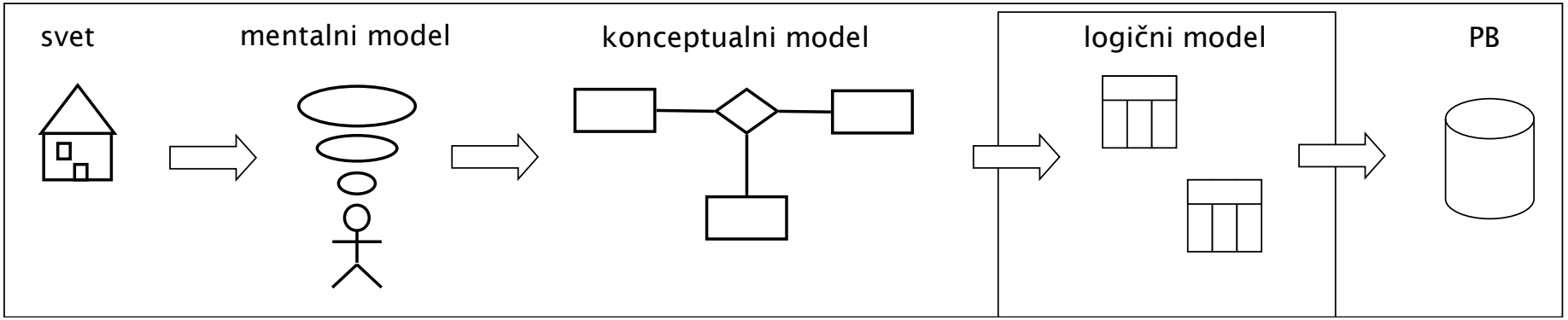
Poglavje 2

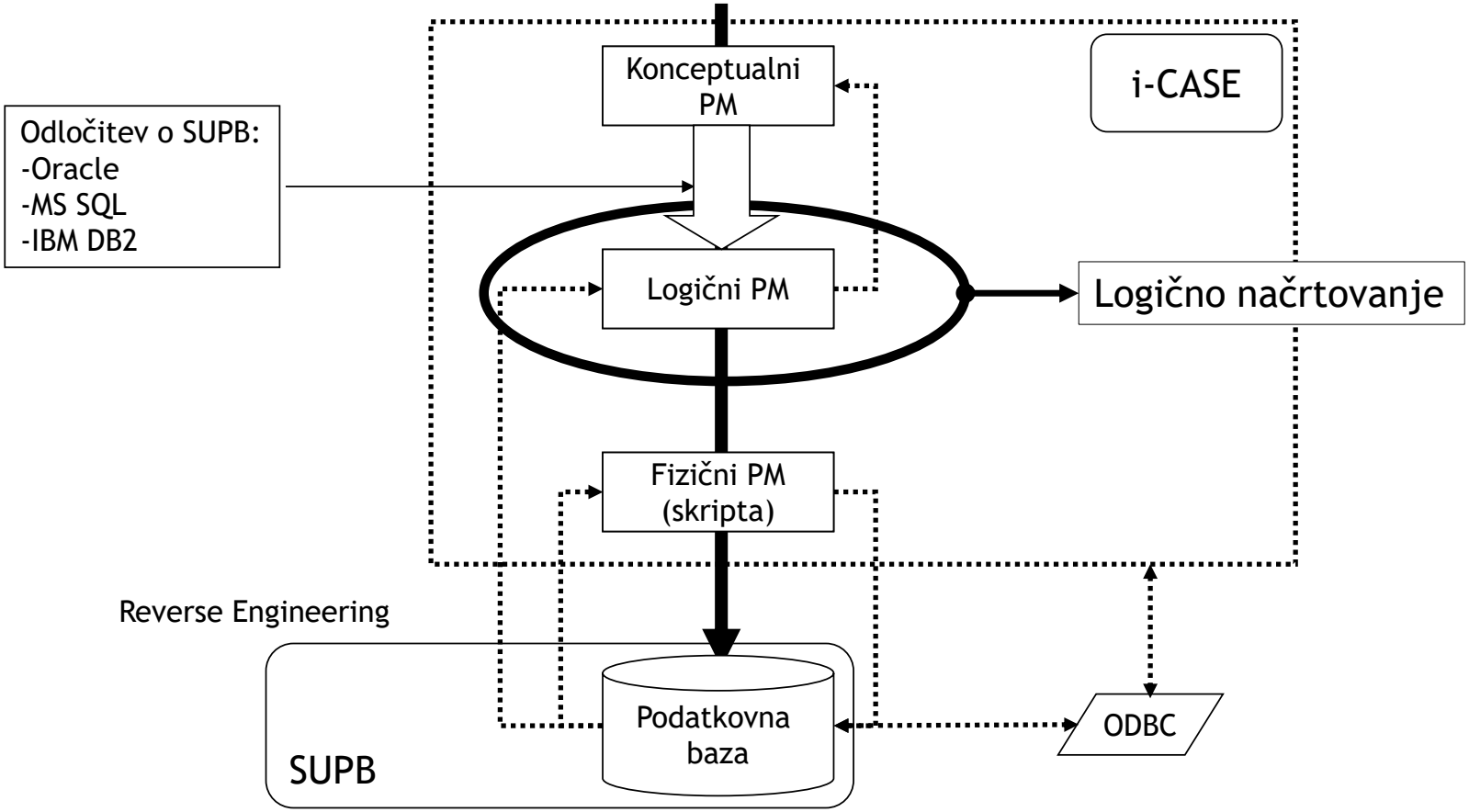
Logično in fizično načrtovanje relacijske podatkovne baze

Logično načrtovanje podatkovne baze

- Logično načrtovanje oz. modeliranje podatkovne baze nastopi za konceptualnim modeliranjem.
- Osnova logičnega modela je jezik, ki je razumljiv ciljnemu SUPB.
- Če izberemo relacijski SUPB, potem na logičnem nivoju govorimo o relacijskem modelu.



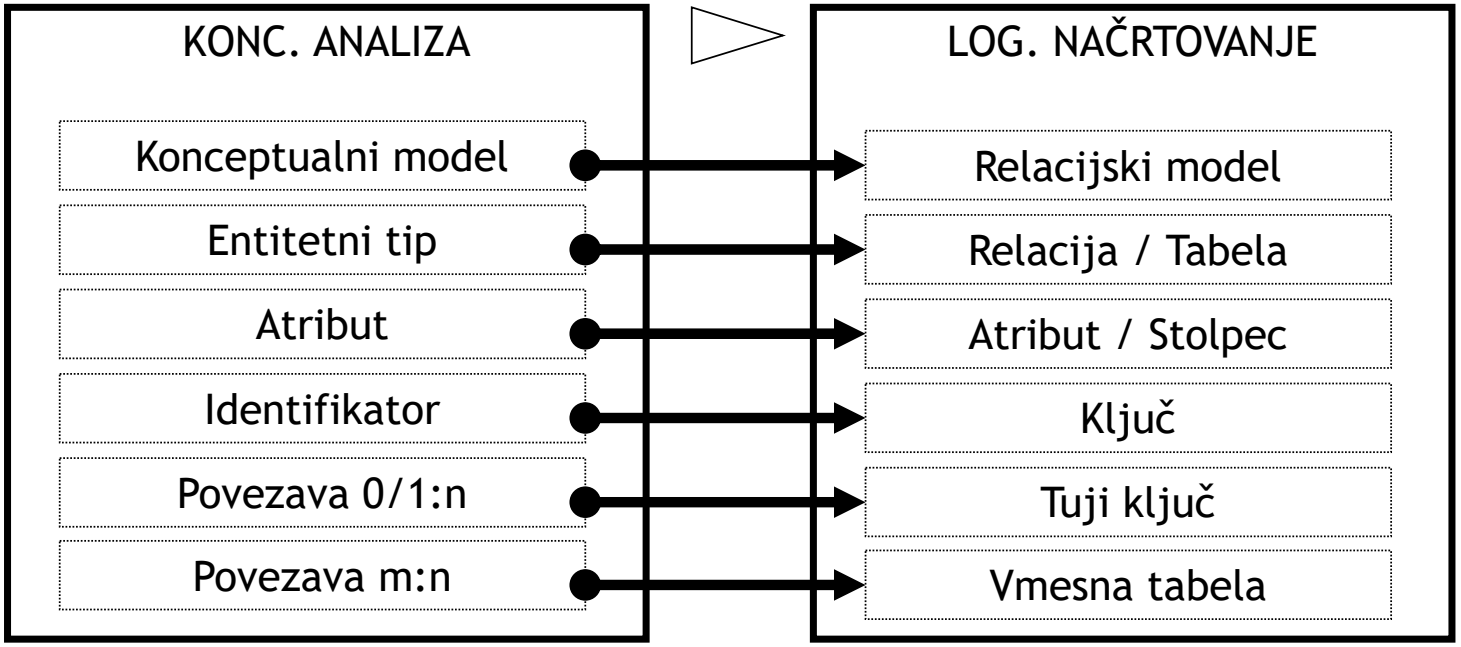
Podpora orodij CASE



Prehod iz konceptualnega v logični model..

- Prehod iz konceptualnega v logični model je navadno avtomatiziran s strani CASE orodij.

  Primer:
  vrsta baze: relacijska
  SUPB: PostgreSQL



Prehod iz konceptualnega v logični model

- Zakaj moramo izbrati SUPB
  - Podatkovni model (relacijski, ...)
  - Značilnosti in zmogljivosti SUPB, potrebne za prehod
- Načelno obstaja tudi metodologija načrtovanja logične podatkovne baze, vendar je dandanašnji le redko priporočljivo začeti z načrtovanjem na logičnem nivoju
  - Izjema: preproste PB, do okvirno 10 relacij oz. tabel

Prehod iz konceptualnega v logični model

- Možni koraki logičnega načrtovanja:
  - K2.1 Preslikaj gradnike konceptualnega modela v gradnike logičnega modela (običajno relacijske sheme)
  - K2.2 Preveri normalne oblike relacij (najmanj 3. NO)
  - K2.3 Preveri relacije z uporabniškimi transakcijami
  - K2.4 Preveri integritetne omejitve
  - K2.5 Preveri logični model skupaj s končnim uporabnikom
  - K2.6 (opcijsko) Združi logične podatkovne podmodele v globalni model
  - K2.7 Preveri skalabilnost modela

K2.1 Preslikava elementov konceptualnega modela v elemente logičnega modela (relacije)

- Preslikajo se:
  - močni entitetni tip
  - šibki entitetni tip
  - razmerja
  - hierarhije

  } enolično

  } ni enolično
- Preslikava v relacijske sheme (glave tabel)

K2.2 – Preveri relacije z normalizacijo...

- Namen tega koraka je preveriti, če so vse pridobljene relacije v ustrezni normalni obliki. To zagotavlja:
  - Da imajo relacije minimalno, vendar zadostno število atributov za potrebe problemske domene;
  - Da ni odvečnih podatkov (razen za potrebe povezovanja)
- Prevedba konceptualnega modela v logični model navadno da relacije, ki ustrezajo 3NO.
  - Če to ne drži, so v konceptualnem modelu ali v postopku prevedbe napake.

Intuitivna normalizacija

1NO tabele predstavljajo entitetne tipe

2NO vsaka tabela predstavlja natanko en entitetni tip

3NO tabele ne vsebujejo atributov vključenih (podrejenih) entitetnih tipov

4NO tri (ali več)-mestne relacije niso predstavljene kot pari dvomestnih relacij

K2.3 – Preveri relacije z vidika transakcij

- Podobno kot konceptualni model preverimo tudi logični model z vidika podpore transakcij, ki jih uporabnik specificira (glej K1.8).
- Če vseh transakcij ni moč izvesti ročno, smo pri pretvorbi naredili napako, ki jo je potrebno odpraviti.

K2.4 – Preveri integritetne omejitve ...

- V tem koraku preverimo pravila za zagotavljanje celovitosti podatkov:
  - Obveznost atributov
  - Omejitve domen atributov
  - Številnost
  - Omejitve entitet (celovitost entitet)
  - Omejitve povezav (celovitost povezav)
  - Splošne omejitve

K2.5 – Preveri model z uporabnikom...

- Namen tega koraka je preveriti model z uporabnikom ter ugotoviti, če ustreza vsem uporabniškim zahtevam.
- Model lahko zajema več uporabniških pogledov. Pri pregledu lahko nastopa več uporabnikov.
- Primeren način za pregled celovitosti podatkovnega modela je specifikacija podatkovnih tokov s pomočjo diagrama podatkovnih tokov.

K2.6 – Združi lokalne modele...

- Namen tega koraka je združiti vse lokalne modele v en globalni model, ki predstavlja vse uporabniške vidike podatkovne baze.
- Čeprav so lokalni modeli preverjeni, lahko pri njihovem združevanju pride do prekrivanja in neskladnosti.
- Globalni model preverimo podobno kot smo preverjali lokalne modele.
- Če pri načrtovanju nismo zajeli več uporabniških vidikov, lahko korak preskočimo.
- Koraki:
  - K2.6.1 – Lokalne modele združi v globalni model
  - K2.6.2 – Preveri globalni model
  - K2.6.3 – Globalni model preveri z uporabniki

K2.7 – Preveri možnosti za razširitve

- V primeru, da so predvidene bodoče razširitve sistema, moramo preveriti, če logični model take razširitve podpira.
- Podatkovni model mora biti prilagodljiv; omogočati mora razširitve skladno z novimi zahtevami ter z minimalnim vplivom na obstoječe uporabnike.
- Popolnoma odprt sistem za razširitve je težko doseči.

Terminologija pri relacijskem modelu..

- Relacijo si predstavljamo kot dvodimenzionalno tabelo s stolpci (atributi) in vrsticami (elementi).
  - Velja za logično strukturo podatkovne baze in ne za fizično (fizična PB: datotečni zapisi).
- Atribut je poimenovani stolpec relacije.
- Domena je množica dovoljenih vrednosti enega ali več atributov, ki so vključeni v to domeno.

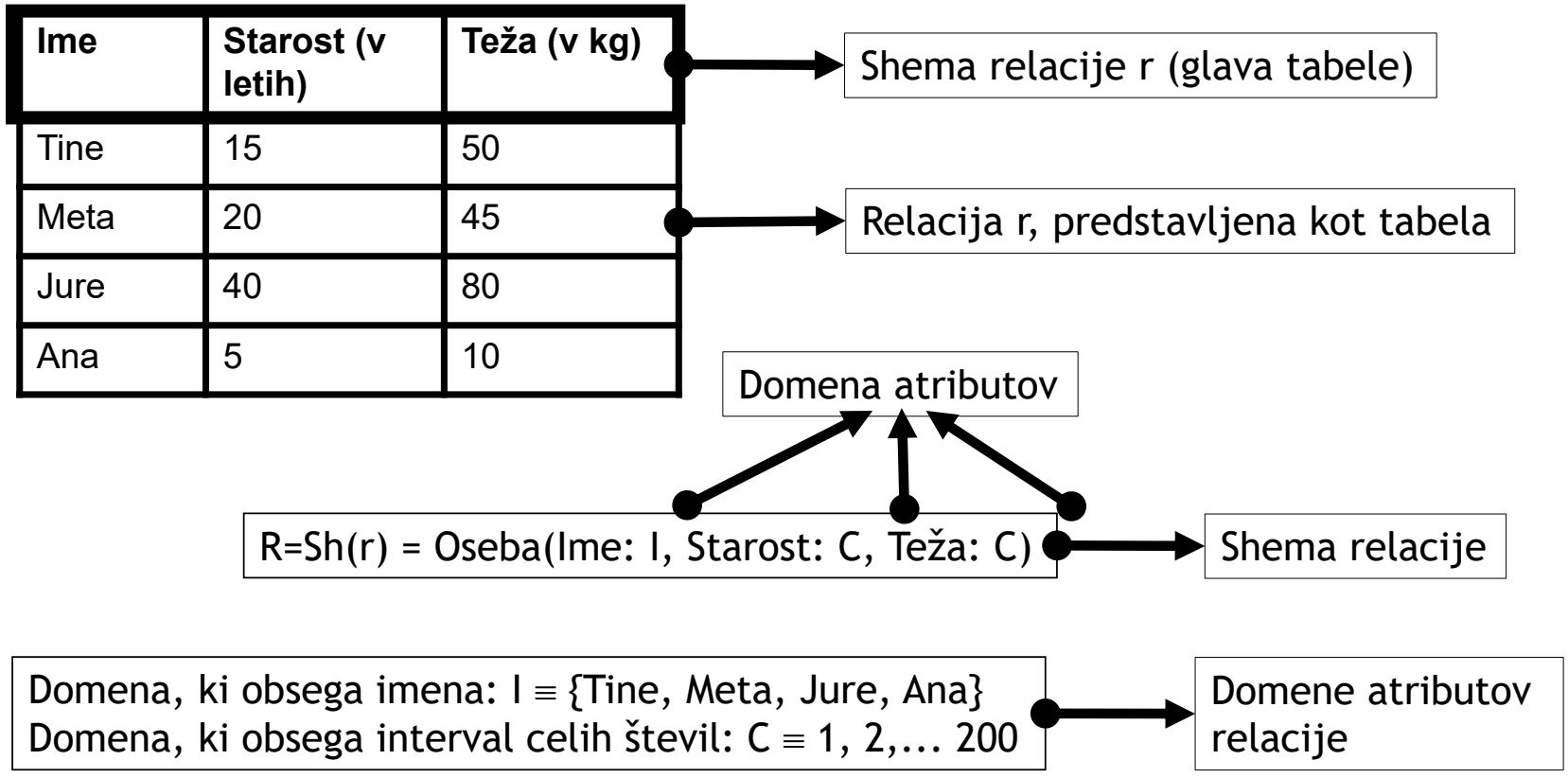
Terminologija pri relacijskem modelu

- N-terica je ena vrstica (element) v relaciji.
- Stopnja relacije je število atributov v relaciji.
- Številnost relacije je število n-teric (elementov) relacije.
- Relacijska podatkovna baza je množica normaliziranih relacij z enoličnimi imeni.

Primeri domen atributov

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Relacijska shema



Lastnosti relacij..

- Ime relacije je enolično. V podatkovni bazi ni dveh relacij z enakim imenom.
- Vsaka celica tabele, ki predstavlja relacijo, vsebuje največ (ali natanko) eno atomarno vrednost.
- Vsak atribut relacije ima enolično ime. V isti relaciji ni dveh atributov, ki bi imel isto ime.
- Vrednosti nekega atributa so vse iz iste domene.

Lastnosti relacij

- Vsaka n-terica relacije je enolična → v relaciji ni dveh enakih n-teric (teoretično je relacija množica).
- Vrstni red atributov v relaciji je nepomemben.
- Vrstni red n-teric v relaciji je nepomemben.

Primer relacije, ki ni niti v 1. NO

Ime	Starost (v letih), teža (v kg)
Tine	S15_T50
Meta	S20_T45
Jure	S40_T80
Ana	S6_T10

Celice ne vsebujejo atomarnih vrednosti

Sodobni SUP: JSON (slovar)

Zakonca	Leto poroke (celo število)
Tine, Meta	1995
Ana, Jure	1980

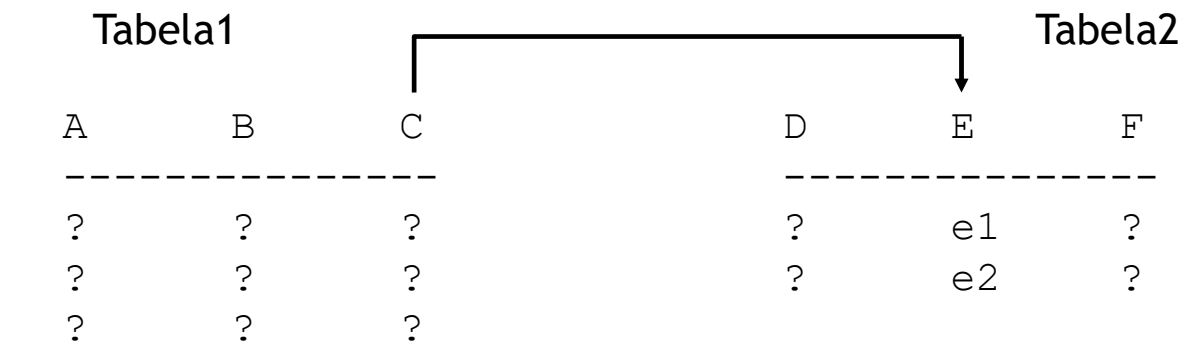
Celice vsebujejo več vrednosti

Sodobni SUP: JSON (seznam)

Pojem tujega ključa

- Tuji ključ je referenčna omejitev vrednosti enega ali več atributov relacije
- Pomen: atribut lahko zavzame le vrednosti, ki jih zavzame nek drug (točno določen) atribut iz druge relacije
- Pogosta uporaba pri implementaciji razmerij
- Oznaka v relacijski shemi s predpono #
  npr. #EMSO

Primer za tuji ključ



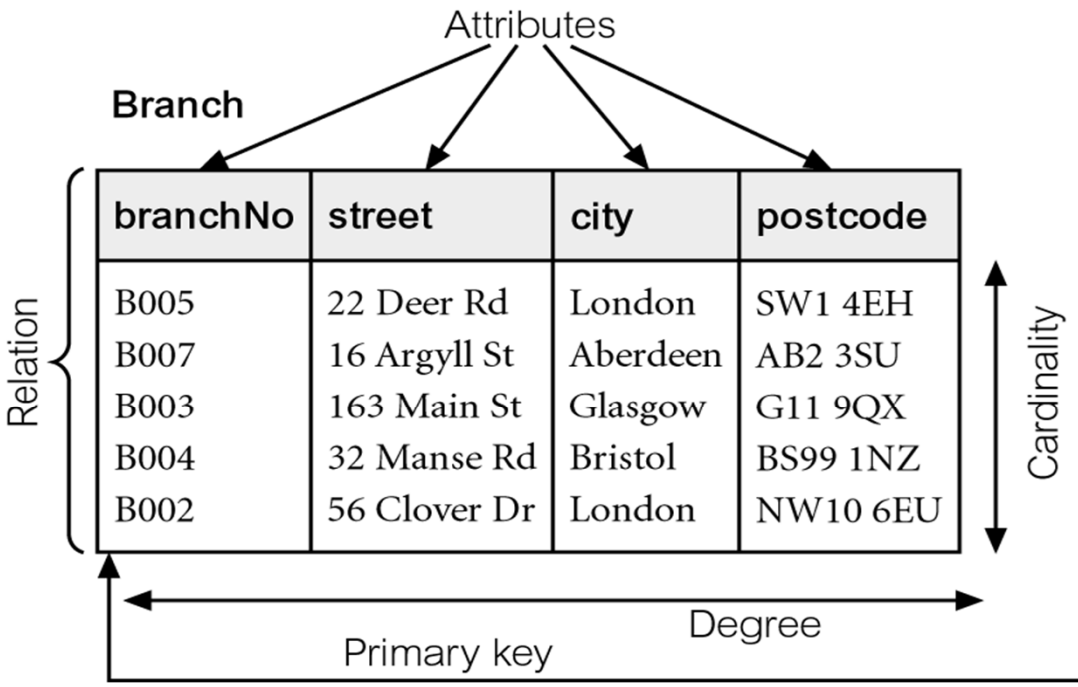

```
CREATE TABLE Tabela1 (
       A INTEGER,
       B INTEGER,
       C CHAR(2) FOREIGN KEY REFERENCES Tabela2(E)
);
```

Atribut C lahko zavzame le vrednosti atributa E, torej e1 in e2!

`Tabela1(A, B, #C)`

Primerki relacije:

Branch (oddelek)
Staff (zaposlen)



Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

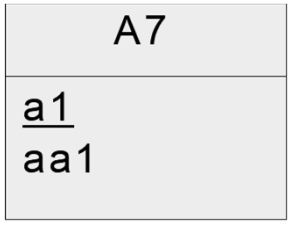
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Preslikava konceptualnih gradnikov v logične

Gradnik konceptualnega modela		**Gradnik logičnega modela (v našem primeru relacijskega)**
entitetni tip	→	relacija (tabela)
entiteta	→	element relacije (vrstica v tabeli)
atribut	→	atribut relacije (stolpec v tabeli)
entitetni identifikator	→	primarni ključ
povezava 1:n	→	referenca in tuji ključ
povezava m:n	→	vmesna relacija (tabela) in povezavi 1:n
šibki entitetni tip	→	relacija (tabela), tuji ključ je del primarnega
hierarhija	→	relacije (tabele) in povezave, ni enolično

Preslikava močnega entitetnega tipa

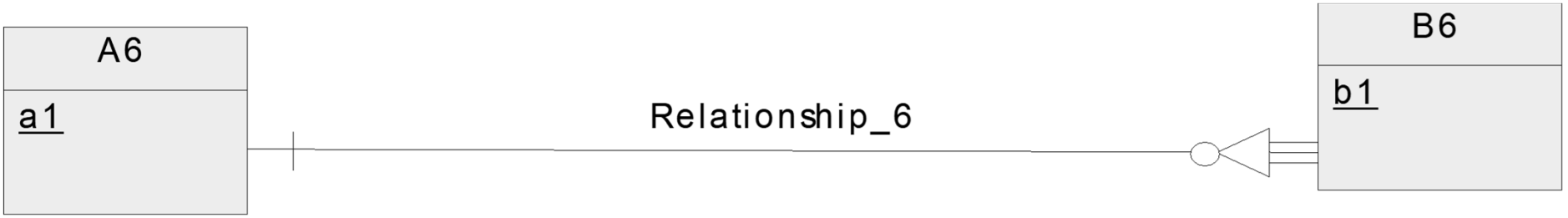
- Preslika se v istoimensko relacijsko shemo z vsemi njegovimi atributi
- Ključ sheme je identifikator močnega entitetnega tipa
- Shemi se potencialno dodajo še atributi, ki implementirajo razmerja (tuji ključi)



A7(<u>a1</u>,aa1)

Preslikava šibkega entitetnega tipa

- Preslika se v istoimensko relacijsko shemo z vsemi njegovimi atributi in identifikatorji entitetnih tipov, od katerih je odvisen
- Ključ sheme je identifikator šibkega entitetnega tipa in identifikatorji entitetnih tipov, od katerih je odvisen
- Shemi se potencialno dodajo še atributi, ki implementirajo druga razmerja (tuji ključi)

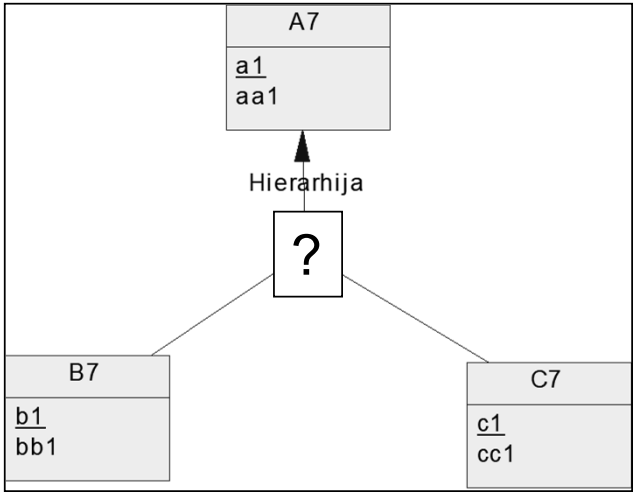


A6(a1)
B6(b1,#a1)

Preslikava hierarhije entitetnih tipov

- Ni enoličnega recepta glede migracije atributov. Uporabljamo dva načelna principa:
  - Združevanje podtipov
  - Ohranjanje podtipov
- Kdaj uporabiti kateri pristop? Odvisno od potreb in vrste pokrivanja, lahko podamo naslednja priporočila:
  - Totalno prekrivno: združevanje (celotna hierarhija v eno samo relacijo)
  - Delno prekrivno: združevanje (ena relacija za nadtip in druga za vse podtipe skupaj. Identifikator nadtipa migriramo k relaciji podtipov.)
  - Totalno ekskluzivno: ohranjanje (po ena relacija za vsak podtip kombiniran z nadtipom, relacija nadtipa je redundantna)
  - Delno ekskluzivno: ohranjanje (po ena relacija za vsak podtip in za nadtip. Identifikator nadtipa migriramo k vsem relacijam podtipov.)
- To niso pravila ampak zgolj priporočila!

Združevanje podtipov



Delno prekrivno:

A7(a1,aa1)

B7C7(#a1,b1,bb1,c1,cc1)

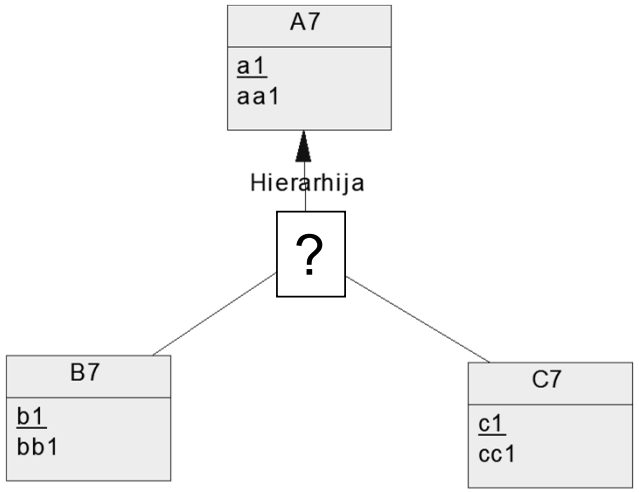
Totalno prekrivno:

A7(a1,aa1,b1,bb1,c1,cc1)

b1, bb1, c1, cc1 so parcialni atributi

#a1: tuji ključ

Ohranjanje podtipov



Delno ekskluzivno:

A7(a1,aa1)

B7(#a1,b1,bb1)

C7(#a1,c1,cc1)

#a1: tuji ključ

Totalno ekskluzivno:

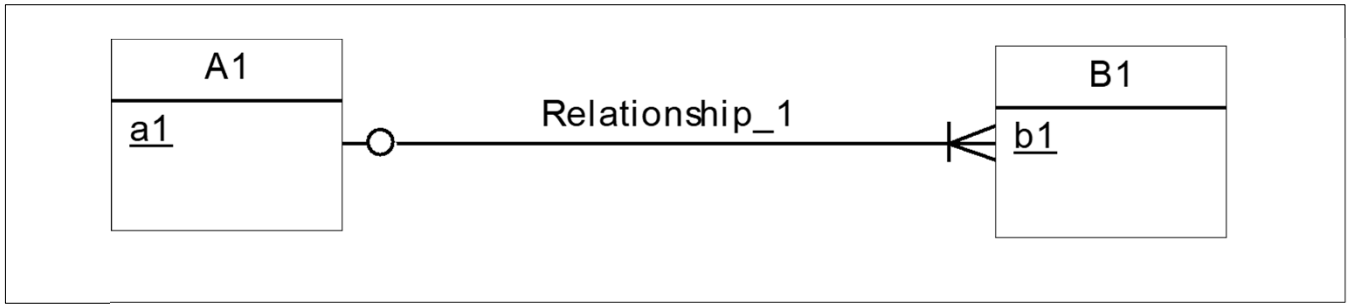
A7B7(a1,aa1,b1,bb1)

A7C7(a1,aa1,c1,cc1)

Preslikava razmerij

- Odvisno od kardinalnosti razmerij
  - Razmerja (0/1,1):(0/1,n) se preslikajo v tuji ključ
  - Razmerja (0/1,1):(0/1,1) se preslikajo v tuje ključe
  - Razmerja (0/1,m):(0/1,n) se preslikajo v šibek entitetni tip (odvisen od obeh entitetnih tipov v razmerju) in nato v relacijsko shemo
- Načeloma bi lahko vsakemu razmerju priredili svojo relacijsko shemo in v končni fazi dobili množico večinoma dvostolpčnih tabel, a se v praksi temu izogibamo z uporabo tujih ključev

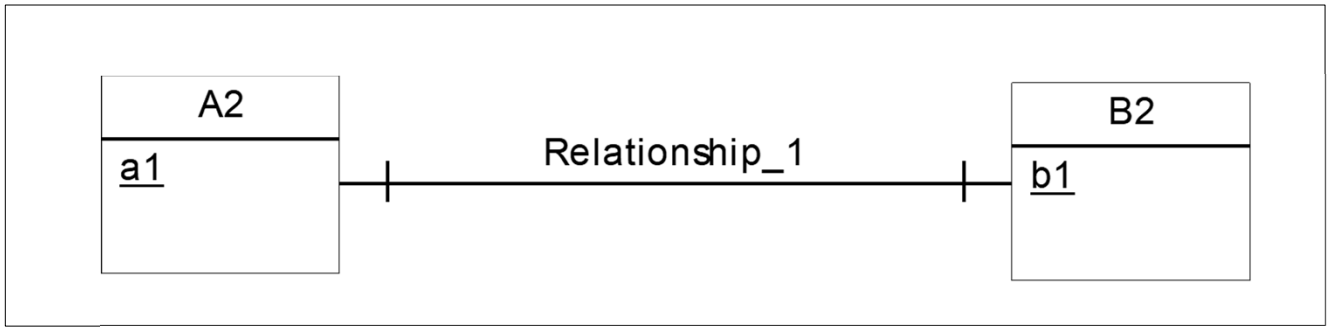
Razmerja (0/1,1):(0/1,n)



- Identifikator *dominantnega* entitetnega tipa - tistega pri ponoru delnega razmerja s kardinalnostjo (0/1,1) – v našem primeru A1 - se kot tuji ključ prenese na drugi entitetni tip (B1)

A1(a1) B1(b1,#a1)

Razmerja (1,1):(1,1)



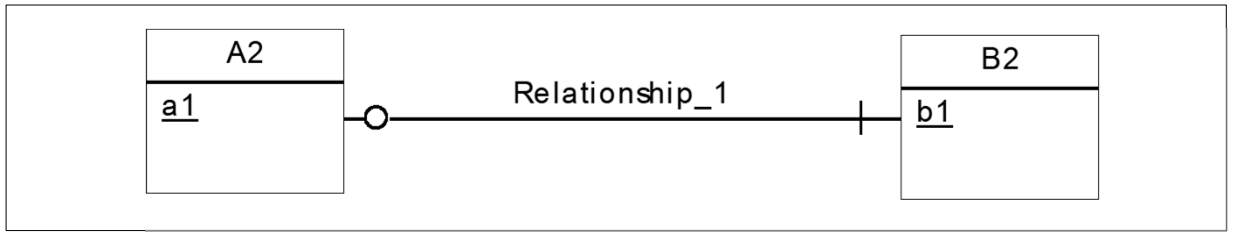
- Razmerje (1,1):(1,1) lahko odpravimo tako da - če je smiselno - združimo s tem razmerjem povezana entitetna tipa in izberemo primeren identifikator (ključ):

  A2B2(a1, b1)

- Če ne, pa dobimo:

  A2(a1, #b1) B2(b1, #a1)

Razmerja (0/1,1):(0/1,1)



- V splošnem se identifikatorja entitetnih tipov kot tuja ključa preneseta z drugega na drugi entitetni tip

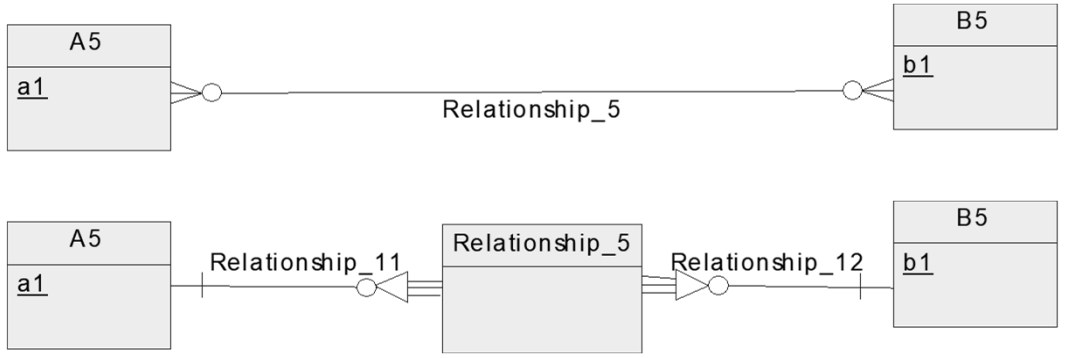
  A2(a1,#b1) B2(b1,#a1)

- Kadar lahko določimo *dominanten* entitetni tip se njegov identifikator kot tuji ključ prenese na drugi entitetni tip

  A2(a1) B2(b1,#a1) -- A je dominanten

- Dominanten tip je tisti pri izvoru delnega razmerja s kardinalnostjo (1,1), v gornjem primeru je to A2, lahko pa ga določimo tudi sami

Razmerja (0/1,m):(0/1,n)



- Na mestu razmerja se (ročno ali avtomatsko) ustvari nov – šibek – entitetni tip, odvisen od obeh entitetnih tipov v razmerju, ki ga ustrezno preslikamo.

A5(a1) **B5(b1)** **A5_B5(#a1,#b1)**

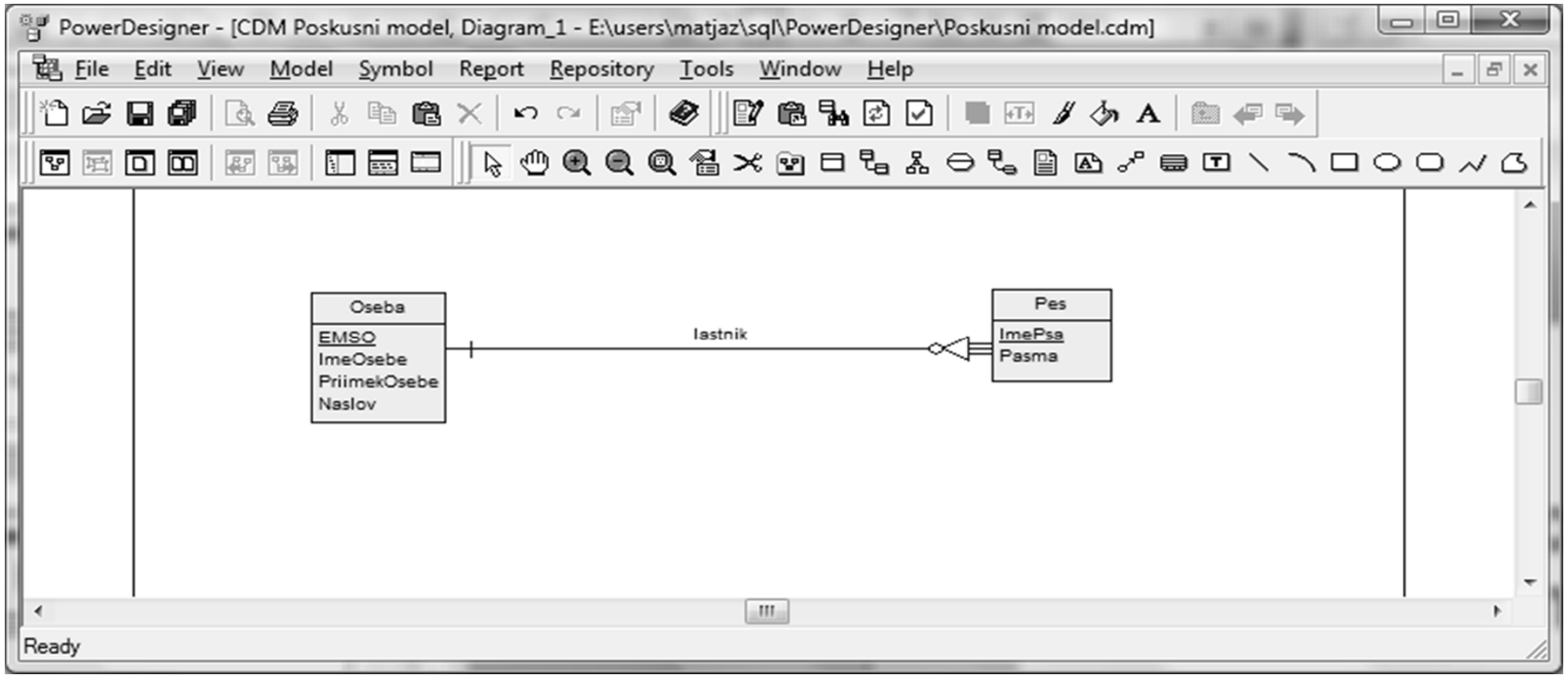
Končni rezultat preslikave

- Množica relacij z določenimi primarnimi ključi
- Povezave med relacijami določene s tujimi ključi
- Opcijsko: določeni pogledi kot navidezne relacije
- Pretvorba v tabele oz. poglede je trivialna (običajno avtomatizirana)
- Pomoč z orodji CASE: npr. Power Designer; pogojno MySQL Workbench

- Kaj še manjka:
  - potrditev, ali je dobljena struktura tabel primerna
  - preverjanje upoštevanja omejitev (poslovnih pravil)
- Pri tem si lahko pomagamo s pojmom normalnih oblik tabel

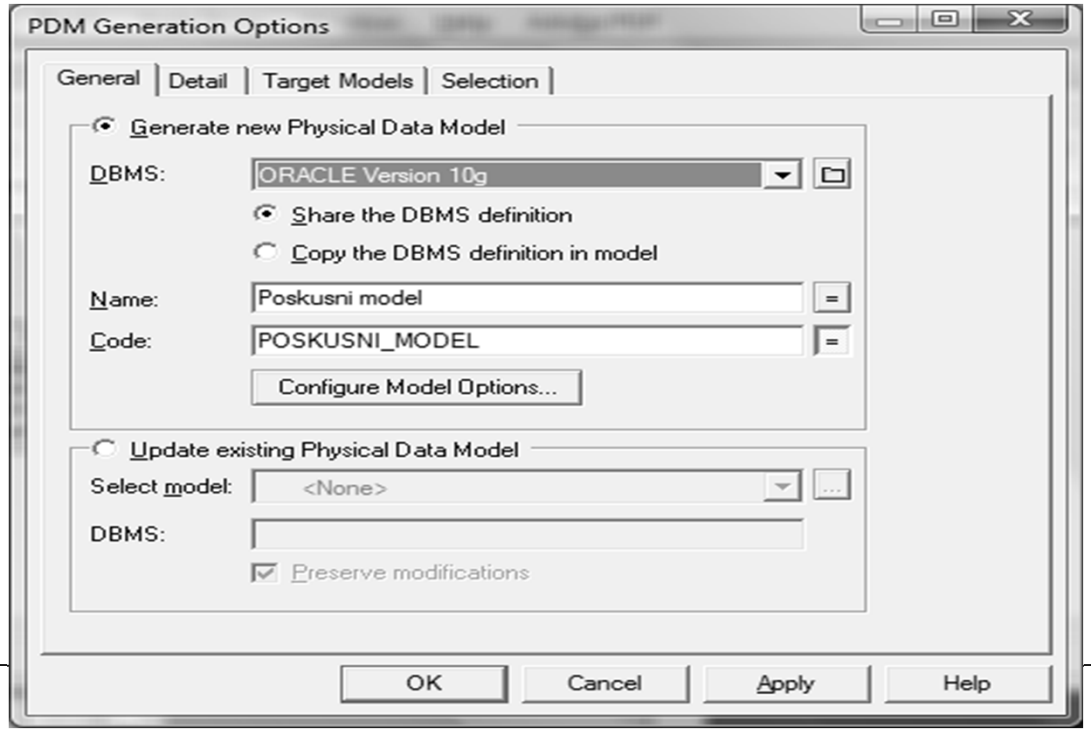
Gradnja konceptualnega modela s pomočjo Power Designerja

- Z uporabo ustrezne strategije zgradimo model.
- Paleta orodij je v posebnem oknu, lahko jo premaknete v glavno okno.



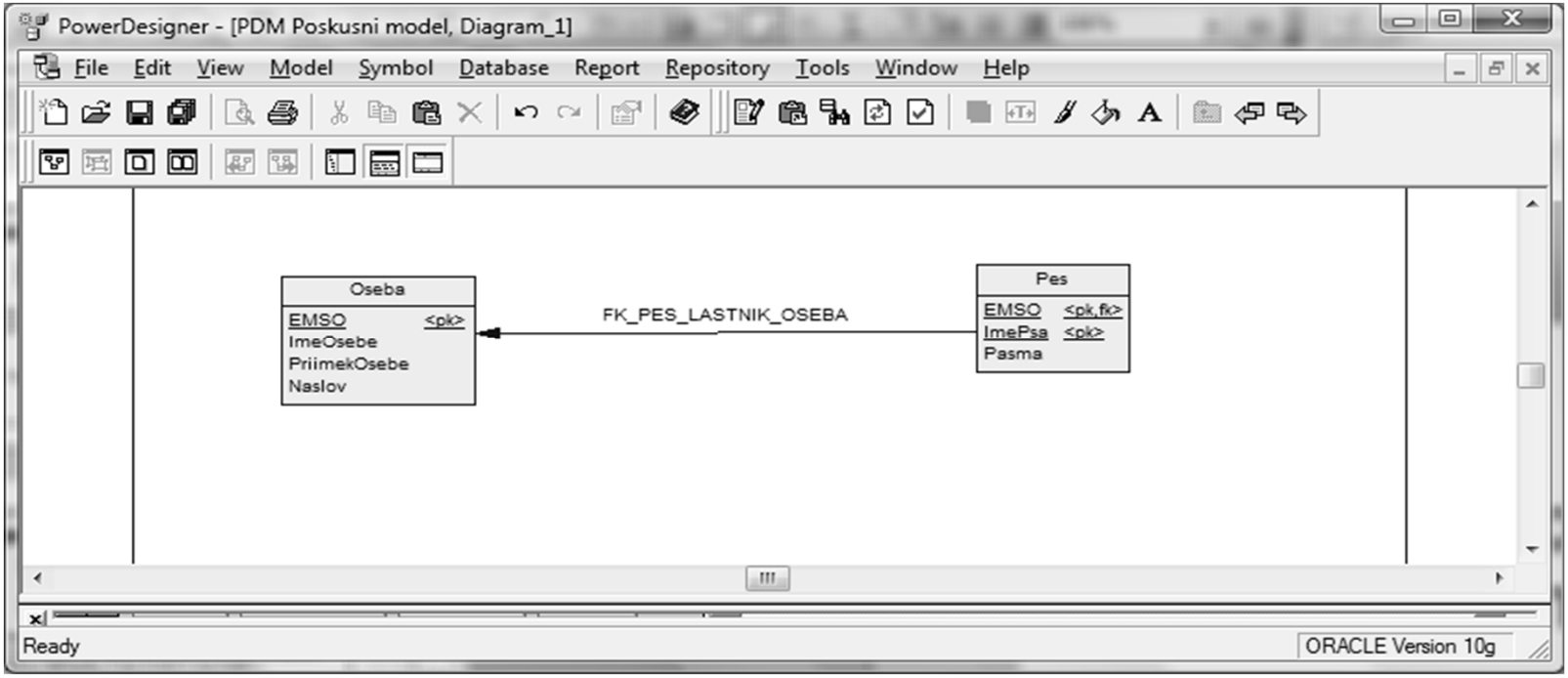
Pretvorba v logični model s pomočjo Power Designerja

- Pretvorba v logični model (Power designer mu pravi fizični):
  `Tools -> Generate Physical Data Model`…
- Pred pretvorbo moramo logični model poimenovati in izbrati SUPB, kjer bomo implementirali bazo



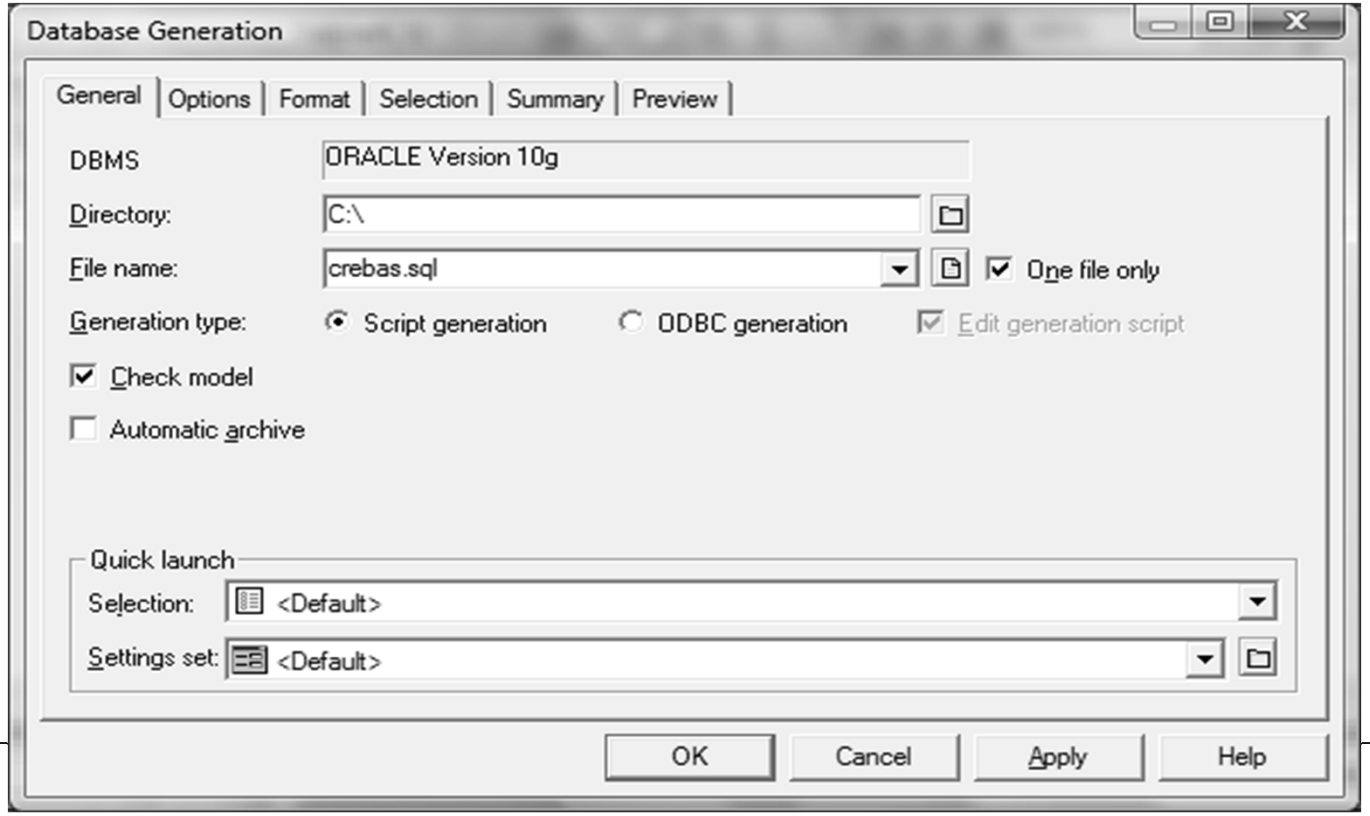
Pretvorba v logični model s pomočjo Power Designerja

- Dobljeni relacijski shemi:
  Oseba(EMSO, ImeOsebe, PriimekOsebe, Naslov)
  Pes(#EMSO, ImePsa, Pasma)



Pretvorba v fizični model (SQL skripto) s pomočjo Power Designerja

- Ko dobimo logični model, se pojavi menu **`Database`**. Izberemo **`Database -> Generate database`**…
  in vpišemo ime SQL skripte, ki se bo zgenerirala.



Pretvorba v fizični model (SQL skripto) s pomočjo Power Designerja

```
create table OSEBA  (
   EMSO                 NUMBER(13)                    not null,
   IMEOSEBE             VARCHAR2(20)                  not null,
   PRIIMEKOSEBE         VARCHAR2(20)                  not null,
   NASLOV               VARCHAR2(20)                  not null,
   constraint PK_OSEBA primary key (EMSO)
);

create table PES  (
   EMSO                 NUMBER(13)                    not null,
   IMEPSA               VARCHAR2(20)                  not null,
   PASMA                VARCHAR2(20)                  not null,
   constraint PK_PES primary key (EMSO, IMEPSA)
);

alter table PES
   add constraint FK_PES_LASTNIK_OSEBA foreign key (EMSO)
      references OSEBA (EMSO);
```

Pretvorba v logični in fizični model (SQL skripto) s pomočjo Power Designerja

- Pazite na terminološko razliko:

Predavanja, vaje	PowerDesigner
Konceptualni model	**Generate conceptual model**
Logični model	**Generate physical model**
Fizični model	**Generate database**

- OPOMBA: "logical model" v PD15 in novejših je nekaj med našim konceptualnim in logičnim modelom (npr. več-več razmerja pretvorjena v entitetne tipe, tuji ključi, specifikacija indeksov, ...)

K2.2

Normalizacija in podatkovne baze

- Spada na širše področje načrtovanja PB
- Samostojna uporaba primerna za manj kompleksne PB (do cca. 10 tabel)
- Imamo problem iz realnega sveta, opisan z atributi združenimi v relacije in omejitvami vrednosti atributov.
- Normalizacija odgovarja na vprašanja:
  - ali je **obstoječa** struktura relacij oz. tabel v PB primerna za obravnavo danega problema (pasivna vloga normalizacije)
  - kako **združiti atribute v tabele**, da dobimo najprimernejšo strukturo PB (aktivna vloga normalizacije)

Modeliranje omejitev s funkcionalnimi odvisnostmi

- Relacija je model nekega stanja v svetu, torej njena vsebina ne more biti poljubna.
- Realne omejitve ne dovoljujejo, da bi bili odnosi v svetu kakršnikoli; možna so le določena stanja (tudi: poslovna pravila).
- Omejitve v relacijskem modelu formalno opišemo s pomočjo odvisnosti
- Odvisnosti so sredstvo, s katerim v relacijskem podatkovnem modelu povemo, katere vrednosti relacij (komb. vrednosti atributov v vrsticah) so veljavne in katere sploh ne morejo obstajati

Funkcionalne odvisnosti..

- Predpostavimo, da obstaja relacijska shema R z množico atributov, katere podmnožici sta X in Y.
- V relacijski shemi R velja X → Y
  (X funkcionalno določa Y oziroma Y je funkcionalno odvisen od X),
  če v nobeni relaciji, ki pripada shemi R, ne obstajata dve n-terici (vrstici), ki bi se ujemali v vrednostih atributov X in se ne bi ujemali v vrednostih atributov Y.
- Preprosto povedano, **obstaja** neka funkcija f, s pomočjo katere lahko iz vrednosti X izračunamo vrednosti Y = f(X).

Primeri funkcionalnih odvisnosti

- Imamo relacijo s shemo

  Stevila(X,Y,Z)

  z naslednjim pomenom:

  X in Y sta celi števili, Z pa njuna vsota

- Funkcionalne odvisnosti relacijske sheme so:

  $F \equiv \{ XY \rightarrow Z, XZ \rightarrow Y, YZ \rightarrow X\}$

- S pomočjo funkcionalnih odvisnosti definiramo tudi pojem ključa, ki implementira določene integritetne omejitve.

Primeri funkcionalnih odvisnosti

- Imamo relacijo s shemo

  Izpit(VpŠt, Priimek, Ime, ŠifraPredmeta, Datum izpita,
  OcenaPisno, OcenaUstno)

  z naslednjim pomenom:

  Študent z vpisno številko VpŠt ter priimkom Priimek in imenom Ime je na DatumIzpita opravljal izpit iz predmeta s šifro ŠifraPredmeta. Dobil je oceno OcenaPisno in OcenaUstno.

- Funkcionalne odvisnosti relacijske sheme Izpit so:

  $F \equiv \{$ VpŠt $\rightarrow$ (Priimek, Ime), (VpŠt, ŠifraPredmeta,
  DatumIzpita) $\rightarrow$ (OcenaPisno, OcenaUstno) $\}$

Iskanje naravnih ključev relacije na podlagi definiranih funkcionalnih odvisnosti

- Izhajamo iz relacijske sheme R (množica atributov) in nad njo definirane množice funkcionalnih odvisnosti F
- Pomožni pojmi (zaprtje množice atributov)
- Različni postopki (algoritmi) za določanje enega ali vseh naravnih ključev
- Zakaj te postopke potrebujemo (integritetne omejitve, normalizacija)
- Kompaktna notacija zapisa: $X \rightarrow Y$
  X in Y sta množici atributov

Zaprtje (closure) množice atributov

- Zaprtje X^+ množice atributov X glede na množico funkcionalnih odvisnosti F
- X^+ = {A: A lahko izračunamo iz X s pomočjo odvisnosti iz F}
- Postopek za izračun X^+

```
Vhod: X, F
Izhod: X+

X+ = X
ponavljaj
    stari X+ = X+
    za vsako odvisnost Y → Z ∈ F naredi
          če Y ⊆ X+ potem
                X+ = X+ ∪ Z
dokler ni stari X+ = X+
```

Primer izračuna zaprtja

R=ABCDEFG

F={A$\rightarrow$B, BE$\rightarrow$G, EF$\rightarrow$A, D$\rightarrow$AC}

Iščemo $\{EF\}^+$:

1. $\{EF\}^+ = EFA$
2. $\{EFA\}^+ = EFAB$
3. $\{EFAB\}^+ = EFABG$
4. $\{EFABG\}^+ = EFABG$

5. Končni rezultat: $\{EF\}^+ = EF^+ = EFABG$

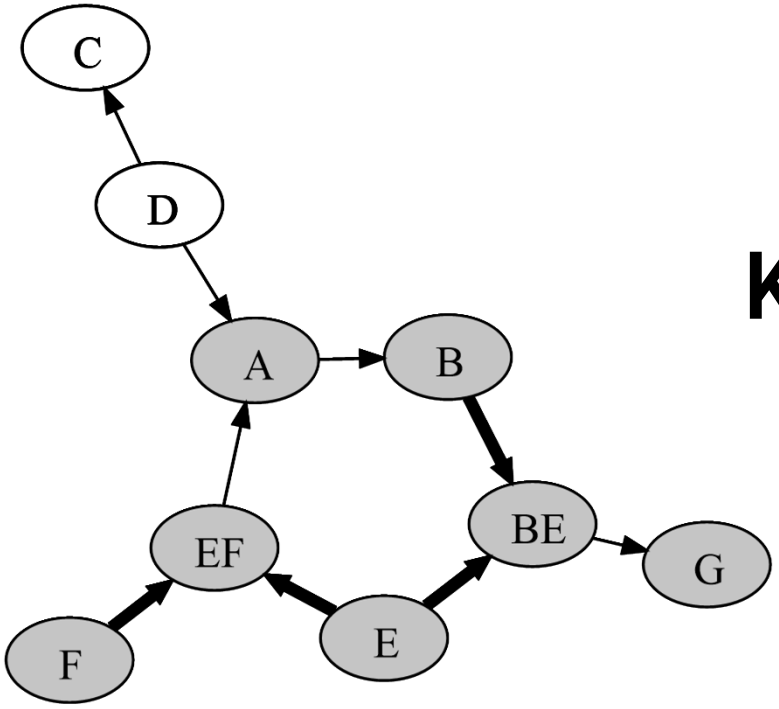
Ali je EF ključ ali vsak kandidat za ključ? Ne, ker $\{EF\}^+ \subset R$.

Za vsakega kandidata za ključ K namreč velja: $K^+=R$.

Primer izračuna zaprtja

R=ABCDEFG
F={A→B, BE→G, EF→A, D→AC}

$\{EF\}^+$



KONEC!

Iskanje naravnih ključev relacije na podlagi podanih funkcionalnih odvisnosti

- Splošni veljavne resnice
  - Atribut, ki ne nastopa na desni strani nobene funkcionalne odvisnosti, mora biti vsebovan v vsakem ključu
  - Atribut, ki nastopa na desni strani neke funkcionalne odvisnosti in ne nastopa na levi strani nobene funkcionalne odvisnosti, ne more biti vsebovan v nobenem ključu
  - Dobri kandidati za ključe so leve strani funkcionalnih odvisnosti in njihove unije

Elmasari-Navathe algoritem za določanje enega ključa

- Vhod: relacijska shema R,
  množica funkcionalnih odvisnosti F

1. Postavi K= začetni kandidat, npr. R (vsi atributi)
2. Za vsak atribut $X \in K$
   Izračunaj $\{K-X\}^+$ glede na F
   Če $\{K-X\}^+ = R$ (vsebuje vse atribute R)
   postavi K= K- {X}
3. Kar ostane v K je ključ.

- Problem: vrne samo en ključ, odvisen od vrstnega reda pregledovanja atributov

Primer (Elmasari-Navathe):

R=ABCDEFG

F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}

- K=ABCDEFG, X=A
  K-X= BCDEFG, $(K\text{-}X)^+$= BCDEFG
  manjka A
- K=ABCDEFG, X=B
  K-X= ACDEFG, $(K\text{-}X)^+$= ABCDEFG ($AG\rightarrow B$)
- K=ACDEFG, X=C
  K-X= ADEFG, $(K\text{-}X)^+$= ABDEFG ($AG\rightarrow B$)
  manjka C
- K=ACDEFG, X=D
  K-X= ACEFG, $(K\text{-}X)^+$= ABCDEFG ($AG\rightarrow B$, $A\rightarrow D$)
- K=ACEFG, X=E
  K-X= ACFG, $(K\text{-}X)^+$= ABCDEFG ($AG\rightarrow B$, $A\rightarrow D$,$B\rightarrow E$)
- K=ACFG, X=F
  K-X= ACG, $(K\text{-}X)^+$= ABCDEFG ($AG\rightarrow B$, $A\rightarrow D$,$B\rightarrow E$,$E\rightarrow F$)
- K=ACG, X=G ($A\rightarrow D$)
  K-X= AC, $(K\text{-}X)^+$= ACD manjkajo BEFG
-

Ključ je
ABCDEFG - BDEF
torej
ACG

Saiedian-Spencer algoritem za določanje vseh ključev

- Vhod: relacijska shema R,
  množica funkcionalnih odvisnosti F

1. Poišči množice $\mathcal{L}$ (atributi samo na levi strani odvisnosti in atributi ki ne nastopajo v nobeni odvisnosti), $\mathcal{R}$ (atributi samo na desni strani odvisnosti) in $\mathcal{B}$ (atributi na levi in desni strani odvisnosti)

2. Preveri množico $\mathcal{L}$. Če $\mathcal{L}^+$=R, je edini ključ in lahko končaš, sicer nadaljuj na koraku 3.

3. Preveri množico $\mathcal{B}$ tako da v $\mathcal{L}$ vstavljaš po vrsti vse možne podmnožice atributov X iz $\mathcal{B}$, začenši s posameznimi atributi. Kadar dobimo $\{\mathcal{L} \cup X\}^+$ =R, smo našli ključ. N-teric, ki vsebujejo X, dalje ne obravnavamo več.

Primer (Saiedian-Spencer):

R=ABCDEFG

F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}

1. $\mathcal{L}$= CAGBE – BGE = CA
   $\mathcal{R}$= DBGEF – BGE = DF
   $\mathcal{B}$ = BGE

2. $\mathcal{L}^+$=CAD $\subseteq$ R

3. X=B $\mathcal{L}$=CAB
   $\mathcal{L}^+$=CABDGEF = R

4. X=G $\mathcal{L}$=CAG
   $\mathcal{L}^+$=CAGDBEF = R

5. X=E $\mathcal{L}$=CAE
   $\mathcal{L}^+$=CAEDBFG = R

Ključi: ABC, ACG, ACE

Integritetne omejitve in ključi

- Integritetna omejitev v splošnem zagotavlja smiselno vsebino podatkov – njihovo celovitost
- Primarni ključ ne služi le potrebam enolične identifikacije vrstic
- Primarni (in tudi vsi alternativni) ključi zagotavljajo tudi integritetne omejitve enoličnosti vključenih atributov, kar zagotavlja možnost enolične identifikacije tudi v prihodnje
- To še posebej velja za naravne ključe.

Normalizacija

- Problemi zaradi redundance podatkov v osnovnih relacijah.
- Namen normalizacije.
- Uporabnost normalizacije pri načrtovanju relacijske podatkovne baze.
- Postopek normalizacije.
- Normalne oblike glede na restriktivnost in stopnjo neredundantnosti:
  - I. normalna oblika
  - II. normalna oblika
  - III. normalna oblika
  - Boyce-Coddova normalna oblika
  - IV. normalna oblika (večvrednostne odvisnosti)
  - V. normalna oblika (stične odvisnosti)
  - VI. Normalna oblika (ni netrivialnih stičnih odvisnosti, upošteva časovni okvir)

Kaj je normalizacija

- Normalizacija je postopek, s katerem pridemo do množice primernih (primerno strukturiranih) relacij, ki ustrezajo potrebam uporabe
- Nekaj lastnosti primernih relacij:
  - Relacije imajo minimalen nabor atributov → zgolj tiste, ki so potrebni za pokritje potreb poslovnega sistema;
  - Atributi, ki so logično povezani, so zajeti v isti relaciji;
  - Med atributi relacij je minimalna redundanca → vsak atribut (razen tujih ključev) je predstavljen samo enkrat.

Načrtovanje PB in normalizacija

- Osnovni cilj načrtovanja relacijske podatkovne baze je smiselno grupirati atribute v relacije na način, da bo med podatki čim manj redundance.
- Potencialne koristi pravilnega načrtovanja so:
  - Spremembe podatkov v podatkovni bazi dosežemo z minimalnim številom operacij → večja učinkovitost; manj možnosti za podatkovne nekonsistentnosti.
  - Manjše potrebe po diskovnih kapacitetah za shranjevanje osnovnih relacij → manjši stroški.
- Normalizacija (oziroma upoštevanje njenih principov) je pomemben korak v postopku načrtovanja.

Primer nenormalizirane relacije

- Relacija Osebje ima odvečne podatke.

Ime	Priimek	Oddelek	Naslov oddelka
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6

Odvečni (ponavljajoči se) naslovi.

Ažurirne anomalije

- Relacije, ki vsebujejo odvečne podatke lahko povzročajo ažurirne anomalije pri operacijah nad podatki.
- Poznamo več vrst anomalij:
  - Anomalije pri dodajanju n-teric (vrstic) v relacijo
  - Anomalije pri brisanju n-teric (vrstic) iz relacije
  - Anomalije pri spreminjanju n-teric (vrstic) v relaciji

Anomalije pri dodajanju vrstic

- Dodajanje novih članov oddelka: ponovno moramo (pravilno) vpisati naslov oddelka
- Dodajanje novega oddelka: za podatke o članu vpišemo NULL; problem: kaj je ključ?

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6
Janko	Jankovič	1A	Tržaška 52
NULL	NULL	3A	Celovška 12

Anomalije pri brisanju vrstic

- Brisanje edinega člana oddelka: izgubimo tudi vse informacije o tem oddelku (šifra oddelka, naslov)

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
~~Marija~~	~~Kovač~~	~~2A~~	~~Dunajska 6~~

Anomalije pri spreminjanju vrstic

- Oddelek 1A se preseli na Večno pot 113. Naslov je treba pravilno popraviti pri vseh članih oddelka!

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6

Postopek normalizacije

- Postopku preoblikovanja relacij v obliko, pri kateri do ažurirnih anomalij ne more priti, pravimo normalizacija.
- Obstaja več stopenj normalnih oblik (form):
  - I. normalna oblika,
  - II. normalna oblika,
  - III. normalna oblika,
  - Boyce-Coddova normalna oblika
  - IV. normalna oblika
  - V. normalna oblika
  - VI. normalna oblika.

Prva normalna oblika..

- Relacija je v prvi normalni obliki, če:
  - Nima večvrednostnih atributov, kar pomeni, da ima vsak atribut lahko le eno vrednost (torej vrednost ne more biti množica). Primer: telefonska številka (fiksna, mobilna, službena)
  - Nima sestavljenih atributov (torej vrednost ne more biti relacija). Primer: naslov (ulica, hišna številka, pošta, kraj, država)
  - Ima definiran primarni ključ in določene funkcionalne odvisnosti
- Koraki normalizacije:
  - Eliminiranje ponavljajočih skupin (večvrednostne in sestavljene atribute skupno obravnavamo kot ponavljajoče skupine)
  - Določitev funkcionalnih odvisnosti
  - Določitev primarnega ključa

Prva normalna oblika..

- Primer relacijske sheme:
  - Študent(VŠ, priimek, ime, pst, kraj, (šifra predmeta, naziv, ocena))
- Določimo primarni ključ:
  - VŠ
- Odpravimo ponavljajoče skupine:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(šifra predmeta, naziv, ocena)
- Določimo primarna ključa obeh relacijskih shem:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(#VŠ, šifra predmeta, naziv, ocena)

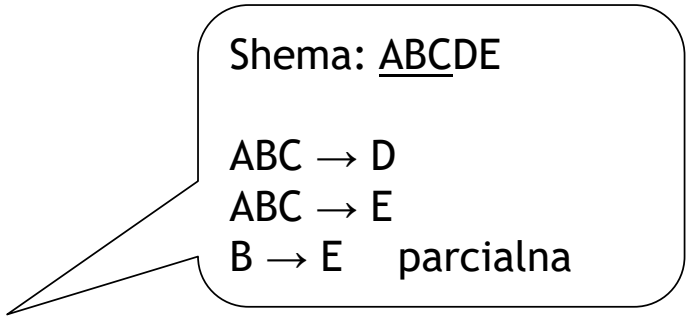
Ključ glavne relacije se kot tuji ključ prenese k ponavljajočim skupinam in postane del njihovega primarnega ključa.

Prva normalna oblika

- Gledamo obe relacijski shemi …
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(#VŠ, šifra predmeta, naziv, ocena)
- … in določimo funkcionalne odvisnosti:
  - VŠ → Ime
  - VŠ → Priimek
  - VŠ → pst
  - VŠ → kraj
  - Pst → kraj

  - Šifra predmeta → naziv
  - VŠ, šifra predmeta → naziv
  - VŠ, šifra predmeta → ocena

Druga normalna oblika..



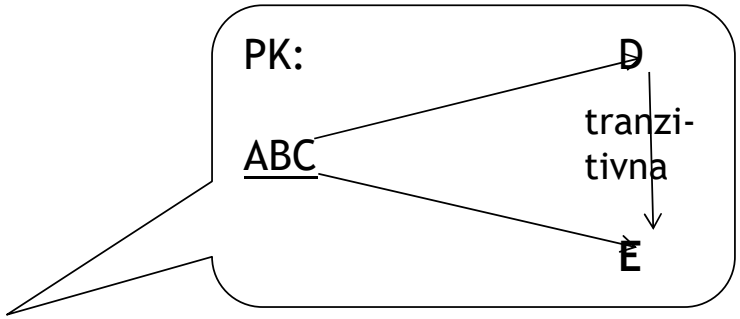
- Relacija je v drugi normalni obliki:
  - Če je v prvi normalni obliki
  - Ne vsebuje parcialnih (delnih) odvisnosti: noben atribut ni funkcionalno odvisen le od dela primarnega ključa, temveč le od celotnega ključa
  - Normalizacija: problematičnim (parcialnim) odvisnostim dodelimo nove relacijske sheme (vsaki svojo). Desne strani izločimo iz originalne sheme.
- Ugotovitev:
  - Relacija, katere primarni ključ je sestavljen le iz enega atributa, je v drugi normalni obliki
  - Relacija, katere primarni ključ je sestavljen iz vseh atributov, je v drugi normalni obliki

Druga normalna oblika

- Nadaljevanje predhodnega primera:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - PredmetOcena(#VŠ, šifra predmeta, naziv, ocena)
- Funkcionalne odvisnosti:
  - VŠ → Ime
  - VŠ → Priimek
  - VŠ → pst
  - VŠ → kraj
  - Pst → kraj
  - **Šifra predmeta → naziv**
  - VŠ, šifra predmeta → naziv
  - VŠ, šifra predmeta → ocena

- Relacijska shema Študent je v drugi normalni obliki
- Relacijsko shemo PredmetOcena normaliziramo :
  - Predmet_študent (#VŠ, #šifra predmeta, ocena)
  - Predmet (šifra predmeta, naziv)

Tretja normalna oblika..



- Relacija je v tretji normalni obliki:
  - Če je v drugi normalni obliki
  - Če ne vsebuje tranzitivnih funkcionalnih odvisnosti: ni funkcionalnih odvisnosti med atributi, ki niso del primarnega ključa oz. ne obstaja atribut, ki ni del primarnega ključa, ki bi bil funkcionalno odvisen od drugega atributa, ki ravno tako ni del primarnega ključa
- Ugotovitev:
  - Relacija, katere primarni ključ je sestavljen iz vseh atributov, je v tretji normalni obliki
  - Relacija, kjer le en atribut izmed vseh ni del primarnega ključa, je v tretji normalni obliki

Tretja normalna oblika

- Nadaljevanje predhodnega primera:
  - Študent(VŠ, priimek, ime, pst, kraj)
  - Predmet (šifra predmeta, naziv)
  - Predmet_študent (#VŠ, #šifra predmeta, ocena)
- Funkcionalne odvisnosti:
  - VŠ → Ime
  - VŠ → Priimek
  - **VŠ → pst**
  - **VŠ → kraj**
  - **Pst → kraj**
  - Šifra predmeta → naziv
  - VŠ, šifra predmeta → naziv
  - VŠ, šifra predmeta → ocena

- Relaciji
  Predmet
  in
  Predmet_Študent
  sta v tretji normalni obliki

- Relacija Študent:
  - Študent (VŠ, priimek, ime, #pst)
  - Pošta (pst, kraj)

Postopek normalizacije v 3. NO

- Normalizacija relacijske sheme R v ρ
  - $\rho = \{\}$
  - Vsaki problematični odvisnosti $X \rightarrow A \in F$ priredimo novo relacijsko shemo XA v ρ, razen v primeru, če že obstaja kakšna shem, ki XA vključuje kot podmnožico. Desno stran odvisnosti (A) izločimo iz originalne sheme.
  - Kar ostane od originalne relacijske sheme dodamo v ρ, razen če v ρ že obstaja kakšna shema, ki jo vsebuje

    Odvisnost je problematična, kadar je tranzitivna (3. NO) ali parcialna (2. NO)

Primer normalizacije v 3. NO

R=ABCDEFG
F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
Primarni ključ: ACG Alternativni ključi: ABC, ACE

1.	ρ={}		
2.	A $\rightarrow$D:	$\rho = \rho \cup$ {AD}	parcialna
3.	AG $\rightarrow$B:	ρ= {AD} $\cup$ {AGB}	parcialna
4.	B $\rightarrow$ G:	ni problematična	
5.	B $\rightarrow$ E:	ρ= {AD, AGB} $\cup$ {BE}	tranzitivna
6.	E $\rightarrow$ B:	{EB} $\subseteq$ {BE}	
7.	E $\rightarrow$ F:	ρ= {AD, AGB, BE} $\cup$ {EF}	tranzitivna
8.	Končamo:	ρ= {AD, AGB, BE, EF} $\cup$ {ACG}	

ρ= {AD, AGB, BE, EF, ACG}

Boyce-Coddova normalna oblika (BCNO)

- V primerjavi s 3.NO vpelje BCNO dodatne omejitve.
- Osnovne normalne oblike preverjajo parcialne (2.NO) in tranzitivne (3.NO) odvisnosti glede na primarni ključ. Ne preverjajo pa se odvisnosti glede na ostale kandidate za ključ (alternativne ključe).
- BCNO razširja omejitve na vse kandidate za ključ.
- Ugotovitev:
  - Če ima shema samo enega kandidata za ključ, potem je njena BCNO enaka 3.NO!

Boyce-Coddova normalna oblika (BCNO)

- Relacija R je v BCNO, če za vsako odvisnost $X \rightarrow A \in F_R$ velja vsaj eden izmed pogojev:

    1. $X \rightarrow A$ je trivialna odvisnost ($A \subseteq X$)
    2. X je (nad)ključ sheme R

- Normalizacija v BCNO je lahko izgubna; s stikom dobljenih relacij ne dobimo nazaj originalne relacije (neizgubni stik ne obstaja).

Postopek normalizacije v BCNO

- Dekompozicija relacijske sheme R v ρ
  - Izračunamo F_{min} (izločimo trivialne in tranzitivne odvisnosti)
  - Postavimo $\rho=\{\}$, S=R (množica preostalih atributov) in F = F_{min} (množica še veljavnih odvisnosti v S)
  - Vsaki problematični odvisnosti $X \rightarrow A \in F$ priredimo novo relacijsko shemo XA v ρ, razen v primeru, če že obstaja kakšna shem, ki XA vključuje kot podmnožico. Desno stran odvisnosti (A) izločimo iz originalne sheme in sproti prilagajamo množico odvisnosti!
    Napotek: v množicah S in F si sproti vodimo trenutno množico atributov in še veljavnih odvisnosti v njej!
  - Kar ostane od originalne relacijske sheme dodamo v ρ, razen če v ρ že obstaja kakšna shema, ki jo vsebuje

Primer normalizacije v BCNO

R=ABCDEFG

F_{min}={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}

Ključ: ACG, ABC, ACE

1. Začetek ρ={} **S=ABCDEFG**
   F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
2. A $\rightarrow$D: $\rho = \rho \cup$ {AD} **S=ABCEFG**
   F={ AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
3. AG $\rightarrow$B: ρ= {AD} $\cup$ {AGB} **S=ACEFG**
   F={E $\rightarrow$F}
4. E $\rightarrow$ F: ρ= {AD, AGB} $\cup$ {EF} **S=ACEG**
   F={}
5. Končamo: ρ= {AD, AGB, EF} $\cup$ {ACEG}

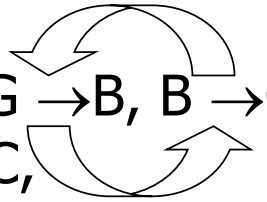
ρ= {AD, AGB, EF, ACEG}

Izgubili smo: B $\rightarrow$E, E $\rightarrow$B.

Primer normalizacije v BCNO

R=ABCDEFG

F_{min}={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}



Ključ: ACG, ABC, ACE

1. Začetek ρ={}
   S=ABCDEFG
   F={A $\rightarrow$D, AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
2. A $\rightarrow$D: $\rho = \rho \cup$ {AD}
   S=ABCEFG
   F={ AG $\rightarrow$B, B $\rightarrow$G, B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
3. B $\rightarrow$ G: ρ= {AD} $\cup$ {BG}
   S=ABCEF
   F={ B $\rightarrow$E, E $\rightarrow$B, E $\rightarrow$F}
4. B $\rightarrow$ E: ρ= {AD, BG} $\cup$ {BE}
   S=ABCF
   F={}
5. Končamo: ρ= {AD, BG, BE} $\cup$ {ABCF}

ρ= {AD, BG, BE, ABCF}

Izgubili smo: AG $\rightarrow$B, E $\rightarrow$F.

Ali so spodnje relacije v BCNO?

- Voznik (ime, priimek, stdov)
  stdov $\rightarrow$ ime, stdov $\rightarrow$ priimek
- Prekršek(stdov, datum, znesek)
  stdov,datum $\rightarrow$ znesek,
- Davek(stdov, davčna)
  stdov $\rightarrow$ davčna, davčna $\rightarrow$ stdov
- PrekrškekDavek (stdov, datum, znesek, davčna)
  stdov,datum $\rightarrow$ znesek, stdov $\rightarrow$ davčna, davčna $\rightarrow$ stdov

Ali je spodnja relacija v BCNO?

- Izpit(<u>student, predavatelj</u>, predmet, ocena)
  student,predavatelj $\rightarrow$ predmet
  student,predavatelj $\rightarrow$ ocena
  predmet $\rightarrow$ predavatelj

- Primarni ključ: student, predavatelj
- Sekundarni (alternativni) ključ: student, predmet
  student,predmet $\rightarrow$ predavatelj
  student,predmet $\rightarrow$ ocena

- Ali je relacija v 3.NO?
- Ali je relacija v BCNO?

Normalizacija v BCNO

- Izpit(student, predmet, ocena)
  Predavanje(predavatelj, predmet)
- Kako je s ključi in funkcionalnimi odvisnostmi?
- Primarni ključ: student, predavatelj
  student,predavatelj $\rightarrow$ predmet
  student,predavatelj $\rightarrow$ ocena
  predmet $\rightarrow$ predavatelj
- Sekundarni (alternativni) ključ: student, predmet
  student,predmet $\rightarrow$ predavatelj
  student,predmet $\rightarrow$ ocena
  predmet $\rightarrow$ predavatelj

Normalizacija v BCNO

- Izpit(<u>student, predmet</u>, ocena)
  student,predmet $\rightarrow$ ocena (izpeljana odvisnost v začetni shemi)

- Predavanje(predavatelj, <u>predmet</u>)
  predmet $\rightarrow$ predavatelj

Večvrednostne odvisnosti

- $A \twoheadrightarrow B$ (A večvrednostno določa B)
- Nastanejo tipično, ko nadomestimo večvrednostne atribute s kartezičnim produktom enovrednostnih
- $A \twoheadrightarrow B$ je trivialna, če:
  - $B \subseteq A$
  - $A \cup B = R$
- Obravnavamo samo netrivialne odvisnosti
- Primer: Oseba(<u>Ime</u>, Jezik, Šport)
  Ime $\twoheadrightarrow$ Jezik, Ime $\twoheadrightarrow$ Šport

Večvrednostne odvisnosti

Ime	Jezik	Šport
Janez	Ang	Tek
Janez	Nem	Plavanje
Janez	Nem	Tek
Janez	Ang	Plavanje

(Janez, (Ang,Nem), (Tek, Plavanje)

Ime ↠ Jezik
Ime ↠ Šport

4NO – četrta (poslovna) normalna oblika

- Relacija je v četrti normalni obliki, če je v Boyce-Coddovi normalni obliki in ne vsebuje netrivialnih večvrednostnih odvisnosti
- Postopek normalizacije: netrivialne večvrednostne odvisnosti zamenjamo z novimi relacijskimi shemami (kot v BCNO)

Primer 1: normalizacija v 4. NO

- Oseba(Ime, Jezik, Šport)
  Ime ↠ Jezik, Ime ↠ Šport

- Oseba(Ime)
  ImeJezik(Ime, Jezik)
  ImeŠport(Ime, Šport)

5. Normalna oblika

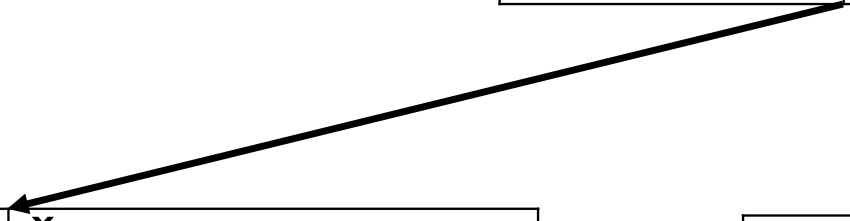
- Stične odvisnosti
- Nastanejo, ko je relacija R rezultat stika dveh ali več relacij
- Anomalije: po ažuriranju relacije R mora ta še vedno imeti obliko stika več relacij

- Relacija je v 5. NO, če je v četrti NO in ne vsebuje stičnih odvisnosti.
- Normalizacija na originalne stične tabele

5. Normalna oblika

Stične odvisnosti:
⋈(Univerza-Študij, Stopnja-Študij)

Univerza	Študij	Stopnja
Ljubljana	Računalništvo	1
Ljubljana	Računalništvo	2
Ljubljana	Bioinformatika	3
Maribor	Računalništvo	1
Maribor	Računalništvo	2



Univerza	Študij
Ljubljana	Računalništvo
Ljubljana	Bioinformatika
Maribor	Računalništvo

Študij	Stopnja
Računalništvo	1
Bioinformatika	3
Računalništvo	2

5. Normalna oblika

Študij računalništva se v **Ljubljani** začne izvajati tudi na **3. stopnji**.

?

Univerza	**Študij**	**Stopnja**
Ljubljana	Računalništvo	1
Ljubljana	Računalništvo	2
Ljubljana	Bioinformatika	3
Maribor	Računalništvo	1
Maribor	Računalništvo	2

Univerza	**Študij**
Ljubljana	Računalništvo
Ljubljana	Bioinformatika
Maribor	Računalništvo

Študij	**Stopnja**
Računalništvo	1
Bioinformatika	3
Računalništvo	2

5. Normalna oblika

Študij računalništva se v **Ljubljani** začne izvajati tudi na **3. stopnji**.

Univerza	**Študij**	**Stopnja**
Ljubljana	Računalništvo	1
Ljubljana	Računalništvo	2
Ljubljana	Bioinformatika	3
Maribor	Računalništvo	1
Maribor	Računalništvo	2
Ljubljana	Računalništvo	3

Univerza	**Študij**
Ljubljana	Računalništvo
Ljubljana	Bioinformatika
Maribor	Računalništvo

Študij	**Stopnja**
Računalništvo	1
Bioinformatika	3
Računalništvo	2
Računalništvo	3

5. Normalna oblika

Študij računalništva se v **Ljubljani** začne izvajati tudi na **3. stopnji**.

Univerza	**Študij**	**Stopnja**
Ljubljana	Računalništvo	1
Ljubljana	Računalništvo	2
Ljubljana	Bioinformatika	3
Maribor	Računalništvo	1
Maribor	Računalništvo	2
Ljubljana	Računalništvo	3
Maribor	Računalništvo	3

Univerza	**Študij**
Ljubljana	Računalništvo
Ljubljana	Bioinformatika
Maribor	Računalništvo

Študij	**Stopnja**
Računalništvo	1
Bioinformatika	3
Računalništvo	2
Računalništvo	3

6. Normalna oblika

- Definicija: stična odvisnost $\bowtie(X_1,X_2, \ldots , X_n)$ je trivialna, kadar vsaj en X_i vsebuje vse atribute
- 6. normalna oblika:
  - ni netrivialnih stičnih odvisnosti (torej je v 5. NO)
  - upošteva časovni okvir veljavnosti vrednosti atributov (del ključa)
  - posledica: vse relacije v obliki $R_i(\underline{ključ_i}, vrednost_i)$ ali $R_i(\underline{ključ_i})$, kar povzroči dodatno izgubo kompleksnejših odvisnosti (poslovnih pravil)
- Neformalno:
  - Relacijo poleg primarnega ključa sestavlja še največ en atribut.
  - Vsi atributi so obvezni