Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Predmet:
Osnove podatkovnih baz

Modul:
Načrtovanje podatkovnih baz

Gradivo:
v.2017

Vsebina

- Različni pristopi k načrtovanju podatkovnih baz
- Trije nivoji načrtovanja:
  - Konceptualni načrt
  - Logični načrt
  - Fizični načrt

Primer: urarna

- Želimo razviti aplikacijo za podporo prodaji in servisu v trgovini z urami:
  - Trgovina prodaja:
    - Artikle (različne ure, jermenčke ipd. ter baterije in druge nadomestne dele);
    - Storitve (popravila in vzdrževanje).
  - Nujno mora voditi evidenco:
    - Nabavljenega blaga
    - Prodanih artiklov in storitev
    - Kupcev (če pravne osebe)
    - Prodajalcev, če gre za prodajo pravnim osebam.

Tabele

- ARTIKLI IN STORITVE
- SKUPINE ARTIKLOV
- IZDANI RAČUNI
- KUPCI
- PRODAJALCI
- PREVZEMI BLAGA
- DOBAVITELJI
- PREVZEMNIKI

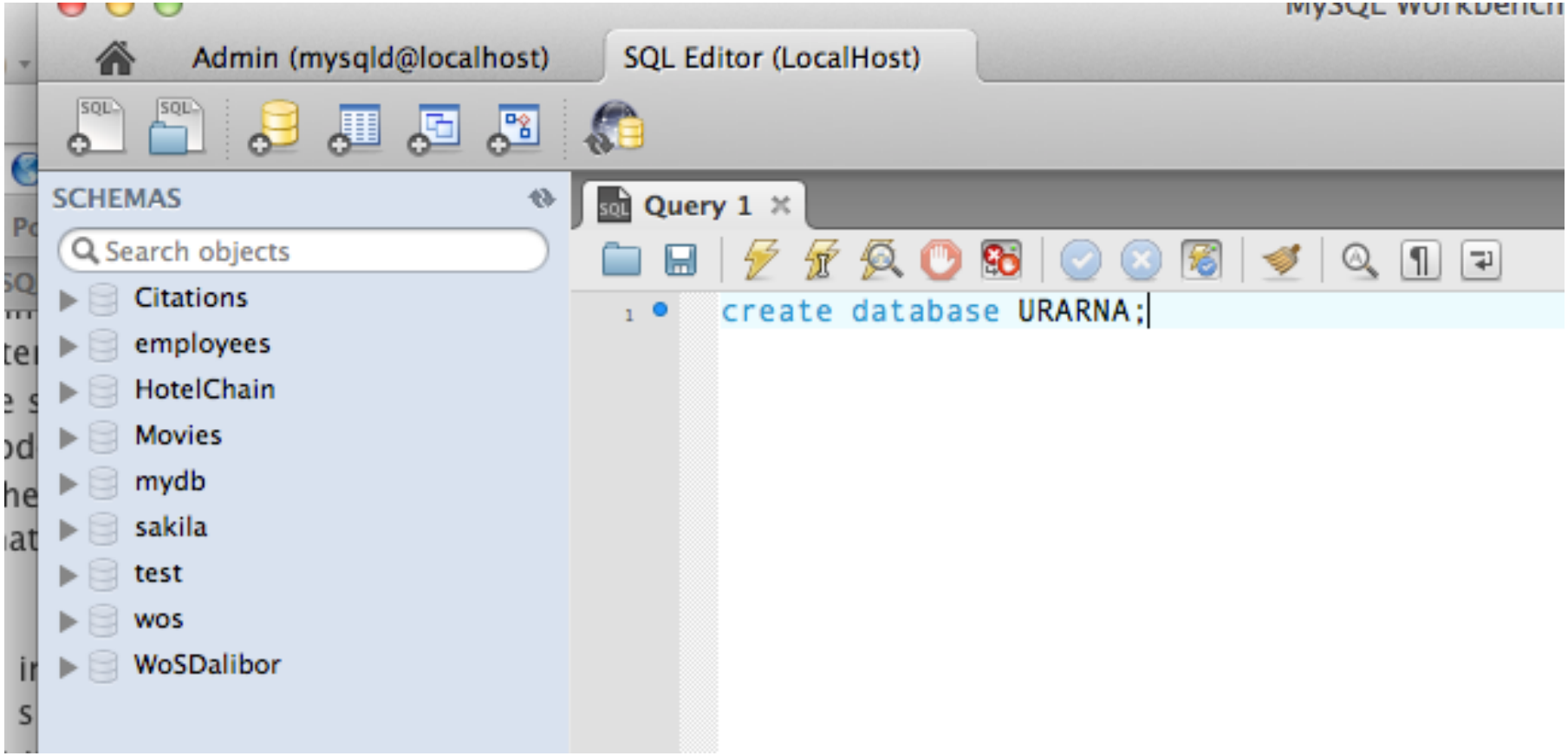
Stolpci tabel (atributi)

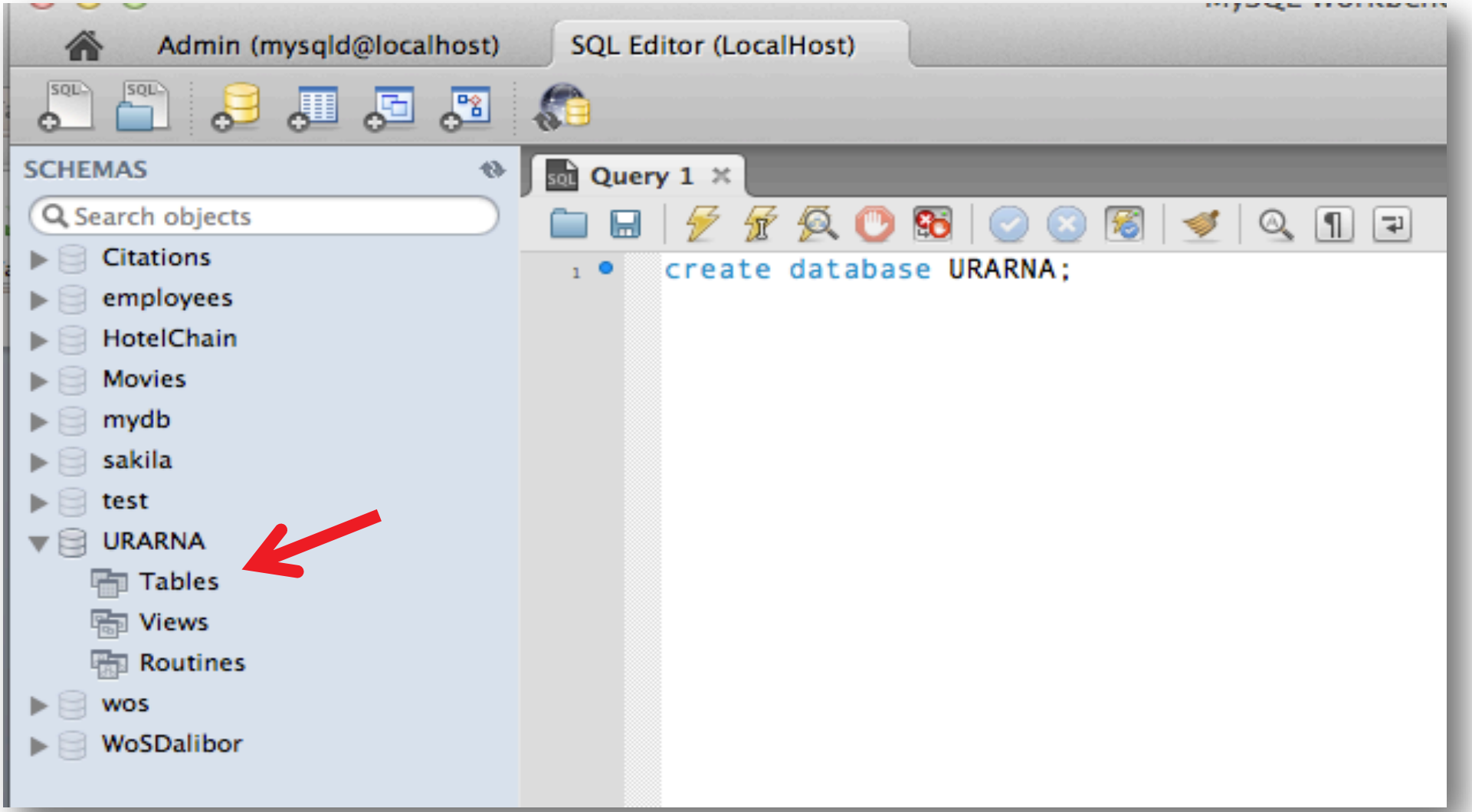
- ARTIKLI IN STORITVE
  - Šifra artikla
  - Naziv artikla
  - Enota mere
  - Zaloga
  - Kritična zaloga
  - Vrsta artikla
  - ...
- ...

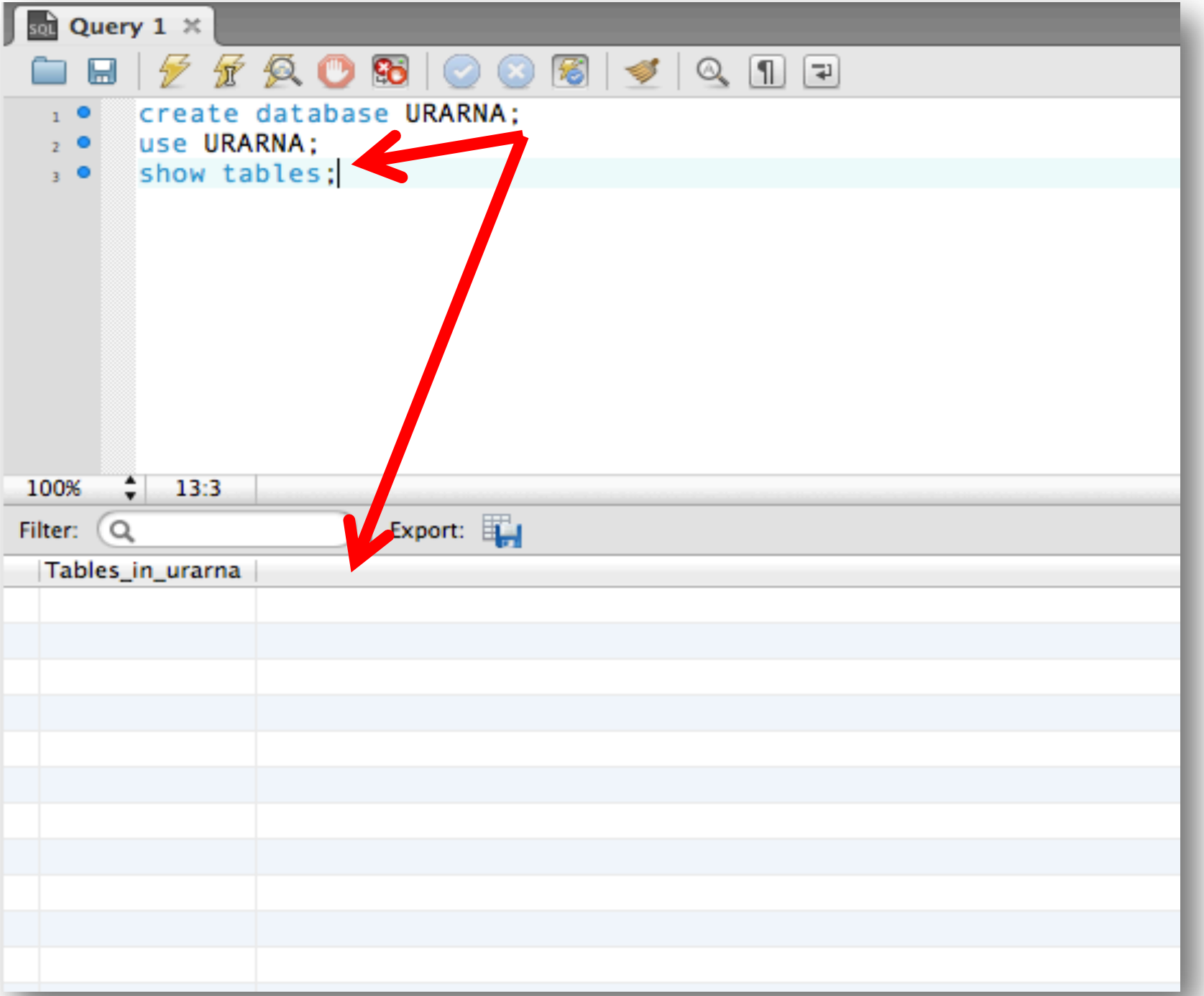
Kreiranje baze z uporabo SQL DDL

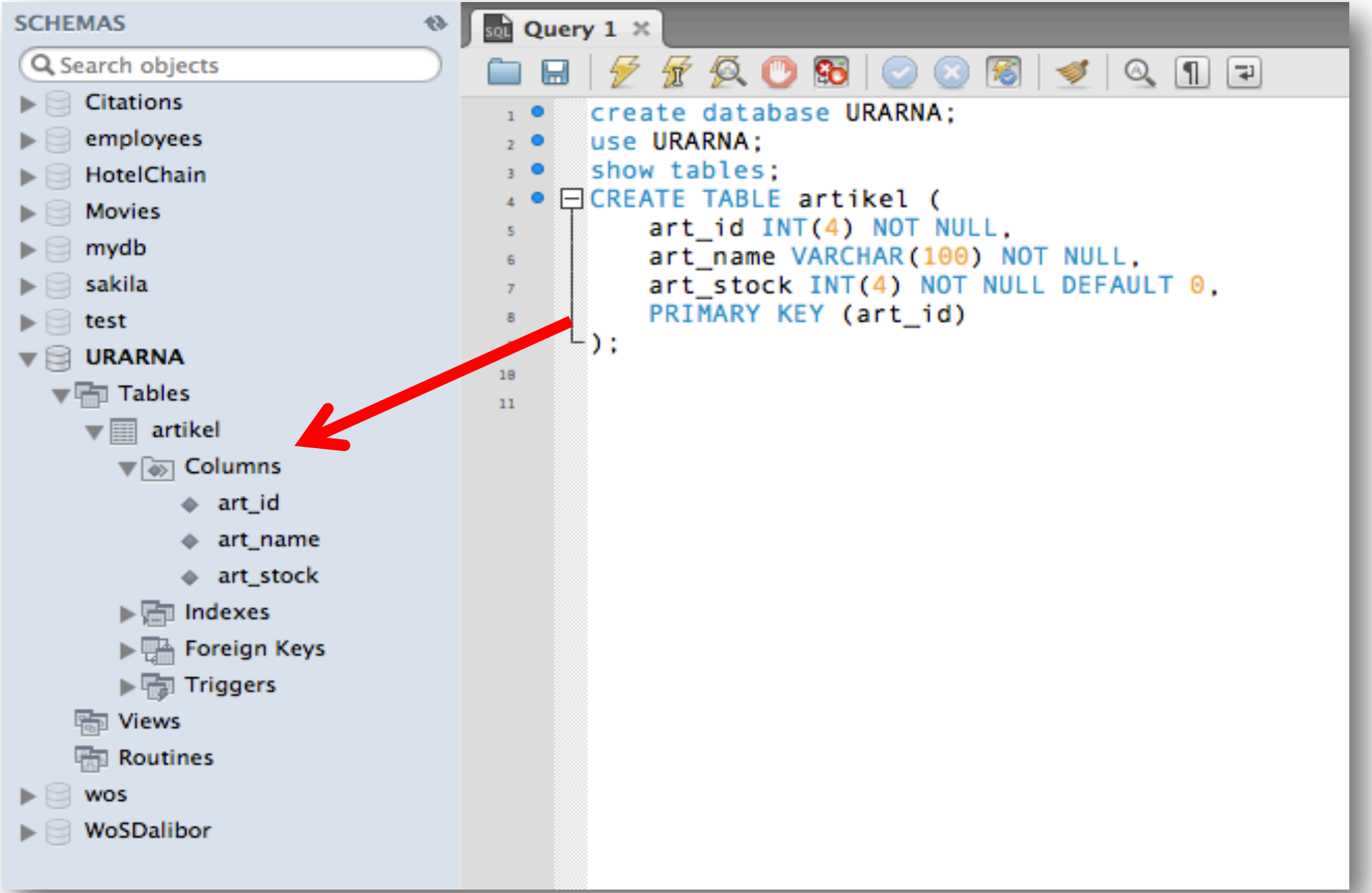
```
CREATE DATABASE urarna;
USE urarna;

CREATE TABLE artikel (
        art_id INT(4) NOT NULL,
        art_name VARCHAR(100) NOT NULL,
        art_stock INT(4) NOT NULL DEFAULT 0,
        PRIMARY KEY (art_id)
...
```

Kreiranje baze z uporabo MySQL Workbench




Admin (mysqld@localhost)
SQL Editor (LocalHost)
SCHEMAS
Search objects
Citations
employees
HotelChain
Movies
mydb
sakila
test
URARNA
Tables
Views
Routines
wos
WoSDalibor
Query 1
1
create database URARNA;


Query 1
create database URARNA;
use URARNA;
show tables;
100%
13:3
Filter:
Export:
Tables_in_urarna


SCHEMAS
Search objects
Citations
employees
HotelChain
Movies
mydb
sakila
test
URARNA
Tables
artikel
Columns
art_id
art_name
art_stock
Indexes
Foreign Keys
Triggers
Views
Routines
wos
WoSDalibor
Query 1
create database URARNA;
use URARNA;
show tables;
CREATE TABLE artikel (
art_id INT(4) NOT NULL,
art_name VARCHAR(100) NOT NULL,
art_stock INT(4) NOT NULL DEFAULT 0,
PRIMARY KEY (art_id)
);

Omejitve direktnega kreiranja baze

- Kaj če je tabel veliko? Več 100 tabel?
  - Zamudno.
  - Tveganje napak – relacijski model zelo razdrobljen…številne povezave med tabelami, ključi lahko sestavljen iz več atributov...
- Kaj če se shema kasneje spreminja?
  - Težko in zamudno vzdrževanje.

Pristopi k načrtovanju PB

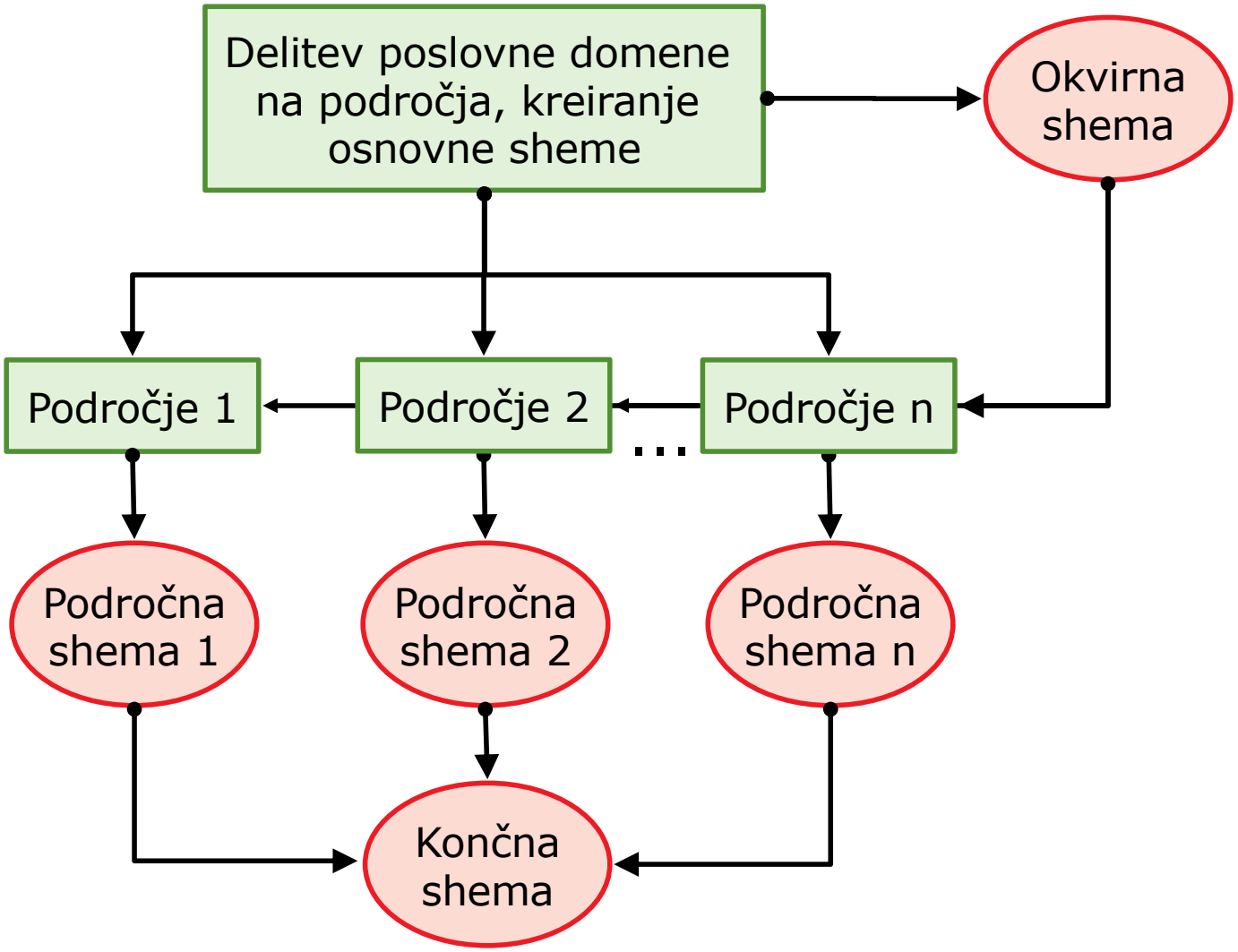
- Dva pristopa k načrtovanju PB:
  - od spodaj navzgor in
  - iz vrha navzdol.
- Pristop od spodaj navzgor:
  - začne z atributi ter jih združuje v skupine.
  - Primeren za enostavne PB (majhno število atributov).
  - Tak pristop predstavlja normalizacija.

Normalizacija = identifikacija potrebnih atributov in njihovih agregacij v normalizirane relacije na osnovi funkcionalnih odvisnosti.

Pristop z vrha navzdol

- Za večje PB primeren pristop z vrha navzdol.
  - Na začetku podatkovni modeli z le nekaj osnovnih entitetnih tipov in razmerij
  - Korakoma razgradnja na pod-entitete, povezave in atribute
  - Tak pristop predstavlja uporaba tehnike Entiteta – Razmerje (E-R).

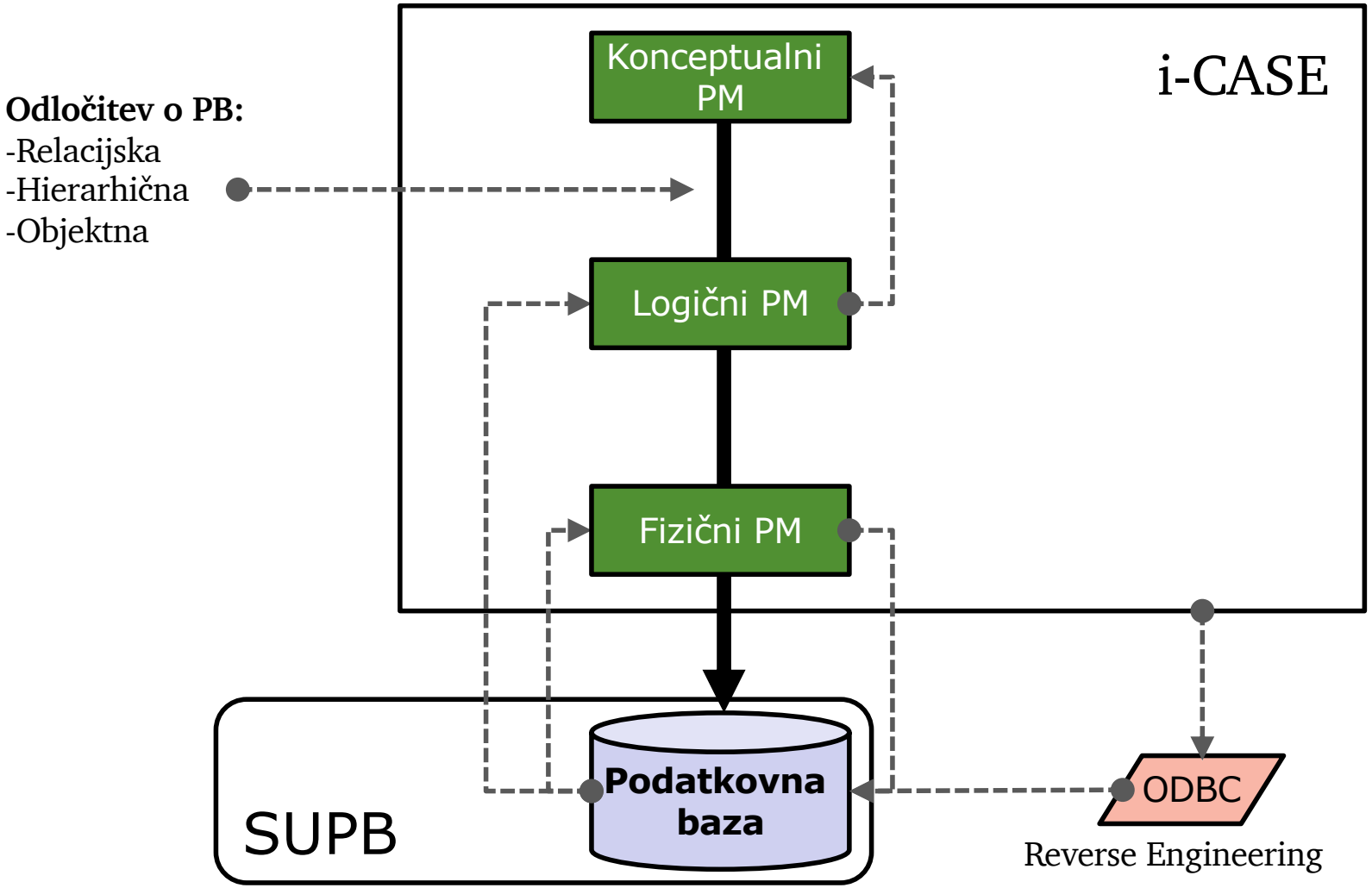
Za kompleksne domene pristop “po delih”



Trije nivoji podatkovnega modeliranja

- Konceptualni podatkovni model
  - Semantika obravnavane domene: entitete, pozvezave med entitetami, atributi entitete, poslovna pravila...
  - Neodviseno od SUPB
- Logični podatkovni model
  - Model v jeziku izbranega SUPB,
  - V primeru izbire relacijskega SUPB → relacijski model,
  - Mehanizmi, ki jih ima ciji SUPB: npr. indeksi, sprožilci, pogledi, omejitve...
- Fizični podatkovni model
  - Optimizacija fizične podatkovne sheme (indeksi, pogledi, sočasnost, velikost medpomnilnika...)

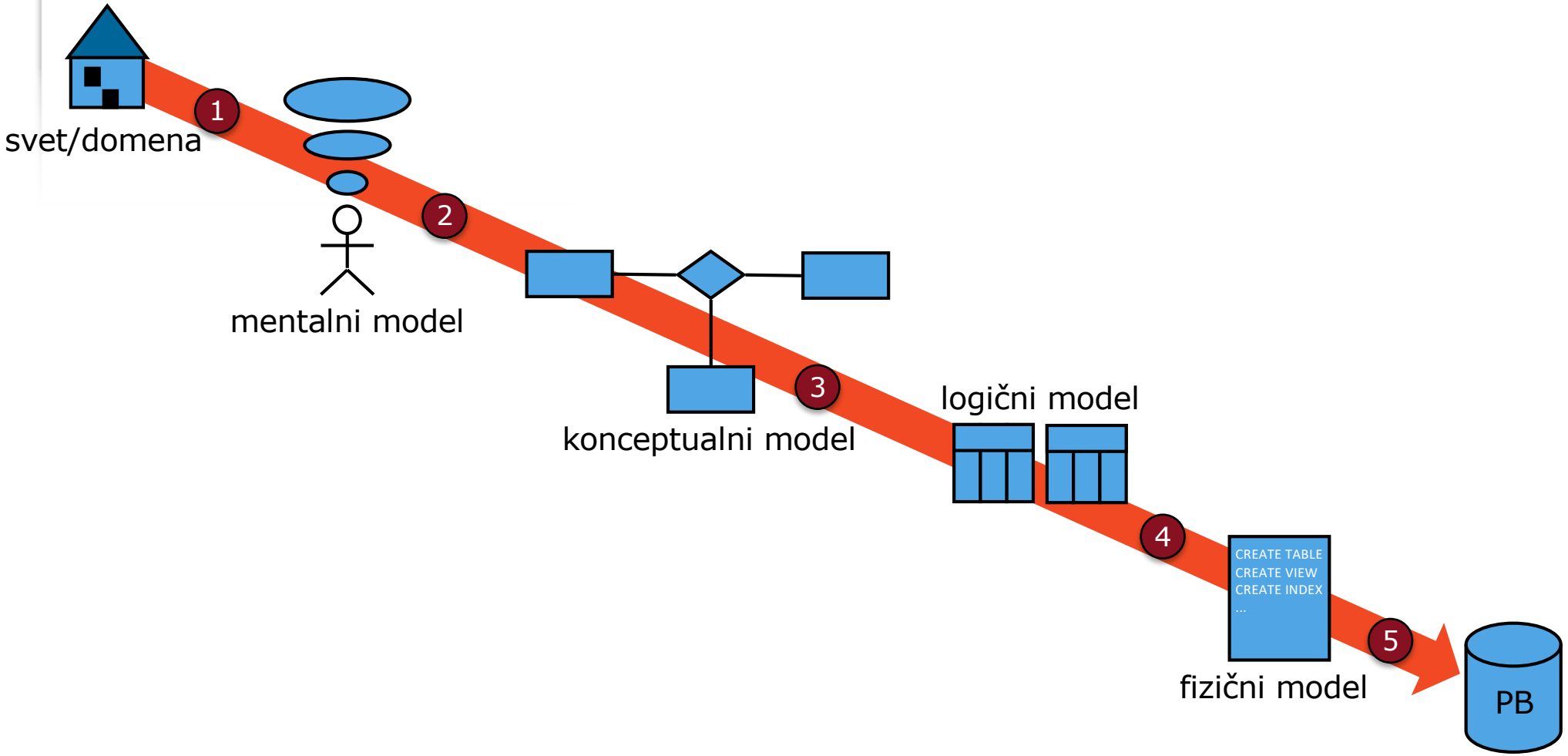
i-CASE orodja



Konceptualno načrtovanje

- Konceptualno načrtovanje je neodvisno od tehnologije – ne ukvarjamo se z vprašanjem, kako hraniti podatke, temveč katere podatke hraniti.
- Z načrtom na formalen način (uporaba diagramske tehnike) opredelimo koncepte domene, za katero gradimo bazo.
- Cilj je razumeti domeno (semantika) ter identificirati koncepte, ki jih želimo hraniti v bazi.
- Rezultat je konceptualni model.

Umestitev konceptualnega načrtovanja



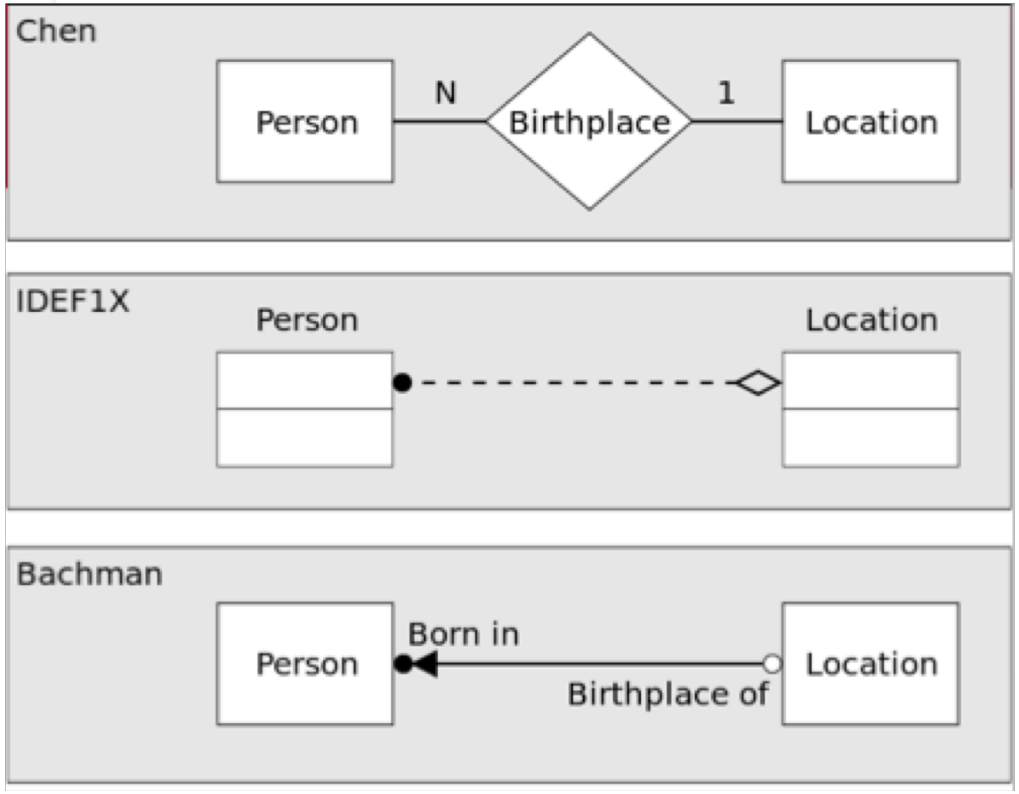
Tehnika modeliranja

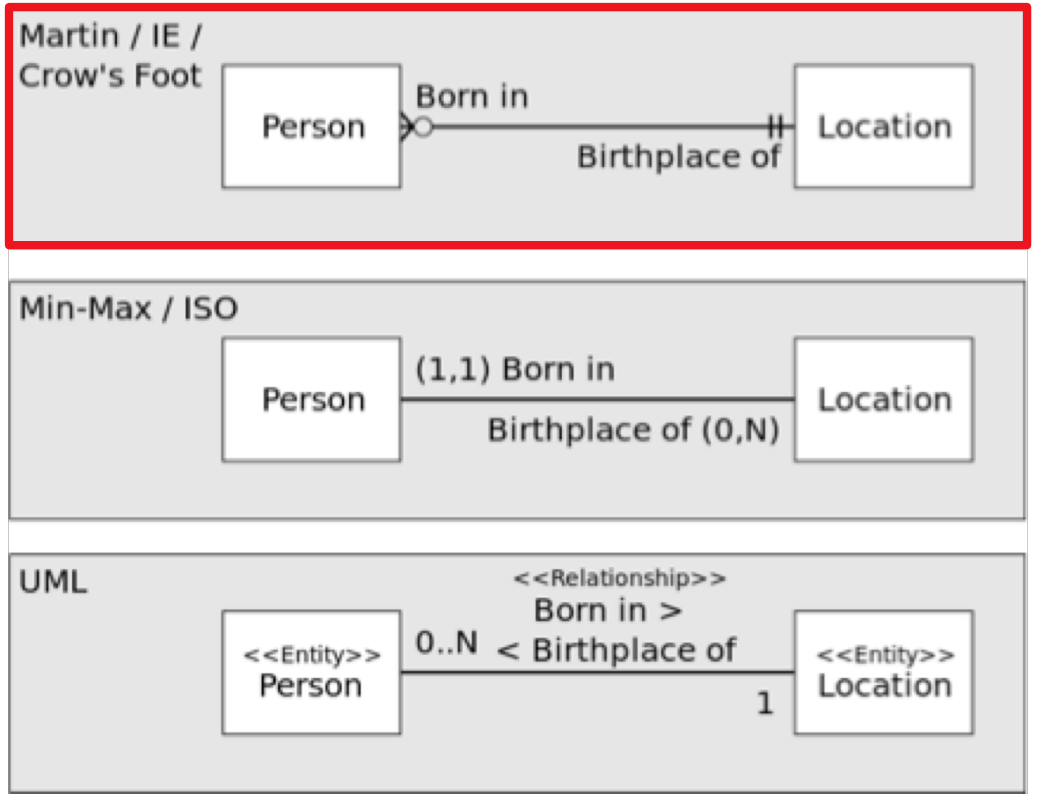
- Za izdelavo konceptualnega načrta/modela lahko uporabimo različne tehnike. Ena najbolj uporabljenih je tehnika entiteta-razmerje (ER).
- Z ER lahko nedvoumno prikažemo koncepte, ki nas zanimajo, njihove lastnosti ter povezave med njimi.

- Ključni koncepti ER:
  - Entitetni tip
  - Razmerje ali povezava
  - Atribut
  - Identifikator

Diagram entiteta-razmerje

- Veliko različnih notacij





Entitetni tipi in entitete...

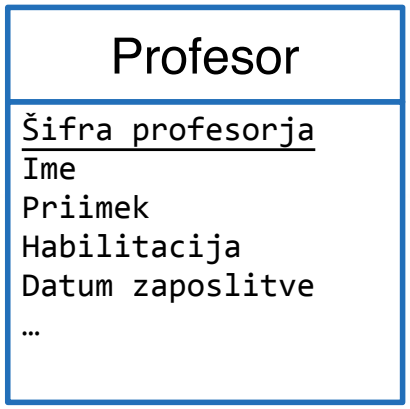
- Entitetni tipi so zbirke istovrstnih objektov, entitete pa posamezni objekti.
- Primeri entitetnih tipov:
  - Avtomobili,
  - Živali,
  - Mobilni telefoni,
  - Profesorji,
  - …
- Primeri entitet:
  - **Avto**: osebni, **Tip**: VW Passat 2.0 TDI, **Letnik**: 2017, **Reg.št**: GO…
  - **Žival**: pes, **Ime**: Spooky, **Pasma**: Zahodnovišavski beli terier…

Opisovanje entitet

- Entitete, ki pripadajo istemu tipu, opisujemo prek istih lastnosti.
- Primer:
  - **Avtomobili**: vrsta, tip, znamka, moč motorja, barva, število sedežev,…
  - **Živali**: vrsta, podvrsta, naziv/ime, starost…

- Lastnosti izberemo glede na to, kaj nas o entitetah nekega tipa zanima (abstrakcija – ne zanimajo nas vse možne lastnosti).
- Izbranim lastnostim pravimo atributi.

Grafična upodobitev entitetnih tipov im entitet

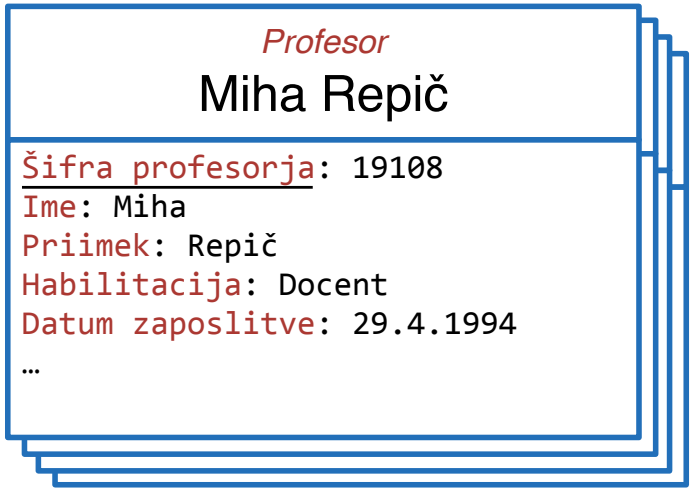


Naziv entitetnega tipa

Prostor za atribute

Entitetni tip

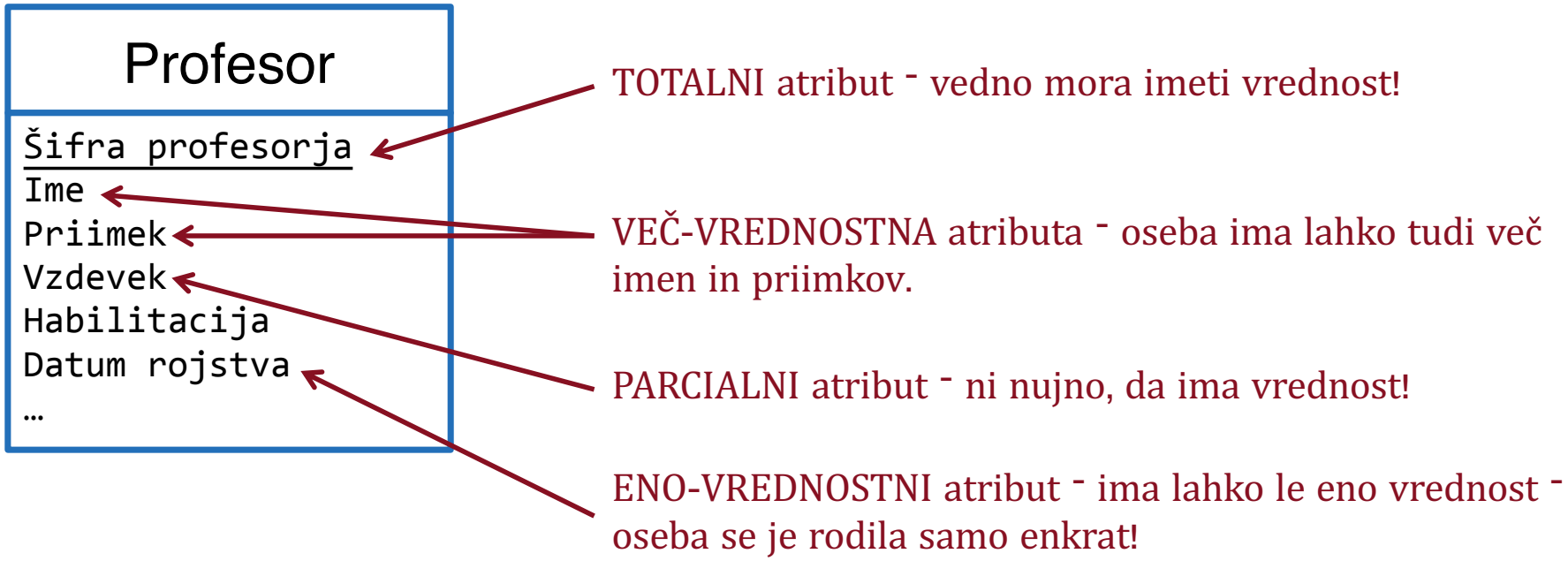
Entitete



Kratka predstavitev brez atributov

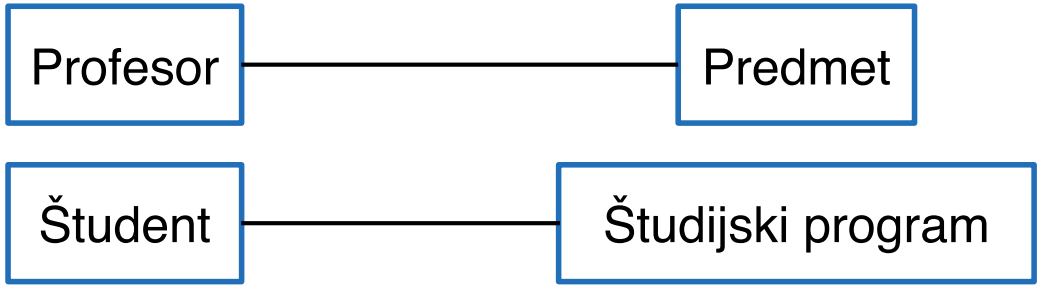
Vrste atributov

- Ločimo totalne, parcialne, eno-vrednostne in več-vrednostne atribute.



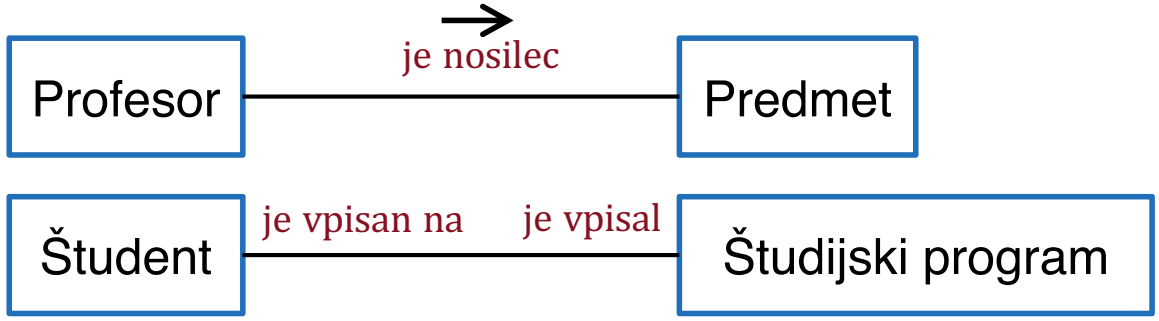
Razmerja med entitetami

- Med entitetami/tipi lahko obstajajo različna razmerja, ki izhajajo iz dejanskih povezav med njimi.
- Primer:
  - PROFESOR *je nosilec* PREDMETA
  - ŠTUDENT *je vpisan na* ŠTUDIJSKI PROGRAM
- Razmerje med dvema entitetama/tipoma prikažemo s črto.



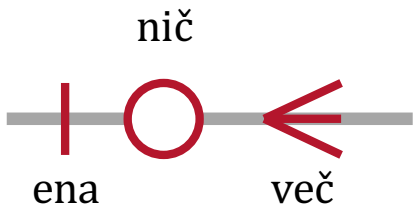
Pomen razmerja

- Vsaka entiteta/tip ima v razmerju z drugo entiteto/tipom svojo vlogo oziroma pomen.
- Primer:
  - PROFESOR *je nosilec* PREDMETA
  - PREDMET *izvaja* PROFESOR
  - ŠTUDENT *je vpisan na* ŠTUDIJSKI PROGRAM
  - ŠTUDIJSKI PROGRAM *je vpisal* ŠTUDENT

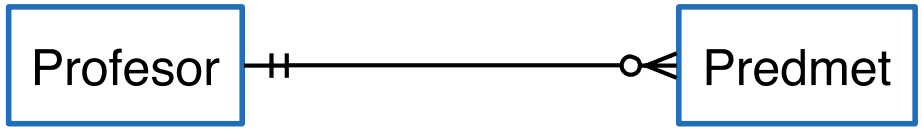


Številnost razmerja

- Številnost razmerja pove, koliko entitet iz ene strani razmerja je lahko povezanih z eno entiteto na drugi strani razmerja.
- Primer:
  - PROFESOR je nosilec *nič enega ali več* PREDMETOV
  - PREDMET izvaja *natanko eden* PROFESOR
- Številnost označujemo s posebno notacijo:

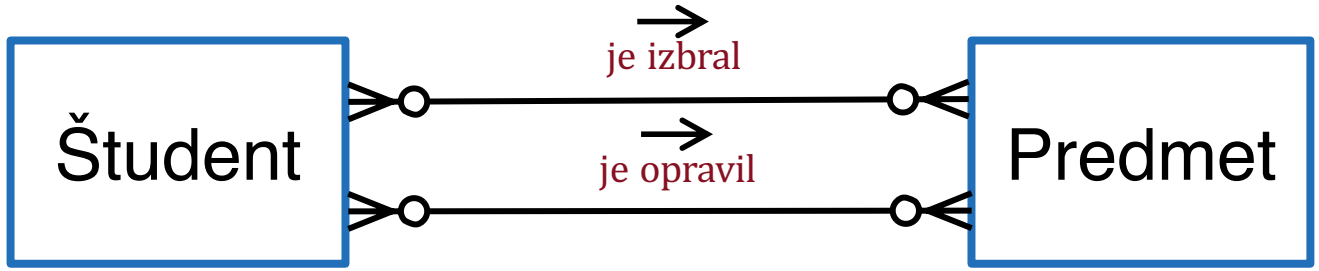


- Na vsaki strani razmerja označimo minimalno in maksimalno številnost!



Število razmerij

- Med opazovanim parom entitet/tipov je lahko več razmerij.
- Primer:
  - ŠTUDENT je izbral nič ali več PREDMETOV
  - ŠTUDENT je opravil nič ali več PREDMETOV

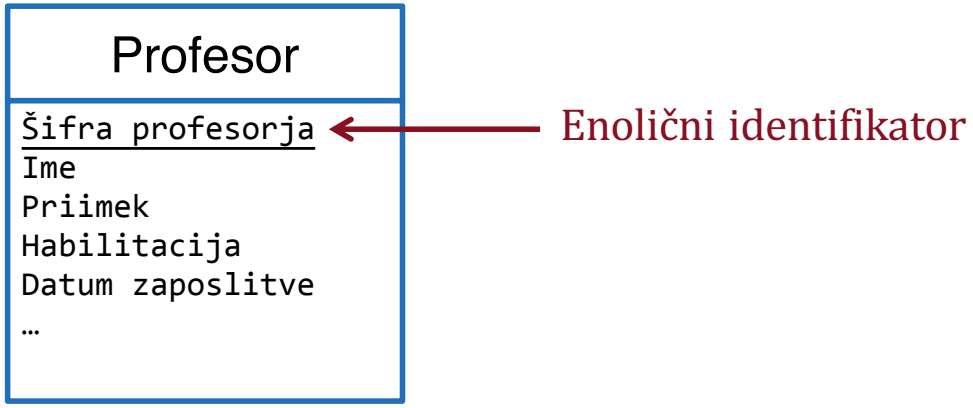


Dodatne lastnosti razmerja

- Razmerje ima lahko svoje lastnosti...

Enolični identifikator entitete

- Z entitetno množico označujemo entitete, ki v nekem trenutku pripadajo entitetnemu tipu.
- Enolični identifikator entitete je podmnožica atributov entitete in razmerij do drugih entitet, ki enolično razlikujejo posamezno instanco entitete znotraj entitetne množice.
- Imamo lahko več enoličnih identifikatorjev; enega izberemo - podlaga za ključ v relacijskem modelu!



Močni entitetni tip

- Kadar je vsako entiteto entitetnega tipa moč enolično identificirati z njenimi atributi, govorimo o močnem entitetnem tipu.
- $\{a_1, \ldots a_k\}$ je enolični identifikator entitete A, če ustreza naslednjim pogojem:
  a. $a_1, \ldots a_k$ so vsi totalni enovrednostni atributi, kar zagotavlja, da imajo vsi identifikacijski atributi definirano natanko eno vrednost (eno dimenzijo)
  b. $T: V_1 \times \ldots \times V_k \rightarrow E_A$ je totalna ali parcialna enovrednostna funkcija, kar zagotavlja, da se vsak element kartezijskega produkta vrednostnih množic V_i, ki so območja identifikacijskih atributov, preslika v največ eno entiteto tipa A
  c. Je minimalna podmnožica: ne obstaja prava podmnožica, za katero bi tudi veljal pogoj b)

Šibki entitetni tip

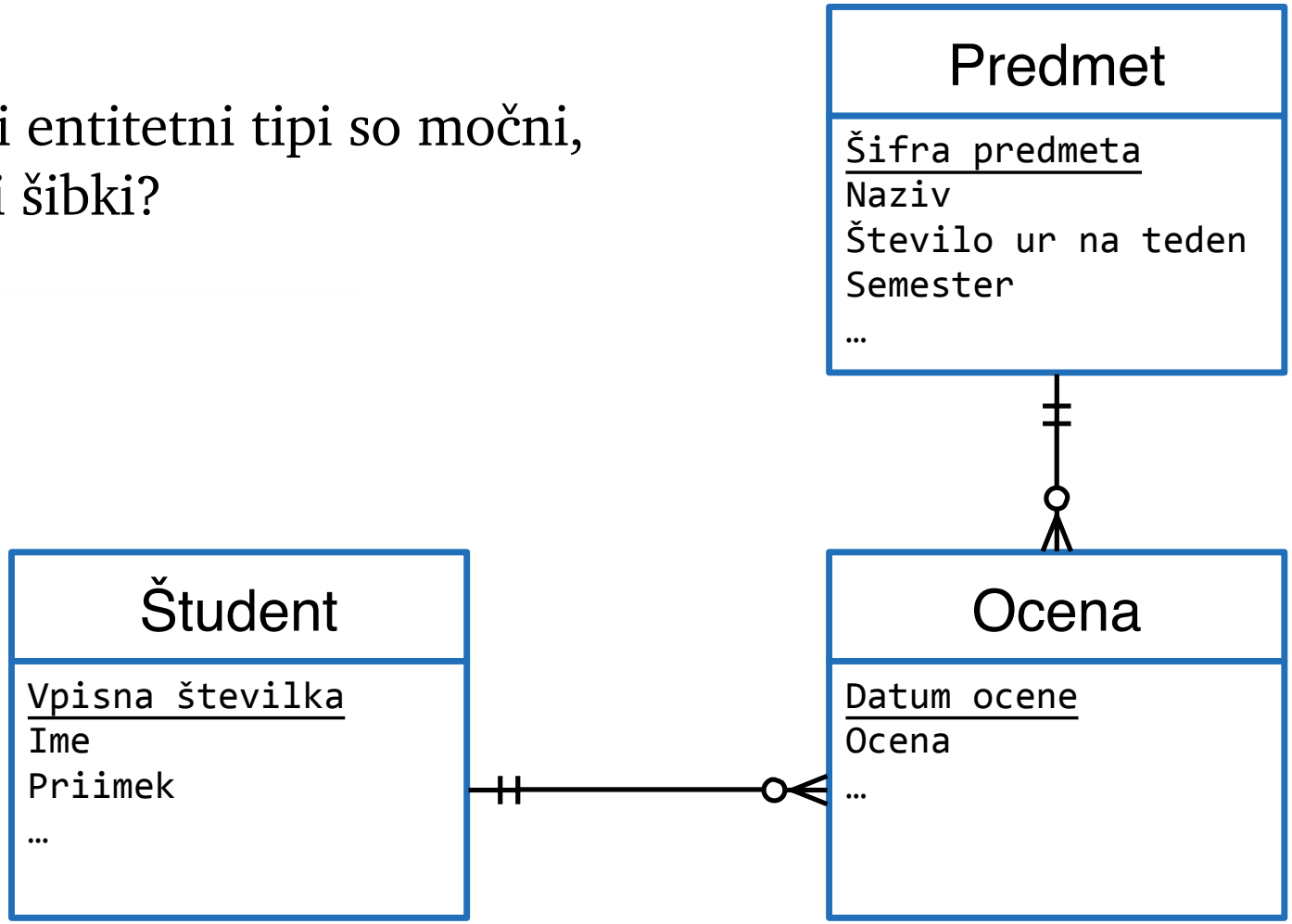
- Kadar enolični identifikator entitetnega tipa ni sestavljen le iz lastnih atributov, temveč tudi iz razmerij oz. drugih entitet v razmerju oz. njenih identifikatorjev, govorimo o šibkem entitetnem tipu.
- $\{a_1, \ldots a_k\} \cup I_{ET1} \cup .. \cup I_{ETn}$ je enolični identifikator entitete A, če ustreza naslednjim pogojem:
  a. $a_1, \ldots a_k$ so vsi totalni enovrednostni atributi, I_{Ti} pa identifikatorji entitetnih tipov
  b. $T: V_{a1} \times \ldots \times V_{ak} \times V_{IET1} \times .. \times V_{IETn} \rightarrow E_A$ je totalna ali parcialna enovrednostna funkcija;
  c. Je minimalna podmnožica, ne obstaja prava podmnožica, za katero bi tudi veljal pogoj b).

Vaja

- Narišimo konceptualni model za PB, v kateri bi želeli hraniti podatke o opravljenih izpitih:
  - *za vsakega študenta bi radi hranili ime in priimek ter prvo leto vpisa, za predmete njihove šifre, nazive, število ur izvajanja tedensko ter semester izvajanja, za vsako opravljanje izpita pa datum opravljanja ter oceno.*

Vaja

- Kateri entitetni tipi so močni, kateri šibki?



Vaja

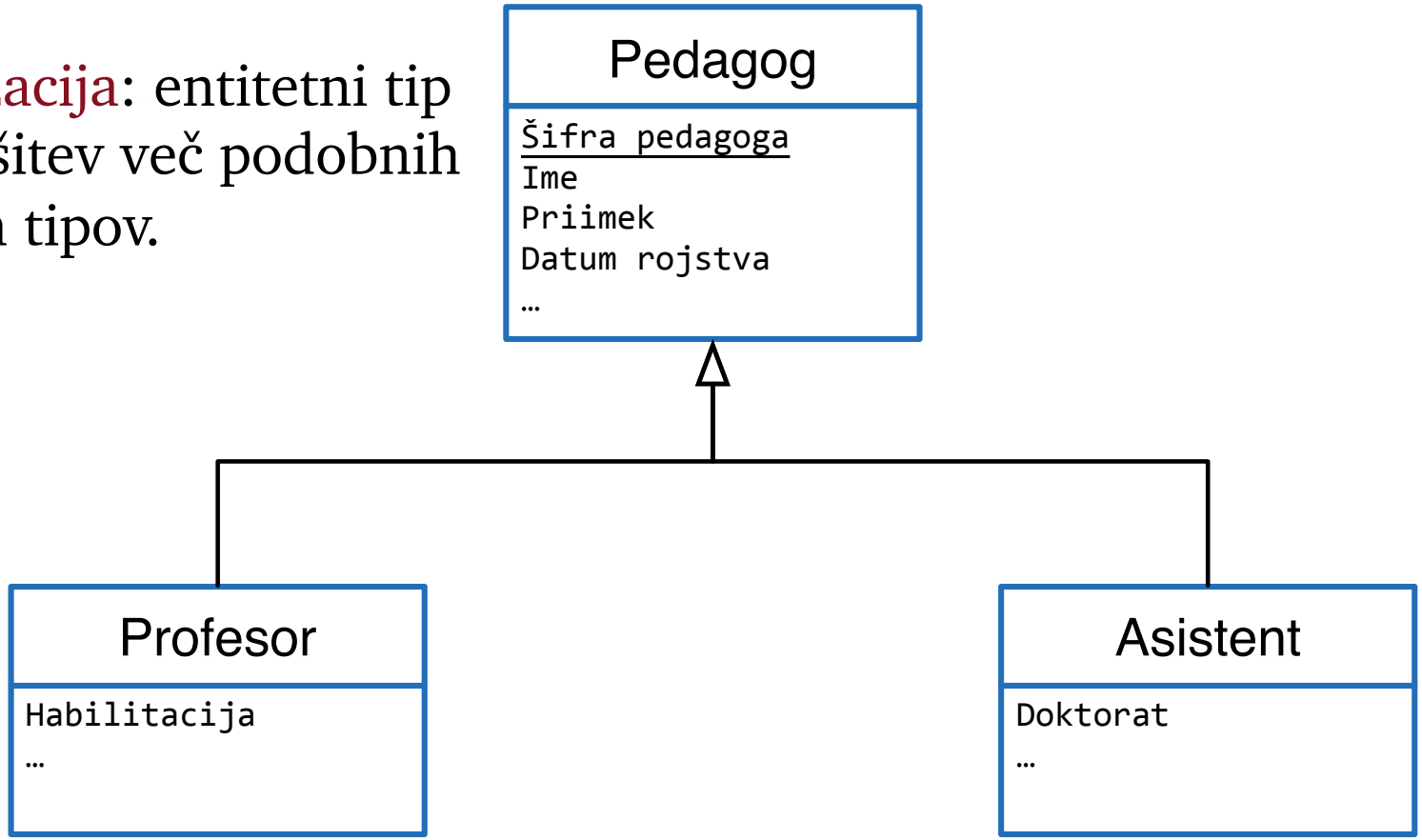
- Narišite podatkovni model urarne, za katero smo identificirali naslednje entitetne tipe:
  - ARTIKEL
  - SKUPINA ARTIKLA
  - STORITEV
  - KUPCI
  - PRODAJALCI
  - PRODAJA
  - PREVZEM
  - DOBAVITELJI
  - ...

Razširjena ER notacija

- Razširjena ER notacija pozna dodatne vrste povezav:
  - Generalizacija
  - Specializacija
  - Agregacija
  - Kompozicija

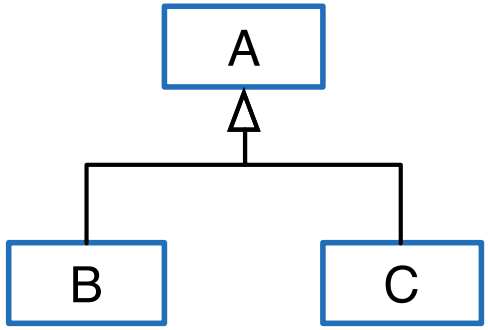
Generalizacija in specializacija

- Specializacija: entitetni tip je posebna vrsta drugega entitetnega tipa.
- Generalizacija: entitetni tip je posplošitev več podobnih entitetnih tipov.



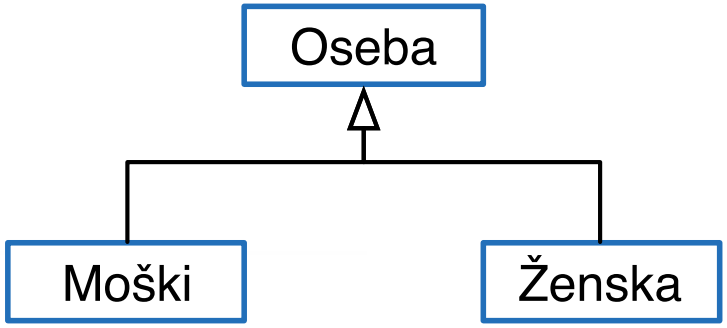
Specifike generalizacije in specializacije

- Imamo entitetni tip A s podtipoma B in C.

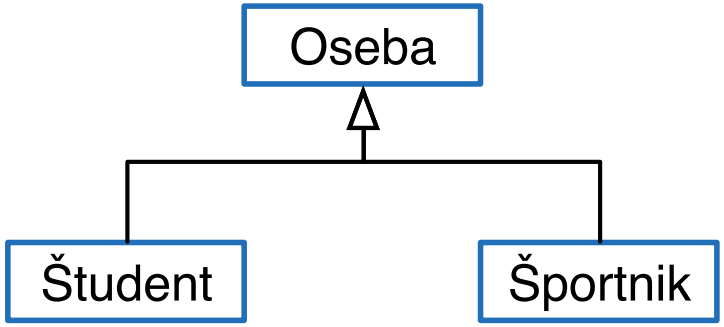


- Možne so naslednje povezave:
  - B in C pokrivata A totalno in ekskluzivno, če velja: $EB \cup EC = EA$ in $EB \cap EC = \{\}$
  - B in C pokrivata A totalno in prekrivno, če velja: $EB \cup EC = EA$ in $EB \cap EC \neq \{\}$
  - B in C pokrivata A delno in ekskluzivno, če velja: $EB \cup EC \subset EA$ in $EB \cap EC = \{\}$
  - B in C pokrivata A delno in prekrivno, če velja: $EB \cup EC \subset EA$ in $EB \cap EC \neq \{\}$

Primer specializacije in generalizacije



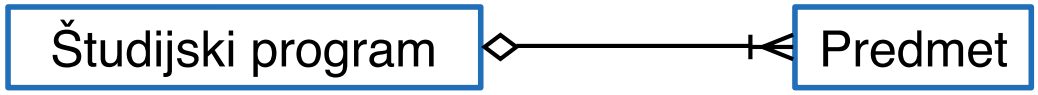
Totalno in ekskluzivno razmerje



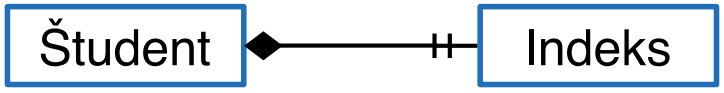
Delno in prekrivno razmerje

Agregacija in kompozicija

- Agregacija predstavlja razmerje, v katerem en entitetni tip predstavlja neko celoto, drugi pa njen del.



- Kadar je povezava med entitetnima tipoma tako močna, da entitetni tip, ki predstavlja del, ne more obstajati brez entitetnega tipa, ki predstavlja celoto, govorimo o kompoziciji.



Vaja

- Ali lahko v konceptualnem modelu URARNE koristno uporabimo povezave tipa *specializacija*, *generalizacija*, *agregacija* ali *kompozicija*?

Metoda konceptualnega načrtovanja

- Možni koraki konceptualnega načrtovanja:
  - K1.1: Identificiraj entitetne tipe
  - K1.2: Identificiraj povezave
  - K1.3: Identificiraj in z entitetnimi tipi poveži atribute
  - K1.4: Atributom določi domene
  - K1.5: Določi kandidate za ključe; izmed kandidatov izberi primarni ključ
  - K1.6: Po potrebi uporabi elemente razširjenega diagrama entiteta – razmerje
  - K1.7: Preveri, če v modelu obstajajo odvečni elementi
  - K1.8: Preveri, če model "zdrži" transkacije
  - K1.9: Preveri model z uporabnikom

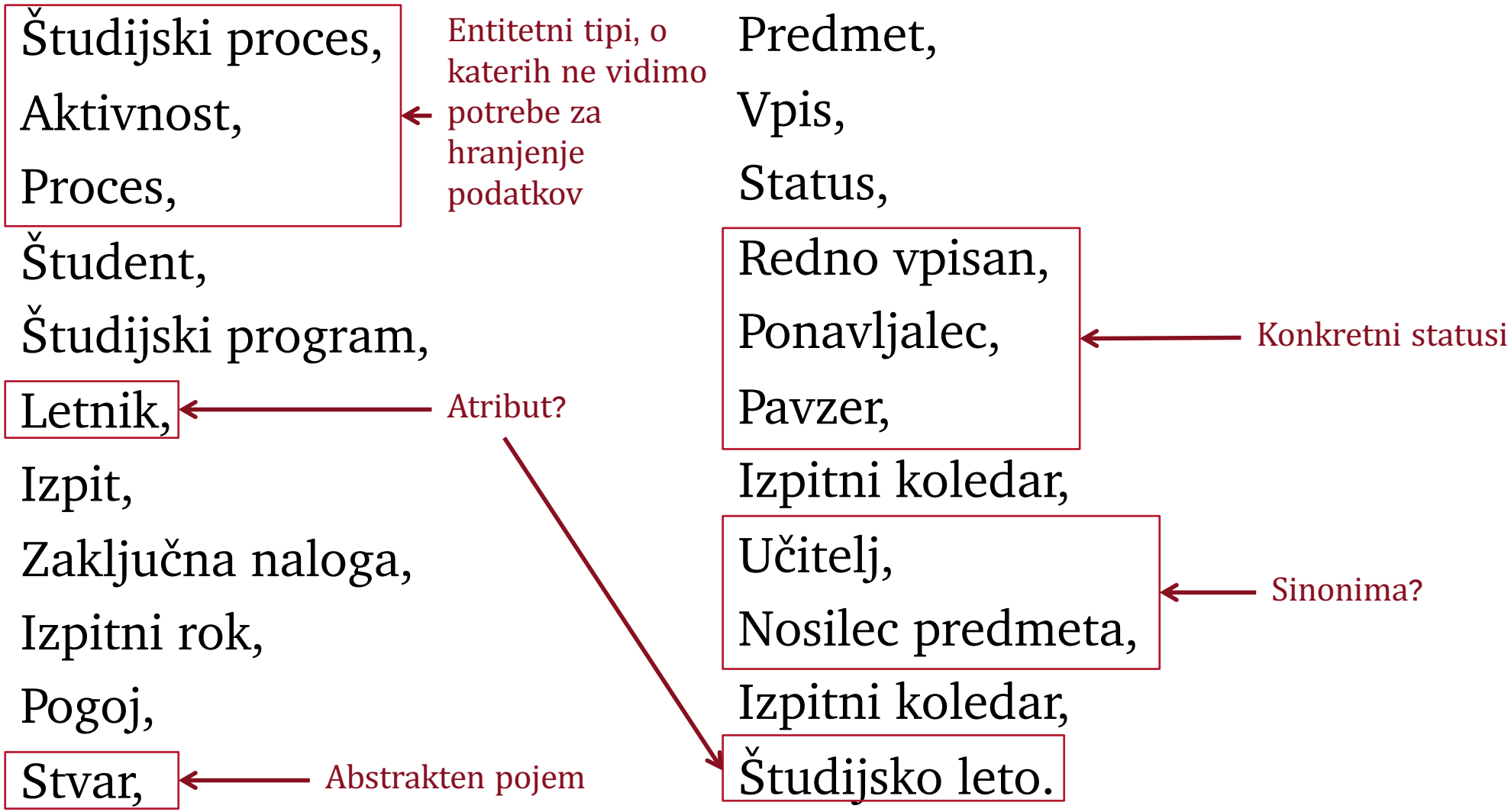
K1.1 – Identificiraj entitetne tipe...

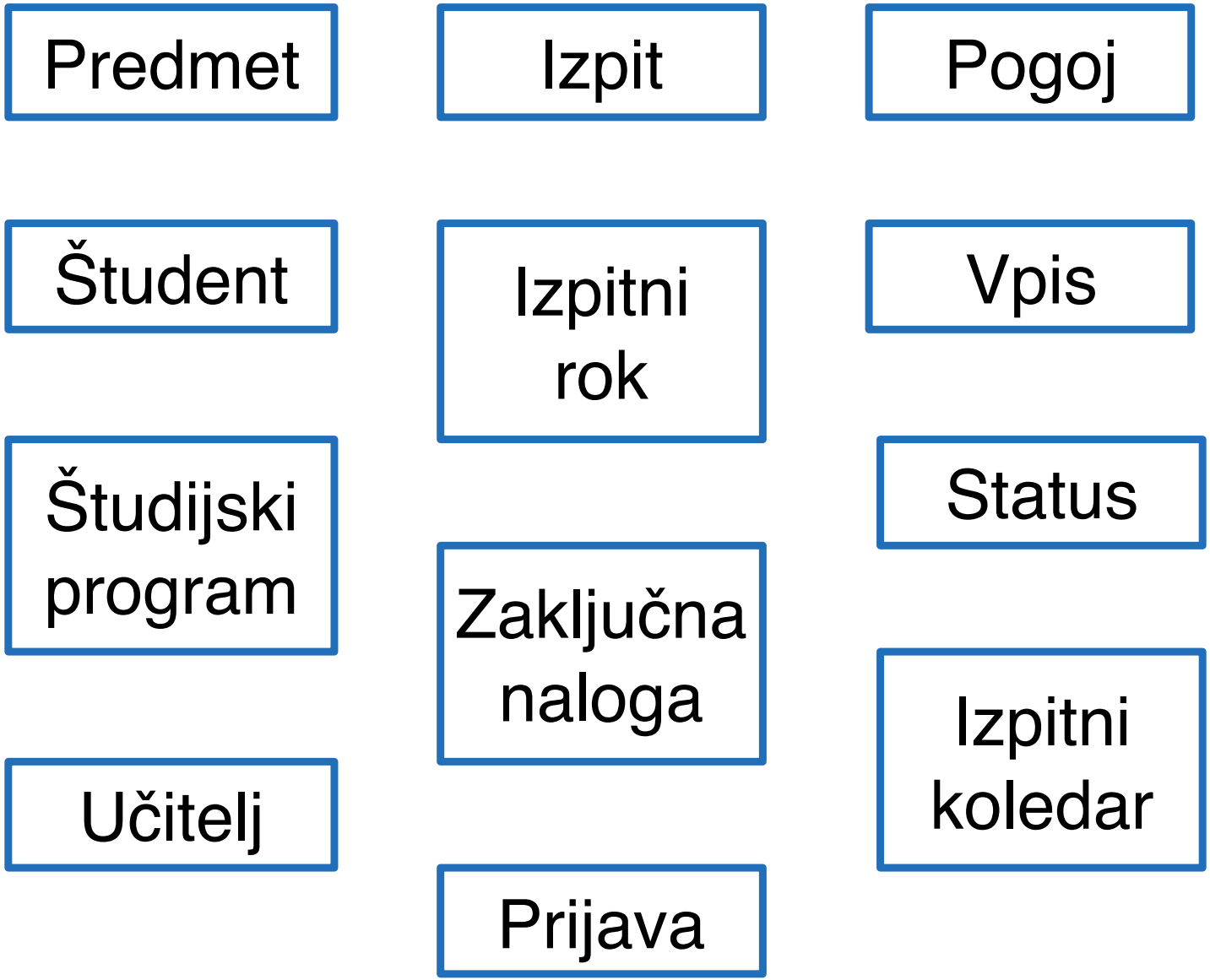
- Na voljo različne tehnike
- Ena izmed tehnik je pregled uporabniških zahtev:
  - Pregledamo vse omenjene samostalnike in fraze (npr. *profesor*, *predmet*, *izpit*, *rok*, *datum izpita*,...)
  - Pozorni smo na pomembne objekte (npr. *ljudje*, *lokacije*...)
  - Skušamo ločiti objekte (npr. *profesor*, *izpit*,...) od lastnosti objektov (*ime*, *vpisna številka*,...)
  - Lastnosti objektov združujemo v entitetne tipe

K1.1 – Identificiraj entitetne tipe...

- Težave:
  - Entitete niso vedno jasno predstavljene v dokumentaciji
    - Uporaba primerov, analogij, sinonimov, homonimov
    - Uporaba konkretnih imen oseb
  - Ni vedno jasno, kaj je entitetni tip, kaj povezava in kaj atribut (npr. *izpit*)
- Načrtovanje je subjektivne narave – možnih je več (pravilnih) rešitev.
- Načrt odvisen od uporabnikove presoje in izkušenj

Študijski proces zajema vse aktivnosti in procese, ki se nanašajo na vpis študenta na nek študijski program, vpis v višji letnik, opravljanje izpitov, opravljanje zaključnih nalog itn. Polaganje izpita za nek predmet zgleda tako, da se študent najprej prijavi na izpitni rok za ta predmet, kar je možno le, če izpolnjuje vse pogoje. Pogoji za prijavo na izpit so odvisni od mnogih stvari. Od tega, kakšen status ima študent (redno vpisan, ponavljalec, pavzer…), v kateri letnik in kateri študijski program je vpisan, kolikokrat je že opravljal ta predmet, katere predmete je že opravil itn. Večina izpitnih rokov je razpisanih z izpitnim koledarjem (ta je znan za celo študijsko leto v naprej), posamezen učitelj, nosilec predmeta…


Študijski proces,
Aktivnost,
Proces,
Entitetni tipi, o katerih ne vidimo potrebe za hranjenje podatkov
Študent,
Študijski program,
Letnik,
Atribut?
Izpit,
Zaključna naloga,
Izpitni rok,
Pogoj,
Stvar,
Abstrakten pojem
Prijava,
Predmet,
Vpis,
Status,
Redno vpisan,
Ponavljalec,
Pavzer,
Konkretni statusi
Izpitni koledar,
Učitelj,
Nosilec predmeta,
Sinonima?
Izpitni koledar,
Študijsko leto.


Predmet
Izpit
Pogoj
Študent
Izpitni rok
Vpis
Študijski program
Status
Zaključna naloga
Izpitni koledar
Učitelj
Prijava

K1.1 – Identificiraj entitetne tipe

- Entitetne tipe je potrebno dokumentirati
- Primer dokumentacije:

Naziv entitetnega tipa	Opis	Sinonim	Število entitet
Profesor	Predstavlja pedagoškega delavca, ki je nosilec enega ali več predmetov	Pedagoški delavec	Vsaka katedra ima enega ali več profesorjev
Izpitni rok	Predstavlja datum, na katerega je za nek predmet in določeno ciljno skupino (letnik, smer,...) razpisan izpitni rok.	Rok, pisni izpit, kolokvij	Na leto se razpiše okrog 300 pisnih izpitov. Vsak predmet mora imeti vsaj tri roke letno
...			

K1.2 – Identificiraj povezave...

- Ko identificiramo entitetne tipe, skušamo opredeliti vse povezave med njimi.
- Uporabimo lahko podoben postopek kot v K1 (pregled uporabniških zahtev):
  - Iščemo glagole (npr. profesor *razpiše* rok, študent *polaga* izpit, študent *izbere* mentorja, študent *se vpiše* v letnik,...)
  - Zanimajo nas samo tiste povezave, ki so res potrebne (očitne povezave ali povezave, ki nas ne zanimajo z vidika hranjenja podatkov, so odveč)

K1.2 – Identificiraj povezave...

- Postopek identifikacije povezav (nadaljevanje)
  - Pozorni smo na povezave, ki niso binarne - povezujejo več kot dve entiteti ali so rekurzivne.
    - Npr. *študent opravi izpit* za nek *predmet* pri nekem *profesorju*.
    - Ali, *pedagoški delavec ima asistenta*, ki *je* tudi *pedagoški delavec*.
  - Preverimo, če smo zajeli vse povezave (načeloma lahko preverimo za vsak par entitetnih tipov, če med njima obstaja povezava) – postopek je lahko zelo potraten, zato ga ne izvajamo vedno (preverjanje modela je stvar K8).

Študijski proces zajema vse aktivnosti in procese, ki se nanašajo na vpis študenta na nek študijski program, vpis v višji letnik, opravljanje izpitov, opravljanje zaključnih nalog itn. Polaganje izpita za nek predmet zgleda tako, da se študent najprej prijavi na izpitni rok za ta predmet, kar je možno le, če izpolnjuje vse pogoje. Pogoji za prijavo na izpit so odvisni od mnogih stvari. Od tega, kakšen status ima študent (redno vpisan, ponavljalec, pavzer…), v kateri letnik in kateri študijski program je vpisan, kolikokrat je že opravljal ta predmet, katere predmete je že opravil itn. Večina izpitnih rokov je razpisanih z izpitnim koledarjem (ta je znan za celo študijsko leto v naprej), posamezen učitelj, nosilec predmeta…

Študent *se vpiše* na študijski program

Študent *se prijavi* na izpitni rok za predmet

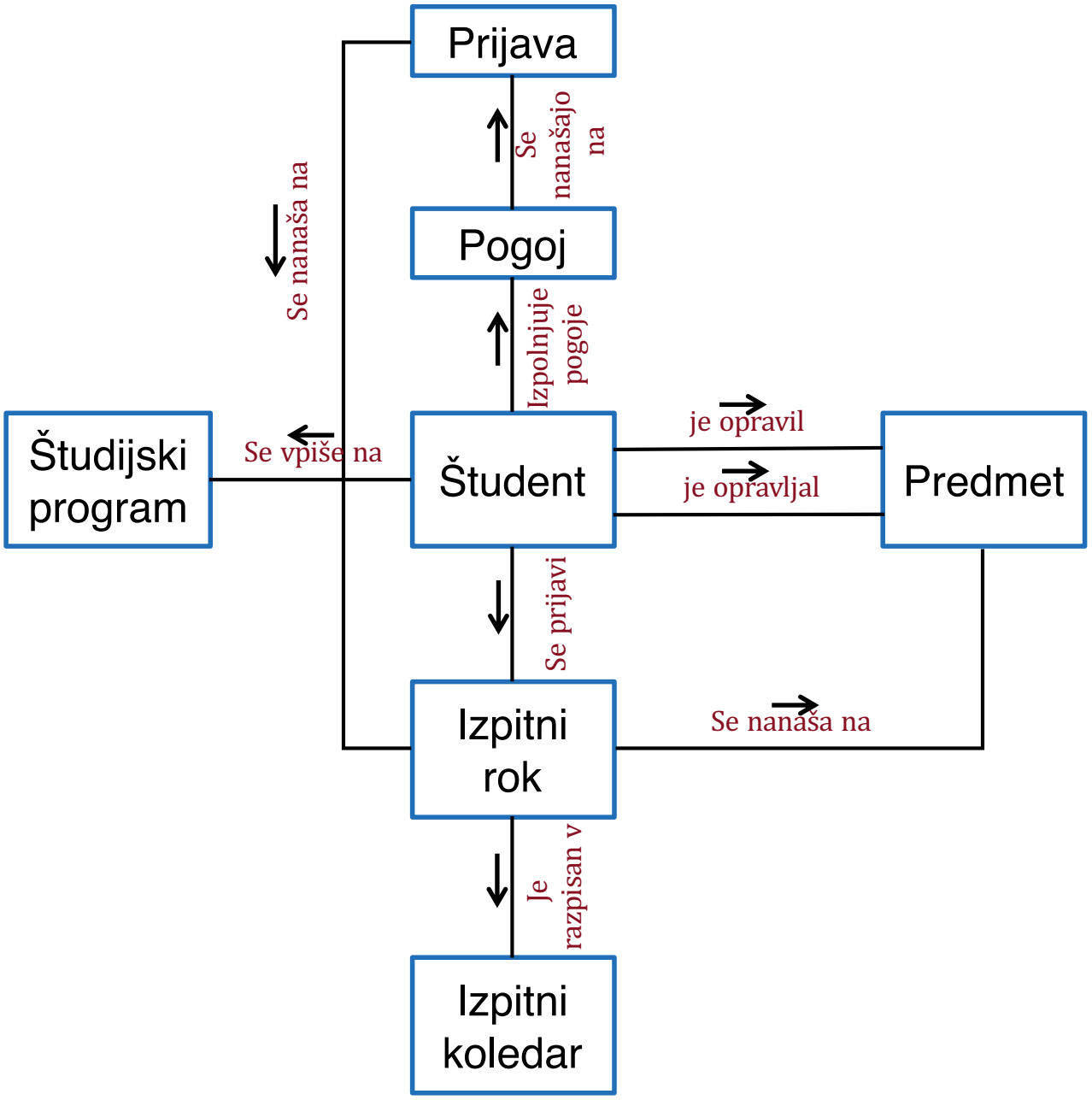
Študent *izpolnjuje* pogoje za prijavo

Študent *je opravljal* predmet

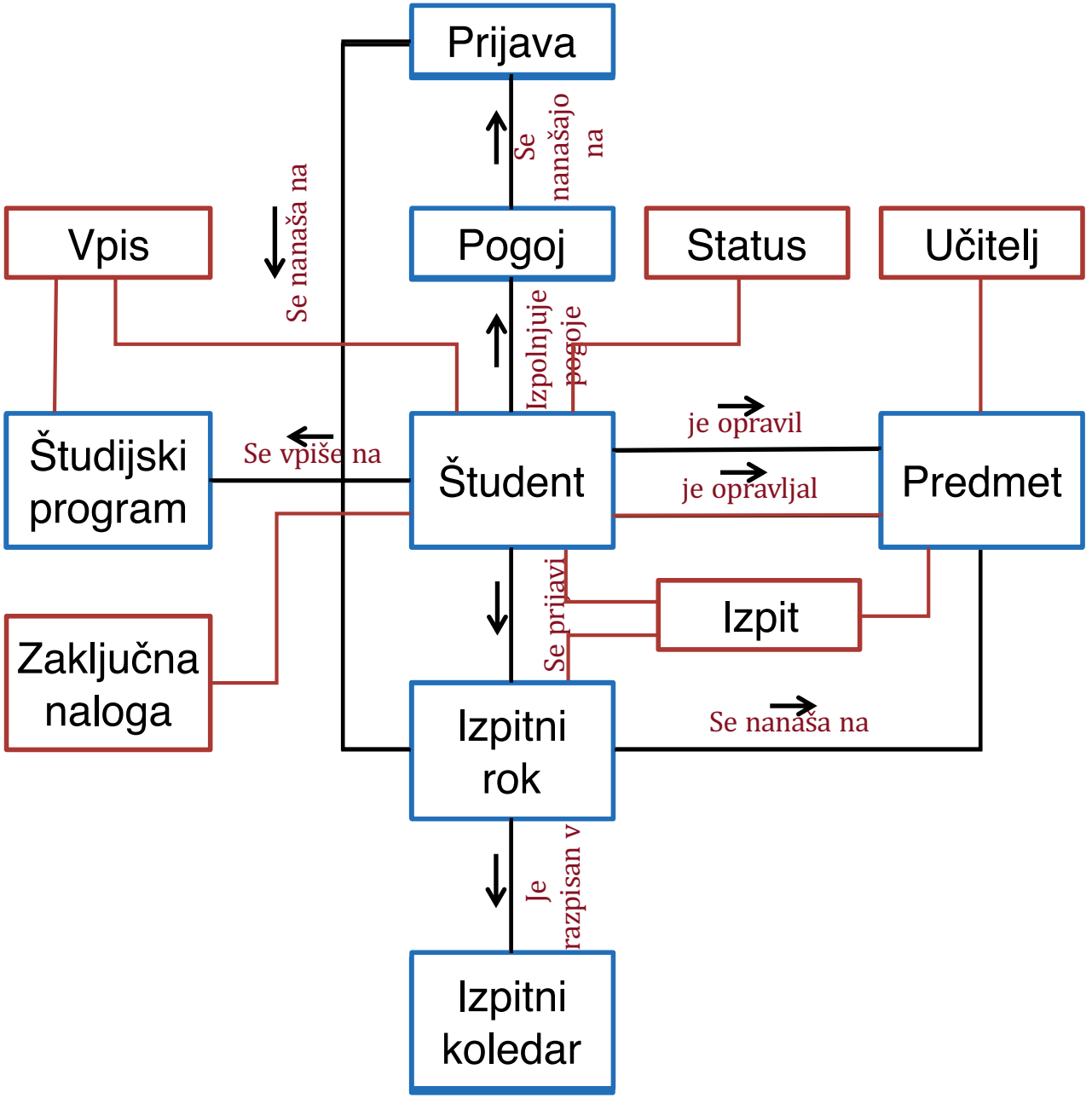
Študent *je opravil* predmet

Izpitni rok *je razpisan* z izpitnim koledarjem

...

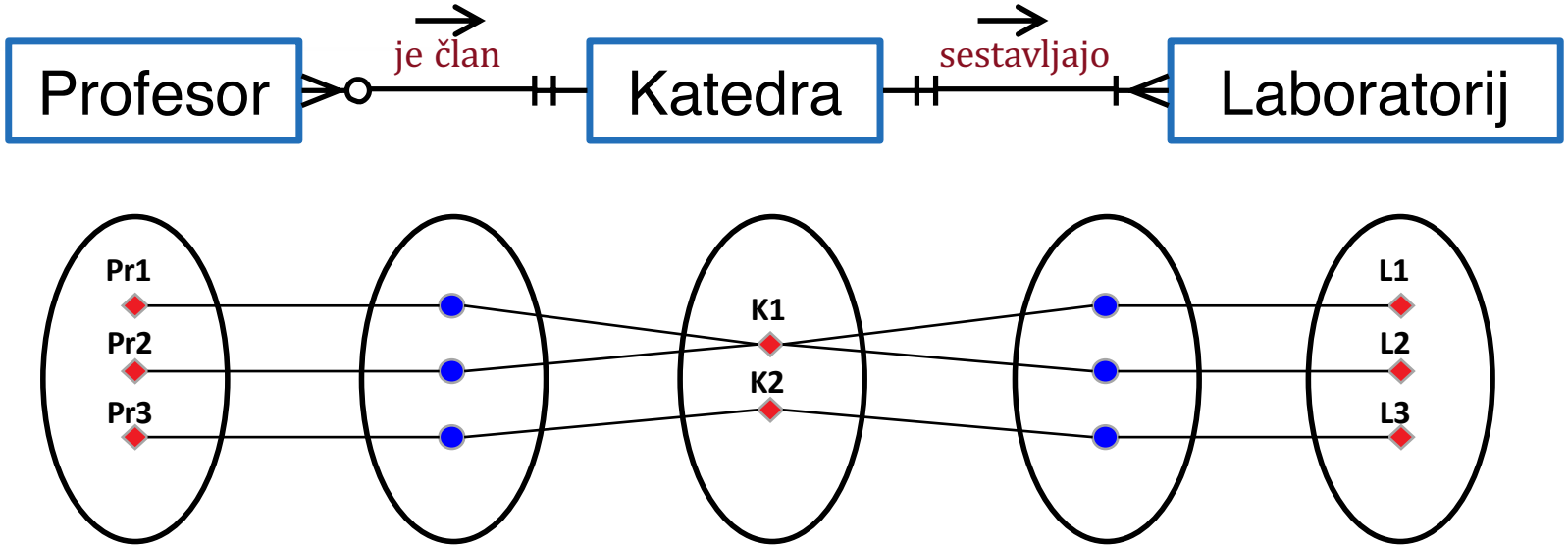

Prijava
Se nanašajo na
Pogoj
Se nanaša na
Izpolnjuje pogoje
Študijski program
Se vpiše na
Študent
je opravil
je opravljal
Predmet
Se prijavi
Izpitni rok
Se nanaša na
Je razpisan v
Izpitni koledar

Kakšne so številnosti povezav?



K1.2 – Identificiraj povezave...

- Pri identifikaciji povezav in njihove številnosti moramo biti pozorni na dvoumne in nepopolne povezave.

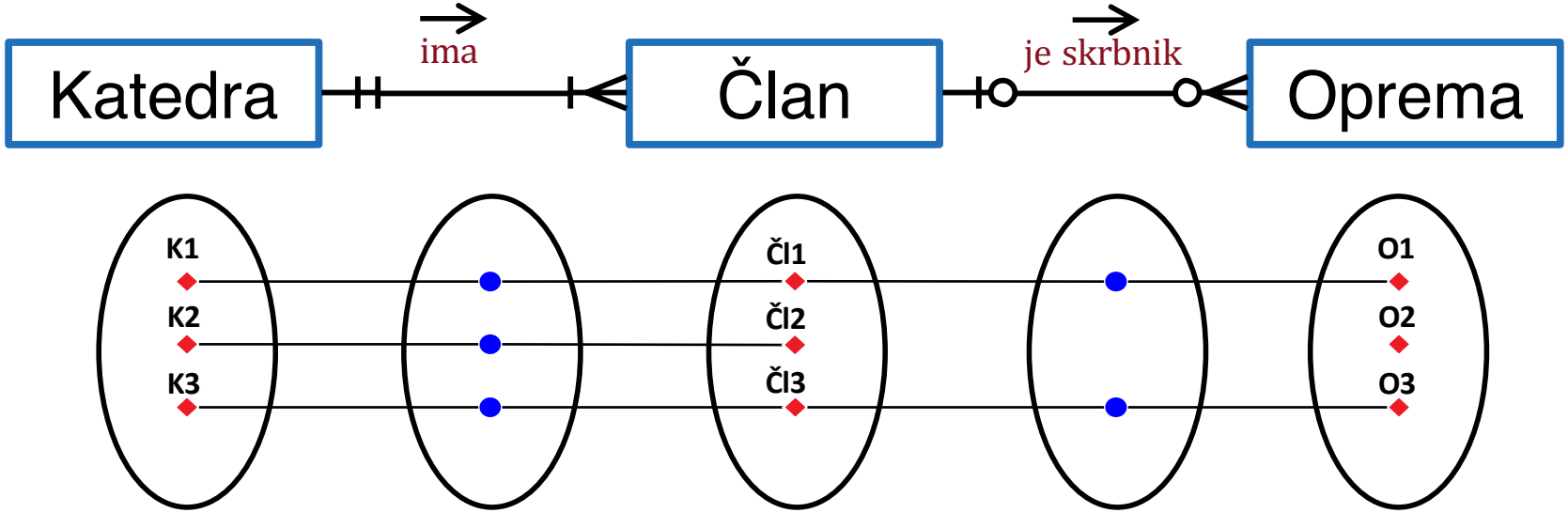


V katerem laboratoriju dela profesor `Pr1`?

K1.2 – Identificiraj povezave...

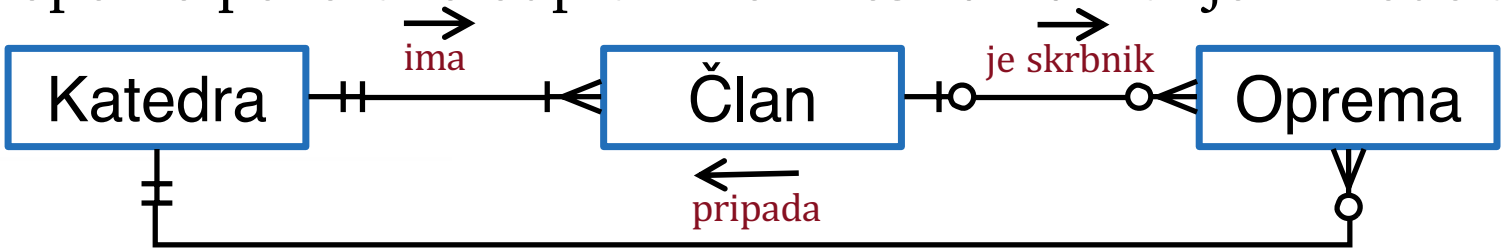
- Primer nepopolne povezave.

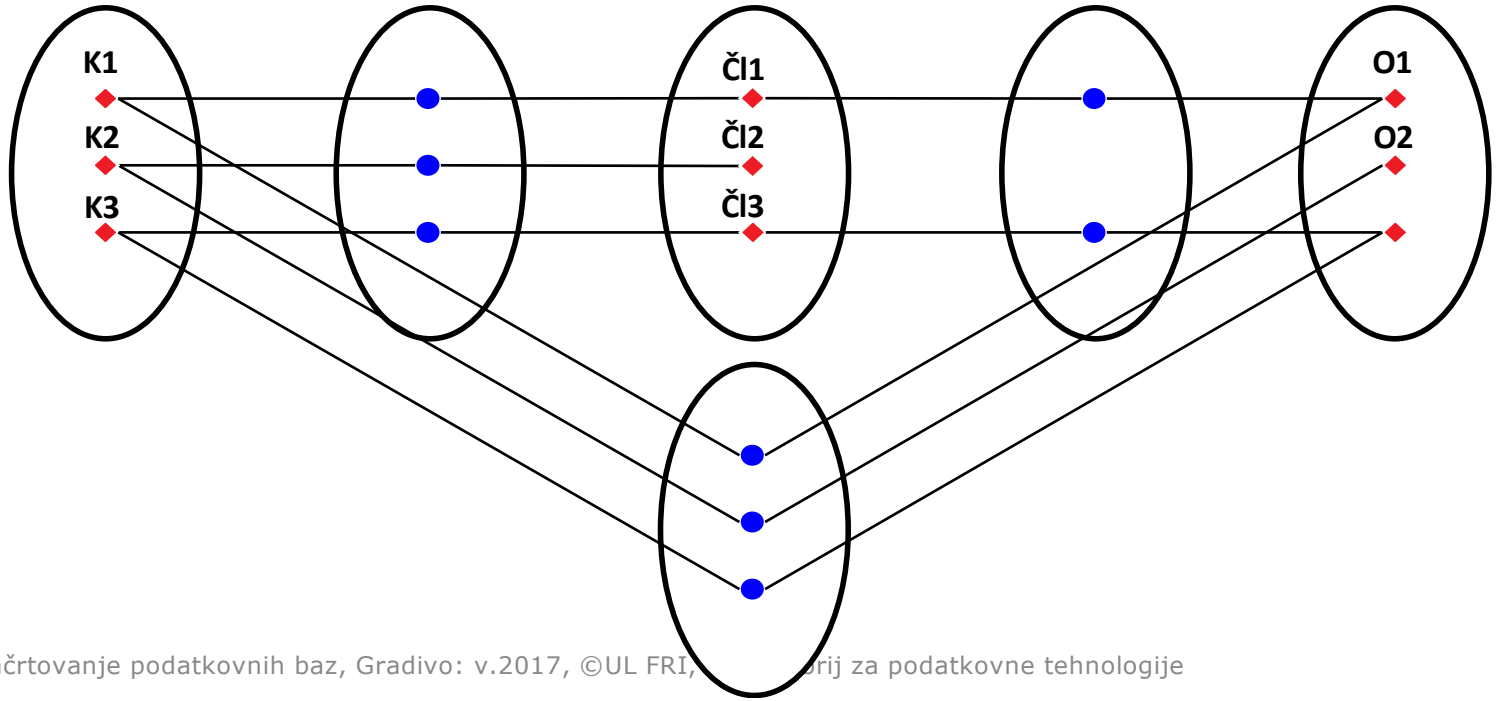
  Kateri katedri pripada oprema o2?



K1.2 – Identificiraj povezave...

- Nepopolno povezavo odpravimo z restrukturiranjem modela





K1.2 – Identificiraj povezave

- Tudi povezave je potrebno dokumentirati!

Entitetni tip	Številnost	Povezava	Številnost	Entitetni tip
Član	1..* 1..1	Pripada Je predstojnik	1..1 0..1	Katedra
Laboratorij	1..*	Sodi v	1..1	Katedra
...				

K1.3 – Identificiraj atribute...

- Skušamo identificirati lastnosti entitet ter povezav
- Uporabimo lahko tehniko proučevanja uporabniških zahtev
  - iščemo samostalnike, ki predstavljajo lastnosti, opisne vrednosti ali identifikatorje objektov
- Korak določanja atributov entitetnih tipov je relativno enostaven

K1.3 – Identificiraj atribute…

- Nekaj primerov, kjer je potrebna pazljivost

Sestavljeni atributi

… za vsakega študenta hranimo podatek o stalnem prebivališču, če ima začasnega pa tudi o začasnem prebivališču. …

Več-vrednostni atributi

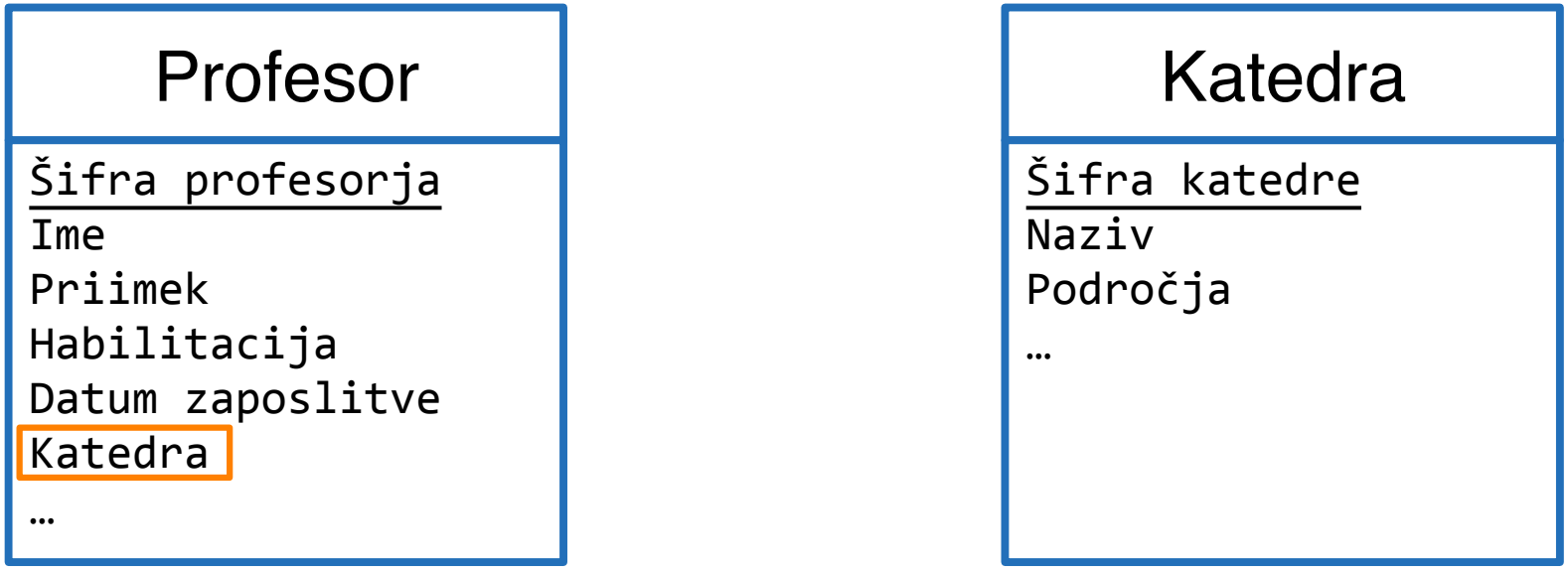
… o študentih in zaposlenih hranimo tudi kontaktne podatke, kot so elektronska pošta, telefonska številka ipd. …

Izpeljani atributi

… eden od pomembnih kriterijev pri preverjanju pogojev za prijavo na izpit je število dosedanjih polaganj izpita in sicer vseh ter v tekočem študijskem letu…

K1.3 – Identificiraj atribute...

- Dodatna priporočila:
  - Če identificiramo atributi, ki navidez pripadajo več entitetam, preverimo:
    - Ali je možno združiti entitete (npr. *asistent* in *profesor* združimo v *pedagoški delavec*);
    - Če imajo entitete več skupnih vendar tudi svoje atribute, razmislimo o uporabi generalizacije (npr. poleg entitetnega tipa *asistent* in *profesor* uvedemo še entitetni tip *pedagoški delavec*, ki prevzame vse skupne atribute.)
  - Če identificiramo atribut, ki odraža povezavo (npr. atribut *katedra* v entitetnem tipu *Profesor* predstavlja povezavo z entiteto katedra):
    - Če povezava obstaja, potem je atribut odveč
    - Če povezava ne obstaja, jo je potrebno dodati ter atribut zbrisati



Pazi: atribut je atomaren, hrani lahko le eno vrednost!!

K1.3 – Identificiraj atribute

- Atribute je potrebno dokumentirati:
  - Naziv atributa, opis, podatkovni tip, dolžina, sinonimi, ali je atribut sestavljen (iz katerih atributov je sestavljen?), ali je atribut izpeljan (iz katerih atributov je izpeljan?),...

Entitetni tip	Atributi	Opis	Podatkovni tip	Dolžina	...
Študent	VpisSt Ime Priimek ...	Vpisna številka študenta Ime študenta Priimek študenta	Number Character Character	8 20 20	...
...					...

K1.4 – Atributom določi domene...

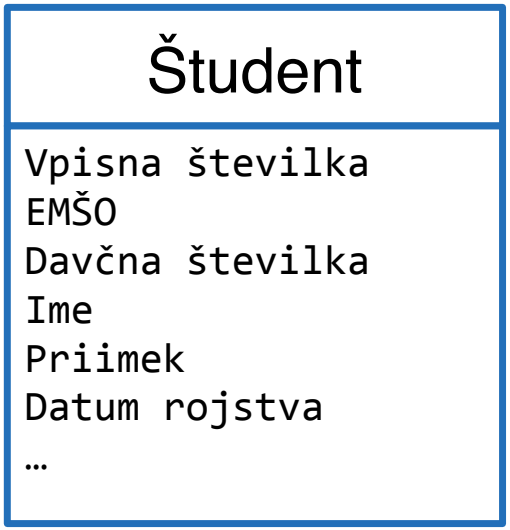
- Domena je množica vrednosti, ki jih lahko zavzamejo atributi, vključeni v to domeno.
- Domeni lahko določimo:
  - Seznam dovoljenih vrednosti
  - Minimalno in maksimalno vrednost
  - Podatkovni tip in dolžino
  - Dovoljene operacije nad atributom (še v raziskavi)
- Primeri domen:
  - `Barva → {bela, rumena, oranžna, rdeča}`
  - `Opis elementa → character 50`
  - `Starost → [0..120]`
  - `EMSO → number 13`

K1.4 – Atributom določi domene

- Tudi domene dokumentiramo
- Zapišemo naziv domene ter lastnosti oz. pravila, ki jih domena določa.

K1.5 - Določi kandidate za ključe...

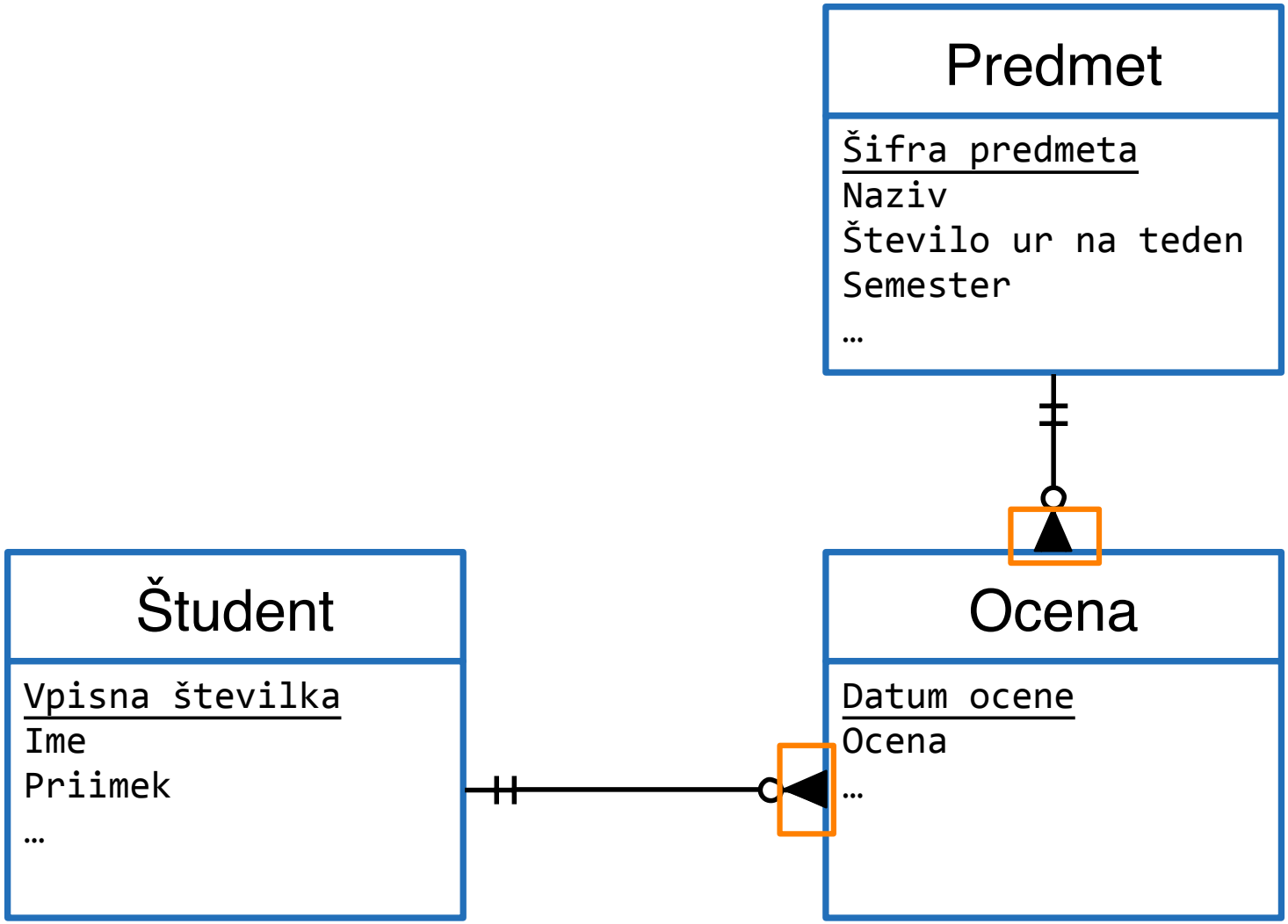
- Za vsak entitetni tip določimo kandidate za ključ ter izberemo enega za primarni ključ.
- Kandidati za ključ so minimalne podmnožice atributov, ki enolično identificirajo vsako entiteto.
- Če je kandidatov več, izberemo enega, ki je primeren za primarni ključ.



K1.5 - Določi kandidate za ključe...

- Nekaj priporočil za izbiro ključa, če je več kandidatov:
  - Kandidat z najmanj atributi;
  - Kandidat, za katerega je najmanj verjetno, da se bodo njegove vrednosti spreminjale;
  - Kandidat z najmanjšo dolžino znakov (za alfanumerične kandidate);
  - Kandidat z najmanjšo maksimalno vrednostjo (za numerične kandidate);
  - Kandidat, ki ga je najlažje uporabiti s stališča uporabnika;
  - Imena navadno niso dober kandidat za ključ;
  - Pazi: šibkim entitetnim tipom v okviru konceptualnega načrtovanja ne moremo določiti ključa.
- Tudi primarne in alternativne ključe je potrebno dokumentirati.

Označitev šibkega entitetnega tipa v konceptualnem modelu



K1.6 – uporabi elemente EER diagrama

- Kdaj uporabiti elemente razširjenega ER diagrama?
  - Elementi razširjenega diagrama povečajo semantiko modela vendar lahko negativno vplivajo na “berljivost” modela
  - Berljivost, enostavnost modela naj bo vodilo pri odločanju o uporabi naprednih modelirnih elementov (predvsem agregacija in kompozicija)

K1.7 - Preveri obstoj odvečnih elementov

- Preverimo, če v modelu obstajajo redundantni elementi:
  - Pregledamo povezava 1 – 1
  - Odstranimo odvečne povezave

K1.7 - Preveri obstoj odvečnih elem....

- Povezave 1 – 1
  - Pri identifikaciji entitetnih tipov smo morda zajeli več tipov, ki predstavljajo iste objekte (npr. Profesor, Pedagoški delavec, Asistent)
  - Če taki tipi obstajajo, jih je potrebno združiti
  - Če so primarni ključi različni, izberemo enega

Profesor
Šifra profesorja
Ime
Priimek
Vzdevek
Habilitacija
Datum rojstva
…

Asistent
Šifra asistenta
Ime
Priimek
Habilitacija
Datum rojstva
…

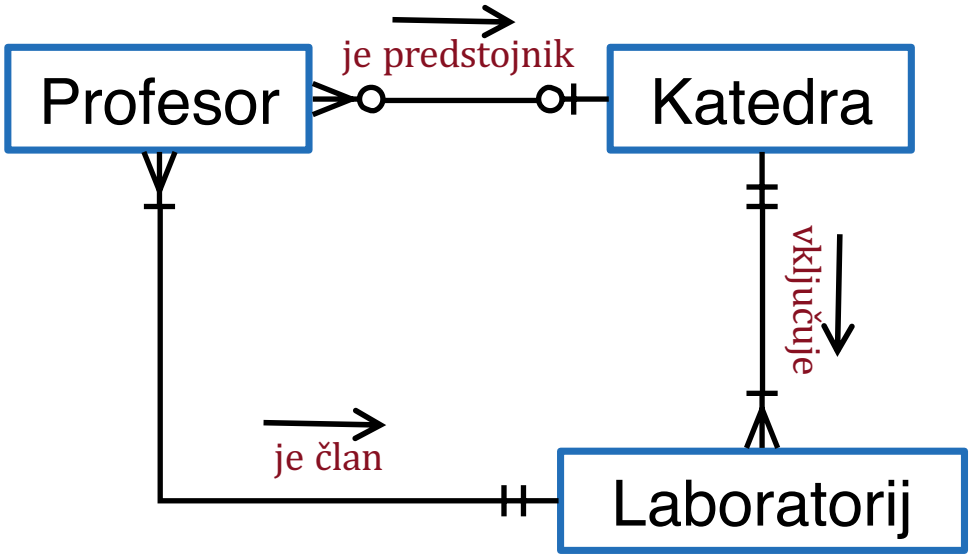
Delavec
Šifra delavca
Ime
Priimek
Vzdevek
Habilitacija
Datum rojstva
…

K1.7 - Preveri obstoj odvečnih elem....

- Odstrani odvečne povezave
  - Povezava je odvečna, če je možno priti do iste informacije prek drugih povezav!
  - Izdelati želimo minimalen podatkovni model → odvečne povezave zato odstranimo.
  - Zgolj pregledovanje poti med entitetnimi tipi ne zadošča (povezave imajo lahko različen pomen)

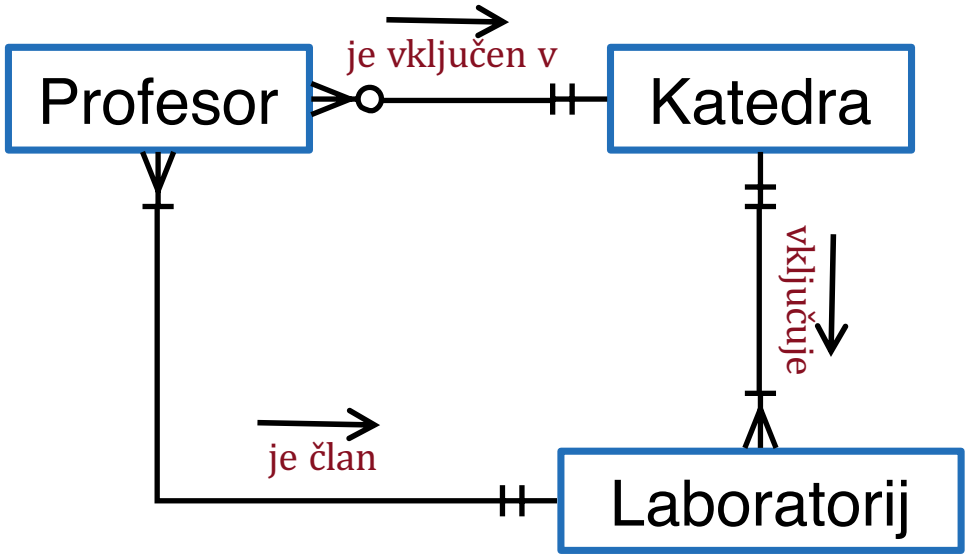
K1.7 - Preveri obstoj odvečnih elem....

- Ali je kakšna povezava odveč?



K1.7 - Preveri obstoj odvečnih elem....

- Ali je kakšna povezava odveč?



K1.8 - Preveri če model zdrži transkacije...

- Preveriti moramo če model, ki smo ga dobili s koraki od K1 do K7, podpira vse zahtevane transakcije.
- Če neke transkacije ne uspemo izvesti, je model pomanjkljiv (manjka bodisi entitetni tip, povezava ali atribut)

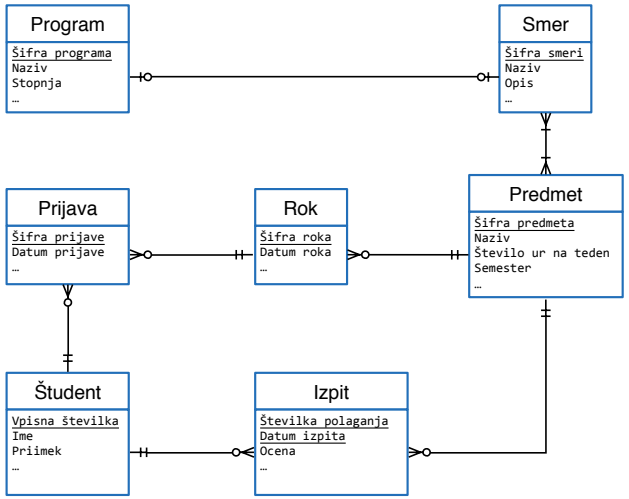
- Preverjanje opisa transakcij
  - Vsako transakcijo opišemo;
  - Preverimo, če model zajema vse entitetne tipe, povezave in atribute, ki jih transakcija potrebuje.

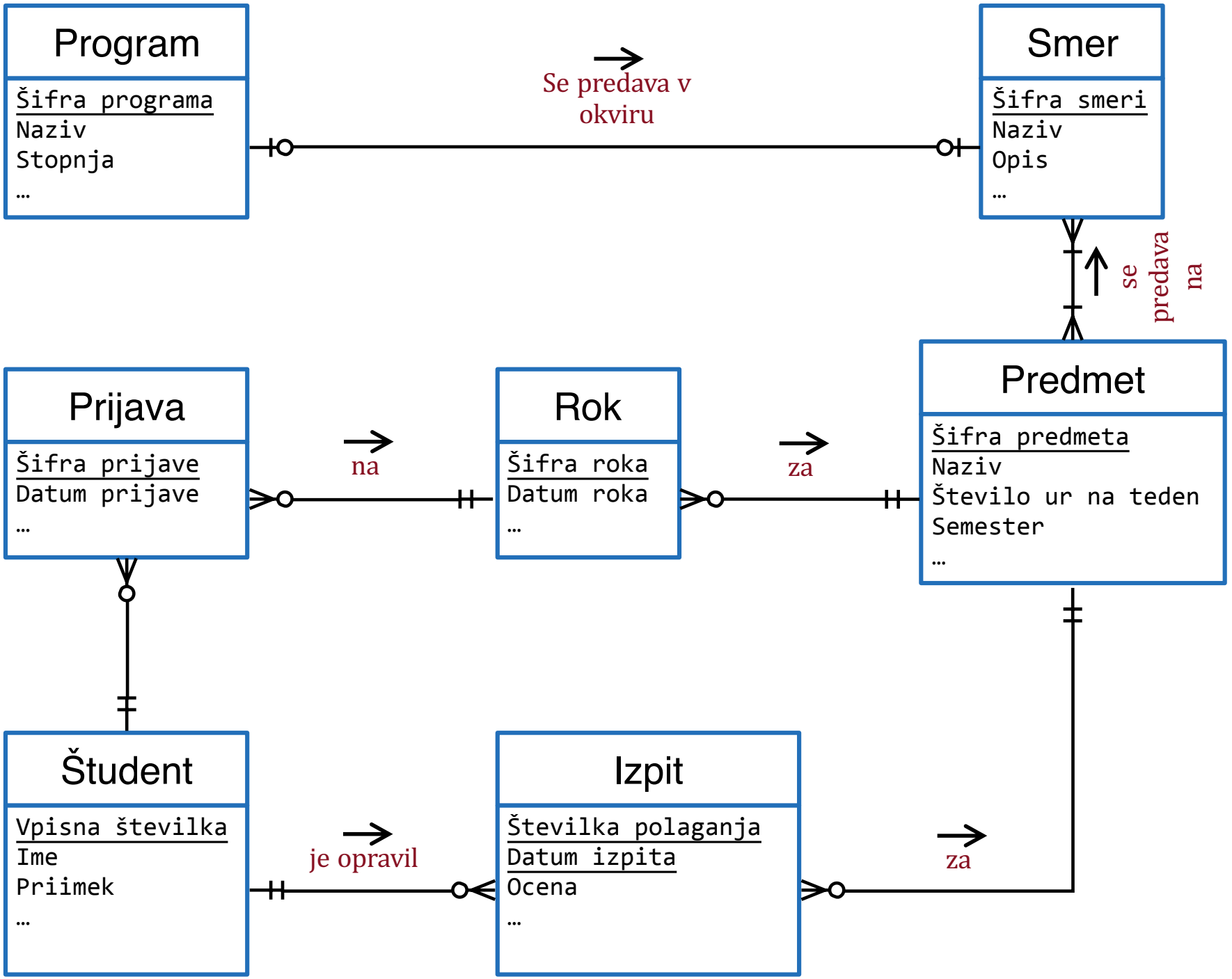
K1.8 - Preveri če model zdrži transkacije...

- Primer opisa transakcijskih zahtev
  - Vnos podatkov:
    - Vnesi podatke o študentih (npr. 24010637, Monika Jemec,...)
    - Vnesi podatke o predmetih (npr. 70029, Razvoj IS, Letni,...)
    - ...
  - Urejanje in brisanje podatkov:
    - Uredi/briši podatke o študentu
    - Uredi/briši podatke o predmetih
    - ...
  - Poizvedbe
    - Izpiši vse študente, ki so se vpisali v določen letnik, določene smeri, določenega programa
    - Izpiši vse predmete, ki jih je opravil določen študent
    - ...

K1.8 - Preveri, če model zdrži transkacije

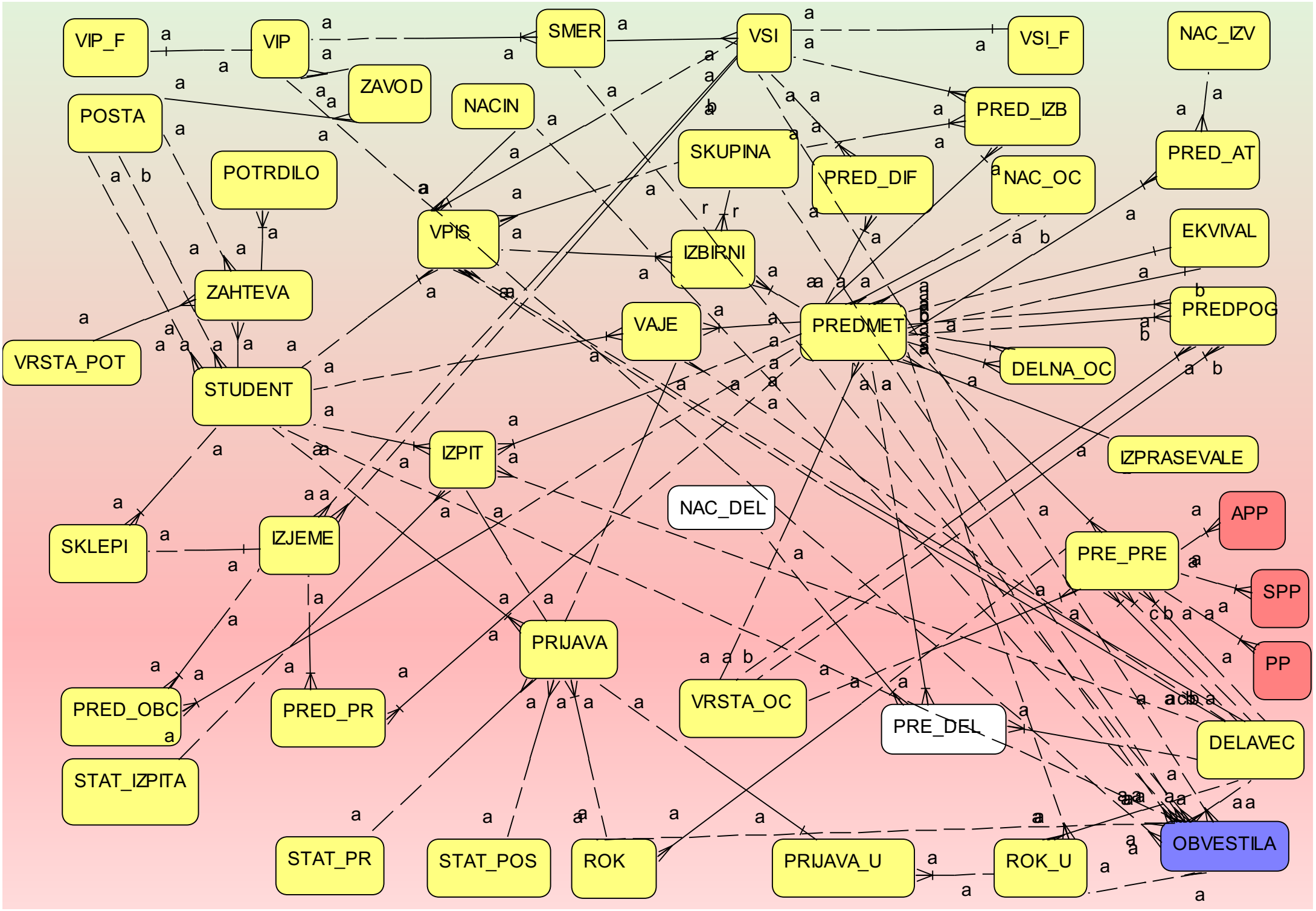
- Preveri naslednji dve transakciji:
  - T1: Izpiši vse predmete, ki jih je opravil določen študent
  - T2: Izpiši vse študente, ki so se vpisali v določen letnik, določene smeri, določenega programa




Program
Šifra programa
Naziv
Stopnja
…
Se predava v okviru
Smer
Šifra smeri
Naziv
Opis
…
se predava na
Predmet
Šifra predmeta
Naziv
Število ur na teden
Semester
…
Prijava
Šifra prijave
Datum prijave
…
na
Rok
Šifra roka
Datum roka
…
za
Študent
Vpisna številka
Ime
Priimek
…
je opravil
Izpit
Številka polaganja
Datum izpita
Ocena
…
za

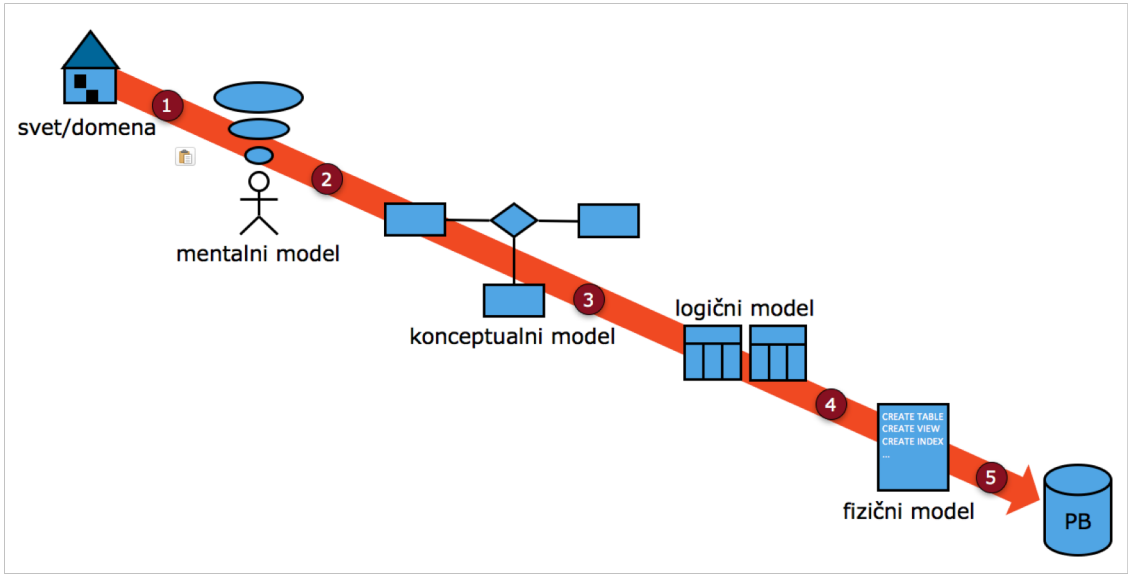
K1.9 – Preveri model z uporabnikom

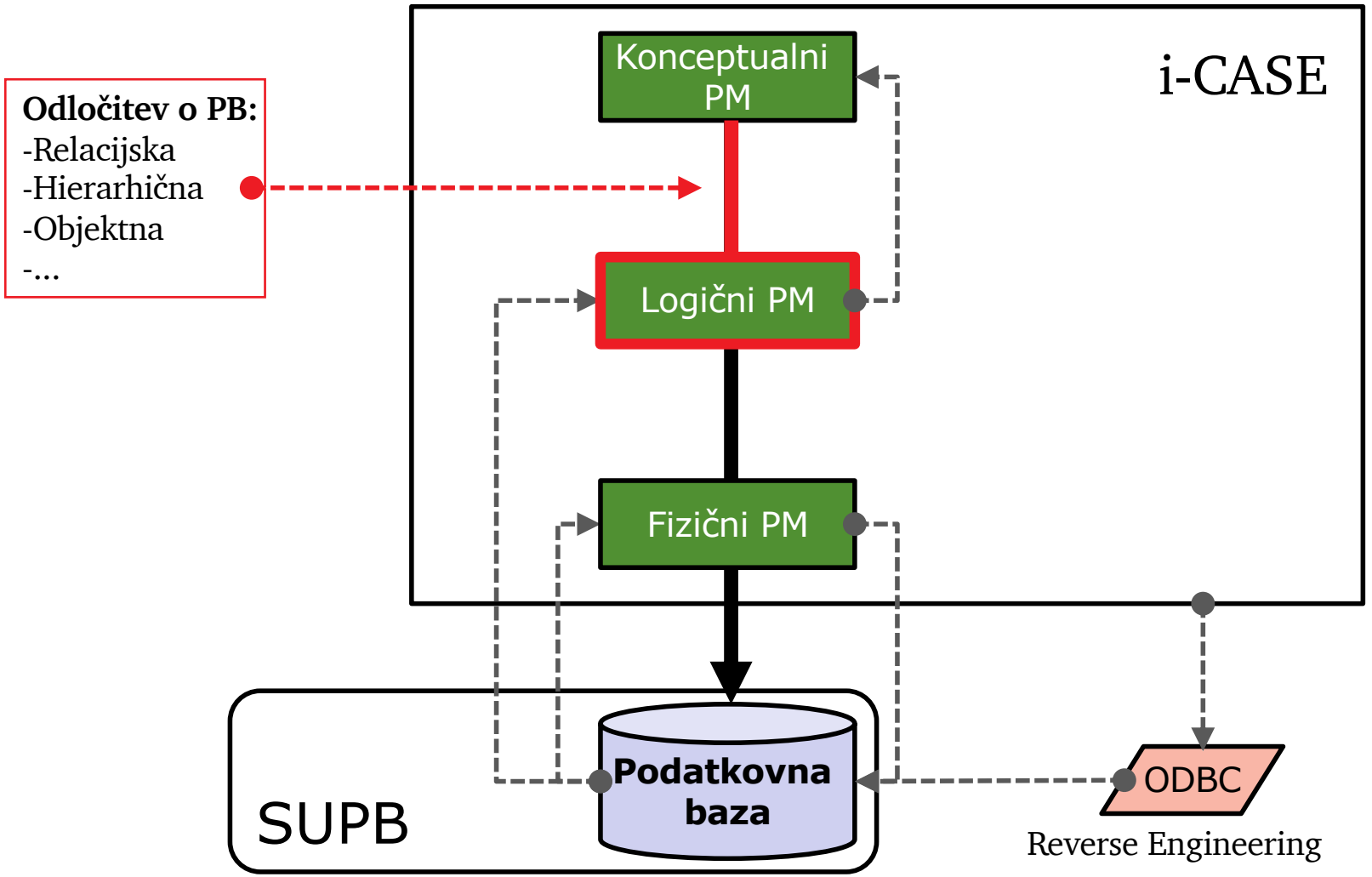
- Na koncu model preverimo z uporabnikom
- Anomalije, pomanjkljivosti, napake,... lahko vodijo v ponovitev korakov od K1 do K9.
- V mnogih podjetjih mora uporabnik podpisati podatkovni model


VIP_F
VIP
SMER
VSI
VSI_F
NAC_IZV
ZAVOD
POSTA
NACIN
PRED_IZB
SKUPINA
PRED_AT
POTRDILO
PRED_DIF
NAC_OC
VPIS
IZBIRNI
EKVIVAL
ZAHTEVA
PREDPOG
VAJE
PREDMET
VRSTA_POT
DELNA_OC
STUDENT
IZPIT
IZPRASEVALE
NAC_DEL
APP
SKLEPI
IZJEME
PRE_PRE
SPP
PRIJAVA
PP
PRED_OBC
PRED_PR
VRSTA_OC
PRE_DEL
DELAVEC
STAT_IZPITA
STAT_PR
STAT_POS
ROK
PRIJAVA_U
ROK_U
OBVESTILA

Logično načrtovanje podatkovne baze...

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

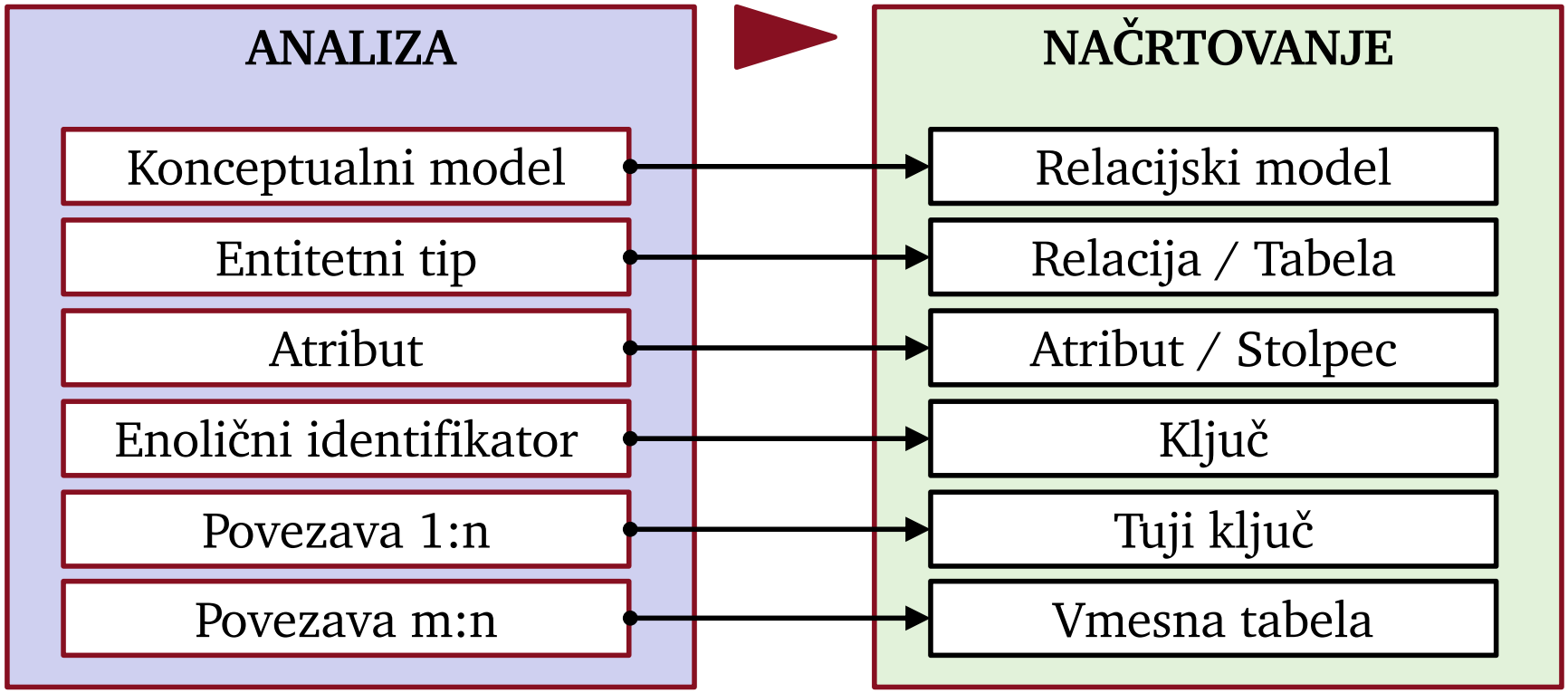



Odločitev o PB:
-Relacijska
-Hierarhična
-Objektna
-...
Konceptualni PM
Logični PM
Fizični PM
i-CASE
Podatkovna baza
SUPB
ODBC
Reverse Engineering

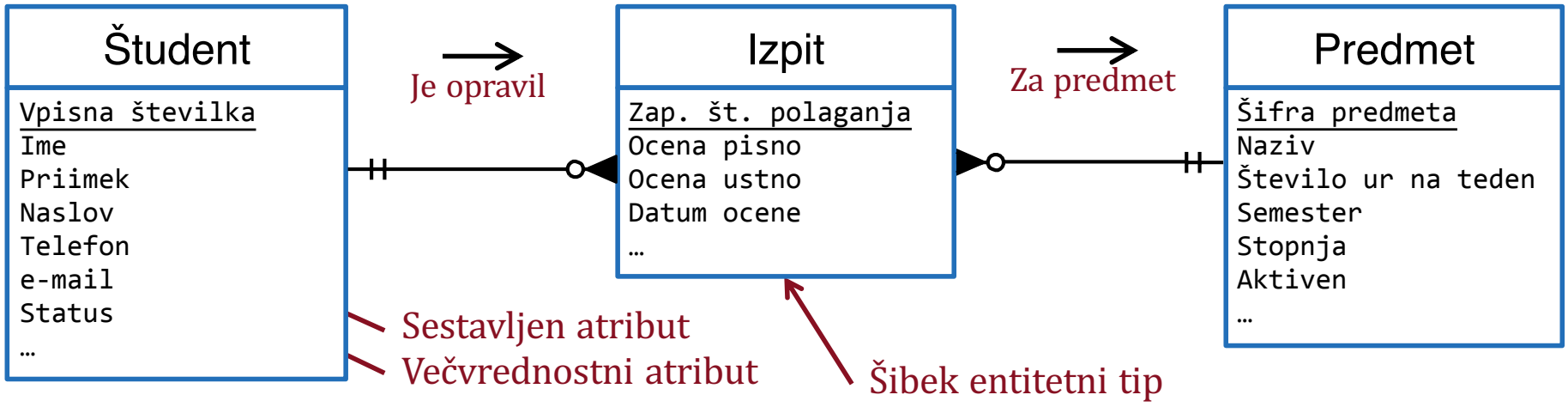
Prehod iz konceptualnega v logični model

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

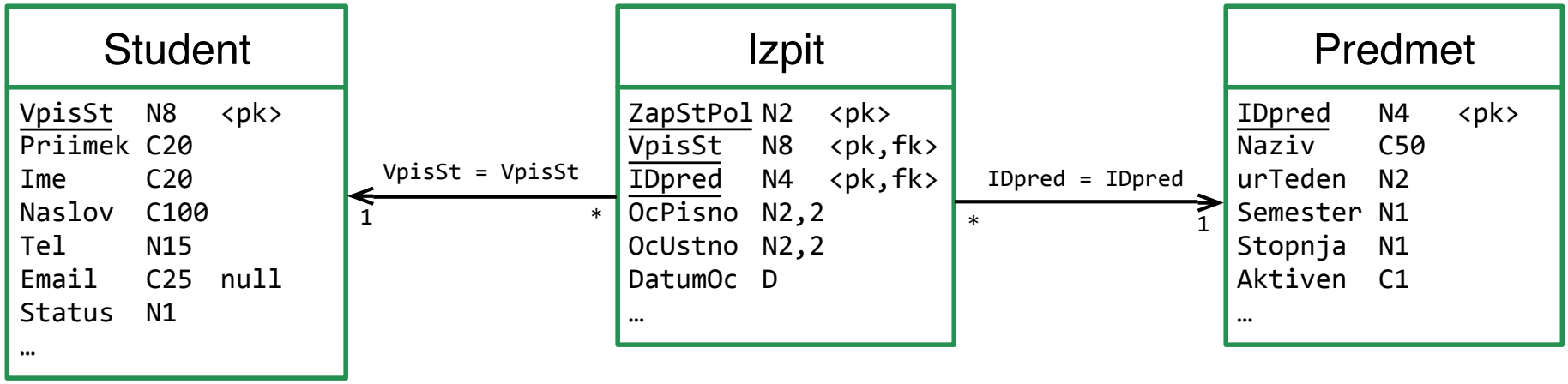
vrsta baze: relacijska
SUPB: Oracle 12c



Konceptualni model



Logični model



Logični načrtovanje

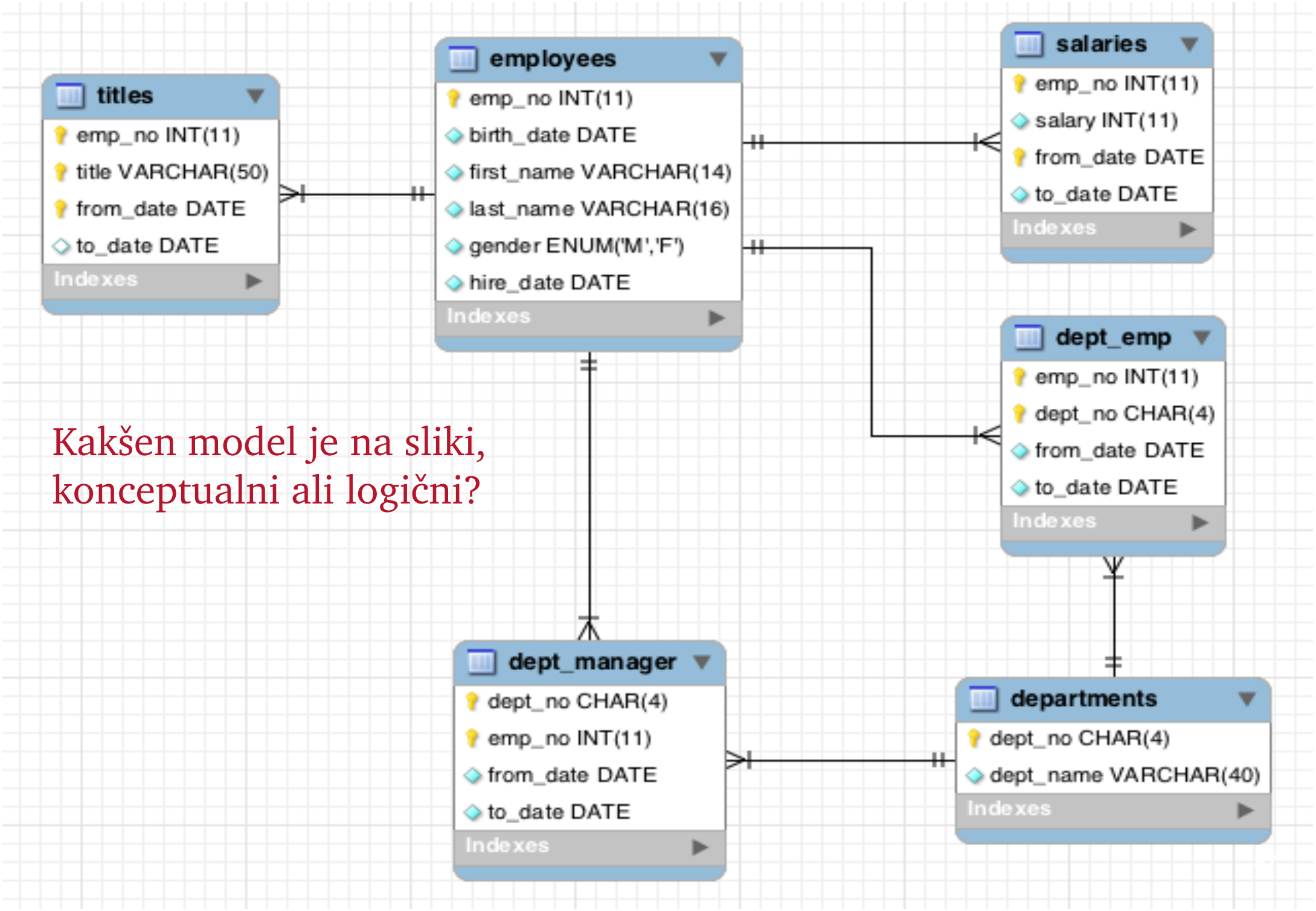
- Po pretvorbi poskrbimo še za:
  - Ustreznost imen
  - Pretvorbo sestavljenih, več-vrednostnih in izpeljanih atributov
  - Indekse po poljih, ki niso primarni ali tuji ključ
  - Določitev pogledov
  - Ustreznost ključev
  - Referencialno integriteto
  - Sprožilce in shranjene procedure
  - …

Student

VpisSt	N8	<pk>
Priimek	C20	
Ime	C20	
Naslov	C100	
Tel	N15	
Email	C25	null
Status	N1	
…		

Student

VpisSt	N8	<pk>
Priimek	C20	
Ime	C20	
Ulica	C25	
Posta	N4	
Drzava	C20	
GSM	N15	
Tel	N15	
Email	C25	null
Status	N1	
…		


titles
emp_no INT(11)
title VARCHAR(50)
from_date DATE
to_date DATE
Indexes
employees
emp_no INT(11)
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
gender ENUM('M','F')
hire_date DATE
Indexes
salaries
emp_no INT(11)
salary INT(11)
from_date DATE
to_date DATE
Indexes
dept_emp
emp_no INT(11)
dept_no CHAR(4)
from_date DATE
to_date DATE
Indexes
dept_manager
dept_no CHAR(4)
emp_no INT(11)
from_date DATE
to_date DATE
Indexes
departments
dept_no CHAR(4)
dept_name VARCHAR(40)
Indexes
Kakšen model je na sliki,
konceptualni ali logični?

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, ki bi se ujemali v vrednostih atributov X in se ne bi ujemali v vrednostih atributov Y.

Funkcionalne odvisnosti

- Množico funkcionalnih odvisnosti, ki veljajo med atributi funkcionalne sheme R in v vseh njenih relacijah, označimo s F

$$X \rightarrow Y \in F \Leftrightarrow \forall r(Sh(r)=R \Rightarrow \forall t, \forall u(t \in r \text{ in } u \in r \text{ in } t.X=u.X \Rightarrow t.Y=u.Y)$$

- kjer
  - t.X, u.X, t.Y in u.Y označujejo vrednosti atributov X oziroma Y v n-tericah t oziroma u.

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 ≡ { VpŠt → (Priimek, Ime), (VpŠt, ŠifraPredmeta, DatumIzpita) → (OcenaPisno, OcenaUstno) }`

Ključi relacije...

- Ker je relacija množica n-teric, so v njej vse n-terice ločene med seboj.
- Za sklicevanje na posamezno n-terico ni potrebno poznati vseh vrednosti atributov n-terice, če v shemi nastopajo funkcionalne odvisnosti.
- Množici atributov, ki določajo vsako n-terico, pravimo ključ relacije oziroma ključ relacijske sheme.

Ključi relacije...

- Predpostavimo, da obstaja relacijska shema z atributi A1 A2 ... An katere podmnožica je množica atributov X.

- Atributi X so ključ relacijske sheme oziroma pripadajočih relacij, če sta izpolnjena naslednja dva pogoja:

  - X → A1 A2 ... An
  - ne obstaja X', ki bi bila prava podmnožica od X in ki bi tudi funkcionalno določala A1, A2 ... An

Normalizacija...

- Normalizacija je postopek, s katerim zagotovimo, da je relacijska shema sestavljena iz primernih relacij, ki ustrezajo potrebam poslovne domene.

- 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.

Primer

- Relacija `Employees` ima odvečne podatke.

emp_no	first_name	last_name	gender	dept_no	dept_name
10002	Bezalel	Simmel	F	d007	Sales
10003	Parto	Bamford	M	d004	Production
10004	Chirstian	Koblick	M	d004	Production
10005	Kyoichi	Maliniak	M	d003	Human Resources
10006	Anneke	Preusig	F	d005	Development
10007	Tzvetan	Zielinski	F	d008	Research
10008	Saniya	Kalloufi	M	d005	Development
10009	Sumant	Peac	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d004	Production
10011	Mary	Sluis	F	d009	Customer Service

Ažurne anomalije

- Relacije, ki vsebujejo odvečne podatke lahko povzročajo anomalije pri spreminjanju podatkov – ažurne anomalije.

- Vrste ažurnih anomalij:
  - Anomalije pri dodajanju n-teric v relacijo
  - Anomalije pri brisanju n-teric iz relacije
  - Anomalije pri spreminjanju n-teric

Anomalije pri dodajanju

- Primeri anomalij:
  - Če želimo dodati podatke o novih zaposlenih, moramo vpisati tudi naziv oddelka.
  - Če želimo dodati podatke o novem oddelku, ki še nima nobenega člana, moramo v vsa polja, ki se nanašajo na zaposlene, vpisati Null.

emp_no	first_name	last_name	gender	dept_no	dept_name
10002	Bezalel	Simmel	F	d007	Sales
10003	Parto	Bamford	M	d004	Production
10004	Chirstian	Koblick	M	d004	Production
10005	Kyoichi	Maliniak	M	d003	Human Resources
10006	Anneke	Preusig	F	d005	Development
10007	Tzvetan	Zielinski	F	d008	Research
10008	Saniya	Kalloufi	M	d005	Development
10009	Sumant	Peac	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d004	Production
10011	Mary	Sluis	F	d009	Customer Service

Anomalije pri brisanju

- Primeri anomalij:
  - Če iz relacije zbrišemo n-terico, ki predstavlja zadnjega zaposlenega v nekem oddelku, izgubimo tudi podatke o tem oddelku.

emp_no	first_name	last_name	gender	dept_no	dept_name
10002	Bezalel	Simmel	F	d007	Sales
10003	Parto	Bamford	M	d004	Production
10004	Chirstian	Koblick	M	d004	Production
10005	Kyoichi	Maliniak	M	d003	Human Resources
10006	Anneke	Preusig	F	d005	Development
10007	Tzvetan	Zielinski	F	d008	Research
10008	Saniya	Kalloufi	M	d005	Development
10009	Sumant	Peac	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d004	Production
10011	Mary	Sluis	F	d009	Customer Service

Anomalije pri spreminjanju

- Primeri anomalij:
  - Če želimo spremeniti vrednost nekega atributa določenega oddelka (npr. naslov), moramo popraviti vse n-terice, v katerih takšna vrednost atributa nastopa.

emp_no	first_name	last_name	gender	dept_no	dept_name
10002	Bezalel	Simmel	F	d007	Sales
10003	Parto	Bamford	M	d004	Production
10004	Chirstian	Koblick	M	d004	Production
10005	Kyoichi	Maliniak	M	d003	Human Resources
10006	Anneke	Preusig	F	d005	Development
10007	Tzvetan	Zielinski	F	d008	Research
10008	Saniya	Kalloufi	M	d005	Development
10009	Sumant	Peac	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d006	Quality Management
10010	Duangkaew	Piveteau	F	d004	Production
10011	Mary	Sluis	F	d009	Customer Service

Postopek normalizacije

- Postopku preoblikovanja relacij v obliko, pri kateri do ažurnih anomalij ne more priti, pravimo normalizacija.

- Obstaja več stopenj normalnih oblik. Obravnavali bomo:
  - 1NO – Prva normalna oblika
  - 2NO – Druga normalna oblika
  - 3NO – Tretja normalna oblika in
  - 4PNO – Četrta poslovna normalna oblika

- Druge NO: Boyce Coddova NO, 4NO, 5NO

1NO – prva normalna oblika

- Relacija je v prvi normalni obliki, če:
  - Nima ponavljajočih atributov → ne obstajajo atributi ali skupine atributov, ki bi imele več vrednosti pri isti vrednosti ostalih atributov (na presečišču ene vrstice in enega stolpca je več vrednosti)
  - Ima definiran primarni ključ in določene funkcionalne odvisnosti

- Koraki:
  - Odstranimo ponavljajoče atribute
  - Določimo funkcionalne odvisnosti
  - Določimo primarni ključ

Primer – relacija v nenormalizirani obliki

Indeks (VŠ, priimek, ime, pošta, kraj, (šifra predmeta, naziv, ocena), šolnina, zaključek)

Skupina ponavljajočih se atributov.

VŠ	priimek	ime	pošta	kraj	šifra predmeta	naziv	ocena	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj	20020 20021 20033	IS TPO IPI	10 8 8	10.1.2016	
64016209	Bizjak	Tadeja	2250	Ptuj	20060 20033	E1 IPI	9 6		1.1.2022

Primer – pretvorba v 1NO...

Indeks (VŠ, priimek, ime, pošta, kraj, (šifra predmeta, naziv, ocena), šolnina, zaključek)

Odpravimo ponavljajoče atribute

Indeks (VŠ, priimek, ime, pošta, kraj, šifra predmeta, naziv, ocena, šolnina, zaključek)

Identificiramo funkcionalne odvisnosti

F ≡ { VŠ → (priimek, ime, pošta, kraj, šolnina, zaključek), šifra predmeta → naziv, pošta → kraj, (VŠ, šifra predmeta) → ocena }

Določimo primarni ključ

Indeks(<u>VŠ</u>, priimek, ime, pošta, kraj, <u>šifra predmeta</u>, naziv, ocena, šolnina, zaključek)

Primer – pretvorba v 1NO

VŠ	priimek	ime	pošta	kraj	šifra predmeta	naziv	ocena	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj	20020 20021 20033	IS TPO IPI	10 8 8	10.1.2016	
64016209	Bizjak	Tadeja	2250	Ptuj	20060 20033	E1 IPI	9 6		1.1.2022

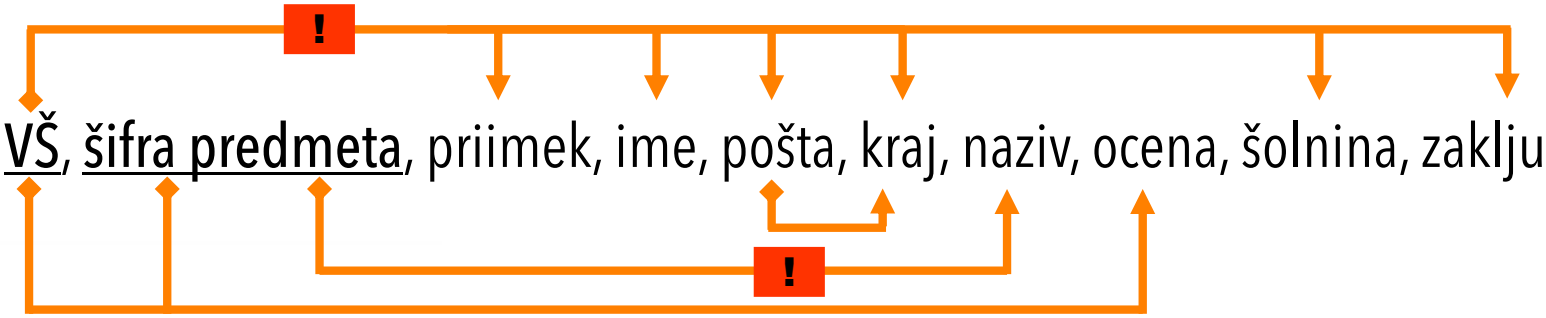
VŠ	priimek	ime	pošta	kraj	šifra predmeta	naziv	ocena	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj	20020	IS	10	10.1.2016	
64010632	Bratina	Simon	4100	Kranj	20021	TPO	8	10.1.2016	
64010632	Bratina	Simon	4100	Kranj	20033	IPI	8	10.1.2016	
64016209	Bizjak	Tadeja	2250	Ptuj	20060	E1	9		1.1.2022
64016209	Bizjak	Tadeja	2250	Ptuj	20033	IPI	6		1.1.2022

2NO – druga normalna oblika

- Relacija je v drugi normalni obliki:
  - Če je v prvi normalni obliki in
  - Ne vsebuje parcialnih odvisnosti → noben atribut, ki ni del ključa, ni funkcionalno odvisen le od dela primarnega ključa, temveč od celotnega ključa

- Druga normalna oblika je odvisna predvsem od ključa relacije. Relacija je avtomatsko v drugi normalni obliki, če:
  - Je njen primarni ključ sestavljen le iz enega atributa,
  - Je njen primarni ključ sestavljen iz vseh atributov relacije ali
  - Je njen primarni ključ sestavljen iz vseh razen enega atributa relacije

Primer – pretvorba v 2NO...



Relacijo razbijemo

Študent (VŠ, priimek, ime, pošta, kraj, šolnina, zaključek)

Predmet (šifra predmeta, naziv)

Indeks (#VŠ, #šifra predmeta, ocena)

Primer – pretvorba v 2NO

VŠ <pk>	priimek	ime	pošta	kraj	šifra predmeta <pk>	naziv	ocena	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj	20020	IS	10	10.1.2016	
64010632	Bratina	Simon	4100	Kranj	20021	TPO	8	10.1.2016	
64010632	Bratina	Simon	4100	Kranj	20033	IPI	8	10.1.2016	
64016209	Bizjak	Tadeja	2250	Ptuj	20060	E1	9		1.1.2022
64016209	Bizjak	Tadeja	2250	Ptuj	20033	IPI	6		1.1.2022

Predmet

šifra predmeta <pk>	naziv
20020	IS
20021	TPO
20033	IPI
20060	E1
20033	IPI

Indeks

VŠ <pk>	šifra predmeta <pk, fk>	ocena
64010632	20020	10
64010632	20021	8
64010632	20033	8
		9
		6

Študent

VŠ <pk>	priimek	ime	pošta	kraj	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj		
64016209	Bizjak	Tadeja	2250	Ptuj		

3NO – tretja normalna oblika

- Relacija je v tretji normalni obliki:
  - Če je v drugi normalni obliki in
  - Če ne vsebuje tranzitivnih funkcionalnih odvisnosti → med atributi, ki niso del primarnega ključa, ni odvisnosti.

- Relacija je avtomatsko v tretji normalni obliki, če:
  - Je njen ključ sestavljen iz vseh atributov relacije
  - Je njen ključ sestavljen iz vseh razen enega atributa relacije.

Primer – pretvorba v 3NO...

Študent (VŠ, priimek, ime, pošta, kraj, šolnina, zaključek)

Predmet (šifra predmeta, naziv)

Indeks (#VŠ, #šifra predmeta, ocena)

Relacijo razbijemo

Študent (VŠ, priimek, ime, #pošta, šolnina, zaključek)

Pošta (pošta, kraj)

Predmet (šifra predmeta, naziv)

Indeks (#VŠ, #šifra predmeta, ocena)

Primer – pretvorba v 3NO

Študent

VŠ <pk>	priimek	ime	pošta	kraj	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj	10.1.2016	
64016209	Bizjak	Tadeja	2250	Ptuj		1.1.2022

Študent

VŠ <pk>	priimek	ime	Pošta <fk>	šolnina	zaključek
64010632	Bratina	Simon	4100	10.1.2016	
64016209	Bizjak	Tadeja	2250		1.1.2022

Pošta

Pošta <pk>	kraj
4100	Kranj
2250	Ptuj

Razbitje po prvih treh korakih

Indeks (VŠ, priimek, ime, pošta, kraj, (šifra predmeta, naziv, ocena), šolnina, zaključek)

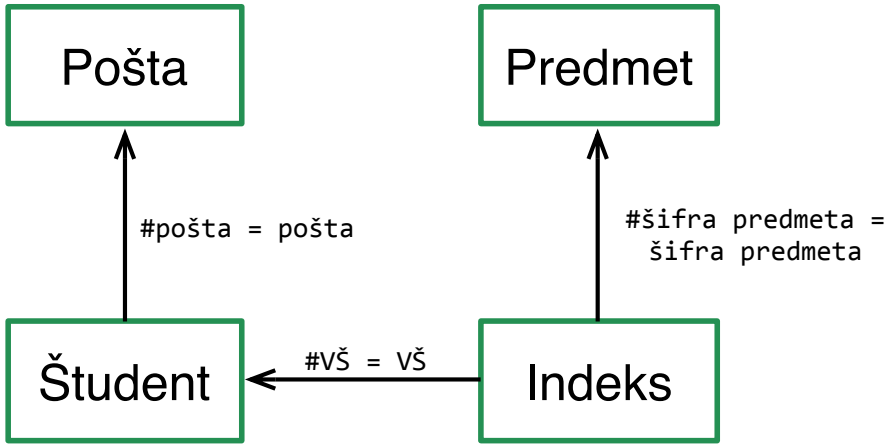
Indeks

Študent (**VŠ**, priimek, ime, #pošta, šolnina, zaključek)

Pošta (**pošta**, kraj)

Predmet (**šifra predmeta**, naziv

Indeks (#**VŠ**, #**šifra predmeta**, ocena)



4PNO – četrta poslovna normalna oblika

- Relacija je v četrti poslovni normalni obliki, če:
  - je v tretji normalni obliki in
  - v relaciji ne obstajajo atributi, ki bi bili odvisni od vrednosti primarnega ključa.

Primer – pretvorba v 4PNO...

Študent (VŠ, priimek, ime, #pošta, šolnina, zaključek)

Študent (VŠ, priimek, ime, #pošta)

Redni študent (#VŠ, zaključek)

Izredni študent (#VŠ, šolnina)

Primer – pretvorba v 4PNO

Študent

VŠ <pk>	Priimek	Ime	Šolnina	Zaključek
64010632	Bratina	Simon	10.1.2016	
64016209	Bizjak	Tadeja		1.1.2022
64010670	Berce	Marjan	12.4.2004	
64620010	Mele	Silvana		1.4.2005
65120987	Leban	Tibor		15.7.2005

Študent

VŠ <pk>	Priimek	Ime
64010632	Bratina	Simon
64016209	Bizjak	Tadeja
64010670	Berce	Marjan
64620010	Mele	Silvana
65120987	Leban	Tibor

Redni študent

VŠ <pk,fk>	Zaključek
64016209	1.1.2022
64620010	1.4.2005
65120987	15.7.2005

Izredni študent

VŠ <pk,fk>	Šolnina
64010670	12.4.2004
64010632	10.1.2016

Normalizirana shema

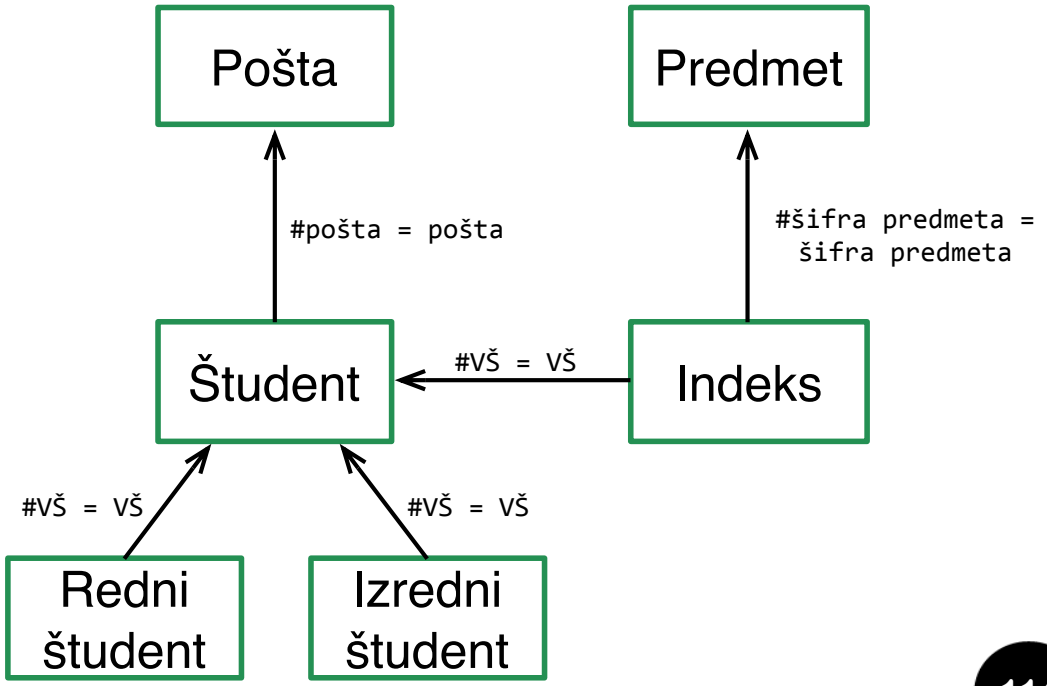
- Začetna shema

Indeks

Indeks (VŠ, priimek, ime, pošta, kraj, (šifra predmeta, naziv, ocena), šolnina, zaključek)

- Končna shema

Pošta (<u>pošta</u>, kraj)
Predmet (<u>šifra predmeta</u>, naziv)
Indeks (#<u>VŠ</u>, #<u>šifra predmeta</u>, ocena)
Študent (<u>VŠ</u>, priimek, ime, #pošta)
Redni študent (#<u>VŠ</u>, zaključek)
Izredni študent (#<u>VŠ</u>, šolnina)



Uporaba nenormaliziranih relacij...

- Včasih zavestno uporabljamo relacije, ki ne ustrezajo najvišjim normalnim oblikam.
- Prve in druge normalne oblike nikoli ne kršimo.
- Višjim normalnim oblikam se včasih odrečemo na račun doseganja boljše učinkovitosti.

Uporaba nenormaliziranih relacij

Primer:

- Rezultat (športnik, tekmovanje, čas prvega teka, čas drugega teka, čas skupaj)
- Relacija ni v tretji normalni formi.
- Čas skupaj je izpeljan atribut → ni odvisen od ključa, temveč je seštevek časov obeh tekov.
- Skupen čas računamo ob vpisu v bazo, zato izboljšamo učinkovitost pri nadaljnji obdelavi podatkov.

Metoda logičnega načrtovanja...

- Možni koraki logičnega načrtovanja:
  - K2.1: Za entitetne tipe kreiraj relacije
  - K2.2: Preveri relacije z normalizacijo
  - K2.3: Preveri relacije s pregledom uporabniških transakcij
  - K2.4: Preveri omejitve integritete
  - K2.5: Preveri model z uporabnikom
  - K2.6: Preveri zmožnosti modela za razširitve

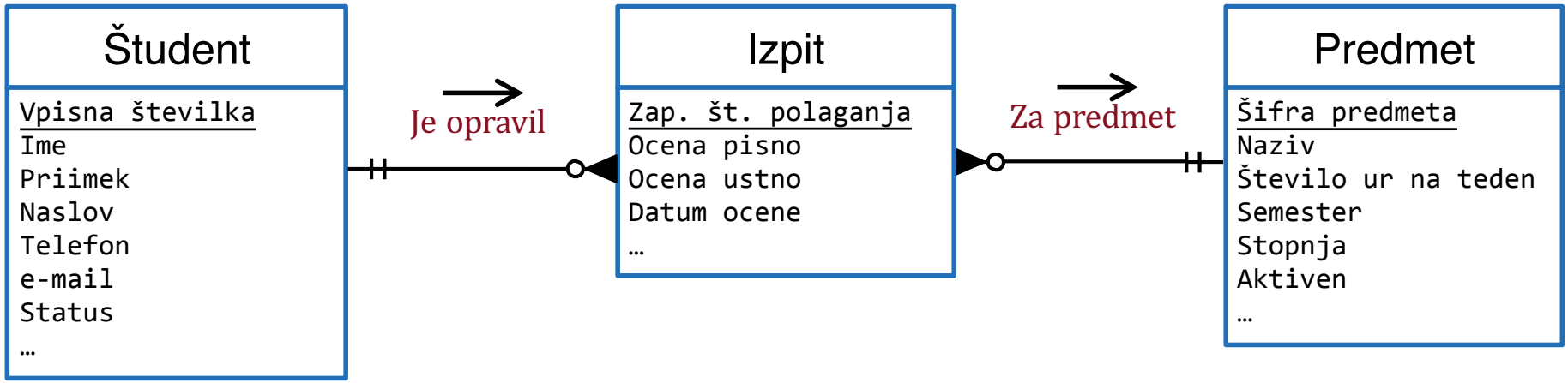
K2.1 – Za entitetne tipe kreiraj relacije...

- Namen
  - Izdelati relacije za logični model, ki bo predstavljal entitete, povezave in atribute, ki smo jih identificirali v okviru konceptualnega modeliranja.
- Ta korak je navadno avtomatiziran → pretvorba iz konceptualnega v logični model je podprta s strani številnih CASE orodij.

K2.1 – Za entitetne tipe kreiraj relacije...

- Ročna pretvorba:
  - Močni entitetni tipi
    - Za vsak močan entitetni tip kreiraj relacijo, ki vključuje vse enostavne atribute tega entitetnega tipa. Namesto sestavljenih atributov vključi njihove atribute, ki jih sestavljajo.
  - Šibki entitetni tipi
    - Za vsak šibki entitetni tip kreiraj relacijo, ki vključuje vse enostavne atribute tega entitetnega tipa. Primarni ključ šibkega entitetnega tipa je delno ali v celoti sestavljen iz atributov, ki so ključ v entitetnih tipih, s katerimi je opazovani entitetni tip povezan. Da bi lahko določili ključ, moramo najprej pretvoriti vse povezave.

Primer pretvorbe šibkega entitetnega tipa



Študent(VpisSt, Ime, Priimek, Ulica, Hst, Mesto, PTT, Drzava, Tel, MobTel, email, Status)

Predmet(IDPred, Naziv, StUrTeden, Semester, Stopnja, Aktiven)

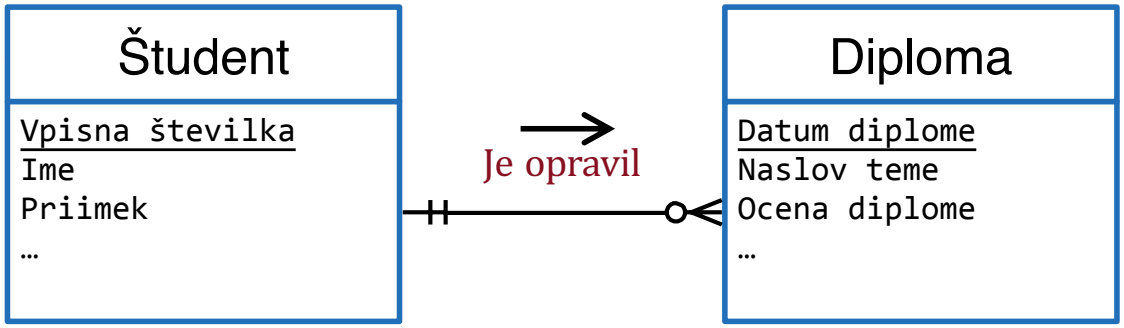
Izpit(ZapStPol, #VpisSt, #IDPred, OcenaPisno, OcenaUstno, DtmOc)

Pretvorbe glede na številnost povezave...

- Povezave 1:n
  - Za vsako povezavo 1:n prenesi ključ entitetnega tipa, ki nastopa v povezavi na strani 1 (oče) v entitetni tip, ki nastopa v povezavi na strani n (otrok). Dobimo tuji ključ.
- Povezave 1:1
  - Obveznost na obeh straneh 1:1 povezave: entitetna tipa združi v eno relacijo.
  - Obveznost na eni strani 1:1 povezave: tuji ključ dobi entitetni tip, ki ima obvezno povezavo z drugim entitetnim tipom.
  - Neobvezna povezava na obeh straneh 1:1 povezave: po presoji.

Primer: povezave 1:n

- Ključ se prenaša v smeri 1:n

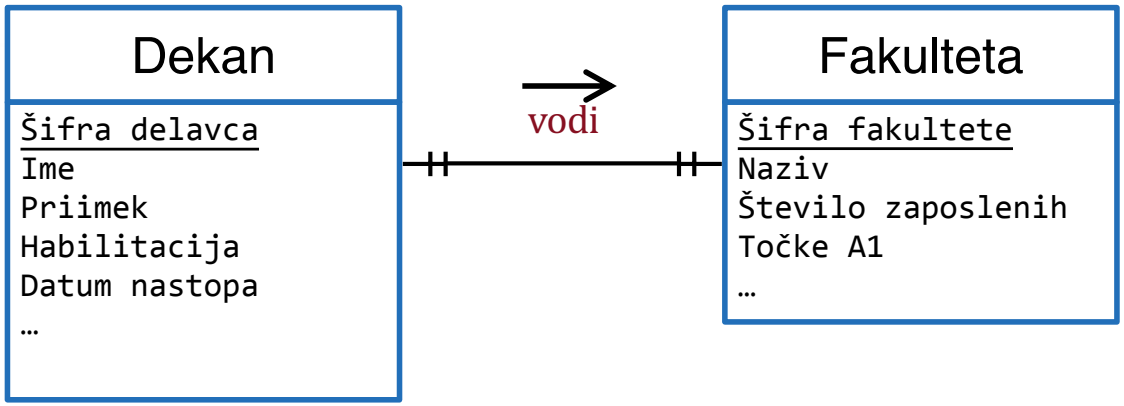


Študent(VpisSt, Ime, Priimek,...)

Diploma(Datum, #VpisSt, NaslovTeme, Ocena...)

Primer: povezave 1:1

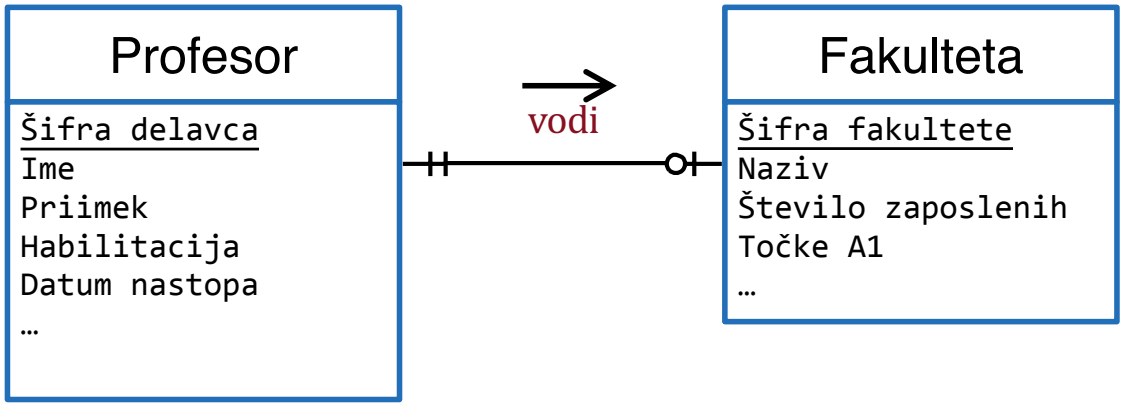
- Obveznost na obeh straneh: združimo v eno entiteto in izberemo ključ!



Dekan(IDdelavca, Ime, Priimek, Habilitacija, DtmNastopa,…, NazivFak, StZapos, A1…)

Primer: povezave 1:1

- Obveznost na eni strani: entitetni tip, ki igra vlogo očeta, dobi tuji ključ.



Profesor(IDdelavca, Ime, Priimek, Habilitacija, DtmNastopa,...)

Fakulteta(IDfak, #dekan, NazivFak, StZapos, A1...)

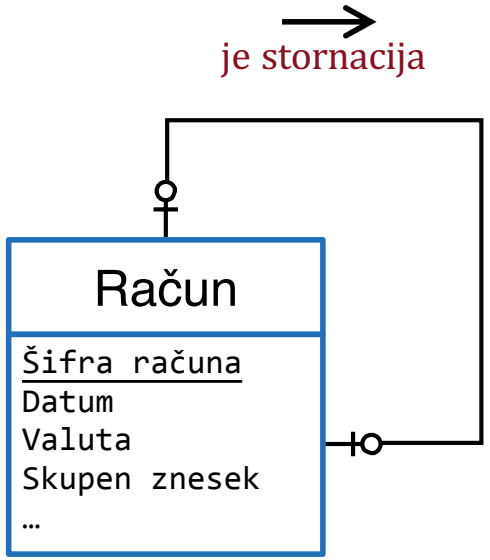
Poimenovani tuji ključ.

Rekurzivne povezave

- Rekurzivna povezava je povezava entitetnega tipa samega s seboj.
- Pravila podobna kot pri navadnih povezavah.
- Kopije primarnih ključev je v rekurzivnih povezavah potrebno ustrezno poimenovati, da lahko ločimo med njimi!

Primer: rekurzivne povezave 1:1

- Račun predstavlja stornacijo drugega računa.



Je številost modela pravilna?

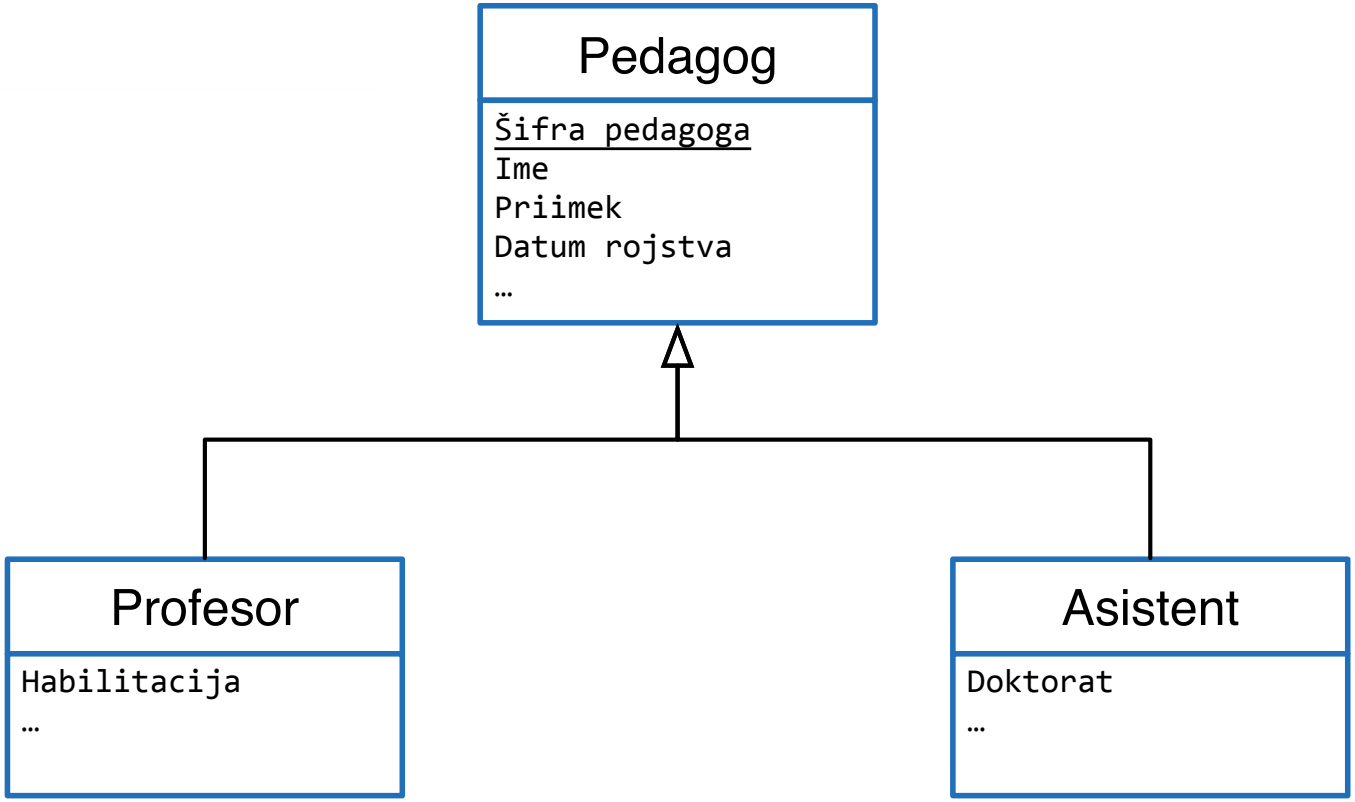
Racun(IDrac, #IDracStorno, Datum, Valuta, Znesek,…)

Povezava med nad- in pod-tipi

- Kadar uporabljamo dedovanje (EER diagram), imamo opravka z entitetnimi nadtipi in podtipi.
- Pravila pretvorbe v relacije:
  - Če imajo podtipi malo svojih atributov, potem vse skupaj prenesemo v nadtip in naredimo samo eno relacijo.
  - Če imajo podtipi veliko svojih atributov, potem naredimo samo podtipe.
  - Redko izdelamo vse tri relacije (za podtipe in nadtip).

Primer: nadtipi, podtipi

- Katere entitetne tipe pretvorimo v relacije?

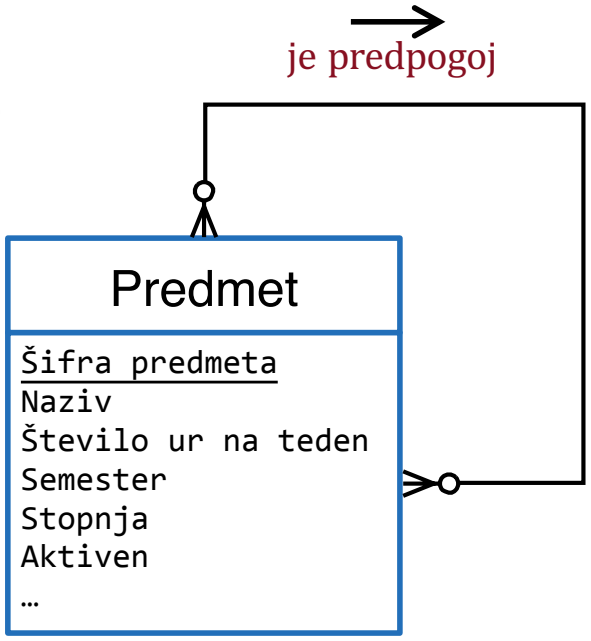


Povezave n:n

- Kadar je številnost povezave n:n (mnogo proti mnogo), upoštevamo naslednja pravila:
  - Kreiraj relacijo, ki predstavlja povezavo ter vse njene atribute.
  - Primarne ključe entitetnih tipov, ki sta povezana s tako povezavo, vključi v novo relacijo kot tuji ključ.
  - Tuji ključi bodo obenem tudi primarni ključi – samostojno ali v kombinaciji z drugimi atributi relacije.

Primer: rekurzivne povezave n:n

- Opravljanje predmeta zahteva, da je študent predhodno opravil druge predmete.



Predmet(IDpred, Naziv, StUrTeden, Semester, Stopnja, Aktiven,…)

PredPogoj(IDpred, #IDpredPogoj)

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.

K2.2 – Preveri relacije z normalizacijo

- Potrebno upoštevati:
  - Normaliziran logični podatkovni model ni dokončen. Specifične potrebe v zvezi z učinkovitostjo lahko zahtevajo drugačen fizični model (denormalizacija).
  - Normaliziran načrt je robusten (razširljiv) in odporen na podatkovne anomalije.
  - Moderni računalniki so veliko zmogljivejši → včasih je upravičeno uporabiti rešitve, ki omogočajo enostavnejšo obdelavo na račun več procesiranja.

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 omejitve integritete...

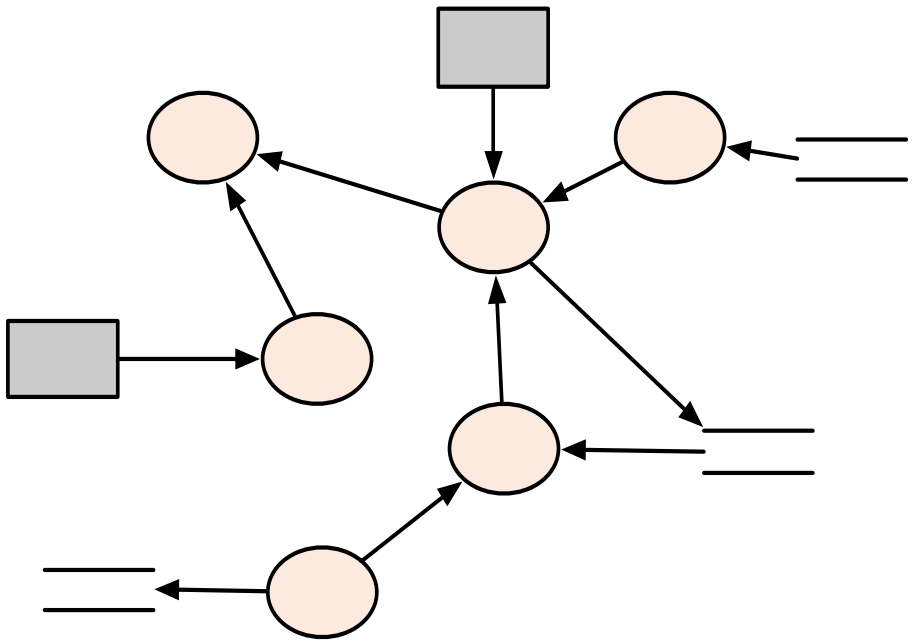
- V tem koraku preverimo pravila za zagotavljanje celovitosti podatkov:
  - Obveznost atributov
  - Omejitve domen atributov
  - Številnost
  - Omejitve entitet (celovitost entitet)
  - Omejitve povezav (celovitost povezav)
  - Splošne omejitve

K2.5 – Preveri model z uporabnikom...

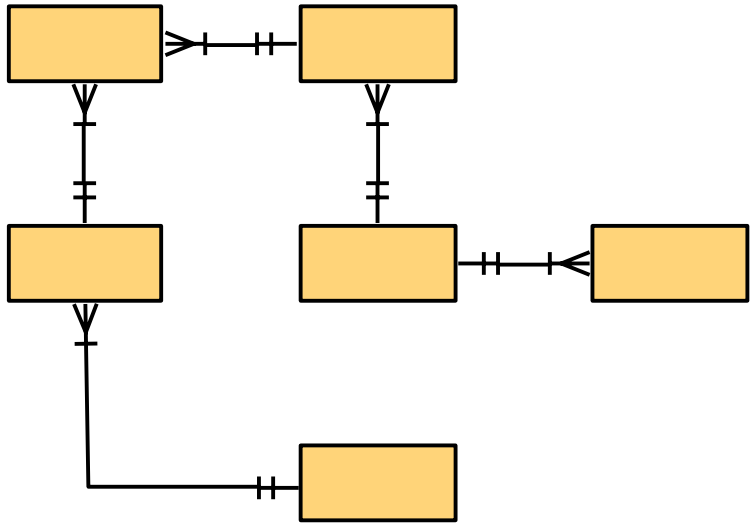
- Namen tega koraka je preveriti model z uporabnikom ter ugotoviti, če ustreza vsem uporabniškim zahtevam.
- Model lahko zajema več uporabniških pogledov. Pri pregledu lahko nastopa več uporabnikov.
- Odličen način za pregled celovitosti podatkovnega modela je specifikacija podatkovnih tokov s pomočjo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

Povezava podatkovni model in DFD

Diagram podatkovnih tokov



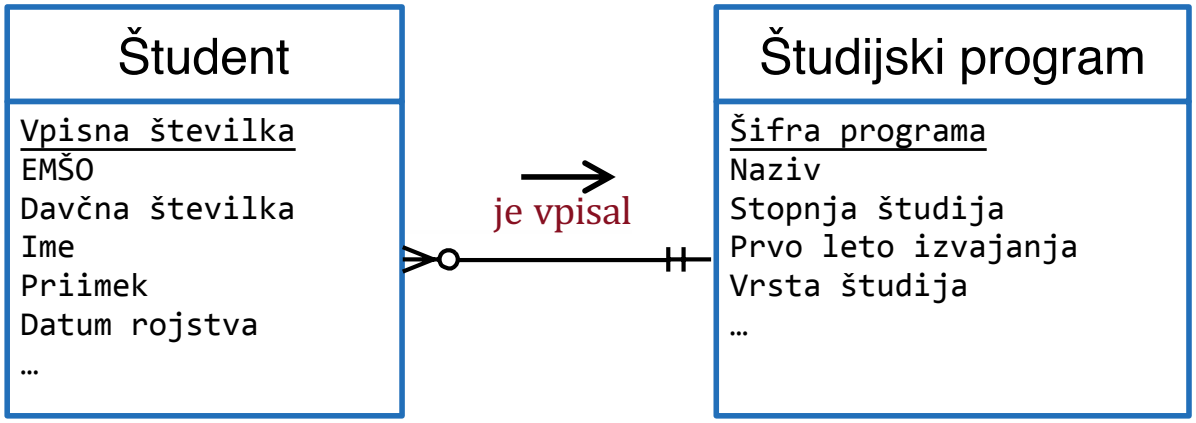
Podatkovni model

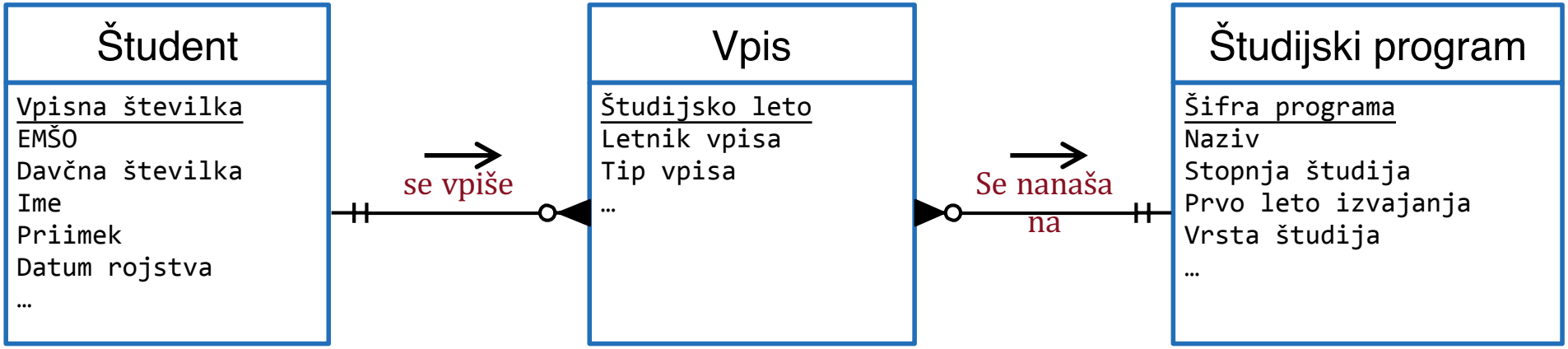


K2.6 – Preveri možnosti za razširitve...

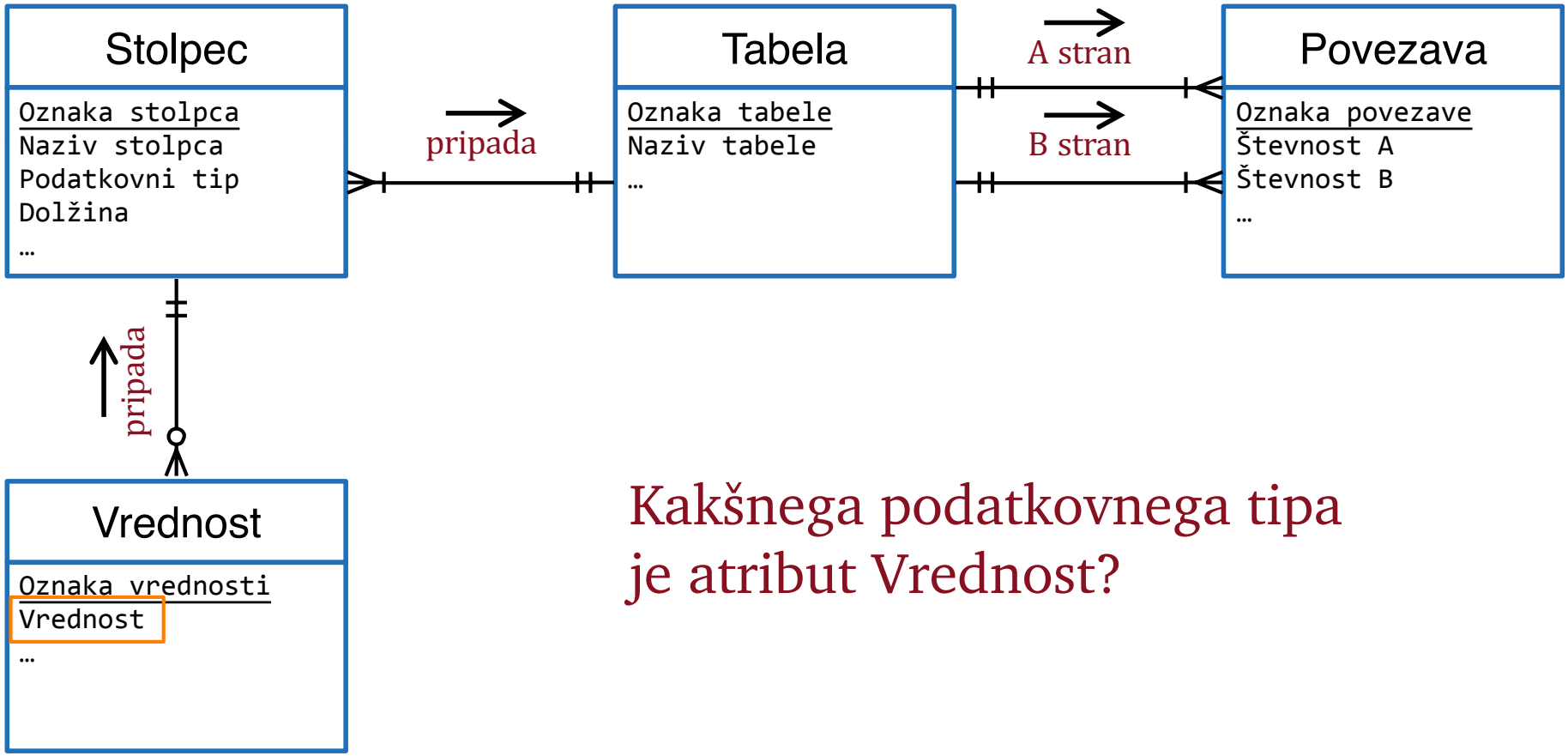
- V primeru, da so predvidene bodoče razširitve sistema, moramo preveriti, če logični model take razširitve podpira.
- Podatkovni model mora biti prilagodljiv; omogočati mora razširitve skladno z novimi zahtevami ter z minimalnim vplivom na obstoječe uporabnike.
- Popolnoma odprt sistem za razširitve je težko doseči.
- Pravilo agilnega načrtovanja:
  - *"Fool me once, shame on you, fool me twice, shame on me!"*

Primer


Študent
Vpisna številka
EMŠO
Davčna številka
Ime
Priimek
Datum rojstva
…
je vpisal
Študijski program
Šifra programa
Naziv
Stopnja študija
Prvo leto izvajanja
Vrsta študija
…


Študent
Vpisna številka
EMŠO
Davčna številka
Ime
Priimek
Datum rojstva
…
se vpiše
Vpis
Študijsko leto
Letnik vpisa
Tip vpisa
…
Se nanaša na
Študijski program
Šifra programa
Naziv
Stopnja študija
Prvo leto izvajanja
Vrsta študija
…

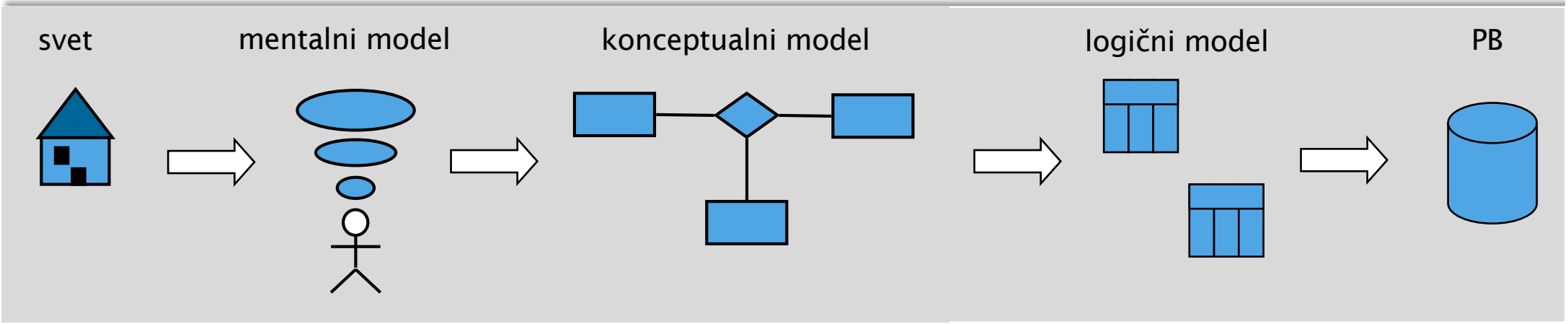
Primer generičnega podatkovnega modela

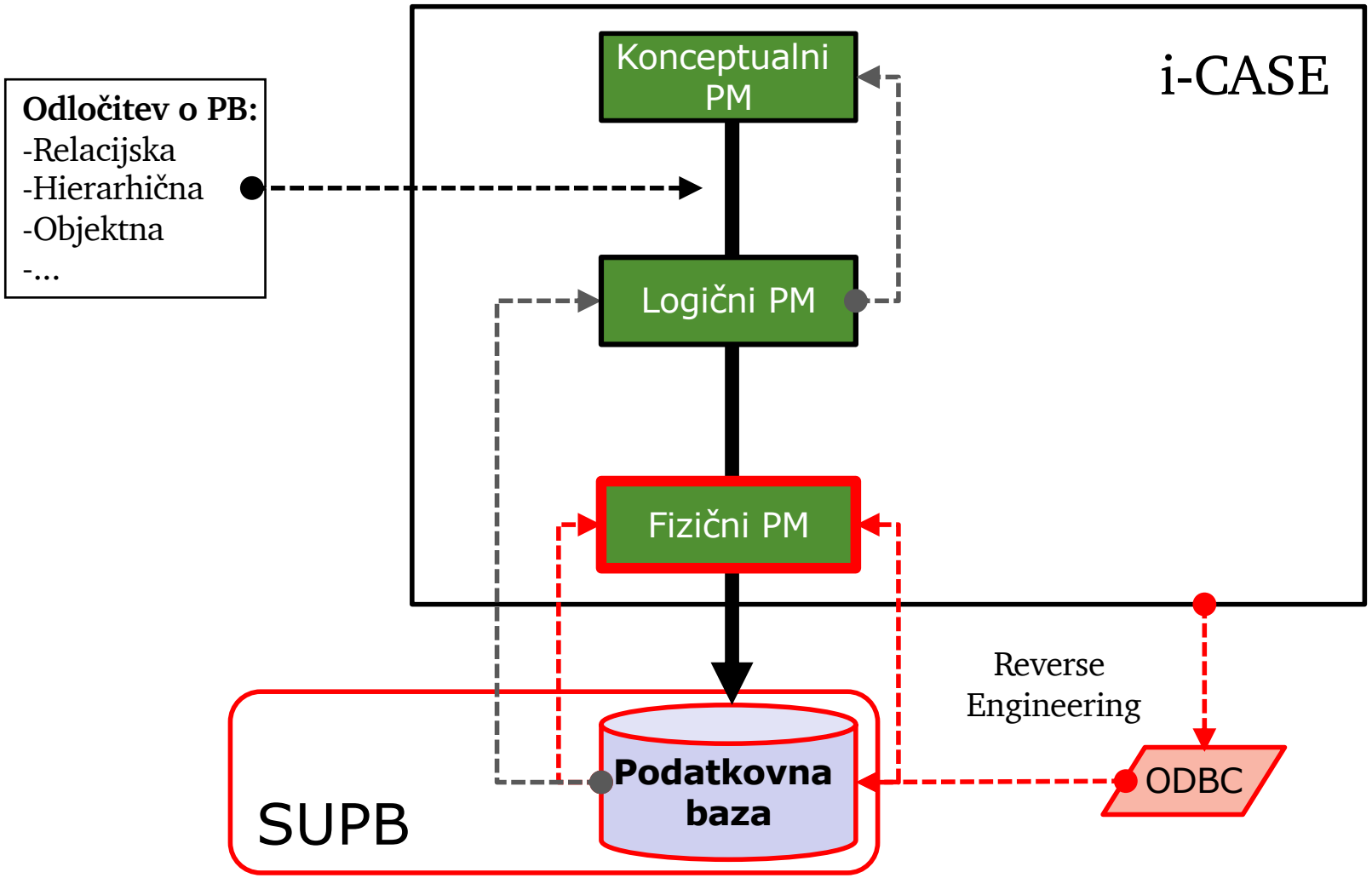


Kakšnega podatkovnega tipa je atribut Vrednost?

Načrtovanje fizične podatkovne baze...

- Načrtovanje fizične PB je korak, ki sledi logičnemu načrtovanju.
- Logični načrt opredeljuje “kaj”, fizični načrt pa “kako”.
- Vhod v načrtovanje fizične PB so:
  - Logični podatkovni model,
  - Relacijska shema
  - dokumentacija




Odločitev o PB:
-Relacijska
-Hierarhična
-Objektna
-...
Konceptualni PM
Logični PM
Fizični PM
i-CASE
Reverse Engineering
ODBC
Podatkovna baza
SUPB

Logični model

Employees

emp_no INT(11) <pk>
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
hire_date DATE
avgSalary DECIMAL(10,2)

Fizični model

skripta

```
CREATE TABLE employees
  emp_no INT(11) NOT NULL,
  birth_date DATE NOT NULL,
  first_name VARCHAR(14) NOT NULL,
  last_name VARCHAR(16) NOT NULL,
  hire_date DATE NOT NULL,
  avgSalary DECIMAL(10,2) DEFAULT NULL,
PRIMARY KEY (emp_no))
ENGINE=InnoDB
DEFAULT CHARSET=latin1
```

fizična tabela

Načrtovanje fizične podatkovne baze

- Fizično načrtovanje PB opredeljuje proces, s katerim izdelamo opis implementacije PB na sekundarnem pomnilnem mediju.
- Opisuje
  - osnovne relacije,
  - datotečno organizacijo,
  - indekse za dosego učinkovitega dostopa do podatkov,
  - povezane omejitve in
  - varnostne mehanizme.

Metoda načrtovanja fizične PB...

- Možni koraki načrtovanja fizične PB:
  - K3 – Pretvori logični model v jezik za ciljni SUPB
    - K3.1 – Izdelaj načrt osnovnih relacij
    - K3.2 – Izdelaj načrt predstavitve izpeljanih atributov
    - K3.3 – Izdelaj načrt splošnih omejitev
  - K4 – Izdelaj načrt datotečne organizacije ter indeksov
    - K4.1 – Analiziraj transakcije
    - K4.2 – Izberi datotečno organizacijo
    - K4.3 – Določi indekse
    - K4.4 – Oceni velikost podatkovne baze

Koraka K1 in K2 se nanašata na izdelavo konceptualnega in logičnega podatkovnega modela

Metoda načrtovanja fizične PB...

- Možni koraki načrtovanja fizične PB (nadaljevanje):
  - K5 – Izdelaj načrt uporabniških pogledov
  - K6 – Izdelaj načrt varnostnih mehanizmov
  - K7 – Preveri smiselnost uvedbe nadzorovane redundance podatkov (denormalizacija)

K3 – Pretvorba v jezik za SUPB

- Namen koraka: iz logičnega modela izdelati podatkovno shemo za ciljni SUPB.
- Poznati moramo funkcionalnosti ciljnega SUPB, npr.:
  - Kako izdelati osnovne relacije?
  - Ali ciljni SUPB podpira primarne, tuje in alternativne ključe?
  - Ali podpira obveznost podatkov (NOT NULL)?
  - Ali podpira domene?
  - Ali podpira pravila omejitve podatkov?
  - Ali podpira sprožilce (triggers) in bazne programe (stored procedures)?

K3.1 – Izdelava osnovnih relacij...

- Namen: določiti, kako bodo osnovne relacije predstavljene v ciljnem SUPB.
- Vir podatkov je podatkovni slovar, jezik za opis pa DDL (database definition language).
- Za vsako relacijo definiramo:
  - Naziv relacije;
  - Listo osnovnih atributov;
  - Primarni ključ ter kjer smiselno alternativne in tuje ključe;
  - Omejitve povezav.

K3.2 – Predstavitev izpeljanih atributov...

- Namen: določiti, kako bodo v SUPB predstavljeni izpeljani atributi.
- Preuči logični podatkovni model in podatkovni slovar; izdelaj seznam izpeljanih atributov.
- Za vsak izpeljani atribut določi:
  - Atribut je shranjen v podatkovni bazi
  - Atribut se vsakokrat posebej izračuna in se ne hrani v podatkovni bazi.

Primeri izpeljanih atributov...

Artikel
IDartikla
CenaNaEnoto
Kolicina
DDV
SkupnaCena
…

Kdaj naj se izrčuna SkupnaCena?

Računska izpeljava

Cena skupaj = (Cena na enoto * količina) * (1+DDV)

Projekt
IDprojekta
Naziv
Vrednost
VelikostProjekta
TipProjekta
…

Logična izpeljava

Velikost projekta : če projekt daljši od 12 mesecev ali vrednost večja od 1MIO EUR, potem velik projekt…

K3.2 – Predstavitev izpeljanih atributov...

- Pri odločitvi, kako predstaviti izpeljane atribute, upoštevaj:
  - "strošek" shranjevanja in vzdrževanja skladnosti izpeljanih atributov z osnovnimi atributi, iz katerih je izpeljan;
  - "strošek" vsakokratnega izračunavanja izpeljanega atributa.
- Izberi stroškovno ugodnejšo rešitev.

Primer

emp_no	birth_date	first_name	last_name	age	gender
10001	1953-09-02	Georgi	Facello	62	M
10002	1964-06-02	Bezalel	Simmel	51	F
10003	1959-12-03	Parto	Bamford	56	M
10004	1954-05-01	Chirstian	Koblick	61	M
10005	1955-01-21	Kyoichi	Maliniak	60	M
10006	1953-04-20	Anneke	Preusig	62	F
10007	1957-05-23	Tzvetan	Zielinski	58	F
10008	1958-02-19	Saniya	Kalloufi	57	M
10009	1952-04-19	Sumant	Peac	63	F
10010	1963-06-01	Duangkaew	Piveteau	52	F
10011	1953-11-07	Mary	Sluis	62	F
10012	1960-10-04	Patricio	Bridgland	55	M
10013	1963-06-07	Eberhardt	Terkki	52	M
10014	1956-02-12	Berni	Genin	59	M
10015	1959-08-19	Guoxiang	Nooteboom	56	M

K3.3 – Načrt splošnih omejitev

- Namen: izdelava načrta splošnih omejitev za ciljni SUPB (povezano s poslovnimi pravili).
- Glede podpore splošnim omejitvam obstajajo velike razlike med SUPB-ji.
- Primer splošne omejitve:

```
CONSTRAINT ManagerConflict
CHECK (NOT EXISTS
           (select dept_no, count(*)
            from dept_manager
            where to_date = "9999-01-01"
            group by dept_no
            having count(*) > 1)
)
```

K4 – Datotečna organizacija in indeksi

- Namen: izbrati optimalno datotečno organizacijo za shranjevanje osnovnih relacij ter potrebne indekse za doseganje ustrezne učinkovitosti.
- Načrtovalec mora dobro poznati, kakšne strukture in organizacije SUPB podpira ter kako deluje.
- Ključnega pomena so uporabniške zahteve v zvezi z želeno/pričakovano učinkovitostjo transakcij.
- Med SUPB-ji velike razlike.

K4.1 – Analiza transakcij...

- Namen: razumeti namen transakcij, ki bodo tekle na SUPB ter analizirati tiste, ki so najpomembnejše.
- Poskušaj določiti kriterije učinkovitosti:
  - Pogoste transakcije, ki imajo velik vpliv na učinkovitost;
  - Transakcije, ki so kritičnega pomena za poslovanje;
  - Pričakovana obdobja (v dnevu/ tednu), ko bo SUPB najbolj obremenjen (peak load).
- Preveri tudi:
  - Atribute, ki jih transakcije spreminjajo
  - "Iskalne" pogoje v transakcijah...

K4.1 – Analiza transakcij...

- Pogosto ni moč analizirati vseh transakcij. Pregledamo zgolj najpomembnejše.
- Za identifikacijo najpomembnejših transakcij lahko uporabimo:
  - Matriko transakcija/relacija, ki kaže, katere relacije se v transakcijah uporabljajo.
  - Diagram uporabe transakcij, ki kaže, katere transakcije bodo potencialno zelo frekventno izvajane.

Primer matrike transakcija/relacija

Table 17.1 Cross-referencing transactions and relations.

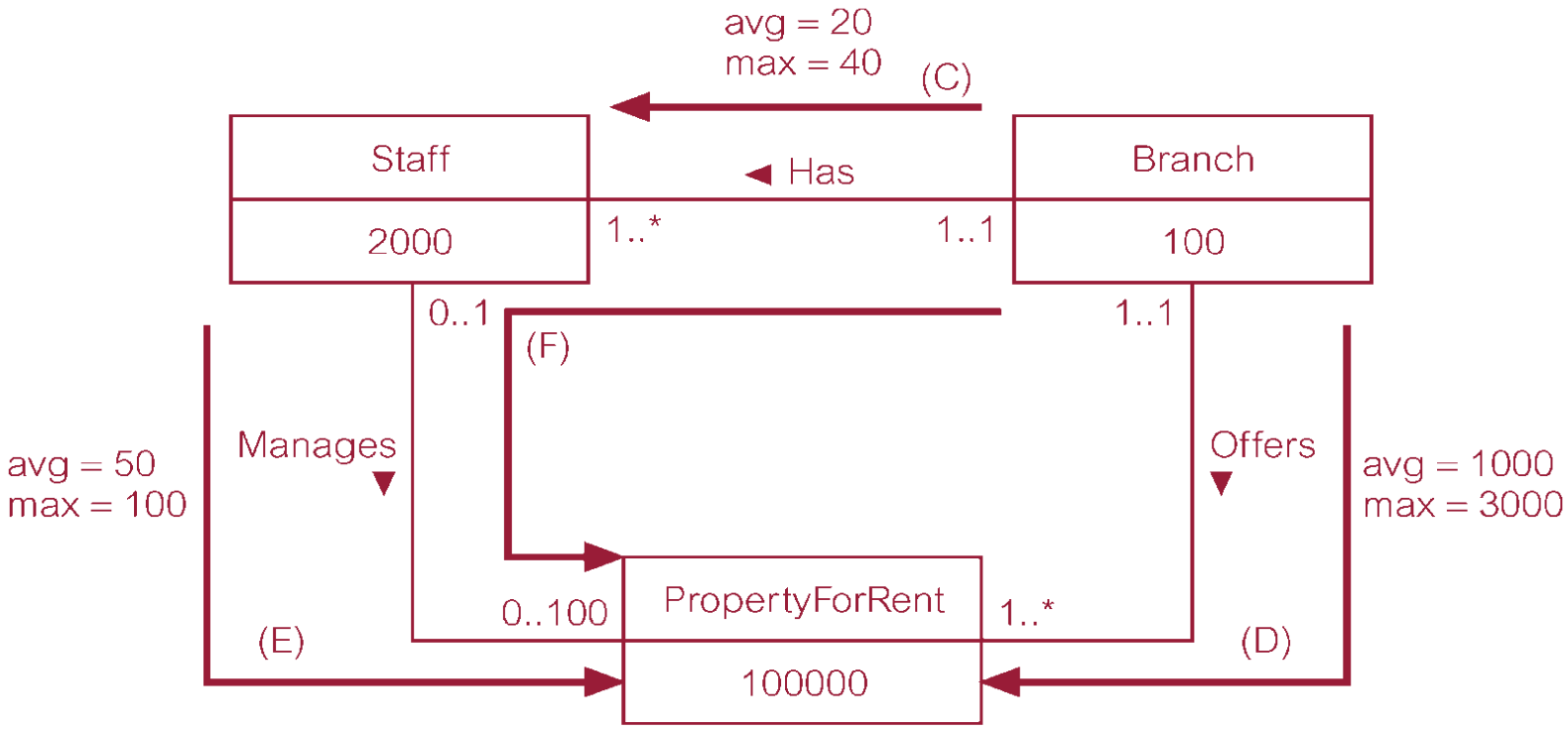
Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X					X	X	X																
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

I = Insert; R = Read; U = Update; D = Delete

Dodatno lahko zapišemo število operacij na časovno enoto

Transakcija: Identify the total number of properties assigned to each member of staff at a given branch.

Primer diagrama uporabe transakcij



V specifikaciji zahtev je ocenjeno:

- 100.000 nepremičnin;
- 2.000 zaposlenih v 100 agencijah;
- Vsaka agencija ima od 1.000 do 3.000 nepremičnin

Primer obrazca za podrobno analizo transakcij

Transaction Analysis Form **1-Sept-2004**

Transaction (D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent

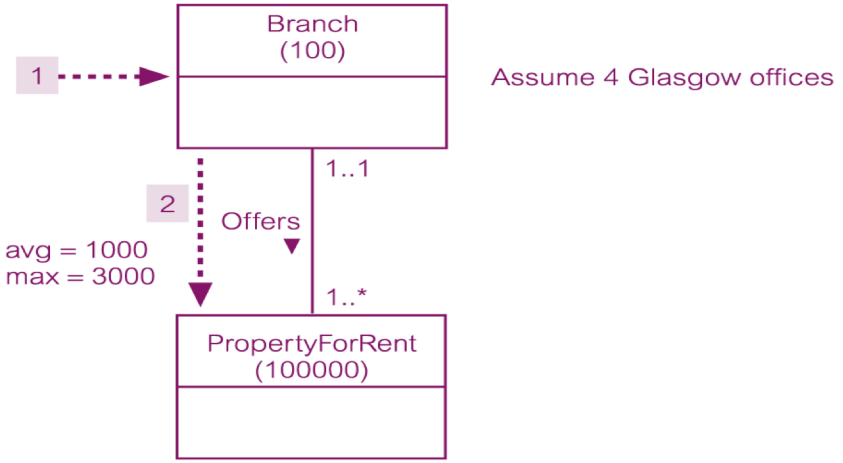
Transaction volume

Average:	50 per hour
Peak:	100 per hour (between 17.00 and 19.00 Monday–Saturday)

```
SELECT propertyNo, p.street, p.postcode, type, rent
FROM Branch b INNER JOIN PropertyForRent p ON
        b.branchNo = p.branchNo
WHERE p.city = 'Glasgow'
ORDER BY rent;
```

Predicate:	**p.city** = 'Glasgow'
Join attributes:	**b.branchNo** = **p.branchNo**
Ordering attribute:	**rent**
Grouping attribute:	none
Built-in functions:	none
Attributes updated:	none

Transaction usage map



Access	Entity	Type of	No. of References

K4.2 – Izbira datotečne organizacije

- Namen: izbrati učinkovito datotečno organizacijo za vse osnovne relacije.
- Datotečne organizacije (podatkovnih in indeksnih datotek):
  - Neurejene datoteke,
  - Urejene datoteke,
  - Razpršene datoteke
  - ISAM in B+ drevesa za indeksiranje
  - Gruče
  - ...

- SUPB-ji podpirajo različne datotečne organizacije.

MySQL Storage Engine...

- MySQL ponuja različne vrste shramb za podatke:
  - MyISAM
  - InnoDB
  - MERGE
  - MEMORY
  - BDB
  - EXAMPLE
  - ARCHIVE
  - CSV
  - BLACKHOLE
  - ISAM
  - ...

```
CREATE TABLE employees
  emp_no INT(11) NOT NULL,
  birth_date DATE NOT NULL,
  first_name VARCHAR(14) NOT NULL,
  last_name VARCHAR(16) NOT NULL,
  gender ENUM('M','F') NOT NULL,
  hire_date DATE NOT NULL,
  avgSalary DECIMAL(10,2) DEFAULT NULL,
PRIMARY KEY (emp_no))
ENGINE=InnoDB
DEFAULT CHARSET=latin1
```

MySQL Storage Engine...

- Z ukazom `show engines` lahko preverimo, katere pogone podpira naša različica MySQL;

Engine	Support	Comment	Transactions	XA	Savepoints
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
PERFORMANC...	YES	Performance Schema	NO	NO	NO

MySQL Storage Engine...

Feature	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	None	384EB
Transactions	No	No	Yes	No	Yes
Locking granularity	Table	Table	Row	Table	Row
MVCC	No	No	Yes	No	No
Geospatial data type support	Yes	No	Yes	Yes	Yes
Geospatial indexing support	Yes	No	No	No	No
B-tree indexes	Yes	Yes	Yes	No	No
T-tree indexes	No	No	No	No	Yes
Hash indexes	No	Yes	No [a]	No	Yes
Full-text search indexes	Yes	No	Yes [b]	No	No
Clustered indexes	No	No	Yes	No	No
Data caches	No	N/A	Yes	No	Yes
Index caches	Yes	N/A	Yes	No	Yes
Compressed data	Yes [c]	No	Yes [d]	Yes	No
Encrypted data [e]	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Replication support [f]	Yes	Yes	Yes	Yes	Yes
Foreign key support	No	No	Yes	No	No
Backup / point-in-time recovery [g]	Yes	Yes	Yes	Yes	Yes
Query cache support	Yes	Yes	Yes	Yes	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes

K4.3 – Izbira indeksov...

- Namen: ugotoviti, ali lahko z dodatnimi indeksi povečamo učinkovitost sistema.
- Možen pristop:
  - Zapise pustimo neurejene.
  - Izdelamo toliko sekundarnih indeksov, kolikor je potrebno.

Sekundarni indeks je indeks po atributu, ki ni obenem tudi atribut, po katerem je urejena relacija

K4.3 – Izbira indeksov...

- Alternativni pristop
  - Zapise uredimo po primarnem ključu ali po indeksu gruče. V tem primeru kot atribut za sortiranje izberemo:
    - Atribut, ki se največkrat uporablja za povezovanja ali
    - Atribut, ki se najpogosteje uporablja za dostop do podatkov v relaciji.
    - Če je izbrani atribut za sortiranje primarni ključ, potem gre za primarni indeks sicer za indeks gruče.
  - Relacija ima lahko primarni indeks ali indeks gruče

Primarni indeks je indeks po primarnem ključu, po katerem je urejena relacija. Vsak zapis ima svojo vrednost.

Indeks gruče je indeks po atributu, ki je obenem tudi atribut, po katerem je urejena relacija, ni pa primarni ključ. Ključ ni unikaten!

K4.3 – Izbira indeksov...

- Sekundarni indeksi so način, kako omogočiti učinkovito iskanje s pomočjo dodatnih ključev.
- Pri določanju sekundarnih indeksov upoštevamo:
  - Povečanje učinkovitosti (predvsem pri iskanju po PB)
  - Dodatno delo, ki ga mora sistem opravljati za vzdrževanje indeksov. To vključuje:
    - Dodajanje zapisa v vsak sekundarni indeks, kadarkoli dodamo nek zapis v osnovno relacijo
    - Spreminjanje sekundarnega indeksa vsakokrat, ko se osnovna relacija spremeni
    - Povečanje porabe prostora v sekundarnem pomnilniku
    - Povečanje časovnega obsega za optimizacijo poizvedb zaradi preverjanja vseh sekundarnih indeksov.

K4.3 – Izbira indeksov...

- Nekaj smernic za uporabo sekundarnih indeksov:
  - Ne indeksiraj majhnih relacij.
  - Če datoteka ni urejena po primarnem ključu, potem kreiraj indeks na osnovi primarnega ključa.
  - Če je tuji ključ pogosto v uporabi, dodaj sekundarni indeks na tuji ključ.
  - Sekundarni indeks dodaj vsakemu atributu, ki se pogosto uporablja kot sekundarni ključ.
  - Sekundarne indekse dodaj atributom, ki nastopajo v pogojih za selekcijo ali stik: ORDER BY; GROUP BY ali v drugih operacijah, ki vključujejo sortiranje (npr. UNION ali DISTINCT).

K4.3 – Izbira indeksov

- Nekaj smernic za uporabo sekundarnih indeksov: (nadaljevanje):
  - Dodaj sekundarni indeks atributom, ki nastopajo v vgrajenih funkcijah;
  - Izogibaj se indeksiranju atributov, ki se pogosto spreminjajo.
  - Izogibaj se indeksiranju atributov v relacijah, nad katerimi se bodo pogosto izvajale poizvedbe, ki bodo vključevale večji del zapisov.
  - Izogibaj se indeksiranju atributov, ki so predstavljeni z daljšimi stringi.

K4.3 – Izbira indeksov

- Z ukazom `explain` preverimo, ali imamo vse potrebne indekse

```
164
165   EXPLAIN select e.first_name, e.last_name, d.dept_name
166   from employees e
167       inner join dept_manager dm on dm.emp_no = e.emp_no
168       inner join departments d on d.dept_no = dm.dept_no
169   where dm.from_date < '1991-01-01' and dm.to_date > '1990-01-01';
178
```

100% 2:165 2 errors found

Filter: Search Export:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	d	index	PRIMARY	dept_name	42	NULL	10	Using index
1	SIMPLE	dm	ref	PRIMARY,emp_no,dept_no	dept_no	4	employees.d.dept_no	1	Using where
1	SIMPLE	e	eq_ref	PRIMARY	PRIMARY	4	employees.dm.emp_no	1	

Primer

```
EXPLAIN SELECT * FROM
orderdetails d
INNER JOIN orders o ON d.orderNumber = o.orderNumber
INNER JOIN products p ON p.productCode = d.productCode
INNER JOIN productlines l ON p.productLine = l.productLine
INNER JOIN customers c on c.customerNumber = o.customerNumber
WHERE o.orderNumber = 10101G
```

```
*********************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: l
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 7
        Extra:
*********************** 2. row ***********************
           id: 1
  select_type: SIMPLE
        table: p
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 110
        Extra: Using where; Using join buffer
*********************** 3. row ***********************
           id: 1
  select_type: SIMPLE
        table: c
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 122
        Extra: Using join buffer
*********************** 4. row ***********************
           id: 1
  select_type: SIMPLE
        table: o
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 326
        Extra: Using where; Using join buffer
*********************** 5. row ***********************
           id: 1
  select_type: SIMPLE
        table: d
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2996
        Extra: Using where; Using join buffer
5 rows in set (0.00 sec)
```

```
ALTER TABLE customers
    ADD PRIMARY KEY (customerNumber);
ALTER TABLE employees
    ADD PRIMARY KEY (employeeNumber);
ALTER TABLE offices
    ADD PRIMARY KEY (officeCode);
ALTER TABLE orderdetails
    ADD PRIMARY KEY (orderNumber, productCode);
ALTER TABLE orders
    ADD PRIMARY KEY (orderNumber),
    ADD KEY (customerNumber);
ALTER TABLE payments
    ADD PRIMARY KEY (customerNumber, checkNumber);
ALTER TABLE productlines
    ADD PRIMARY KEY (productLine);
ALTER TABLE products
    ADD PRIMARY KEY (productCode),
    ADD KEY (buyPrice),
    ADD KEY (productLine);
ALTER TABLE productvariants
    ADD PRIMARY KEY (variantId),
    ADD KEY (buyPrice),
    ADD KEY (productCode);
```

```
*********************** 1. row ***********************
           id: 1
  select_type: SIMPLE
        table: o
         type: const
possible_keys: PRIMARY,customerNumber
          key: PRIMARY
      key_len: 4
          ref: const
         rows: 1
        Extra:
*********************** 2. row ***********************
           id: 1
  select_type: SIMPLE
        table: c
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: const
         rows: 1
        Extra:
*********************** 3. row ***********************
           id: 1
  select_type: SIMPLE
        table: d
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: const
         rows: 4
        Extra:
```

```
*********************** 4. row ***********************
           id: 1
  select_type: SIMPLE
        table: p
         type: eq_ref
possible_keys: PRIMARY,productLine
          key: PRIMARY
      key_len: 17
          ref: classicmodels.d.productCode
         rows: 1
        Extra:
*********************** 5. row ***********************
           id: 1
  select_type: SIMPLE
        table: l
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 52
          ref: classicmodels.p.productLine
         rows: 1
        Extra:
5 rows in set (0.00 sec)
```

$$7 \times 110 \times 122 \times 326 \times 2996 = 91{,}750{,}822{,}240$$

Vrste indeksov v MySQL...

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX
index_name
    [index_type]
    ON tbl_name (index_col_name,...)
    [index_option]
    [algorithm_option | lock_option] ...


index_type:
USING {BTREE | HASH | RTREE}

index_col_name:
col_name [(length)] [ASC | DESC]
```

Storage Engine	Permissible Index Types
MyISAM	BTREE
InnoDB	BTREE
MEMORY/HEAP	HASH, BTREE
NDB	BTREE, HASH (see note in text)

odvisno od uporabljenega pogona

Vrste indeksov v MySQL

- BTREE Indexes
  - Večina indeksov v MySQL
- RTREE Indexes
  - Za GIS, samo MyISAM
- HASH Indexes
  - MEMORY, NDB
- FULLTEXT Indexes
  - MyISAM, Innodb v ver. MySQL 5.6

K4.4 – Ocena velikosti podatkovne baze

- Namen: oceniti, koliko prostora v sekundarnem pomnilniku zahteva načrtovana podatkovna baza.
- Ocena je odvisna
  - od velikosti in števila zapisov ter
  - od ciljnega SUPB.
- Primer: ocena velikosti podatkovne baze s pomočjo orodja PowerDesigner.

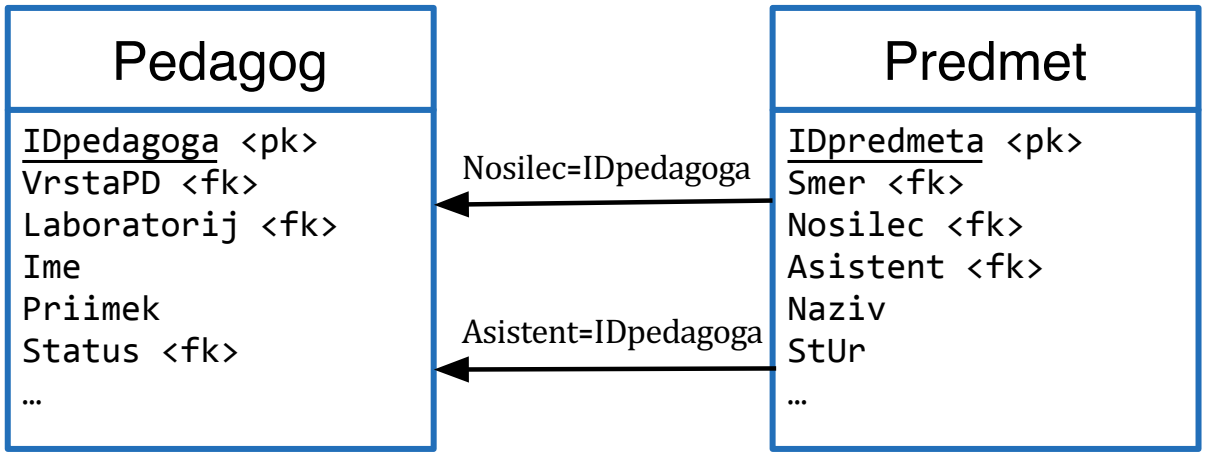
K5 – Načrt uporabniških pogledov

- Namen: izdelati načrt uporabniških pogledov, ki so bili opredeljeni v okviru zajema uporabniških zahtev.
- Uporabimo mehanizem pogledov (view).
- Pogled je navidezna relacija, ki fizično ne obstaja v PB, temveč se vsakokratno kreira s pomočjo poizvedbe.

Kreiranje pogledov v MySQL

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Primer pogleda




```
create or replace view obremenitev as
select  pmet.IDpedagoga as Predavatelj,
    (pdg.Priimek + ' ' + pdg.Ime) as Naziv_predavatelja,
    count(*) as Stevilo_Predmetov,
    sum(pmet.StUr) as Stevilo_ur_tedensko
from Predmet as pmet, Pedagog as pdg
where pmet.Nosilec = pdg.IDPedagoga
group by pmet.Nosilec, (pdg.priimek + pdg.ime);
```

Obremenitev

Predavatelj
Naziv_predavatelja
Stevilo_Predmetov
Stevilo_ur_tedensko

Primer pogleda

- Trenutni vodje posameznih oddelkov:

```
create view managers as
    select e.emp_no, e.first_name, e.last_name, d.dept_name, dm.from_date
    from employees e inner join dept_manager dm on e.emp_no = dm.emp_no
    inner join departments d on dm.dept_no = d.dept_no
    where dm.to_date = '9999-01-01';
```

```
select * from managers;
```

emp_no	first_name	last_name	dept_name	from_date
111939	Yuchang	Weedman	Customer Service	1996-01-03
110567	Leon	DasSarma	Development	1992-04-25
110114	Isamu	Legleitner	Finance	1989-12-17
110228	Karsten	Sigstam	Human Resources	1992-03-21
110039	Vishwani	Minakawa	Marketing	1991-10-01
110420	Oscar	Ghazalie	Production	1996-08-30
110854	Dung	Pesch	Quality Management	1994-06-28
111534	Hilary	Kambil	Research	1991-04-08
111133	Hauke	Zhang	Sales	1991-03-07

K6 – Načrt varnostnih mehanizmov

- Namen: izdelati načrt varnostnih mehanizmov skladno z zahtevami naročnika.
- SUPB-ji tipično podpirajo dve vrsti varnosti:
  - Sistemsko varnost: varnost dostopa in uporabe podatkovne baze (navadno zagotovljeno s pomočjo uporabniških imen in gesel);
  - Podatkovno varnost: varnost uporabe podatkov – kdo lahko uporablja določene relacije ter kako.
- Med SUPB-ji so velike razlike v mehanizmih, ki jih imajo na voljo za dosego varnosti.

MySQL in varnost

- Gesla
  - Ne zapisuj gesel v prostem tekstu v bazo. Uporabi enega od hash algoritmov, npr MD5, SHA1 ipd. Uporabi dvojni hash z dodatkom: `hash(hash(password)+salt)`
  - Od MySQL ver 4.1.1 geslo kriptirano pri komunikaciji
  - Možnost uporabe SSL povezave za ostalo komunikacijo
- Dodeljevanje privilegijev oziroma pravic (`GRANT`)
  - Dodeljuj previdno (`show grants`)
  - Pozor: tabela `USER`

```
use mysql;
select * from user;
```

K7 – Uvedba nadzorovane redundance...

- Namen: ugotoviti, ali je smiselno dopustiti določeno mero redundance (denormalizacija) ter tako izboljšati učinkovitost.
- Rezultat normalizacije je načrt, ki je po strukturi konsistenten ter minimalen.
- Včasih normalizirane relacije ne dajo zadovoljive učinkovitosti.
- Razmislimo, ali se zavoljo izboljšanja učinkovitosti odpovemo določenim koristim, ki jih prinaša normalizacija.

K7 – Uvedba nadzorovane redundance...

- Upoštevamo tudi:
  - Implementacija denormaliziranih relacij je težja;
  - Z denormalizacijo velikokrat zgubimo na prilagodljivosti modela;
  - Denormalizacija navadno pospeši poizvedbe, vendar upočasni spreminjanje podatkov.

K7 – Uvedba nadzorovane redundance...

- Denormalizacija:
  - Denormaliacija se nanaša na dopolnitev relacijske sheme, tako da eni ali več relacij znižamo stopnjo normalne oblike (npr. 3NO → 2NO).
  - Nanaša se tudi na primere, ko dve normalizirani relaciji združimo v eno, ki je še vedno normalizirana, vendar zaradi združitve vsebuje več nedefiniranih vrednosti (NULL).
    (4PNO → 3NO).

Primer: združitev relacij

employees

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18

salaries

emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24
10001	74333	1992-06-24	1993-06-24
10001	75286	1993-06-24	1994-06-24
10001	75994	1994-06-24	1995-06-24
10001	76884	1995-06-24	1996-06-23
10001	80013	1996-06-23	1997-06-23
10001	81025	1997-06-23	1998-06-23

employees with salaries

emp_no	birth_date	first_name	last_name	gender	hire_date	avgSalary	salary	from_date	to_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	60117	1986-06-26	1987-06-26
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	62102	1987-06-26	1988-06-25
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	66074	1988-06-25	1989-06-25
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	66596	1989-06-25	1990-06-25
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	66961	1990-06-25	1991-06-25
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	71046	1991-06-25	1992-06-24
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	74333	1992-06-24	1993-06-24
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	75286	1993-06-24	1994-06-24
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	75994	1994-06-24	1995-06-24
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	76884	1995-06-24	1996-06-23
10001	1953-09-02	Georgi	Facello	M	1986-06-26	76700.67	80013	1996-06-23	1997-06-23

Primer: podvajanje neosnovnih atributov

employees

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18

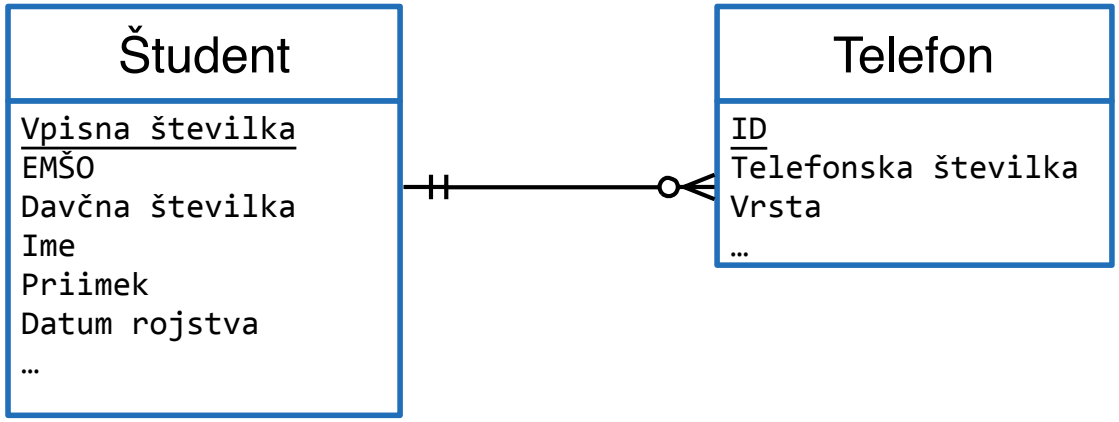
salaries

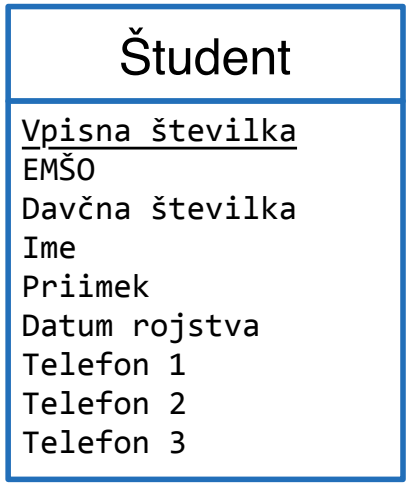
emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24
10001	74333	1992-06-24	1993-06-24
10001	75286	1993-06-24	1994-06-24
10001	75994	1994-06-24	1995-06-24
10001	76884	1995-06-24	1996-06-23
10001	80013	1996-06-23	1997-06-23
10001	81025	1997-06-23	1998-06-23

employees with current salary

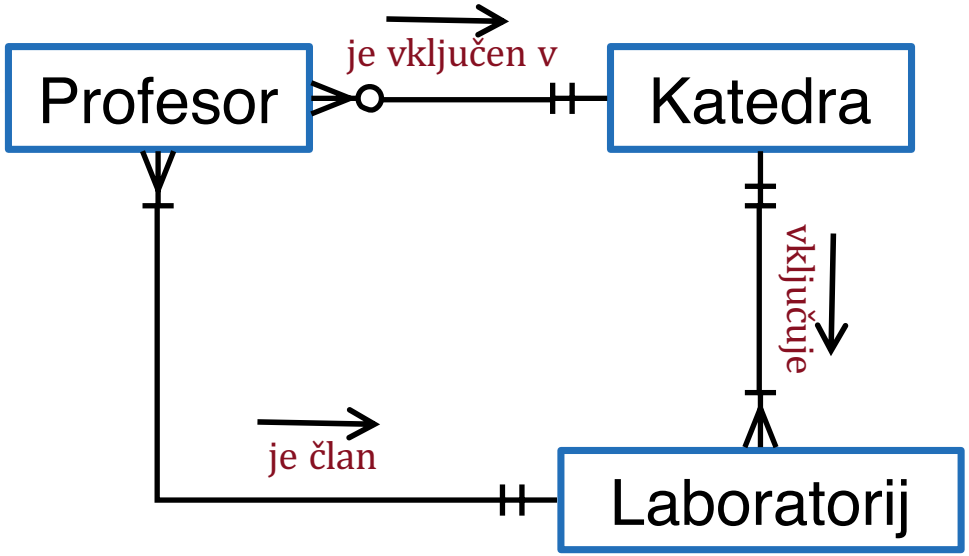
emp_no	birth_date	first_name	last_name	gender	hire_date	salary
10001	1953-09-02	Georgi	Facello	M	1986-06-26	88958
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	72527
10003	1959-12-03	Parto	Bamford	M	1986-08-28	43311
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	74057
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	94692
10006	1953-04-20	Anneke	Preusig	F	1989-06-02	59755
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10	88070
10009	1952-04-19	Sumant	Peac	F	1985-02-18	94409
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24	80324
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18	54423

Primer: ponavljajoče skupine



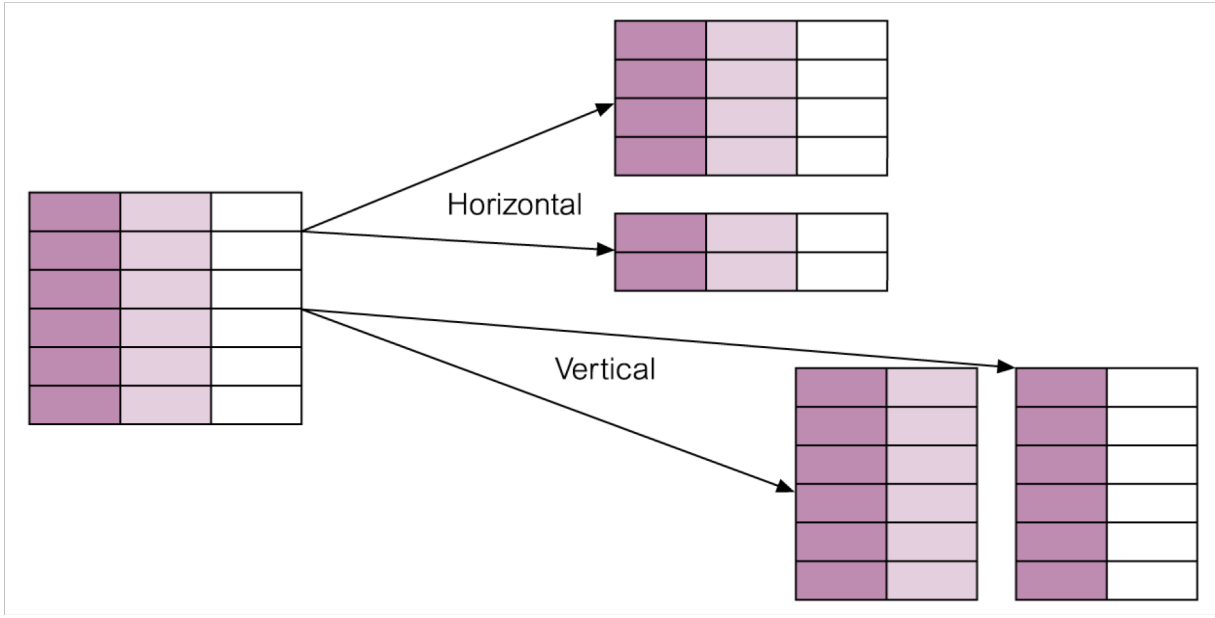


Primer: podvajanje tujih ključev



Razbitje relacij

- Za povečanje učinkovitosti nad relacijami z zelo velikim številom n-teric uporabimo pristop, kjer relacijo razbijemo na manjše dele - particije.
- Dva načina delitve sta:
  - Horizontalni in
  - Vertikalni.



Prednosti razbitja relacije na particije

- Uporaba particioniranja prinaša številne prednosti:
  - Boljša porazdelitev vnosa (load balancing)
  - Večja učinkovitosti (manj podatkov za obdelavo, paralelnost izvajanja,...)
  - Boljša razpoložljivost
  - Boljša obnovljivost
  - Več možnosti za zagotavljanje varnosti

Slabosti razbitja relacije na particije

- Particioniranje ima tudi slabosti:
  - Kompleksnost (particije niso vedno transparentne za uporabnike...)
  - Slabša učinkovitost (pri poizvedbah, kjer je potrebno poizvedovati po več particijah, je učinkovitost slabša)
  - Podvajanje podatkov (pri vertikalnem particioniranju)

Primer – particije v MySQL

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
```

Primer – particije v MySQL

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY HASH(YEAR(hired))
PARTITIONS 4;
```

Pregledovanje fizične sheme v MySQL

- Ukazi v MySQL za pregled kataloga
  - SHOW
  - show [full] tables;
  - show columns;
  - show index;
  - ...
  - DESCRIBE table_name;

SHOW BINARY LOGS Syntax
SHOW BINLOG EVENTS Syntax
SHOW CHARACTER SET Syntax
SHOW COLLATION Syntax
SHOW COLUMNS Syntax
SHOW CREATE DATABASE Syntax
SHOW CREATE FUNCTION Syntax
SHOW CREATE PROCEDURE Syntax
SHOW CREATE TABLE Syntax
SHOW CREATE VIEW Syntax
SHOW DATABASES Syntax
SHOW ENGINE Syntax
SHOW ENGINES Syntax
SHOW ERRORS Syntax
SHOW FUNCTION CODE Syntax
SHOW FUNCTION STATUS Syntax
SHOW GRANTS Syntax
SHOW INDEX Syntax
SHOW INNODB STATUS Syntax
SHOW MASTER STATUS Syntax
SHOW MUTEX STATUS Syntax
SHOW OPEN TABLES Syntax
SHOW PRIVILEGES Syntax
SHOW PROCEDURE CODE Syntax
SHOW PROCEDURE STATUS Syntax
SHOW PROCESSLIST Syntax
SHOW PROFILE Syntax
SHOW PROFILES Syntax
SHOW SLAVE HOSTS Syntax
SHOW SLAVE STATUS Syntax
SHOW STATUS Syntax
SHOW TABLE STATUS Syntax
SHOW TABLES Syntax
SHOW TRIGGERS Syntax
SHOW VARIABLES Syntax
SHOW WARNINGS Syntax

Primer ukaza SHOW TABLES

```
use employees;
show tables;
```

Tables_in_employees
departments
dept_emp
dept_manager
employees
managers
salaries
titles

Primer pregleda sheme

```
use information_schema;
show tables;
```

Tables_in_information_schema
CHARACTER_SETS
COLLATIONS
COLLATION_CHARACTER_SET_APPLICABILITY
COLUMNS
COLUMN_PRIVILEGES
ENGINES
EVENTS
FILES
GLOBAL_STATUS
GLOBAL_VARIABLES
KEY_COLUMN_USAGE
PARAMETERS
PARTITIONS
PLUGINS
PROCESSLIST
PROFILING

```
select * from schemata;
```

CATALOG_NAME	SCHEMA_NAME	DEFAULT_CHA...	DEFAULT_COLL...	SQL_PATH
def	information_schema	utf8	utf8_general_ci	NULL
def	Citations	latin1	latin1_swedish_ci	NULL
def	HotelChain	utf8	utf8_slovenian_ci	NULL
def	Movies	utf8	utf8_unicode_ci	NULL
def	WoSDalibor	latin1	latin1_swedish_ci	NULL
def	employees	latin1	latin1_swedish_ci	NULL
def	mydb	utf8	utf8_general_ci	NULL
def	mysql	latin1	latin1_swedish_ci	NULL
def	performance_schema	utf8	utf8_general_ci	NULL
def	sakila	latin1	latin1_swedish_ci	NULL
def	studis	latin1	latin1_swedish_ci	NULL
def	test	latin1	latin1_swedish_ci	NULL
def	wos	utf8	utf8_slovenian_ci	NULL

Primer ukaza DESCRIBE

```
describe employees;
```

	Field	Type	Null	Key	Default	Extra
▶	emp_no	int(11)	NO	PRI	NULL	
	birth_date	date	NO		NULL	
	first_name	varchar(14)	NO	MUL	NULL	
	last_name	varchar(16)	NO		NULL	
	gender	enum('M','F')	NO		NULL	
	hire_date	date	NO		NULL	

Primer ukaza SHOW INDEXES

```
show indexes from salaries;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
salaries	0	PRIMARY	1	emp_no	A	711128	NULL	NULL		BTREE
salaries	0	PRIMARY	2	from_date	A	2844513	NULL	NULL		BTREE
salaries	1	emp_no	1	emp_no	A	711128	NULL	NULL		BTREE

Primer ukaza SHOW CREATE TABLE

```
show create table employees;
```

Table	Create Table
employees	CREATE TABLE `employees` (`emp_no` int(11) NOT NULL, `birth_date` date NOT NULL, `first_name` varchar(14) NOT NULL, `last_name` varch

Za netabelarični izpis lahko uporabljate ukaz `\G` - na voljo samo v terminalu.

```
mysql> show create table employees\G;
*************************** 1. row ***************************
       Table: employees
Create Table: CREATE TABLE `employees` (
  `emp_no` int(11) NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`),
  KEY `idx_nameEmployee` (`first_name`(1))
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```