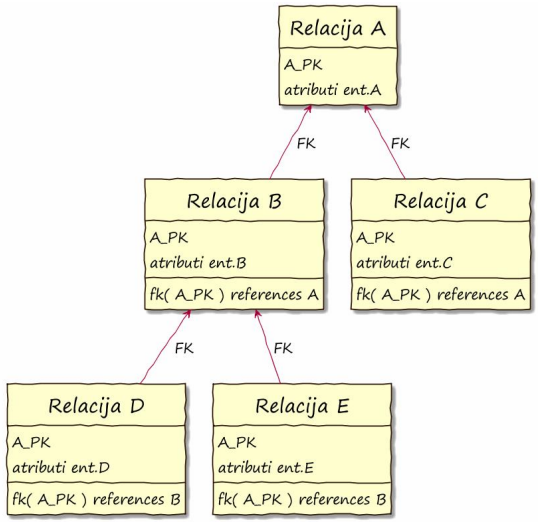
Пример хијерархије



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



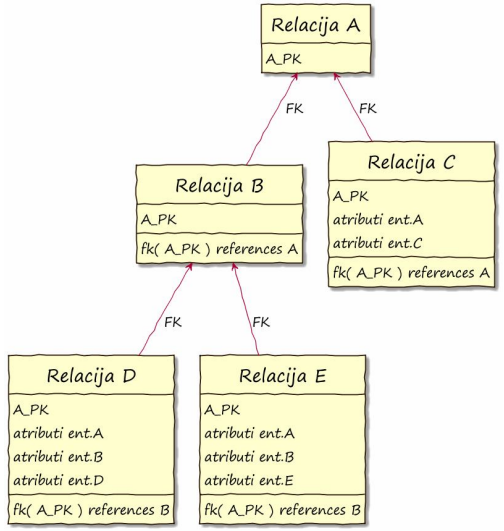
Превођење хијерархија - Сваки ентитет посебно (2)

- 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

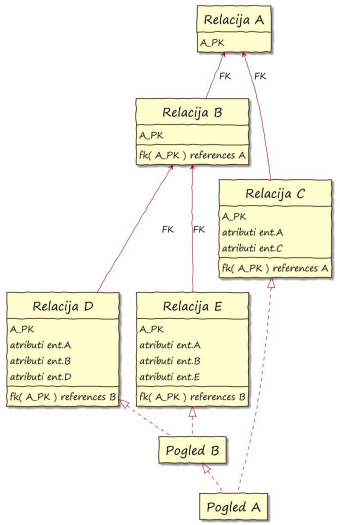
- За сваки ентитет-лист се прави по релација, са страним кључем на непосредну базну релацију
  - Свака релација обухвата атрибуте који су за њу специфични
    - ...и **СВЕ** атрибуте **СВИХ** базних класа
    - ... атрибути кључа базне релације се користе као страни кључ
- За ентитет који није лист
  - 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)

Превођење хијерархија – Сваки лист посебно



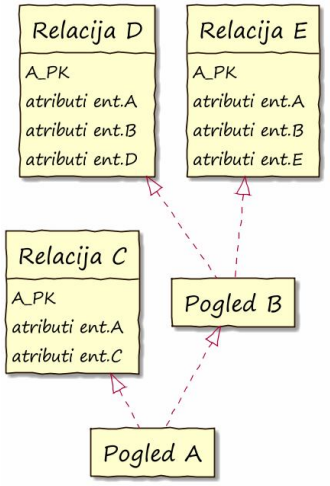
- У релацијама које одговарају листовима наводе се сви атрибути
- У релацијама које одговарају апстрактним класама наводе се само примарни кључеви

Превођење хијерархија – Сваки лист посебно



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

Превођење хијерархија – Сваки лист посебно



- Технички гледано, за апстрактне ентитете ни не морају да се праве релације
- У пракси су ипак потребне ради старања о интегритету, посебно у случајевима без преклапања
  - ако не постоје додатне релације и страни кључеви, онда је отежана провера да ли постоје редови са истим ИД у различитим табелама

Превођење хијерархија – Сваки лист посебно (2)

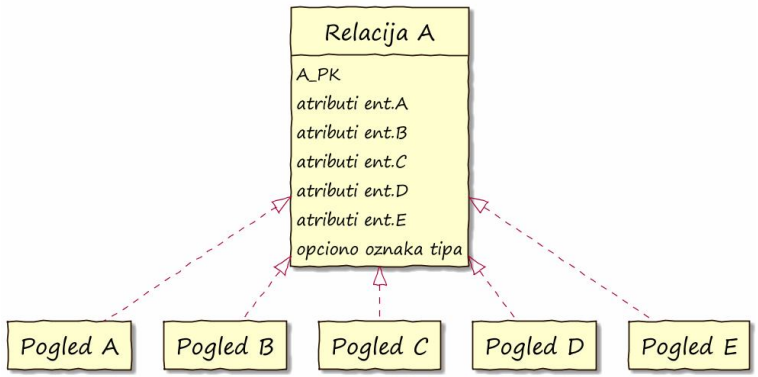
- Прекривање
  - ако је хијерархија *без прекривања* онда се практично своди на претходни случај, зато што свака класа може да буде лист
    - праве "унутрашње" класе су само оне које су апстрактне
  - ако је хијерархија *са прекривањем* онда је потребно да се обезбеди додатно средство за проверу покривености
    - т.ј. да сваки ред базног ентитета има одговарајући ред у неком листу
    - 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?”

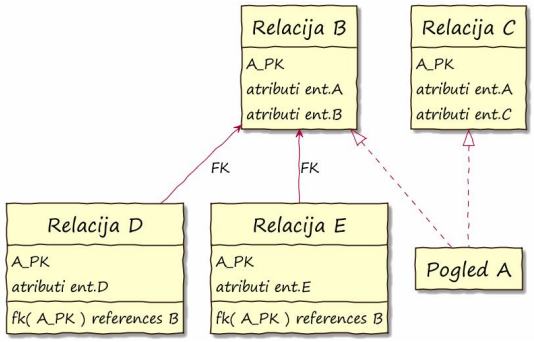
Превођење хијерархија – Резиме варијанти

	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
  - …

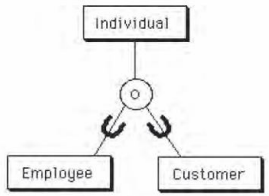
Превођење хијерархија – Пример комбиноване имплементације



- 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*

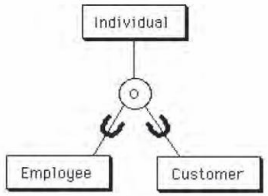
Хијерархија – пример

- Већ смо видели да може на различите начине:
  - Цела хијерархија у једну релацију
  - Сваки ентитет у посебну релацију, тј. као да се ради о “обичном” односу ентитета
  - Сваки ентитет-лист у посебну релацију, али тако да укључи све наслеђене атрибуте
- Пример:
  - Особа може да буде службеник или муштерија или обоје или ништа од тога



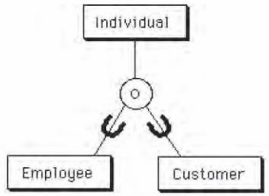
Хијерархија – сваки ентитет у релацију

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



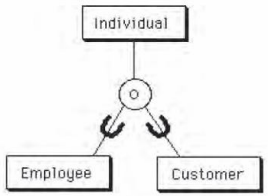
Хијерархија – листови у релације

```
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
);
```



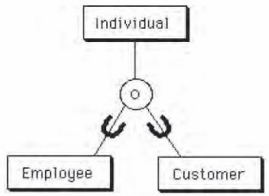
Хијерархија – листови у релације (2)

```
-- може и без табеле 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 )
);
```

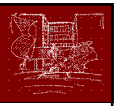


Хијерархија – једна табела

```
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 table customer ...;
```



Остали случајеви односа



- Агрегација
  - обично се своди 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
  - однос са јаким ент. се обично преводи у страни кључ

Резиме

1. Ентитети постају релације
2. Прости атрибути постају атрибути релација
3. Сложени атрибути се преводе у релације са страним кључем према родитељу
4. Бинарни односи се преводе на описане начине, зависно од кардиналности
5. Односи са више од 2 учесника се преводе у *везне* релације које имају одговарајуће односе према свим полазним релацијама
6. Хијерархије се преводе на неки од описаних начина
7. 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.*