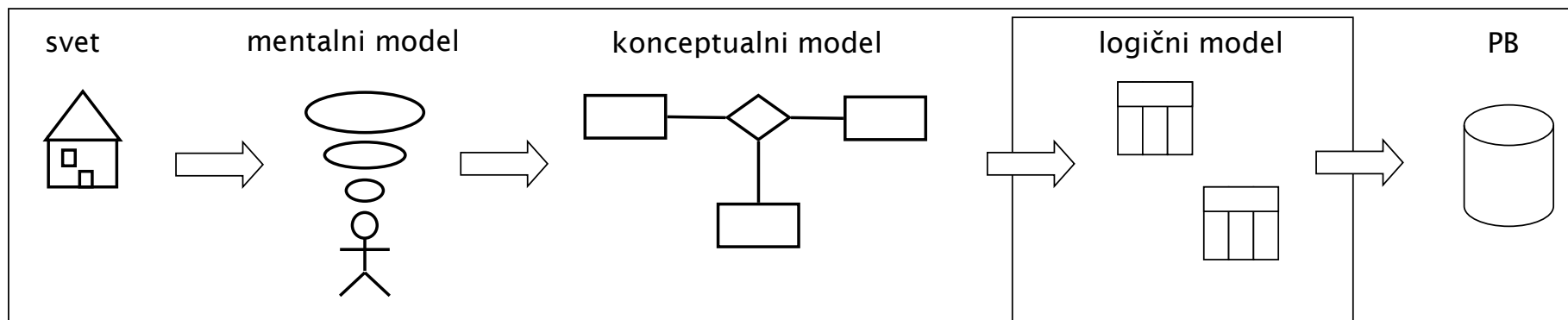


Poglavje 2

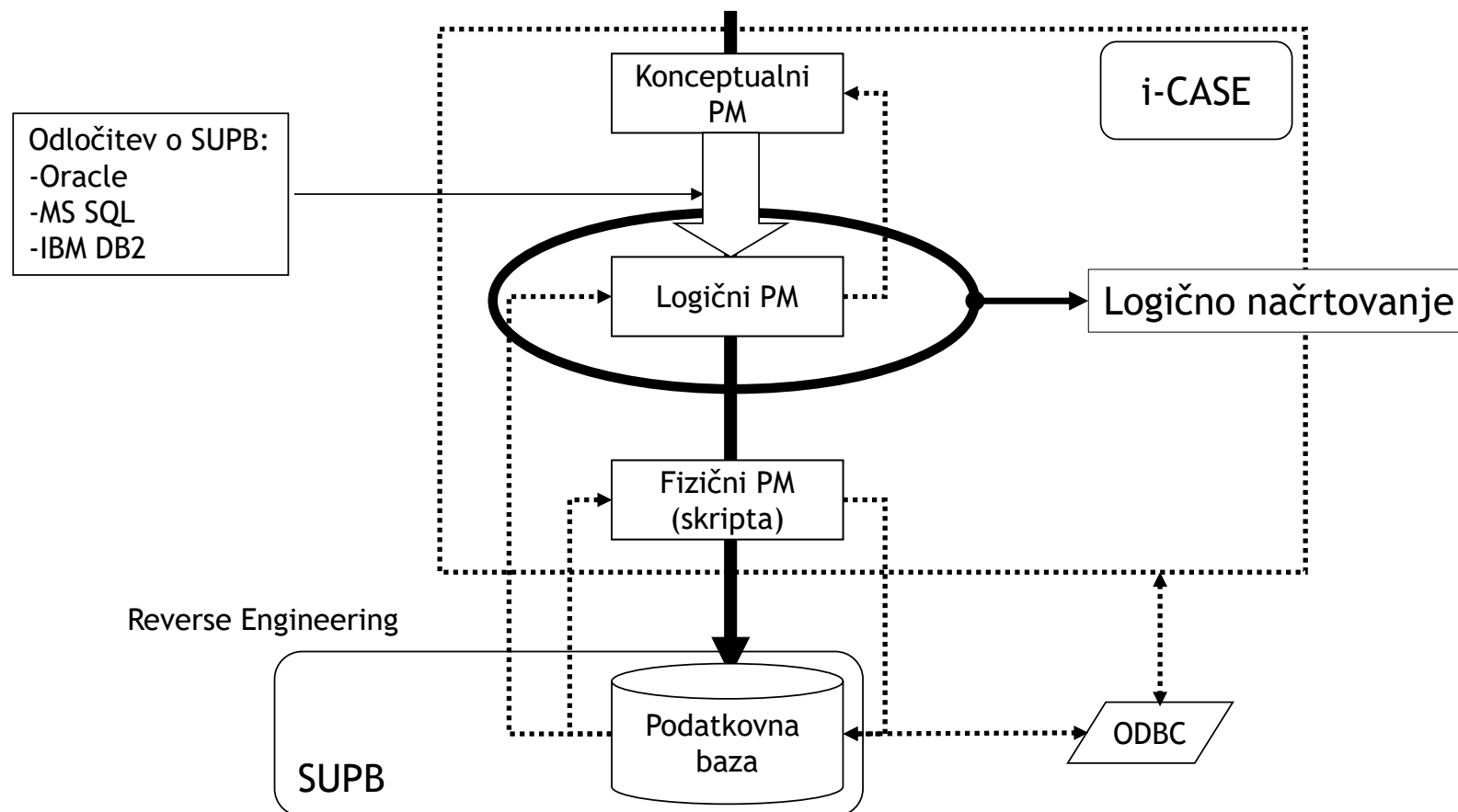
**Logično in fizično načrtovanje
relacijske podatkovne baze**

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



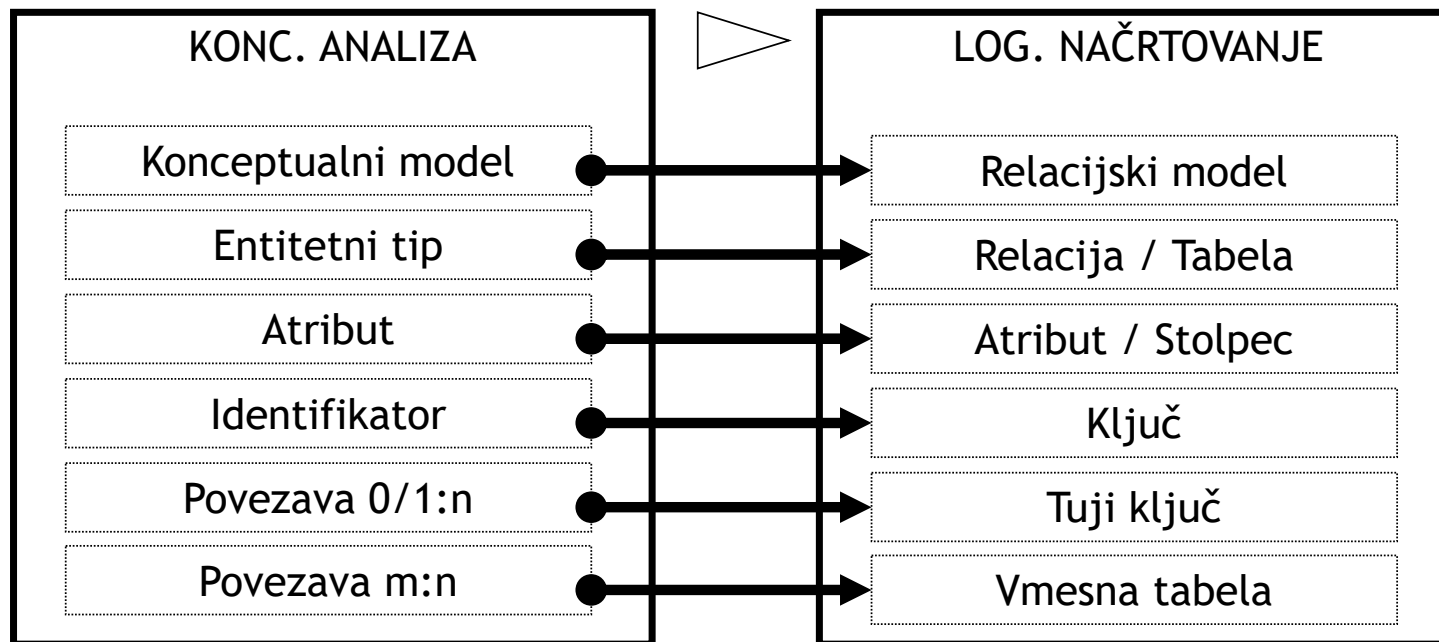
Prehod iz konceptualnega v logični model..

- 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
 - Števnost
 - 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.

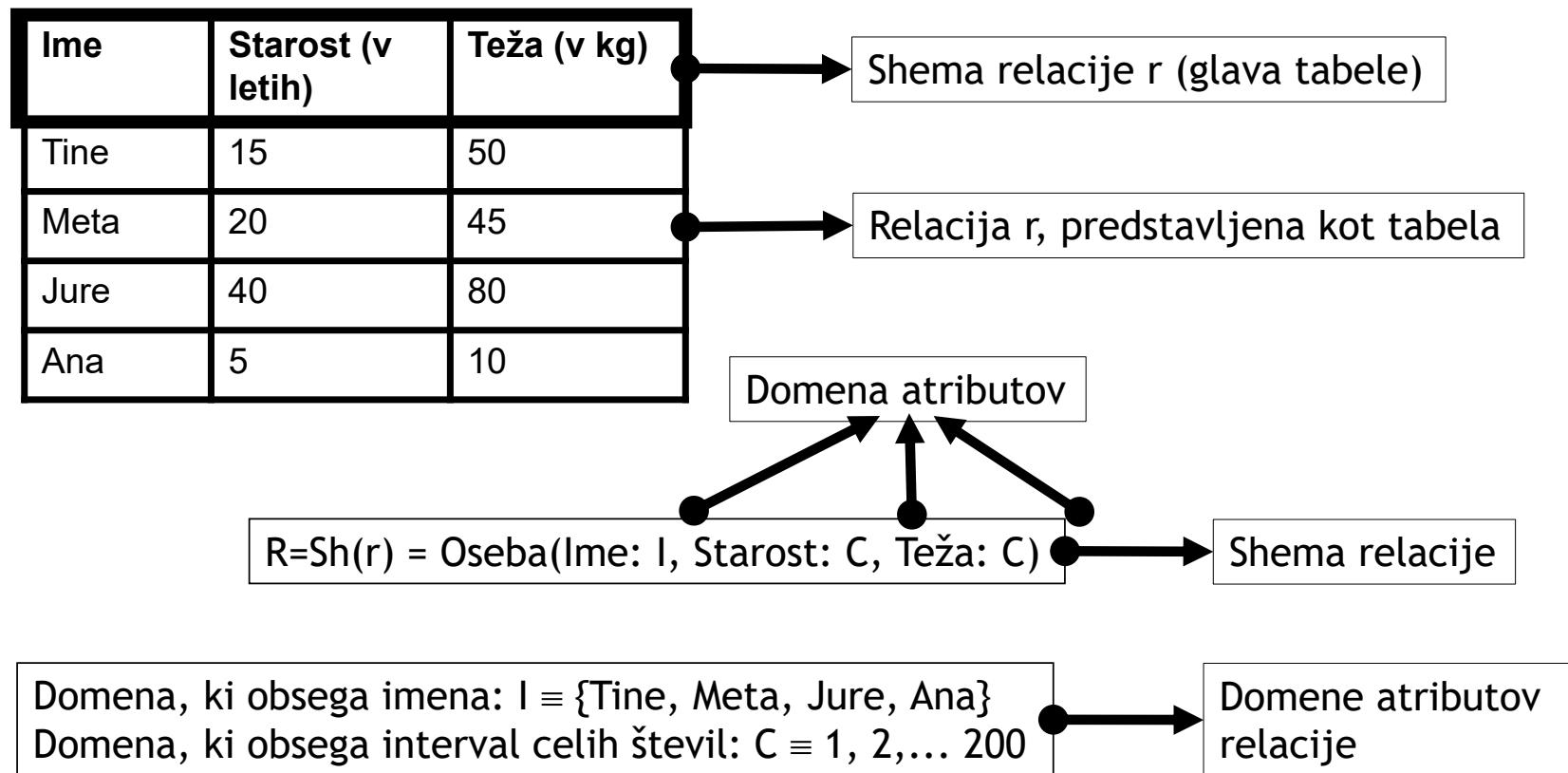
Terminologija pri relacijskem modelu

- N-terica je ena vrstica (element) v relaciji.
- Stopnja relacije je število atributov v relaciji.
- Števnost 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 \rightarrow 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)

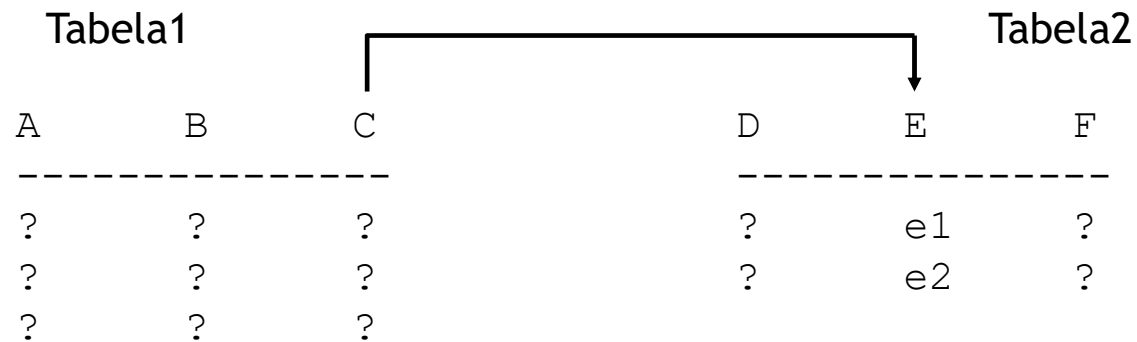
Zakon ca	Leto poroke (celo število)
Tine Meta	1995
Ana, Jure	1980

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



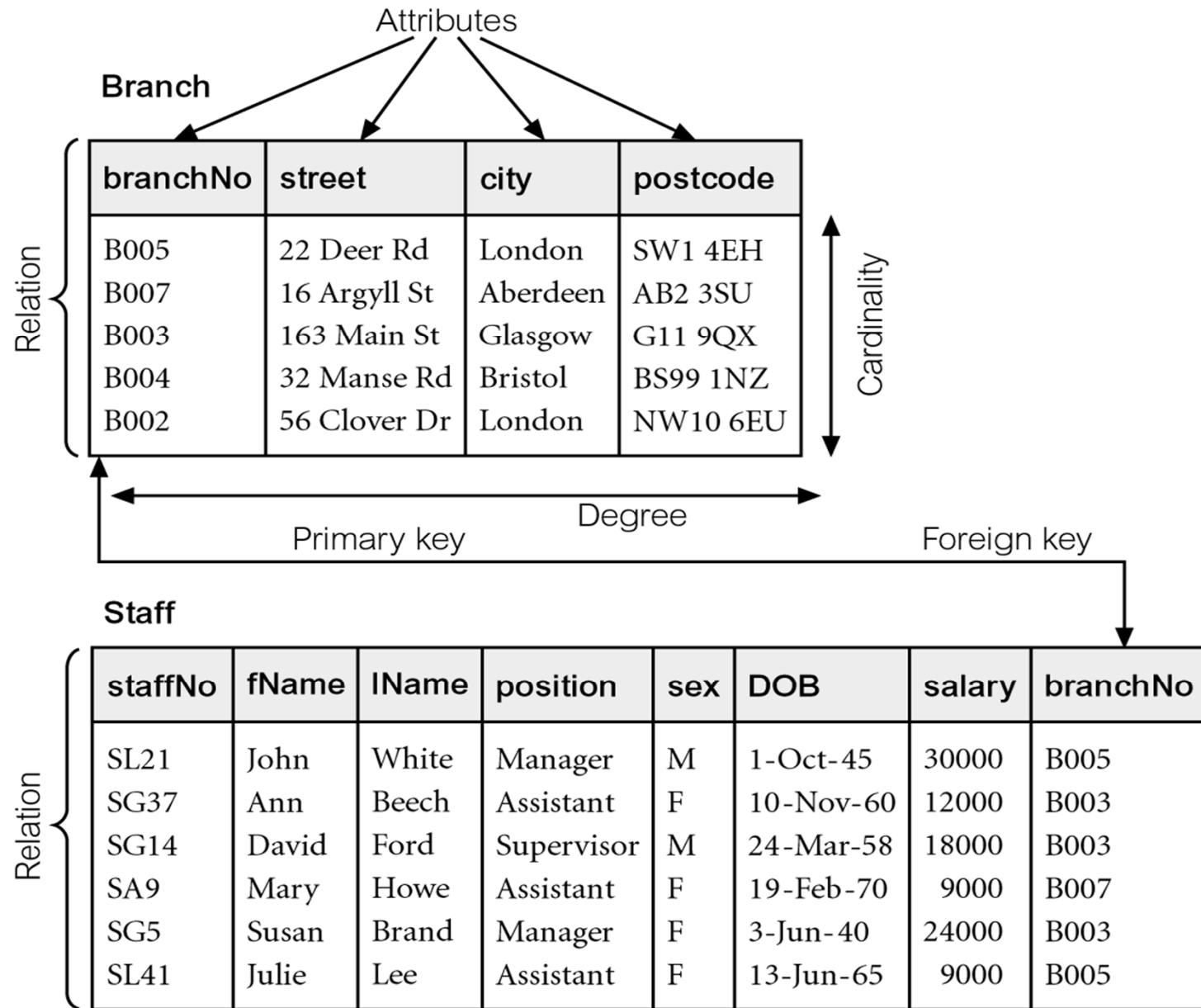
```
CREATE TABLE Tabela1 (  
    A INTEGER,  
    B INTEGER,  
    C CHAR(2) FOREIGN KEY REFERENCES Tabela2(E)  
);
```

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

Tabela1(A, B, #C)

Primerki relacije:

Branch (oddelek)
Staff (zaposlen)



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

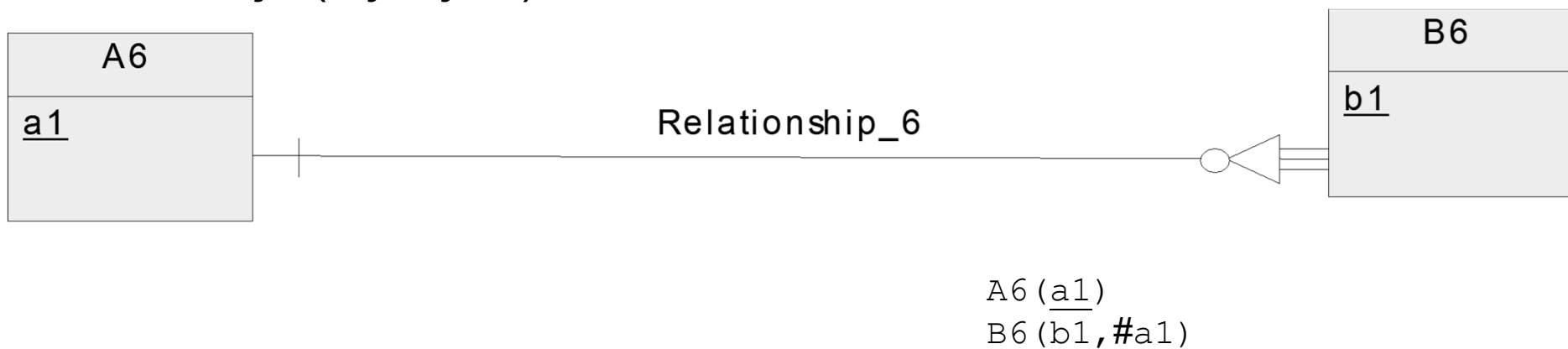
- Preslika se v istoimensko relacijsko shemo z vsemi njegovimi atributi
- Ključ sheme je identifikator močnega entitetnega tipa
- Shemi se potencialno dodajo še atributi, ki implementirajo razmerja (tuji ključi)

A7
<u>a1</u> aa1

A7 (a1, aa1)

Preslikava šibkega entitetnega tipa

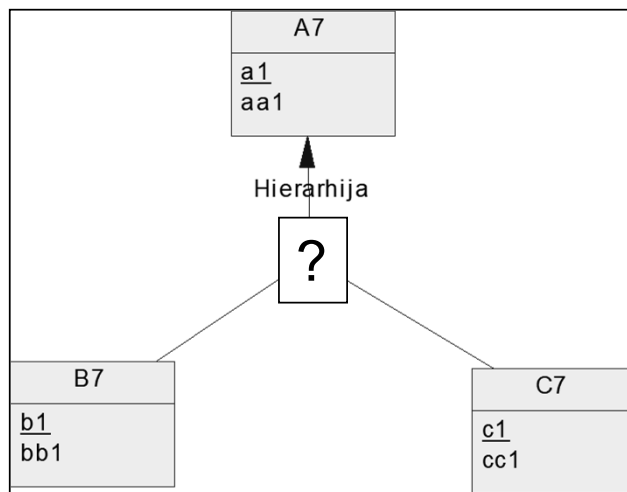
- Preslika se v istoimensko relacijsko shemo z vsemi njegovimi atributi in identifikatorji entitetnih tipov, od katerih je odvisen
- Ključ sheme je identifikator šibkega entitetnega tipa in identifikatorji entitetnih tipov, od katerih je odvisen
- Shemi se potencialno dodajo še atributi, ki implementirajo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



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)

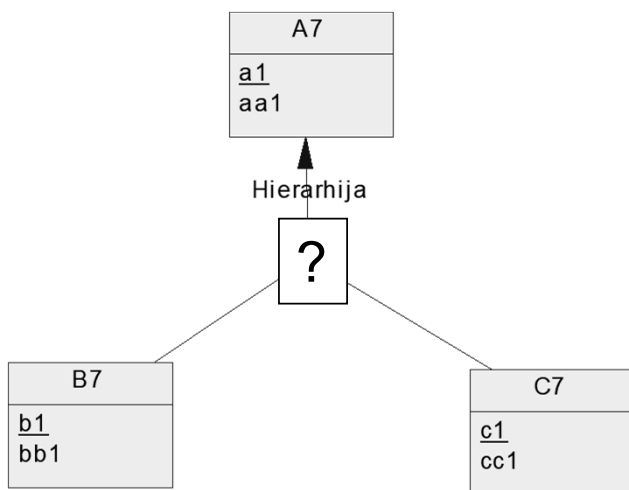
Totalno prekrivno:

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:

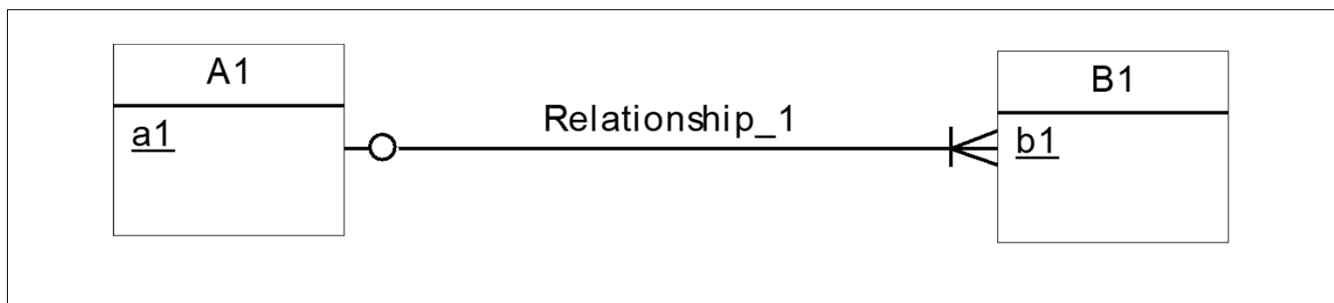
A7B7 (a1, aa1, b1, bb1)

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

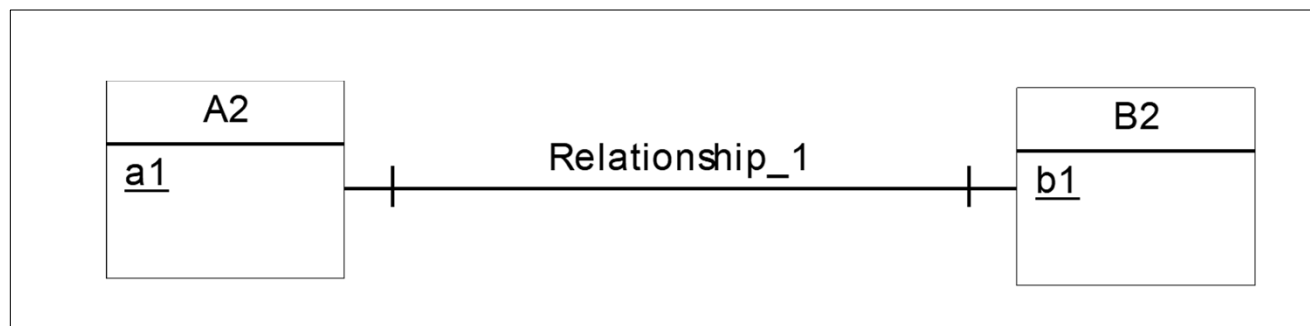
Razmerja (0/1,1):(0/1,n)



- 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):

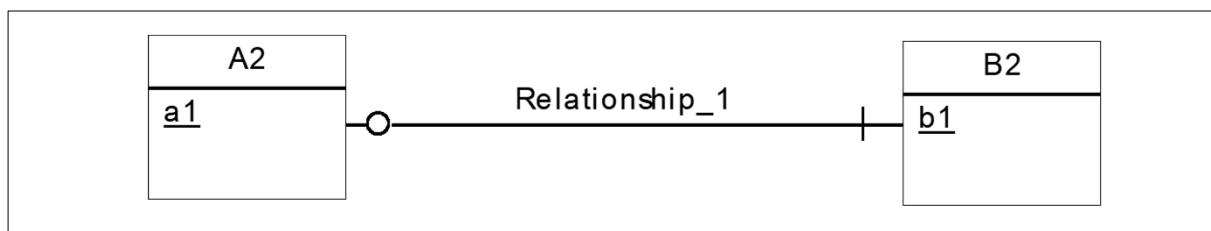
A2B2(a1, b1)

- Če ne, pa dobimo:

A2(a1, #b1)

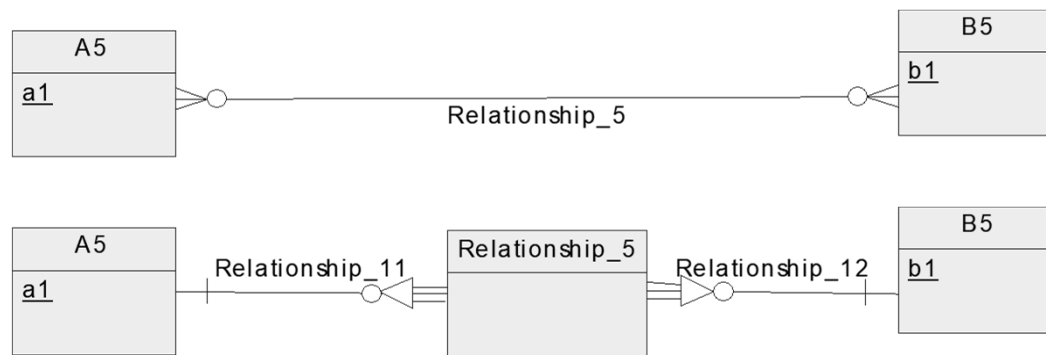
B2(b1, #a1)

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)

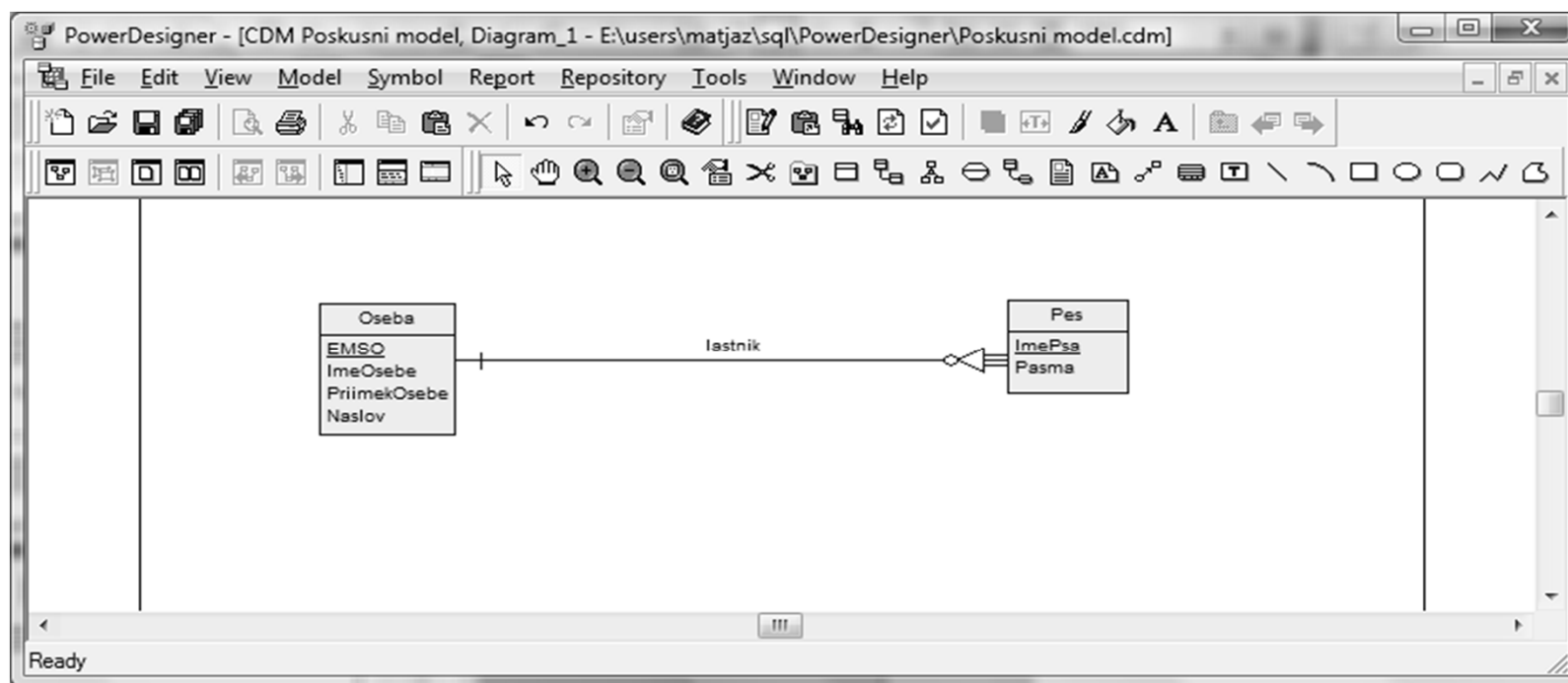
Končni rezultat preslikave

- 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

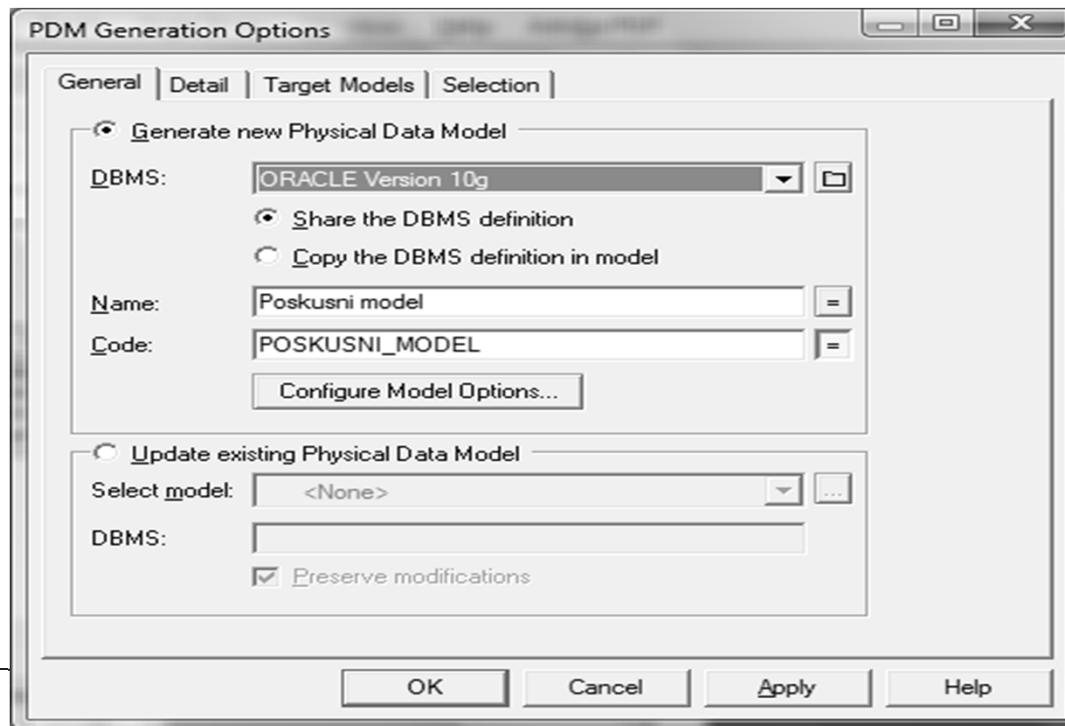
Gradnja konceptualnega modela s pomočjo Power Designerja

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



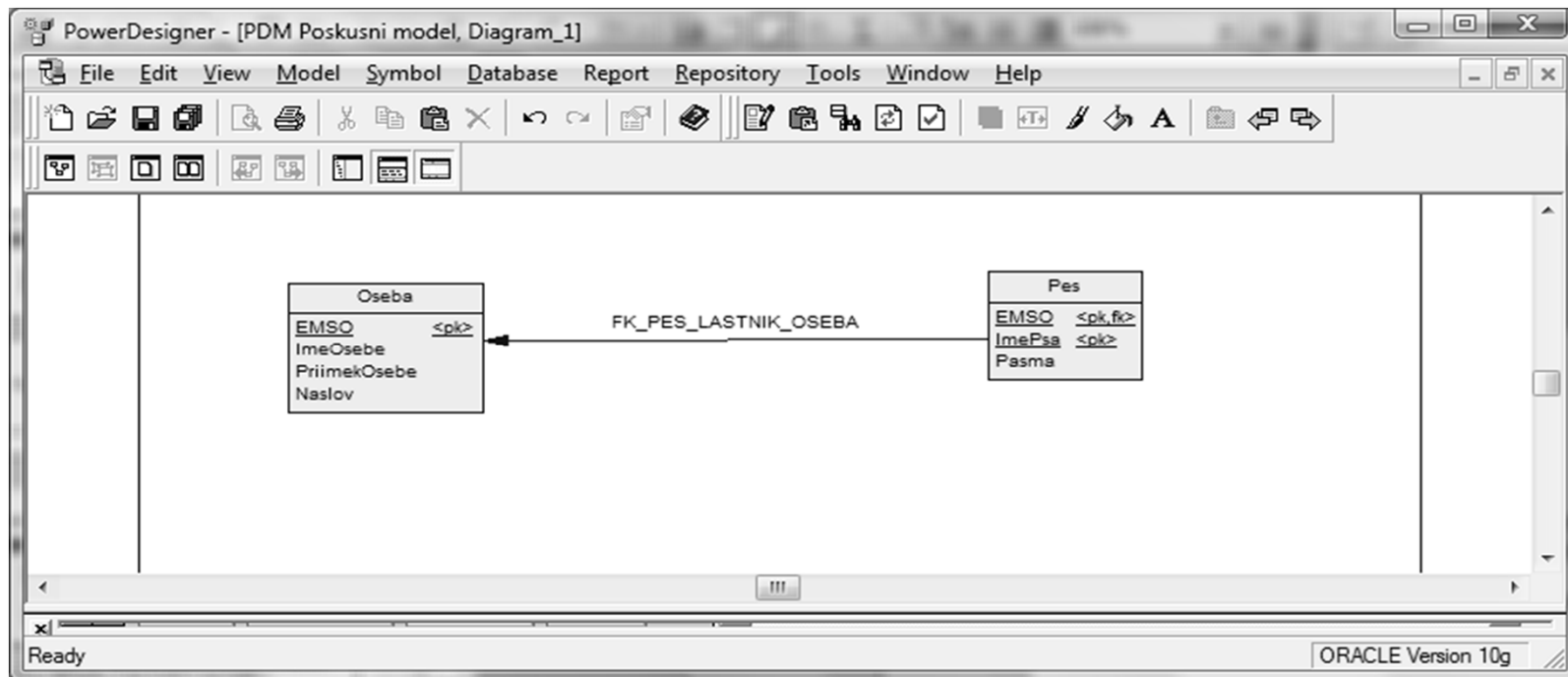
Pretvorba v logični model s pomočjo Power Designerja

- 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



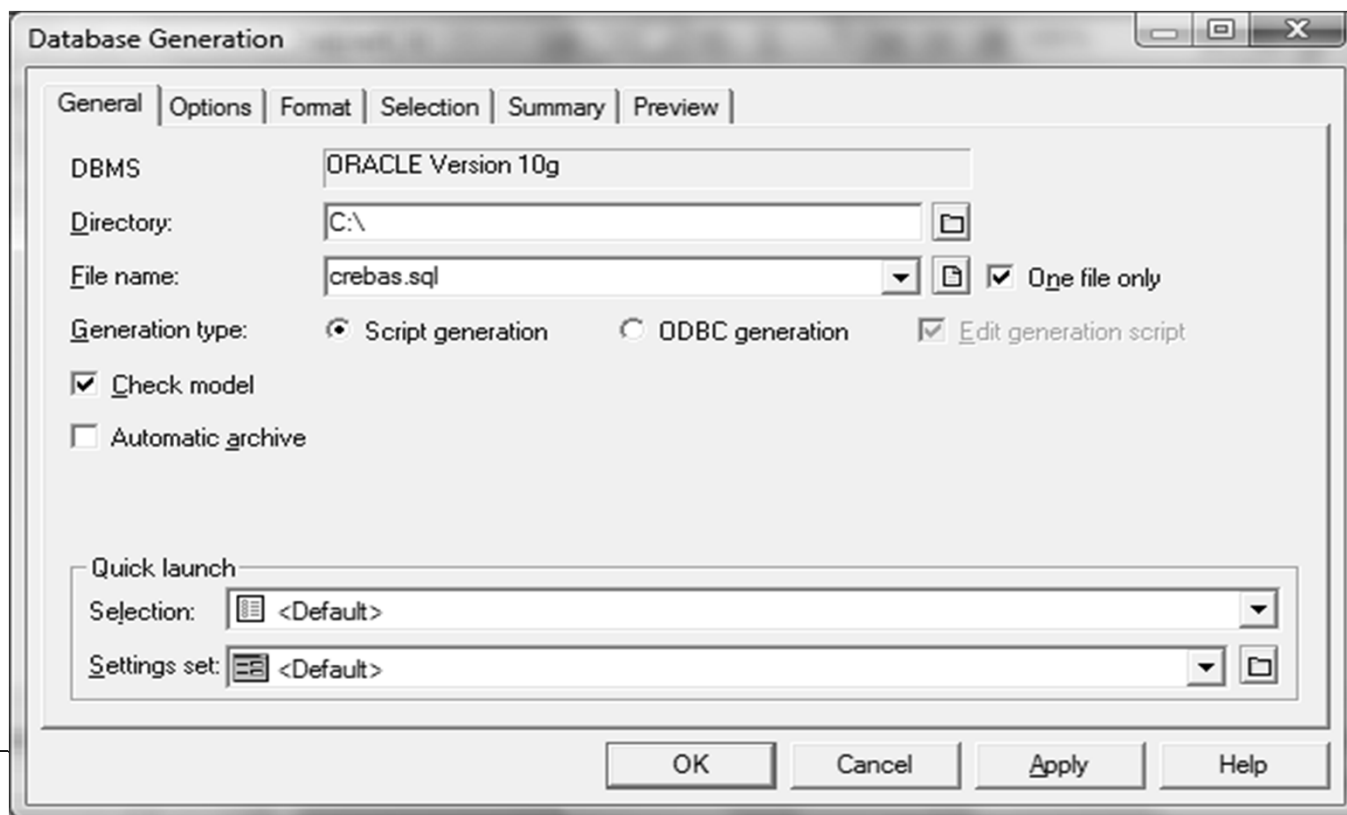
Pretvorba v logični model s pomočjo Power Designerja

- 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, ...)