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



Predmet:
Osnove PB

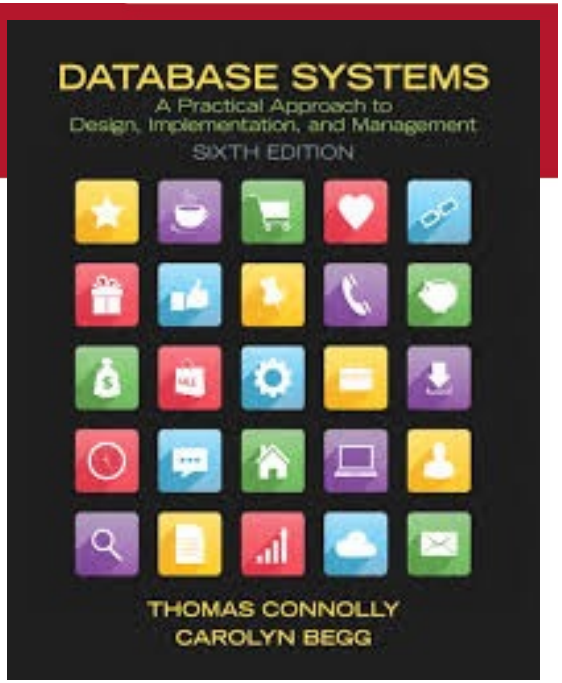
Modul:
O predmetu

Gradivo:
v.2016

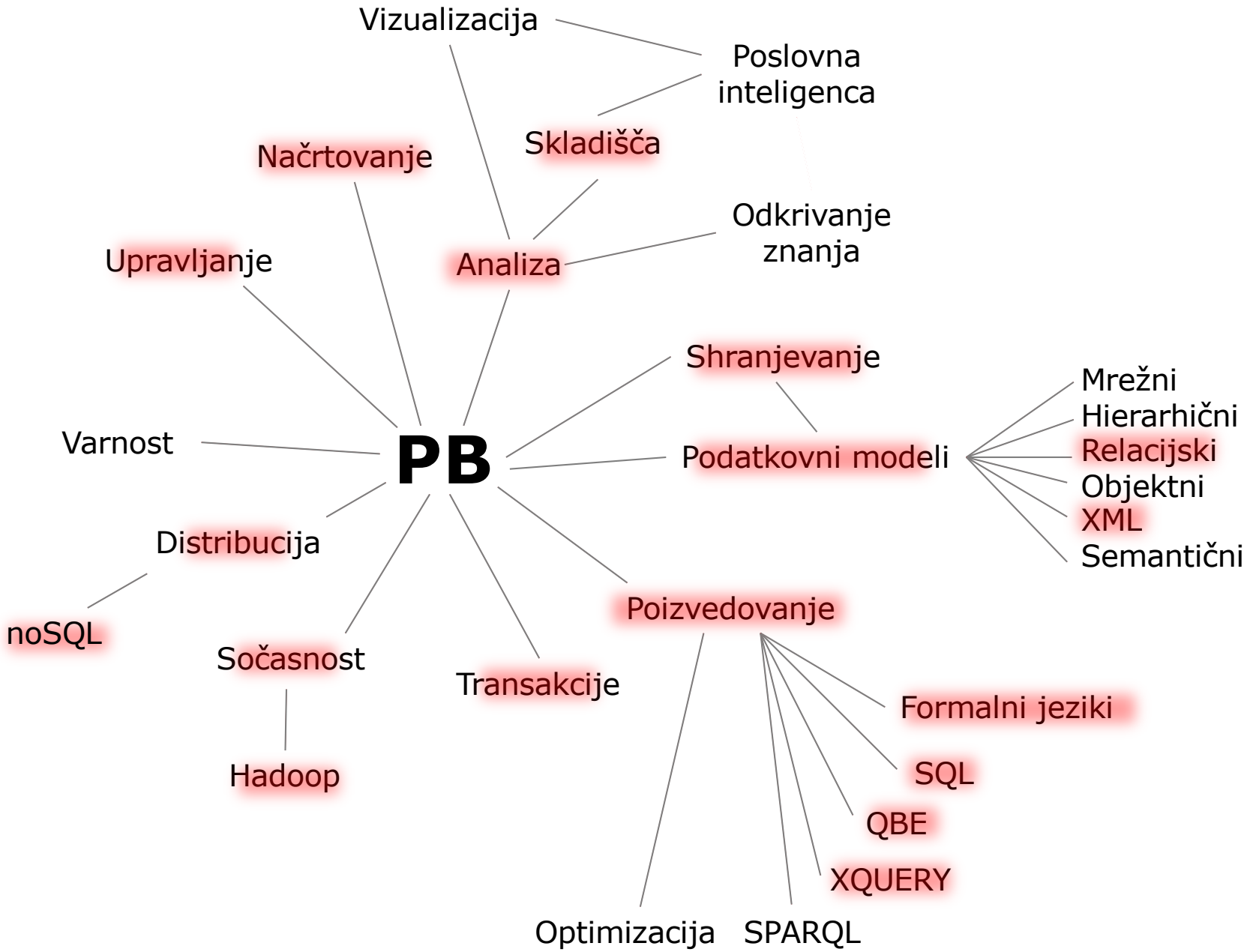
Splošne informacije

- Predavatelj:
  - Prof. dr. Marko Bajec, marko.bajec@fri.uni-lj.si
- Asistenti – vaje:
  - doc. dr. Lovro Šubelj, lovro.subelj@fri.uni-lj.si
  - asist. Timotej Knez
  - asist. Marko Rus

Splošne informacije



- Literatura:
  - Raghu Ramakrishnan, Johannes Gehrke (2003). Database Management Systems, Third Edition, McGraw-Hill
  - Thomas M. Connolly, Carolyn E. Begg (2015). Database Systems, A Practical Approach to Design, Implementation and Management, Sixth Edition, Addison-Wesley
  - Ramez Elmasri, Shamkant B. Navathe (2003). Fundamentals of Database Systems, Fourth Edition, Addison-Wesley
  - Peter Rob, Carlos Coronel (2005). Database Systems: Design, Implementation and Management, Sixth Ed., Addison Wesley.
  - …


Vizualizacija
Poslovna inteligenca
Skladišča
Načrtovanje
Odkrivanje znanja
Upravljanje
Analiza
Shranjevanje
Varnost
PB
Podatkovni modeli
Mrežni
Hierarhični
Relacijski
Objektni
XML
Semantični
Distribucija
noSQL
Poizvedovanje
Sočasnost
Transakcije
Formalni jeziki
SQL
QBE
XQUERY
Hadoop
Optimizacija
SPARQL

Vsebina predavanj…

- Uvod v PB
- Opisovanje in shranjevanje podatkov
  - Zgradba SUPB
  - Diski in diskovna polja
  - Upravljanje z diskom in pomnilnikom
  - Organizacija datotek in indeksiranje
- Poizvedovanje v PB
  - Relacijski podatkovni model
  - Relacijska algebra
  - SQL – strukturiran poizvedovalni jezik
  - QBE – poizvedba na osnovi primera

Vsebina predavanj

- Dodatne teme:
  - Uvod v XML, XML baze,
  - Poizvedbe z XQuery
  - Načrtovanje PB
  - Podatkovna skladišča
  - noSQL podatkovne baze

Terminski načrt

Termin		Predavanja	Vaje
1	4.10.-8.10.2021	Uvod v relacijske baze	/
2	11.10.-15.10.2021	Relacijska algebra & račun	/
3	18.10.-22.10.2021	Poizvedovalni jezik SQL	Poizvedovanje z RA
4	25.10.-29.10.2021	Poizvedovalni jezik QBE	Poizvedovanje z RA & RR
5	1.11.-5.11.2021	Diski & pomnilnik	Poizvedovanje z SQL
6	8.11.-12.11.2021	Organizacija & indeksiranje	Poizvedovanje z SQL & QBE
7	15.11.-19.11.2021	Konceptualno načrtovanje	*1. sprotno preverjanje*
8	22.11.-26.11.2021	Logično načrtovanje	Organizacija & indeksiranje
9	29.11.-3.12.2021	Fizično načrtovanje	Konceptualno načrtovanje
10	6.12.-10.12.2021	XML podatkovne baze	Logično načrtovanje
11	13.12.-17.12.2021	NoSQL podatkovne baze	Formalna normalizacija
12	20.12.-24.12.2021	Podatkovna skladišča	*2. sprotno preverjanje*
13	27.12.-31.12.2021	/	Poizvedovanje po XML
14	3.1.-7.1.2022	Hadoop ekosistem	Poizvedovanje po NoSQL
15	10.1.-14.1.2022	*Vabljeno predavanje*	*3. sprotno preverjanje*

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



Predmet:
Osnove PB

Modul:
Uvod v PB

Gradivo:
v.2018

PB in sistemi za upravljanje s PB

- Danes zbiramo podatke o skoraj vsem...











Sistemi za upravljanje s PB...

- Potrebujemo posebne mehanizme, ki omogočajo učinkovito shranjevanje in iskanje po podatkih → Sistemi za upravljanje PB (SUPB).

- SUPB – programska oprema za delo z večjimi količinami podatkov:
  - Varnost,
  - Konsistentnost,
  - Sočasnost,
  - Administracija
  - ...

Sistemi za upravljanje s PB

- Obstajajo različne vrste SUPB:
  - Relacijski, objektni, dokumentni, XML, noSQL…

- Primeri SUPB:
  - Oracle, Sybase, IBM DB2, MS SQL, MySQL, Postgres, ObjectStore, Jasmine, Objectivity/DB, Cassandra, MongoDB, Neo4J…













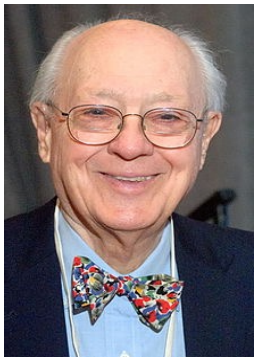






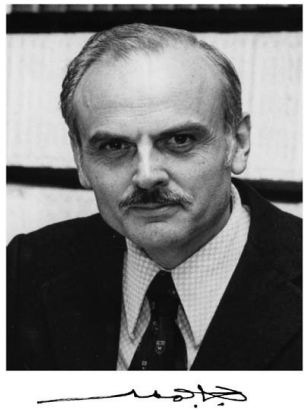


Zgodovina PB...



- Zgodnja 60' – začetki:
  - Prvi splošno namenski SUPB (Integrated Data Store), Charles Bachman (General Electric).
    ACM Turing Award
- Pozna 60' – vstop v poslovni svet:
  - Information Management System (IBM),
  - Hierarhični podatkovni model,
  - SABRE, sistem za rezervacijo letalskih kart (American Airlines in IBM) …

Zgodovina PB…

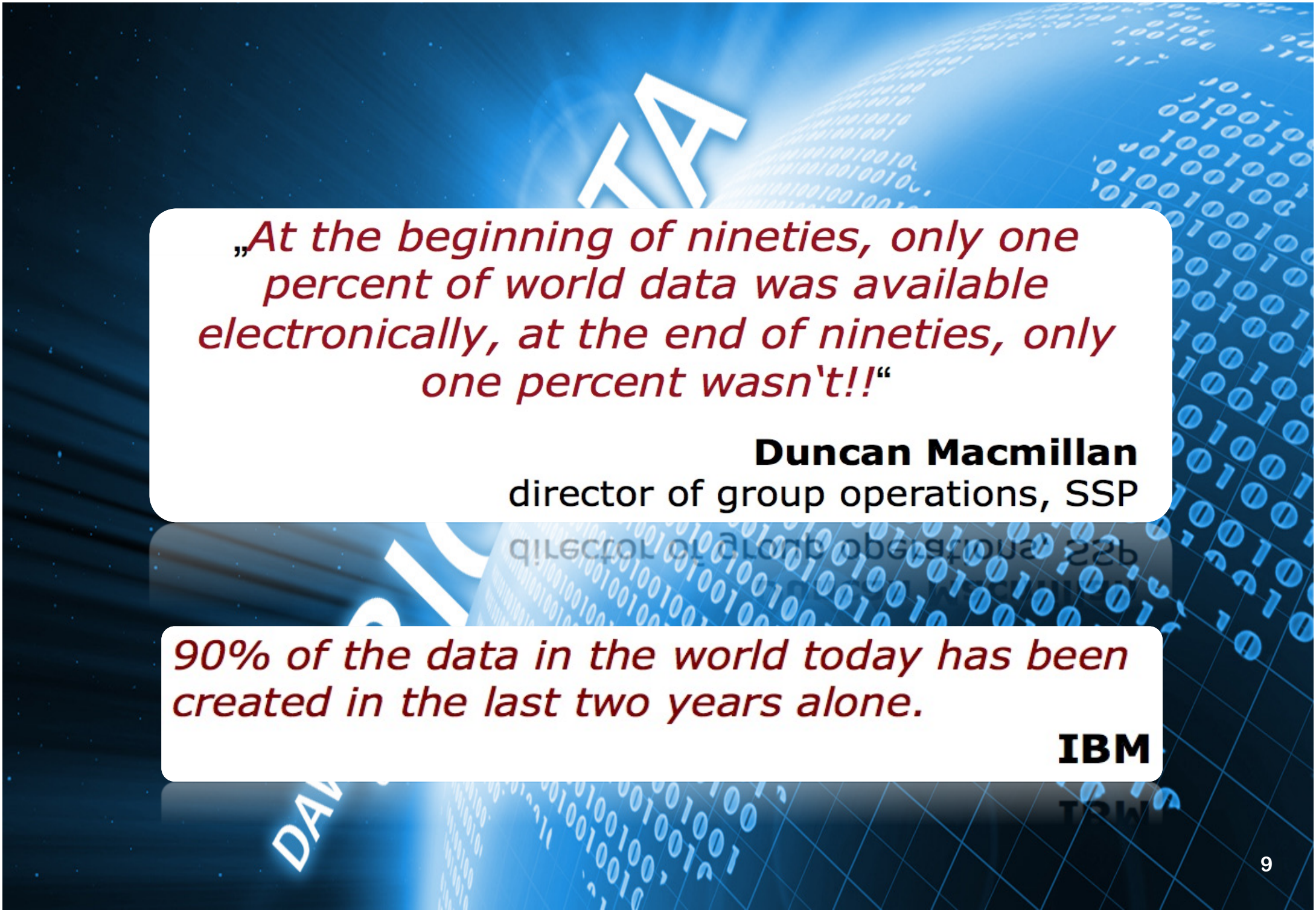


- 70' – relacijski podatkovni model:
  - Relacijski podatkovni model (Edgar Codd, ACM Turing Award),
  - Pojav večjega števila SUPB,
  - Širok interes s strani raziskovalcev,
  - Relacijske PB postanejo standard.
- 80' – SQL:
  - SQL (System R/IBM)
  - SQL standardizacija … SQL-92, ANSI, ISO.
  - Sočasni dostop do podatkov.

Zgodovina PB...

- Pozna 80' in 90':
  - Podatkovna skladišča,
  - Odkrivanje znanja iz podatkov,
  - ERP sistemi, centralna PB,
  - PB in internet.

- Danes:
  - Multimedijske PB, PB za interaktivni video,
  - Digitalne knjižnice,
  - Analiza podatkov, podatkovno rudarjenje, odkrivanje znanja,
  - Veliki podatki, umetna inteligenca.


„At the beginning of nineties, only one percent of world data was available electronically, at the end of nineties, only one percent wasn't!!“
Duncan Macmillan
director of group operations, SSP
90% of the data in the world today has been created in the last two years alone.
IBM

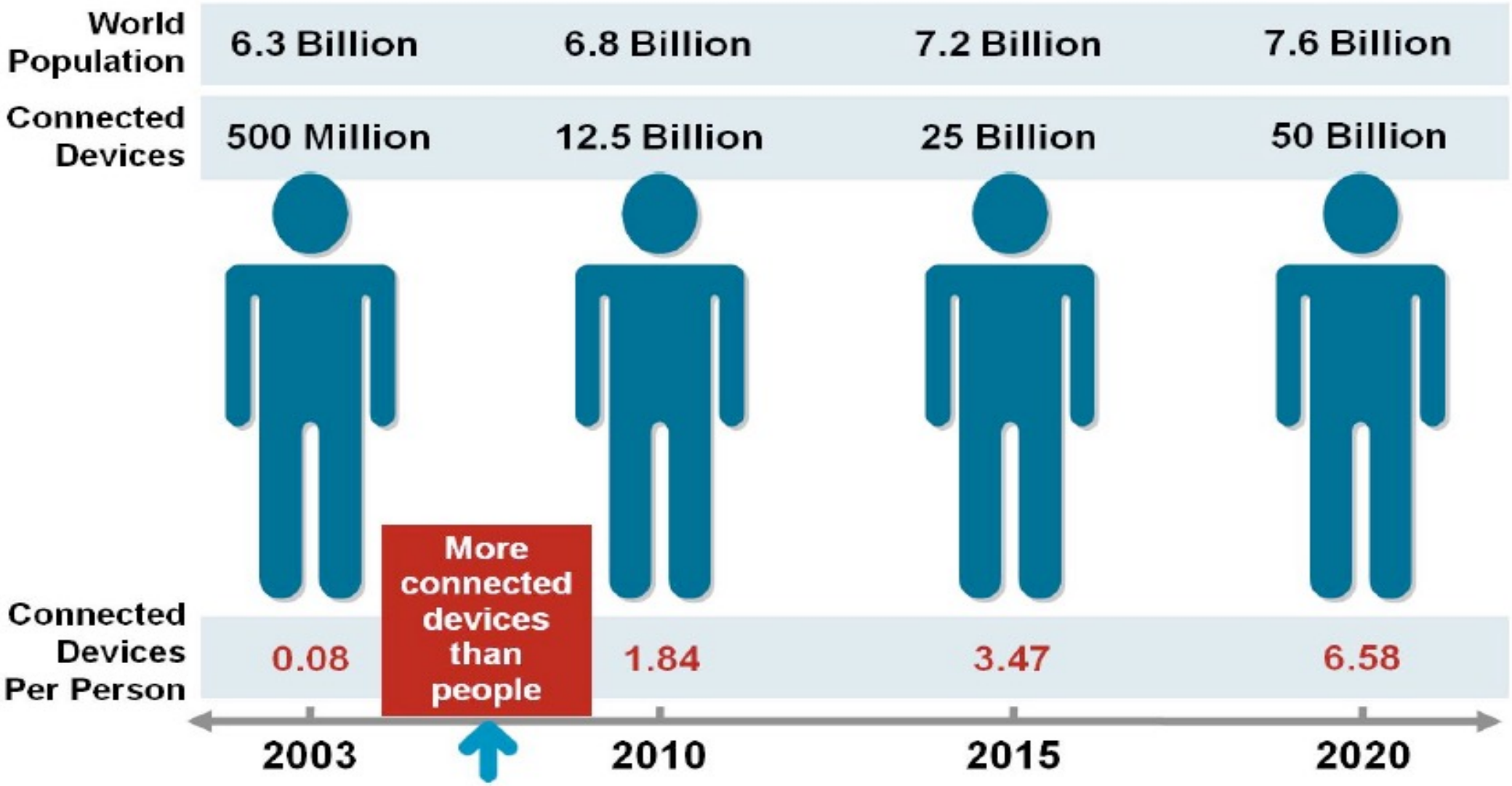





Accelerating Science
Accélérateur de science


The Social Web
Social Networking
Ning
Micro-blogging
twitxr
twitter
PLURK
Publishing
WIKIPEDIA
slideshare
Conversation Apps.
Photo Sharing
Picasa
flickr
photobucket
Search
Ask
bing
Google
Aggregators
digg
Google Reader
Gaming
WORLD WARCRAFT
Audio
Rhapsody
iTunes
p dcast.com
Virtual Worlds
kaneva
LIFE
Video
YouTube
metacafe
viddler
Crowd sourcing
YAHOO! ANSWERS
TweetBrain
Mobile
Aol Mobile.
RSS
FeedBurner
Live-casting
justin.tv
blogtalkradio
USTREAM
callwave
Created by Mirna Bard
www.mirnabard.com


44,000,000
MESSAGES PROCESSED
486,000
PHOTOS
26
NEW REVIEWS
POSTED ON YELP
120
NEW ACCOUNTS
OPENED ON
LINKEDIN
MORE THAN
140
SUBMISSIONS
ON REDDIT
MORE THAN
2,315,000
SEARCHES
3,125,000
243,055
MORE THAN
21,000,000
MESSAGES SENT
70,000
VIDEO MESSAGES
SHARED
MORE THAN
195,000
MINUTES OF AUDIO CHATTING
ON WECHAT
Share
MORE THAN
3,000,000
ITEMS ARE
SHARED
MORE THAN
18,000
MATCHES MADE
972,000
DAILY SWIPES
ON TINDER
IN
60
SECONDS
GO-Globe
CUSTOM WEB DEVELOPMENT
MORE THAN
69,500
HOURS OF
VIDEO WATCHED
ON NETFLIX
NETFLIX
MORE THAN
150,000,000
E-MAILS ARE SENT
MORE THAN
48,000
APPS DOWNLOADED
ON IPHONE
MORE THAN
95,000
APPS DOWNLOADED
ON ANDROID
YouTube
MORE THAN
430,000
TWEETS SENT
MORE THAN
2,700,000
VIDEO VIEWS AND
139,000 HOURS
OF VIDEO WATCHED
MORE THAN
39,300
HOURS OF MUSIC
LISTENED
AROUND
56,000
PHOTOS
UPLOADED
9,800
ARTICLES PINNED
ON PINTEREST
MORE THAN
280,000
SNAPS SENT
ON SNAPCHAT
MORE THAN
100
NEW DOMAINS
REGISTERED
14 NEW
SONGS ADDED
ON SPOTIFY
MORE THAN
300 HOURS
OF VIDEO ARE UPLOADED



Source: Cisco IBSG, April 2011

Apple iMac vs iPhone

včaraj in danes



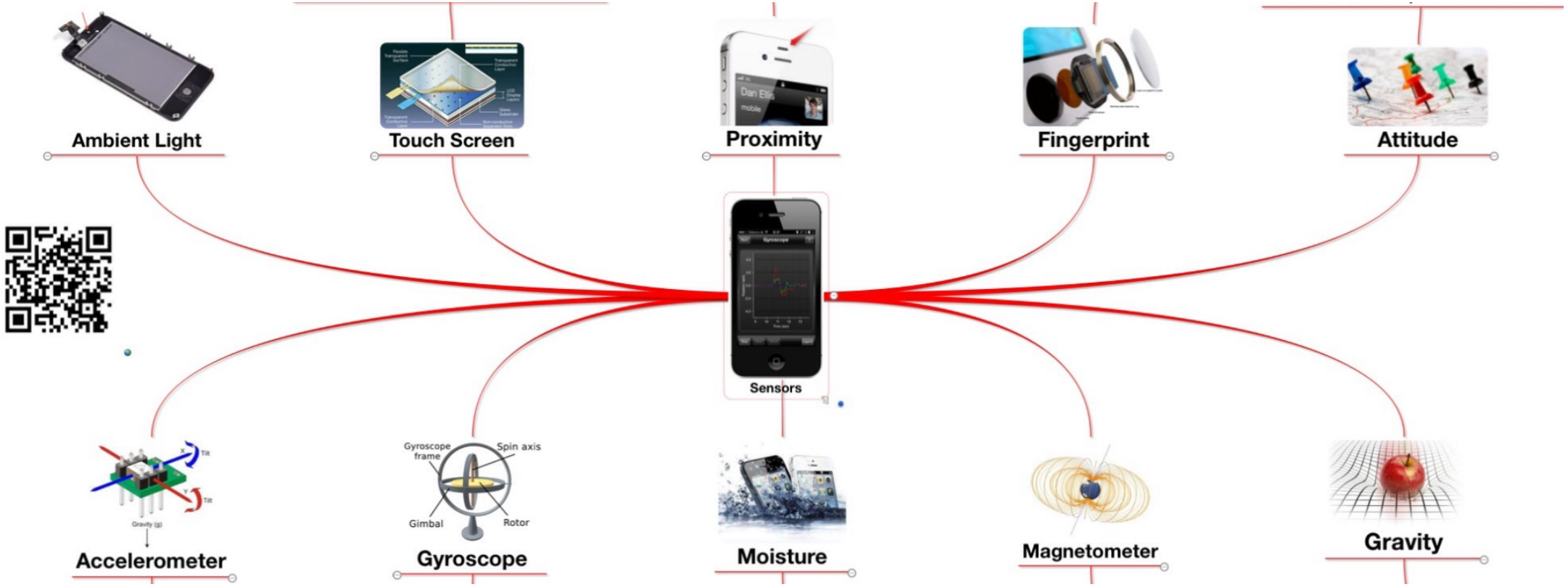
Apple iMac, 2001



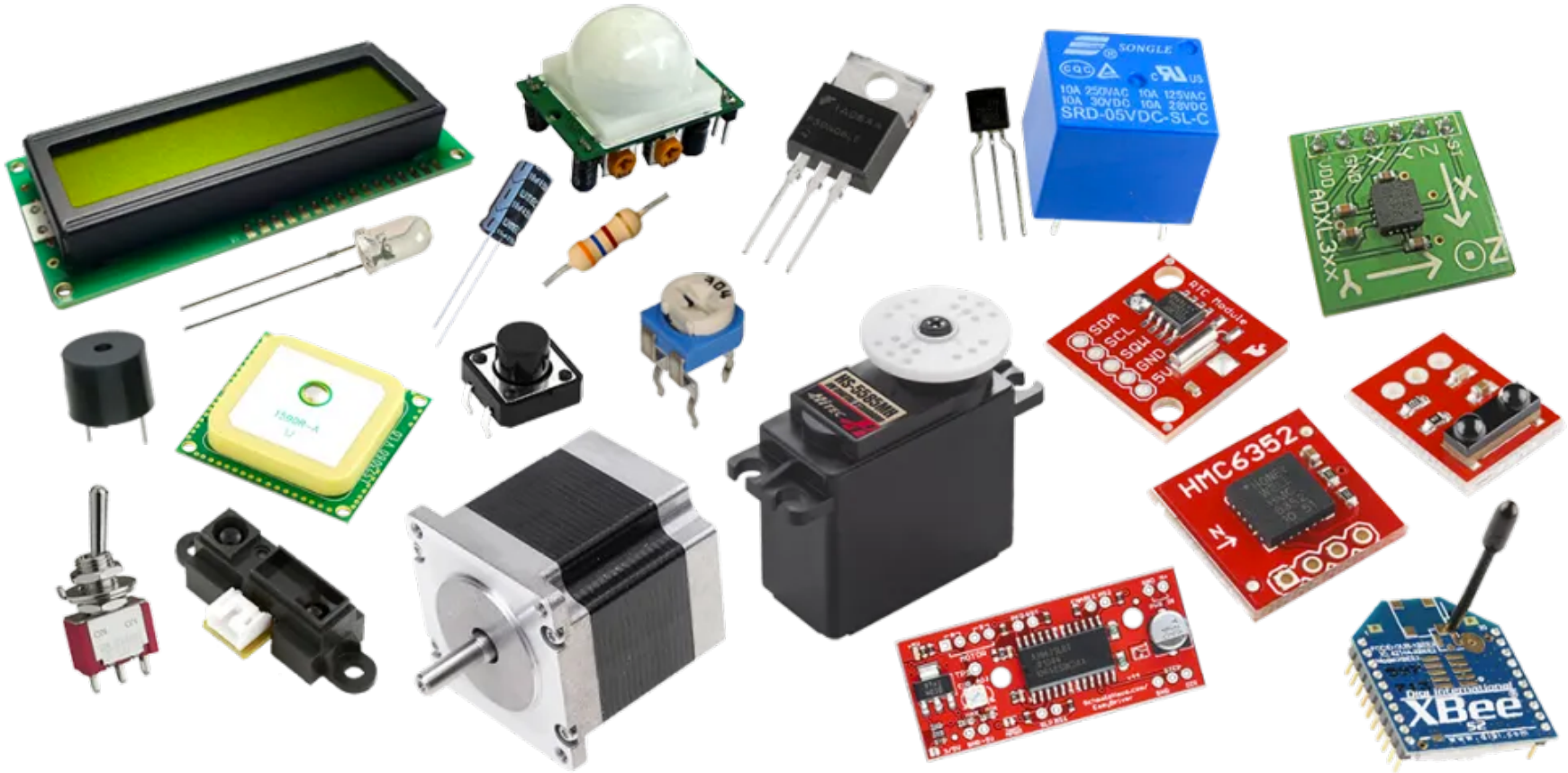
Apple iPhone xS, 2019

Procesor	0,6 GHz	2,49 GHz
RAM	256 MB	4.000 MB
Grafika	800 x 600 pixels	1125 x 2436 pixels
Disk	40 GB	256 GB
Cena	cca 1.200 EUR	cca 1.200 EUR

Apple iPhone senzorji



Senzorji in aktuatorji

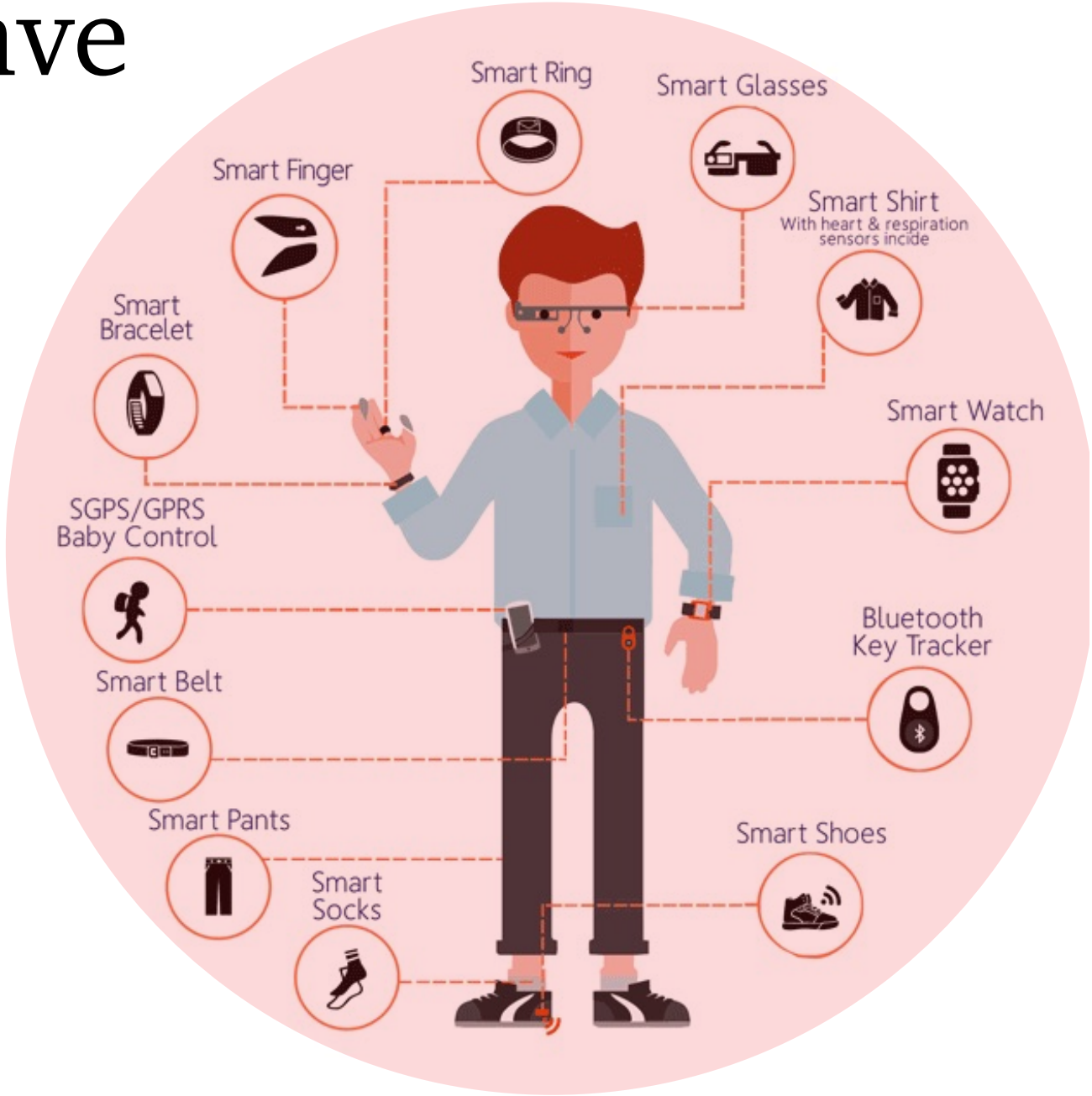


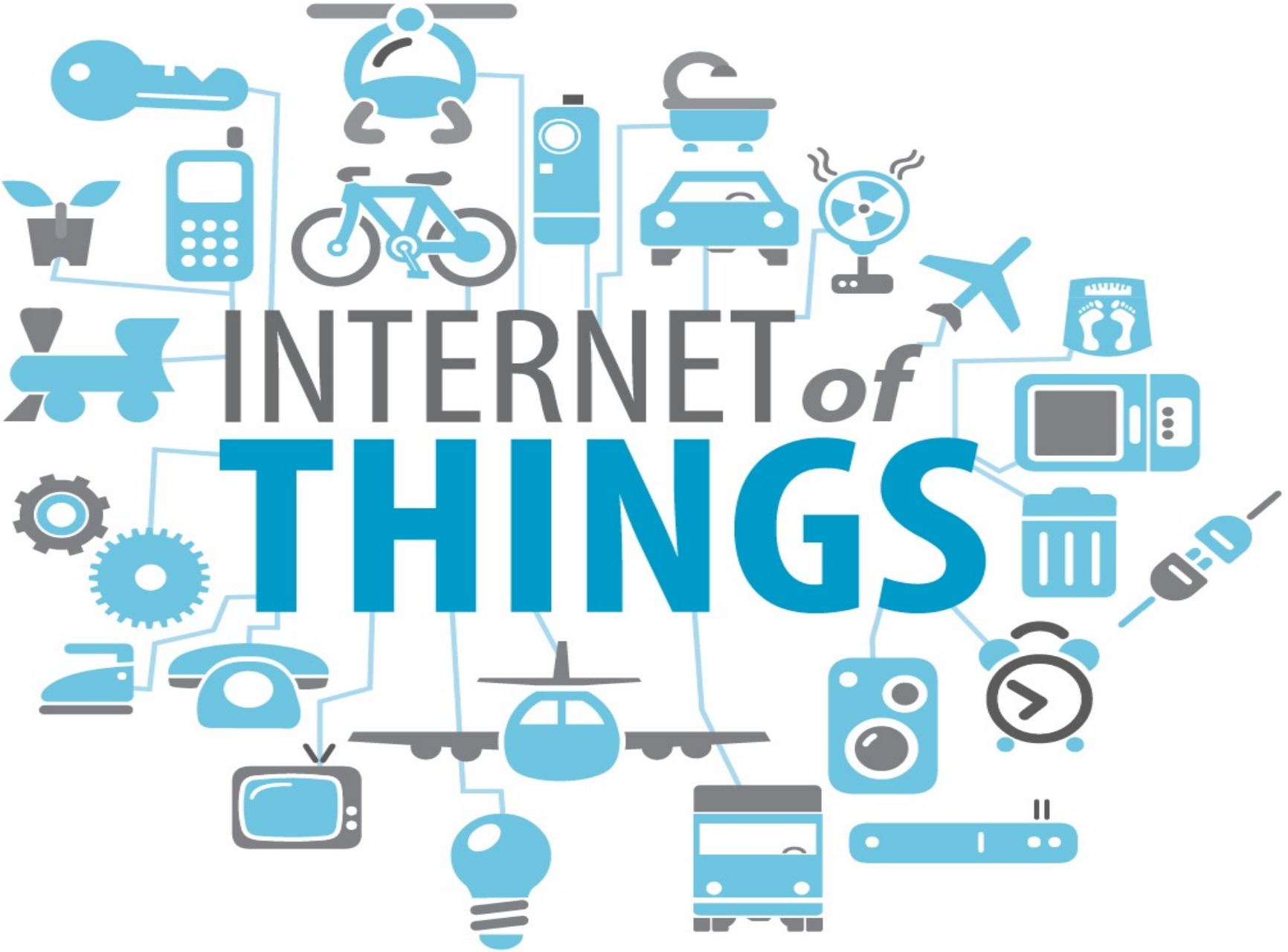
Nosljive naprave



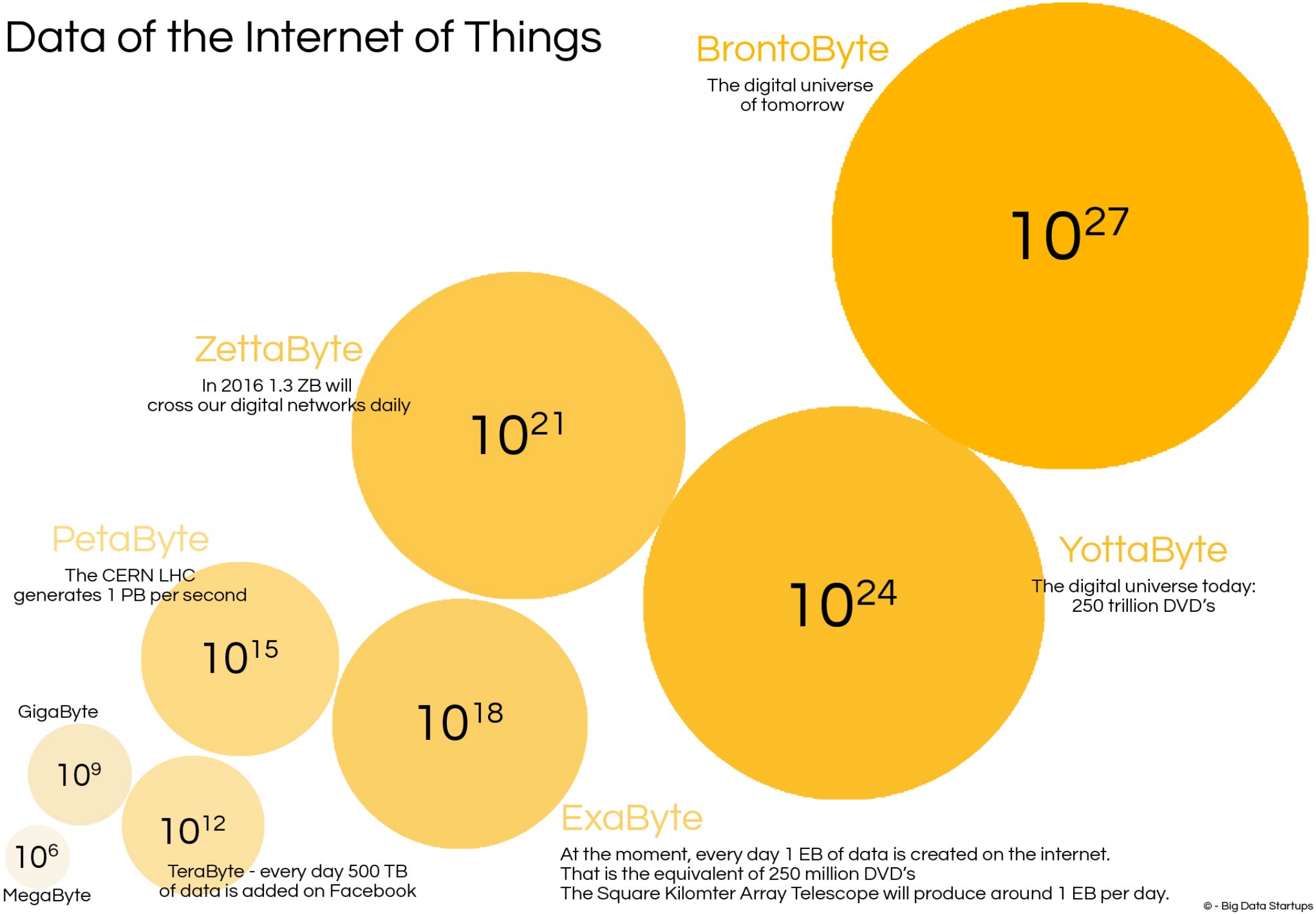







INTERNET of
THINGS

Data of the Internet of Things



© - Big Data Startups

Namen predmeta

- Razumeti, kaj je PB, SUPB
- Se naučiti dela s podatki v SUPB (dostop, indeksiranje, poizvedovanje, ...)
- Se naučiti razvoja PB
- Spoznati povezana področja, kot so noSQL, podatkovna skladišča, XML PB, Xquery...

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



Predmet:
Podatkovne baze

Modul:
Relacijski podatkovni model

Gradivo:
v.2016

Vsebina

- O relacijskem modelu
- Terminologija pri relacijskem modelu
- Matematična definicija relacije
- Relacijska shema
- Lastnosti relacij
- Funkcionalne odvisnosti
- Ključi relacije
- Omejitve nad podatki
- Pogledi

O relacijskem podatkovnem modelu…

- 1970, Edgar Codd.
- Revolucija, nadomesti starejše modele.

- Ključne lastnosti:
  - Enostaven za razumevanje: relacijo si predstavljamo kot tabelo
  - Formalno definiran - matematična osnova;
  - Podatkovno neodvisen - neodvisen od fizičnega shranjevanja podatkov;
  - Močni poizvedovalni jeziki.

Terminologija pri relacijskem modelu...

- Pri relacijskem modelu uporabljamo določeno terminologijo:
  - Relacija
  - Atribut
  - Domena
  - n-terica
  - Stopnja relacija
  - Številnost relacije
  - Relacijska PB

Terminologija pri relacijskem modelu...

- Relacija - dvodimenzionalna tabela s stolpci in vrsticami.
  - Velja za logično strukturo podatkovne baze in ne za fizično.

Ime	Starost (v letih)	Teža (v kg)
Tine	15	50
Meta	20	45
Jure	40	80
Ana	5	10

Relacija

Terminologija pri relacijskem modelu...

- Atribut - poimenovani stolpec relacije.

Ime	Starost (v letih)	Teža (v kg)
Tine	15	50
Meta	20	45
Jure	40	80
Ana	5	10

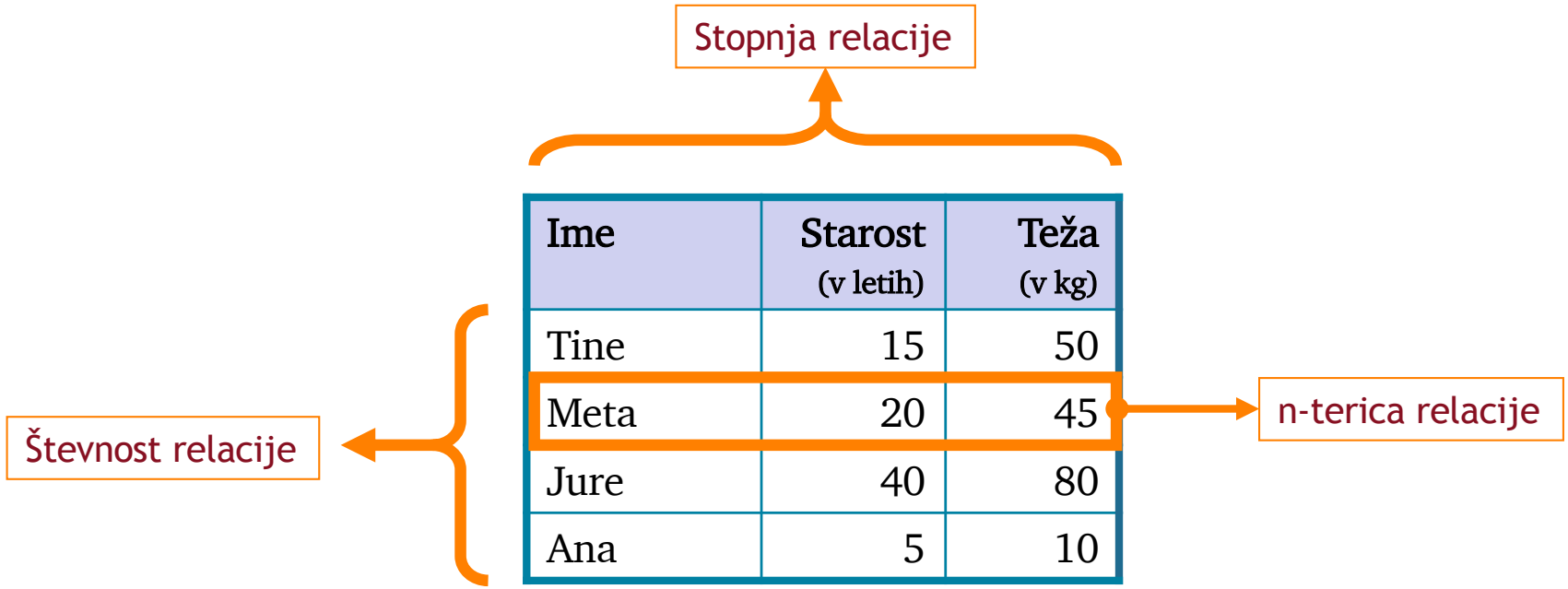
Atribut relacije

Terminologija pri relacijskem modelu...

- Domena - množica dovoljenih vrednosti enega ali več atributov.
- Primeri domen:
  - Barva: {rumena, rdeča, črna, bela, modra, zelena}
  - EMŠO: število, 13 cifer
  - Datum rojstva: datum
  - Priimek: text, max 30 znakov
  - ...

Terminologija pri relacijskem modelu...

- N-terica - ena vrstica v relaciji.
- Številnost relacije - število n-teric relacije.
- Stopnja relacije - število atributov v relaciji.



Terminologija pri relacijskem modelu

- Relacijska podatkovna baza - množica normaliziranih relacij z enoličnimi imeni.

- Kaj so normalizirane relacije? (v nadaljevanju).

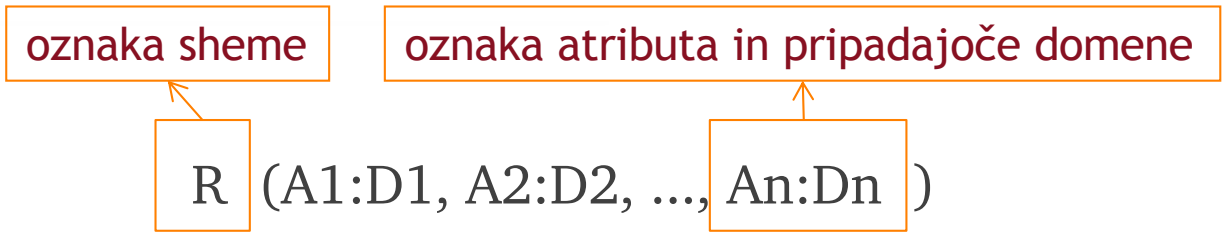
Matematična definicija relacije

- Tabela z n stolpci - matematična relacija stopnje n nad domenami atributov; podmnožica kartezičnega produkta domen atributov.

$$r \subseteq (dom(A1) \times dom(A2) \times \ldots \times dom(An))$$

Relacijska shema...

- Relaciji pripada relacijska shema.



- Relacijska shema predstavlja semantiko ali pomen relacije:
  - Del konceptualnih ali zunanjih shem
  - Semantika ni bogata!

Relacijska shema...

- Primeri relacijske sheme in relacije
  - r (VpŠt, Ime, Priimek, Pošta, Kraj, Spol);
  - Študent (VpŠt: number(8), Ime: char(20), Priimek: char(20), Pošta: number(4), Kraj: char(30), Spol: char(1));

Študent

VpŠt number(8)	**Ime** char(20)	**Priimek** char(20)	**Pošta** number(4)	**Kraj** char(30)	**Spol** char(1) = ('M', 'Ž')
24010632	Marko	Bric	5270	Ajdovščina	M
25089888	Iztok	Jerin	2000	Maribor	M
24135344	Maja	Klepec	1000	Ljubljana	Ž
24090909	Anita	Terčelj	4000	Kranj	Ž

Shema relacije

Lastnosti relacij...

- Ime relacije enolično (znotraj sheme).
- Celica tabele vsebuje natančno eno atomarno vrednost.
- Ime atributa relacije enolično (znotraj relacije)
- Vrednosti atributa iz iste domene.
- n-terica enolična (znotraj relacije).
- Vrstni red atributov v relaciji nepomemben.
- Vrstni red n-teric v relaciji nepomemben.

Primeri

Ali ti dve tabeli predstavljata relaciji?

Ime	Starost (v letih), teža (v kg)
Tine	S15_T50
Meta	S20_T45
Jure	S40_T80
Ana	S5_T10

Oseba	Telefon
Tine Mikuž	1 47 68 819; 041 467 766
Ana Pregelj	5 36 61 234; 5 36 61 235

Funkcionalne odvisnosti...

- Poznamo več vrst odvisnosti:
  - Funkcionalne odvisnosti (functional dependency)
  - Večvrednostne odvisnosti (multivalued dependency)
  - Stične odvisnosti (join dependency)

- Večvrednostne in stične odvisnosti pomembne za razumevanje višjih ravni normalizacije.

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 relacijske 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

Domene atributov so zaradi preglednosti izpuščene.

- Imamo relacijo s shemo

  `Izpit (VpŠt, Priimek, Ime, ŠifraPredmeta, DatumIzpita, 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...

- Relacija je množica med seboj različnih n-teric...
- Katere vrednosti atributov moramo poznati, da se lahko sklicujemo na neko relacijo?
- Ključ relacije: množica atributov, ki enolično določa vsako n-terico.

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

Ključi relacije...

- Poznamo več konceptov, ki jih imenujemo ključ:
  - Kandidat za ključ (*a key candidate*)
  - Primarni ključ (*primary key*)
  - Superključ ali nadključ (*superkey*)
  - Tuji ključ (*foreign key*)

Primeri ključev

ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

Primarni ključ v tabeli Artikel

Primarni ključ v tabeli Račun

RAČUN

Račun	Šifra artikla	Količina
15/05	A10	1
15/05	X12	1

Tuji ključ v tabeli Račun → kaže na primarni ključ v tabeli Artikel

Omejitve nad podatki

- Kakovost podatkov ključnega pomena
- Celovitost/skladnost podatkov - skrbimo s pomočjo omejitev.
- Vrste omejitev:
  - Omejitve domene (*Domain constraints*)
  - Pravila za celovitost podatkov (*Integrity constraints*)
    - Celovitost entitet (*Entity Integrity*)
    - Celovitost povezav (*Referential Integrity*)
  - Številnost (*Multyplicity*)
  - Splošne omejitve (*General constraints*)

Posebnost - oznaka “Null”

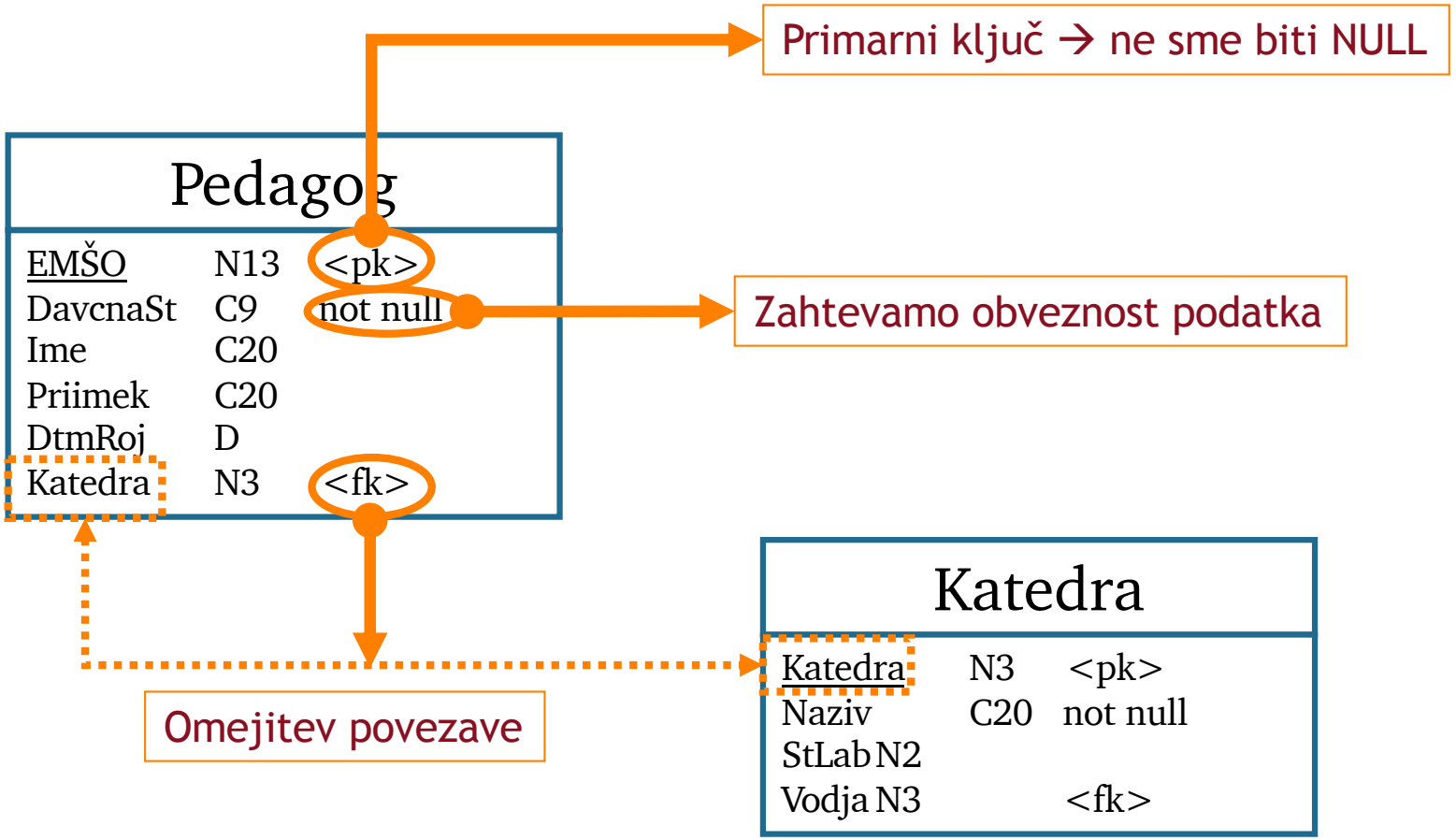
- Oznaka `Null`:
  - Predstavlja vrednost atributa, ki je trenutno neznana ali irelevantna za n-terico.
  - Gre za nepopolne podatke ali podatke pri izjemnih primerih.
  - Predstavlja odsotnost podatka. Ni enako kot `0` ali prazen znak, kar je dejansko vrednost.

- Oznaka `Null` problematična pri implementaciji;
  - relacijski model osnovan na predikatnem računu prvega reda (*Boolean logic*) → edini možni vrednosti `true` in `false`.

Omejitve entitet in povezav

- Omejitev entitete
  - V osnovni relaciji ne sme biti noben atribut, ki je del ključa, enak `Null`.
- Omejitve povezav
  - Če v relaciji obstajajo tuji ključi, potem morajo:
    - (a) njihove vrednosti ustrezati tistim, ki so v obliki ključa zapisane v eni izmed n-teric neke druge ali iste relacije
    - (b) ali pa mora biti tuji ključ v celoti enak Null.
- Splošne omejitve
  - Dodatna pravila, ki jih določi uporabnik ali skrbnik podatkovne baze, ki definirajo ali omejujejo nek vidik področja, za katerega je narejena podatkovna baza.

Primeri omejitev



Pogledi...

- Osnovna relacija (*base relation*)
  - Poimenovana relacija, n-terice fizično shranjene v podatkovni bazi.

- Pogled (*view*)
  - Rezultat ene ali več operacij nad osnovnimi relacijami z namenom pridobitve nove relacije.
  - Navidezna relacija, dinamično kreiranje ob povpraševanju.
  - Pogledi dinamični - spremembe nad osnovnimi relacijami takoj vidne.
  - Spreminjanje pogledov ni vedno možno!

Namen uporabe pogledov

- Mehanizem za zagotavljanje varnosti → enkapsulacija
- Prilagojen dostop za uporabnike – iste relacije vidne na različne načine.
- Poenostavitev kompleksnih operacij nad osnovnimi relacijami.

Primer pogleda

ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

RAČUN

Račun	Šifra artikla	Količina
15/05	A10	1
15/05	X12	1

```
SELECT A.sifra, A.naziv, sum(R.kolicina) AS Prodanih
FROM artikel A, racun R
WHERE A.sifra = R.sifra
GROUP BY A.sifra, A.naziv
```

Šifra	Naziv	Prodanih
A10	Telovadni copati Nike	5
X12	Ženska jakna QuickSilver	1
…		

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



Predmet:
Osnove PB

Modul:
Relacijska algebra

Gradivo:
v.2019

O relacijskih poizvedovalnih jezikih

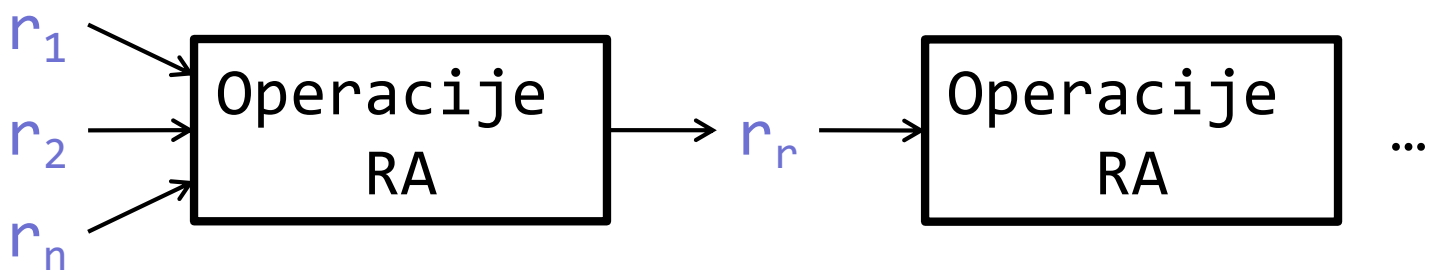
- Relacijska algebra in relacijski račun formalna poizvedovalna jezika...
  - Relacijska algebra visoko-nivojski postopkovni jezik,
  - Relacijski račun nepostopkovni ali deklarativni jezik.

- Formalno ekvivalentna.

- Relacijsko popolni jeziki

Relacijska algebra…

- Vhod in izhod relacijske algebre so relacije.
- Vhodne relacije se ne spremenijo!
- Možno gnezdenje izrazov.



Operacije relacijske algebre…

- Relacijska algebra vsebuje pet osnovnih operacij:
  - Selekcija,
  - Projekcija,
  - Kartezijski produkt
  - Unija in
  - Razlika.
- S temi operacijami se izvede večina poizvedb.
- Možne so tudi izpeljane operacije:
  - Stik,
  - Presek in
  - količnik.

Selekcija

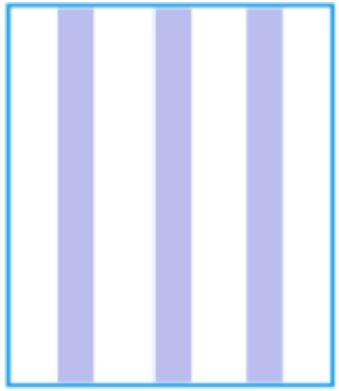
$$\sigma_{predikat}(R)$$



- Deluje na enojni relaciji R; vrne relacijo, ki vsebuje samo tiste n-terice (vrstice) iz relacije R, ki zadoščajo določenemu pogoju (predikat).

Projekcija

$$\Pi_{s1, \ldots, sn} (R)$$



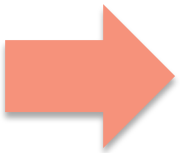
- Deluje na enojni relaciji R; vrne relacijo, ki vsebuje samo tiste atribute (stolpce), ki so določeni s seznamom.
- Projekcija eliminira duplikate.

Primer selekcije

- Izpiši vse artikle z zalogo manjšo od 2
- $\sigma_{\text{zaloga}<2}(R)$

R=ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0



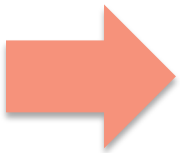
Šifra	Naziv	Zaloga
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

Primer projekcije

- Izpiši nazive in zalogo vseh artiklov
- $\Pi_{\text{naziv, zaloga}}(R)$

R=ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0



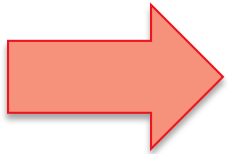
Naziv	Zaloga
Telovadni copati Nike	10
Trenerka Bali	4
Moška jakna QuickSilver	1
Ženska jakna QuickSilver	0

Primer selekcije in projekcije

- Izpiši šifre artiklov z zalogo manjšo od 2
- $\Pi_{\text{šifra}}(\sigma_{\text{zaloga}<2}(R))$

R=ARTIKEL

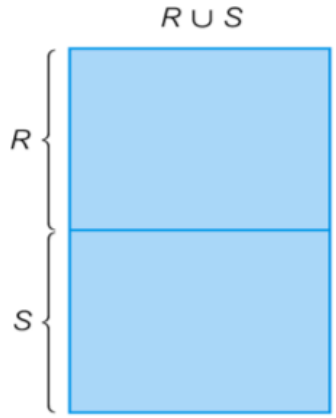
Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0



Šifra
BC80
X12

Unija

$$R \cup S$$



- Unija dveh relacij R in S je relacija, ki vsebuje vse n-terice (vrstice) relacije R in relacije S.
- Operacija eliminira duplikate.

- R in S se morata ujemati po atributih in podatkovnih tipih.
  - ~~R(ime: text, EMSO: num(13)) ∪ S (priimek: text, starost: int)~~

Primer unije

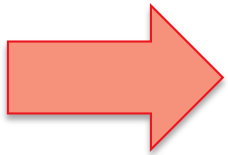
- Izpiši vsa mesta, kjer se nahajajo skladišča ali stranke
- $\Pi_{kraj}(R) \cup \Pi_{kraj}(S)$

R=ARTIKEL

Šifra	Naziv	Kraj
A10	Telovadni copati Nike	LJ
A12	Trenerka Bali	MB
BC80	Moška jakna QuickSilver	LJ
X12	Ženska jakna QuickSilver	GO

S=STRANKA

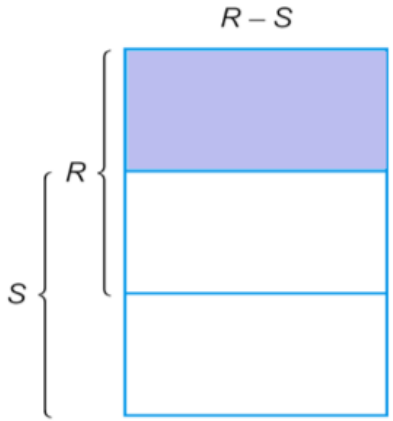
Šifra	Naziv	Kraj
S1	Novak Janez	LJ
S2	Krašna Miha	CE
S3	Bele Simon	PO
S4	Šuc Vilma	GO



Kraj
LJ
CE
PO
GO
MB

Razlika

$$R - S$$

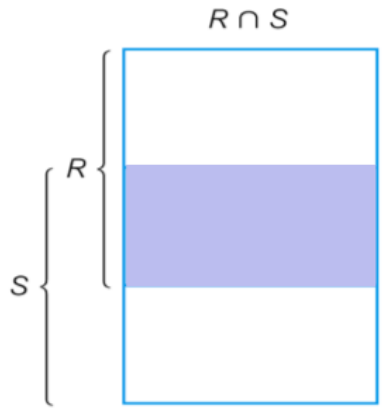


- Razlika med relacijama R in S (R-S) vrne relacijo, ki vsebuje samo tiste n-terice (vrstice), ki so v R in jih ni v S.

- R in S se morata ujemati po atributih in njihovih podatkovnih tipih.
  - R(ime: text, EMSO: int) - S (priimek: text, starost: int) ✓

Presek

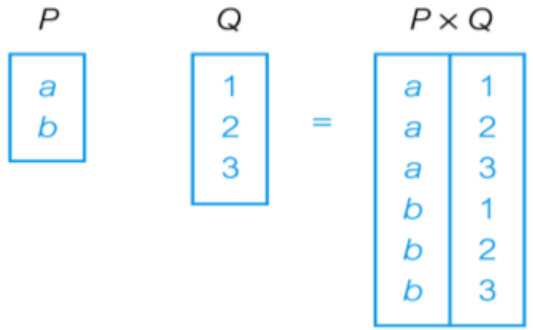
$$R \cap S$$



- Presek med relacijama R in S (R∩S) vrne relacijo, ki vsebuje tiste n-terice (vrstice), ki se nahajajo v obeh relacijah.

- R in S se morata ujemati po atributih in njihovih podatkovnih tipih.
  - ~~R(ime: text, EMSO: num(13)) S (priimek: text, starost: int)~~

Kartezijski produkt



$$R \times S$$

- Kartezijski produkt relacij R in S vrne vse možne kombinacije med n-tericami (vrsticami) relacije R in n-tericami (vrsticami) relacije S.

Primer kartezijskega produkta

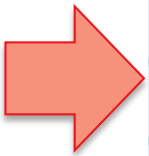
- Izpiši šifre, nazive in količino artiklov, ki se pojavljajo na računih
- $(\Pi_{\text{šifra,naziv}}(R)) \times (\Pi_{\text{šifra artikla, količina}}(S))$

R=ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

S=RAČUN

Račun	Šifra artikla	Količina
15/05	A10	1
15/05	X12	1



Šifra	Naziv	Šifra artikla	Količina
A10	Telovadni copati Nike	A10	1
A10	Telovadni copati Nike	X12	1
A12	Trenerka Bali	A10	1
A12	Trenerka Bali	X12	1
BC80	Moška jakna QuickSilver	A10	1
BC80	Moška jakna QuickSilver	X12	1
X12	Ženska jakna QuickSilver	A10	1
X12	Ženska jakna QuickSilver	X12	1

Stične operacije…

- Kartezijski produkt s selekcijo združimo v eno operacijo, ki jo imenujemo stik.

- Stik je ena najbolj časovno kompleksnih operacij s stališča implementacije v relacijskih SUPB; eden ključnih "krivcev" za probleme z učinkovitostjo.

Stične operacije…

- Obstaja več vrst stičnih operacij:
  - Stik Theta (Theta join)
  - Ekvistik (Equijoin) poseben primer stika Theta)
  - Naravni stik (Natural join)
  - Odprti stik (Outer join)
  - Delni stik (Semijoin)

Stik Theta (θ stik)…

$$R \bowtie_F S$$

- Stik Theta med relacijama R in S vrne n-terice (vrstice), ki zadoščajo predikatu F kartezijskega produkta R in S.
- Predikat F je oblike `R.ai θ S.bi`, kjer je θ primerjalna operacija ($<$, $\leq$, $>$, $\geq$, $=$, $\neq$).

Če se omejimo na enakost, dobimo EKVISTIK!

Primer ekvistika

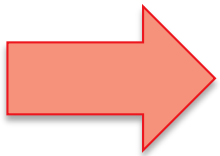
- Izpiši šifre, nazive in količino artiklov, ki se pojavljajo na računih, kjer je šifra artikla na računu enaka šifri artikla v artiklu
- $(\Pi_{\text{šifra, naziv}}(R)) \bowtie_{R.\text{šifra} = S.\text{šifra artikla}} (\Pi_{\text{šifra artikla, količina}}(S))$

R=ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

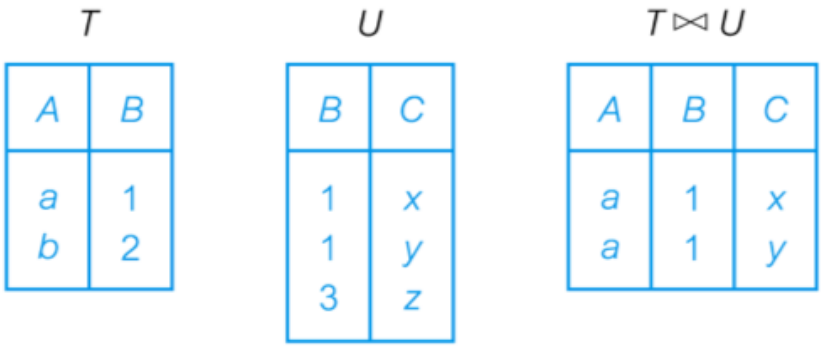
S=RAČUN

Račun	Šifra artikla	Količina
15/05	A10	1
15/05	X12	1



Šifra	Naziv	Šifra artikla	Količina
A10	Telovadni copati Nike	A10	1
X12	Ženska jakna QuickSilver	X12	1

Naravni stik



$R \bowtie S$

- Naravni stik relacij R in S je posebna vrsta ekvistika prek skupnih atributov relacij R in S.

- Pri vsakem stiku se vzame le en primerek skupnega atributa.

Primer naravnega stika

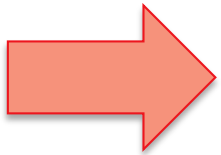
- Izpiši šifre, nazive in količino artiklov, ki se pojavljajo na računih, kjer je šifra artikla na računu enaka šifri artikla v artiklu
- $(\Pi_{\text{šifra, naziv}}(R)) \bowtie (\Pi_{\text{šifra, količina}}(S))$

R=ARTIKEL

Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

S=RAČUN

Račun	Šifra	Količina
15/05	A10	1
15/05	X12	1



Šifra	Naziv	Količina
A10	Telovadni copati Nike	1
X12	Ženska jakna QuickSilver	1

Odprti stik

- Odprti stik omogoča, da prikažemo tudi n-terice (vrstice), ki nimajo vrednosti v stičnem atributu (stolpcu).
- Obstajata levo odprti in desno odprti stik.

$R \rtimes S$

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \rtimes_C U$

A	B	C
a	1	x
a	1	y
b	2	

- Levo odprti stik med relacijama R in S je stik, kjer so n-terice relacije R, ki nimajo para v S z enakim stičnim atributom, vključene v rezultat.

Primer odprtega stika

- Izpiši osebe in njihova začasna prebivališča
- $(\Pi_{\text{Priimek in ime, PTT}}(R)) \rtimes S$

R=OSEBA

ID	Priimek in ime	PTT
1	Kante Janez	5270
2	Tratnik Jože	5000
3	Mali Mihael	
4	Brecelj Jana	1000

S=KRAJ

PTT	Naziv
1000	Ljubljana
5000	Nova Gorica
5270	Ajdovščina



Priimek in ime	PTT	Naziv zač. preb.
Kante Janez	5270	Ajdovščina
Tratnik Jože	5000	Nova Gorica
Mali Mihael		
Brecelj Jana	1000	Ljubljana

Delni stik

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \triangleright_B U$

A	B
a	1

$$R \triangleright_F S$$

- Delni stik predstavlja relacijo, ki vsebuje tiste n-terice (vrstice) relacije R, ki nastopajo v stiku z relacijo S po stolpcu F.

Primer delnega stika

- Izpiši vse podrobnosti o artiklih, ki se nahajajo v skladišču v Ljubljani
- $R \triangleright_{R.skladišče = S.šifra} (\sigma_{šifra = 'LJ'} (S))$

R=ARTIKEL

Šifra	Naziv	Skladišče	Dobavitelj	Zaloga
A10	Telovadni copati Nike	LJ	Nike	10
BC80	Moška jakna QuickSilver	LJ	Karma	1
X12	Ženska jakna QuickSilver	GO	Karma	0

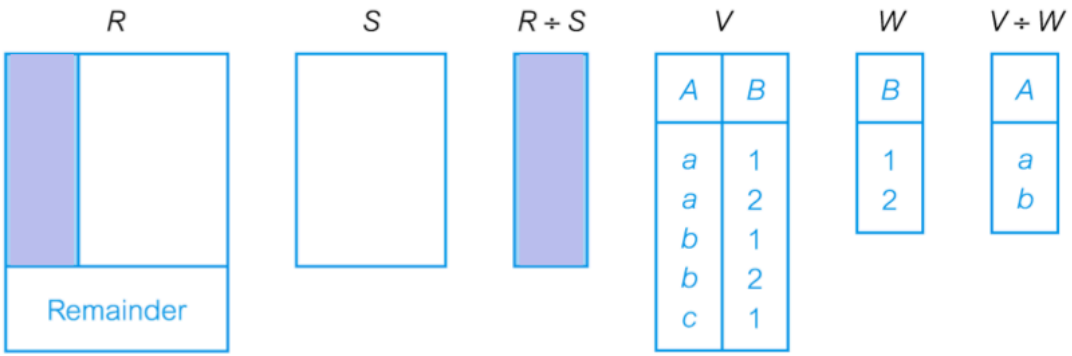
S=SKLADIŠČE

Šifra	Kraj
LJ	Ljubljana, Tržaška 33
MB	Maribor, Prešernov trg 2
GO	Nova Gorica, Cankarjeva 2



Šifra	Naziv	Skladišče	Dobavitelj	Zaloga
A10	Telovadni copati Nike	LJ	Nike	10
BC80	Moška jakna QuickSilver	LJ	Karma	1

Količnik



R / S

- Količnik med relacijama R in S vrne relacijo z atributi C, ki jo sestavljajo n-terice (vrstice) iz R, ki ustrezajo kombinaciji vsake n-terice v S.

Primer količnika

- Izpiši vse kupce, ki so kupili vse izdelke dobavitelja Karma.
- $(\Pi_{\text{šifra,kupec}}(R)) \;/\; (\Pi_{\text{šifra artikla}}(\sigma_{\text{dobavitelj = 'Karma'}}(S)))$

$\Pi_{\text{šifra, kupec}}(R)$

Šifra	Kupec
A10	K1
A12	K1
BC80	K2
X12	K3
A10	K3
BC80	K3
BC80	K4
X12	K4
A12	K5

$\Pi_{\text{šifra artikla}}(\sigma_{\text{dobavitelj = 'Karma'}}(S))$

Šifra
BC80
X12



Kupec
K3
K4

Agregatne operacije

$$\Gamma_{AL}(R)$$

- Agregat Γ_{AL} aplicira seznam agregatnih funkcij AL na relaciji R in vrne takó agregirano relacijo.

- AL vsebuje enega ali več parov (<agregatna funkcija>, <atribut>).

- Osnovne agregatne funkcije so COUNT, SUM, AVG, MIN in MAX.

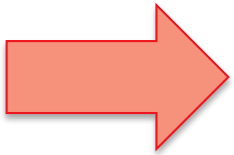
Primer agregatnih operacij

- Kakšna je povprečna cena artiklov v skladišču?
- $\rho_R(\text{AvgCena})\ \boldsymbol{\Gamma}_{\text{AVG Cena}}\ (\sigma_{\text{Zaloga} > 0}\ (R))$

Operacija ρ_R(ime) Op poimenuje atribut, ki ga dobimo z operacijami Op nad relacijo R, z imenom ime

R=ARTIKEL

Šifra	Naziv	Skladišče	Cena	Zaloga
A10	Telovadni copati Nike	LJ	17.990	10
A12	Trenerka Bali	MB	6.750	4
BC80	Moška jakna QuickSilver	LJ	14.290	1
X12	Ženska jakna QuickSilver	GO	14.290	0



AvgCena
13.010

Združevalne operacije

$$_{GA}\Gamma_{AL}(R)$$

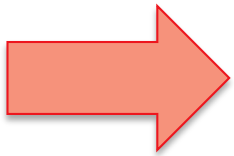
- Združevalne operacije združijo n-terice relacije R tako, da jih grupirajo po atributih GA.
- Relacija, ki jo tako pridobimo, vsebuje atribute GA ter vrednosti, ki jih pridobimo z apliciranjem agregatov iz AL nad vsako posamezno skupino iz GA.

Primer združevalnih operacij

- Izpiši vrednosti posameznih računov?
- ρ_R(račun,SkupnaCena) $\Gamma_{\text{SUM cena*količina}}$ (R)

R=RAČUN

ŠifraR	Šifra artikla	Količina	Cena
1/05	X12	1	12.000
1/05	XC80	2	5.000
1/05	Z55	3	4.500
2/05	X12	2	12.000
2/05	Y15	3	550
3/05	HH8	2	50.000
3/05	X12	1	12.000
3/05	XXZ4	1	990



račun	SkupnaCena
1/05	35.500
2/05	25.650
3/05	112.990

Vaja…

- Napiši relacijsko shemo za hranjenje podatkov o hotelski verigi Across.

Opis: hotelska veriga Across ima v lasti večje število hotelov, lociranih po celem svetu. Za vsak hotel bi želeli vedeti vsaj:

- kakšen naziv ima,
- kje je lociran,
- koliko sob ima na voljo,
- kakšen je tip posamezne sobe (eno posteljna, dvoposteljna…),
- kakšna je cena posamezne sobe,
- imena, priimke in naslove gostov, ki so oziroma so bili v posameznem hotelu in sobi,
- podatke o rezervacijah (poteklih in trenutno veljavnih).

Vaja…

- Imamo naslednje relacije:
  - Hotel (hotelNo, hotelName, address)
  - Room (roomNo, hotelNo, type, price)
  - Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)
  - Guest (guestNo, guestName, guestAddress)

Kaj so ključi posameznih relacij?

- Opiši relacije, ki jih pridobimo z naslednjimi operacijami relacijske algebre:
  - $\sigma_{\text{Hotel.hotelNo}=\text{Room.hotelNo}}(\text{Hotel} \times \text{Room}))$
  - $\Pi_{\text{hotelName}}(\text{Hotel} \bowtie_{\text{Hotel.hotelNo} = \text{Room.hotelNo}} (\sigma_{\text{price} > 50}(\text{Room})))$
  - $\text{Guest} \bowtie (\sigma_{\text{dateTo} \geq \text{'1.1.2002'}}(\text{Booking}))$
  - $\Pi_{\text{guestName}}(\text{Booking} \bowtie_{\text{Booking.guestNo}=\text{Guest.guestNo}} \text{Guest}) \; / \; \Pi_{\text{hotelNo}}(\sigma_{\text{address} = \text{'London'}}(\text{Hotel}))$

Vaja

- Z uporabo relacijske algebre naredite pogled, ki bo vključeval vse podatke o sobah iz Grosvenor hotela, razen podatka o ceni.

- $\Pi_{\text{roomNo, hotelNo, type}}(\text{Room} \bowtie_{\text{hotelNo}} (\sigma_{\text{hotelName}=\text{'Grosvenor Hotel'}} (\text{Hotel})))$

- Kaj je prednost takšnega pogleda?

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



Predmet:
Osnove PB

Modul:
Relacijski račun

Gradivo:
v.2019

Relacijski poizvedovalni jeziki

- Relacijska algebra in relacijski račun formalna poizvedovalna jezika…
  - Relacijska algebra visoko-nivojski postopkovni jezik,
  - Relacijski račun nepostopkovni ali deklarativni jezik.

- Formalno ekvivalentna.

- Relacijsko popolni jeziki

Deklarativno poizvedovanje

- Relacijski račun je deklarativni poizvedovalni jezik.
  - V nasprotju z relacijsko algebro, kjer (implicitno) določimo postopek, s katerim pridemo do rezultata, pri relacijskem računu določimo le, kaj nas zanima.

- Temelji na simbolični logiki imenovani predikatni račun prvega reda.
  - V naravni jeziku so stavki sestavljeni iz besed, ki sledijo sintaksi. Besede, ki označujejo trditve, imenujemo *predikati* in besede o katerih trditve držijo, imenujemo *konstante*.
    - Objekt a ima lastnost P.
    - Objekta a_1, a_2 sta v odnosu Q.

Stavki v predikatnem računu

- Stavki v predikatnem računu so sestavljeni iz predikatov in konstant. Predikat je logična funkcija z argumenti.

- Primer:
  - Marija je ženska.
  - Janez je moški.
  - Marija in Janez sta sestra in brat.



  - Ženska(marija)
  - Moški(janez)
  - SestraBrat(marija,janez)

Spremenljivke in kvantifikatorji

- Predikati imajo lahko spremenljivke kot argumente, katerih vrednosti so lahko kvantificirane.

- Primer:
  - Vsakdo je bodisi moški ali ženska.
  - Moški ni ženska.

$\forall x.\ Moški(x) \vee Ženska(x)$

$\forall x.\ Moški(x) \Rightarrow \neg Ženska(x)$



Funkcije

- V predikatnem računu so domene objektov lahko določene s funkcijami, ki so aplicirane na objekte.

- Primer:
  - Oče osebe je moški

$\forall x.\ \text{Moški}(\text{Oče}(x))$

Funkcija

Sintaksa predikatnega računa

- Če predikat vsebuje spremenljivko (npr. `x je študent`), mora za x obstajati domena vrednosti.
  - Za nekatere vrednosti iz domene je trditev resnična za druge neresnična.

- Sintaksa: če je `P` predikat, potem lahko zapišemo množico vseh `x`, za katere je `P` resničen, takole:

  `{x | P(x)}`

- Predikate lahko povezujemo z logičnimi operatorji: IN ($\wedge$), ALI ($\vee$), NEGACIJA ($\neg$)

Vrste relacijskega računa

- V povezavi s podatkovnimi bazami poznamo dve vrsti relacijskega računa:
  - N-terični relacijski račun (*Tuple Relational Calculus*)
    - Spremenljivke se nanašajo na n-terice ali vrstice relacije.
  - Domenski relacijski račun (*Domain Relational Calculus*).
    - Spremenljivke se nanašajo na domene atributov.

N-terični relacijski račun

- Temelji na uporabi n-teričnih spremenljivk.
  - N-terična spremenljivka je spremenljivka, katere domena je določena z relacijo, t.j. spremenljivka, katere dovoljene vrednosti so n-terice relacije.

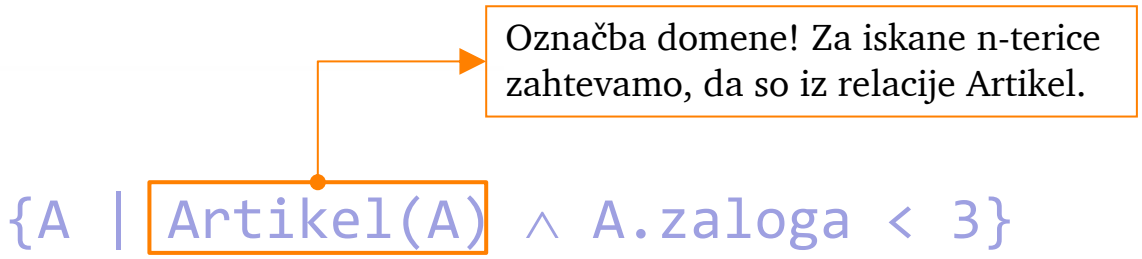
- Splošna oblika zapisa n-teričnega računa je:

```
{t | COND(t)}
```

  - kjer `t` prestavlja n-terično spremenljivko, `COND(t)` pa nabor pogojev, ki naj jih n-terice, ki jih iščemo, izpolnjujejo.

Primeri

- Poišči vse podatke o artiklih, ki imajo kritično zalogo manjšo od tri (zaloga < 3):



- Če nas zanima samo določen atribut (npr. naziv artikla), zapišemo:

$$\{A.naziv \mid Artikel(A) \wedge A.zaloga < 3\}$$

Uporaba kvantifikatorjev

- Na koliko primerkov se predikat nanaša, določimo s pomočjo kvantifikatorjev.
- Obstajata dva kvantifikatorja:
  - Eksistencialni kvantifikator ∃ (beremo 'obstaja')
  - Univerzalni kvantifikator ∀ (beremo 'za vse')

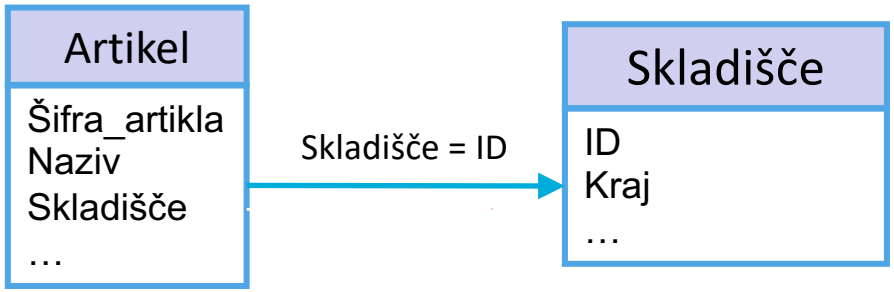
- N-terične spremenljivke, ki so kvantificirane s kvantifikatorjema ∀ ali ∃, imenujemo vezane spremenljivke, ostale pa proste spremenljivke.

Eksistencialni kvantifikator

- Eksistencialni kvantifikator uporabimo v izrazih, ko želimo povedati, da mora obstajati vsaj en primerek, za katerega je predikat resničen.

- Primer: iščemo zapise iz relacije `Artikel`, ki se nahajajo v skladišču v Ljubljani:

$$\{A \mid Artikel(A) \wedge (\exists S)(Skladišče(S) \wedge (A.skladišče = S.ID) \wedge (S.kraj = \text{'Ljubljana'}))\}$$

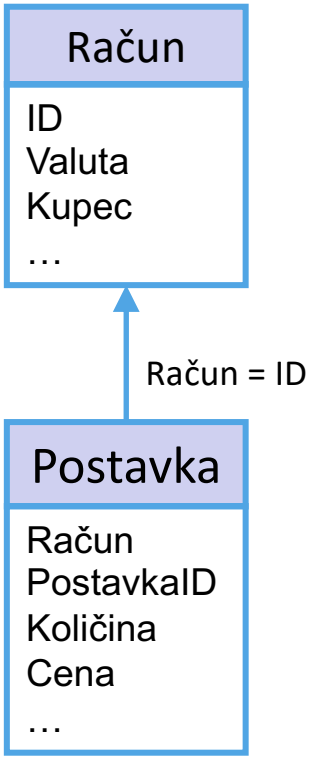


Univerzalni kvantifikator

- Univerzalni kvantifikator uporabimo v izrazih, ki se nanašajo na vse primerke…
- Primer: iščemo račune, ki imajo ceno vsake postavke večjo od 1.000 EUR

$$\{R \mid Račun(R) \wedge (\forall P)(Postavka(P) \wedge (R.ID = P.račun) \wedge (P.cena > 1000))\}$$

Ali lahko isti rezultat pridobimo z uporabo negacije?



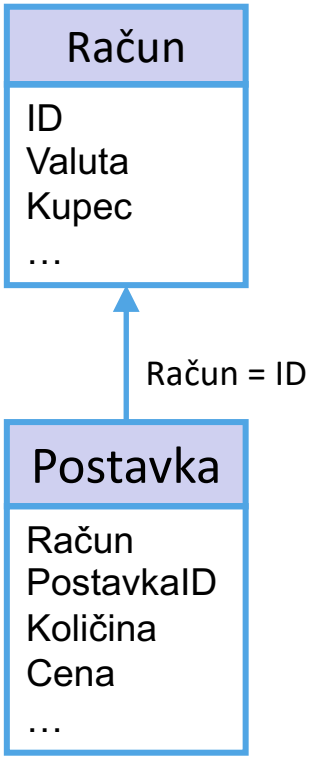
Univerzalni kvantifikator

- Univerzalni kvantifikator uporabimo v izrazih, ki se nanašajo na vse primerke…
- Primer: iščemo račune, ki imajo ceno vsake postavke večjo od 1.000 EUR

$$\{R \mid Račun(R) \wedge (\forall P)((Postavka(P) \wedge (R.ID = P.račun)) \Rightarrow (P.cena > 1000))\}$$

$$\{R \mid Račun(R) \wedge (\forall P)(\neg(Postavka(P) \wedge (R.ID = P.račun)) \vee (P.cena > 1000))\}$$

$$\{R \mid Račun(R) \wedge \neg(\exists P)(Postavka(P) \wedge (R.ID = P.račun) \wedge (P.cena <= 1000)\}$$



Proste in vezane spremenljivke

- Proste spremenljivke v izrazih relacijskega računa so lahko samo spremenljivke, ki so definirane oz. se nahajajo na levi strani znaka | v izrazu.

- Primer:

$$\{A.naziv, A.cena \mid Artikel(A) \wedge (\exists S)(Skladišče(S) \wedge (A.skladišče = S.ID) \wedge S.kraj = \text{‘Ljubljana’})\}$$

  - A je prosta spremenljivka, S pa vezana!

Dobri izrazi

- Izrazi v relacijskem računu morajo biti dobri, t.j. nedvoumni in smiselni.
- Splošna oblika dobrega izraza je naslednja:

  `{S1.a1, S2.a2,..., Sn.an | F(S1, S2,..., Sm)}  m≥n`

- kjer je pomen naslednji:
  - `S1, S2,..., Sm` so n-terične spremenljivke
  - `Sj.ai` so atributi relacije, ki je domena n-terične spremenljivke `Si`
  - `F` je formula.

Dobro definirana formula

- Dobro definirana formula je sestavljena iz naslednjih atomov:
  - `R(Si)`, kjer je `Si` n-terična spremenljivka in `R` relacija.
  - `Si.a1 θ Sj.a2`, kjer sta `Si` in `Sj` n-terični spremenljivki, `a1` atributi relacije, ki je domena `Si`, `a2` atributi relacije, ki je domena `Sj`, `θ` pa primerjalni operator (`<`, `≤`, `>`, `≥`, `=`, `≠`).
  - `Si.a1 θ c`, kjer je `Si` n-terična spremenljivka, `a1` atributi relacije, ki je domena `Si`, `θ` primerjalni operator, `c` pa konstanta iz domene atributov `a1`.

Pravila gradnje formul

- Formule gradimo rekurzivno iz atomov.

- Upoštevamo naslednja pravila:
  - Atom je že sam formula
  - Če sta F1 in F2 formuli, so formule tudi konjunkcija F1∧F2, disjunkcija F1∨F2, in negacija ¬F1
  - Če je F formula s prosto spremenljivko X, potem sta (∃X)(F) in (∀X)(F) tudi formuli.

Primeri

- Imamo naslednje relacije:

```
Hotel   (hotelNo, hotelName, address)
Room    (roomNo, #hotelNo, type, price, free)
Booking (#hotelNo, #guestNo, dateFrom, dateTo, #roomNo)
Guest   (guestNo, guestName, guestAddress)
```

- Izpiši nazive hotelov, ki se nahajajo v Ljubljani.

- Izpiši nazive hotelov, ki imajo prosto vsaj eno dvoposteljno sobo (type=2).

Primeri

- Izpiši nazive hotelov, ki so trenutno brez gostov.

Varni izrazi

- V relacijskem računu je možno zapisati stavke, ki vračajo neskončne množice.
- Varnost izrazov dosežemo z omejitvijo, da morajo biti vse vrednosti, ki se pojavijo v rezultatu, vrednosti iz domene izraza `E` `(dom(E))`.
  - `dom(E)`: vrednosti, ki se eksplicitno pojavijo v izrazu `E` kot konstante ali pa kot zapis ene od relacij, ki nastopajo v `E`.

- Primer nevarnega izraza:

```
{A | ¬(Artikel(A)) }
```

Sintaksa domenskega računa

- Splošna oblika izraza v domenskem relacijskem računu je:

  `{d1, d2,..., dn | F(d1, d2,..., dm)}`

- kjer je:

  `d1, d2,..., dm` množica domenskih spremenljivk in

  `F(d1, d2,..., dm)` formula.

Formule

- Formula `F` je sestavljena iz atomov oblike:
  - `R(d1, d2,..., dn)`, kjer je `R` relacija, di pa domenske spremenljivke
  - `di θ dj`, kjer sta di in dj domenski spremenljivki, `θ` pa primerjalni operator (`<`, `≤`, `>`, `≥`, `=`, `≠`)
  - `di θ c`, kjer je `di` domenska spremenljivka, `θ` primerjalni operator, `c` pa konstanta iz domene `di`

Gradnja izrazov

- Formule gradimo rekurzivno iz atomov.

- Upoštevamo naslednja pravila:
  - Atom je že sam formula
  - Če sta F1 in F2 formuli, so formule tudi konjunkcija F1∧F2, disjunkcija F1∨F2, in negacija ¬F1
  - Če je F formula z nevezano domensko spremenljivko X, potem sta (∃X)(F) in (∀X)(F) tudi formuli.

Primeri

- Imamo naslednje relacije:

```
Hotel   (hotelNo, hotelName, address)
Room    (roomNo, #hotelNo, type, price, free)
Booking (#hotelNo, #guestNo, dateFrom, dateTo, #roomNo)
Guest   (guestNo, guestName, guestAddress)
```

- Izpiši imena hotelov v Ljubljani.

Primeri

- Izpiši nazive hotelov, ki imajo prosto vsaj eno dvoposteljno sobo (type=2)

N-terični in domenski relacijski račun

- N-terični relacijski račun *(Edgar F. Codd, 1972)*:
  - uporabljamo spremenljivke, katerih zaloga vrednosti so relacije.
  - SQL temelji na n-teričnem relacijskem računu!
  - IBM (raziskovalni laboratorij v San Jose, California)

- Domenski relacijski račun *(M. Lacroix in A. Pirotte, 1977)*:
  - uporabljamo spremenljivke, katerih zaloga vrednosti so domene atributov.
  - QBE temelji na domenskem relacijskem računu!
  - IBM (raziskovalni laboratorij v Yorktown Heights, New York).

Primerjava jezikov

- N-terični relacijski račun

```
{H.hotelName | Hotel(H) ∧ (∃R)(Room(R) ∧
H.hotelNo=R.hotelNo ∧ R.type=2 ∧ R.free = true)}
```

- Domenski relacijski račun

```
{hName | ∃hNo, hName, hAddress, rNo, rPrice:
(Hotel (hNo, hName, hAddress) ∧
(Room(rNo, hNo, 2, rPrice, true) ) }
```

- Relacijska algebra

$$\pi_{H.hotelName}(\sigma_{R.type=2 \wedge R.free=true}(H \bowtie_{H.hotelNo=R.hotelNo} R))$$

- SQL

```
SELECT h.hotelName FROM Hotel h, Room r
WHERE
    h.hotelNo=r.hotelNo AND r.type=2 AND r.free=true
```

Moč formalnih jezikov

- Ob uporabi varnih izrazov so si relacijska algebra, n-terični relacijski račun in domenski relacijski račun po moči enakovredni:
  - Kar lahko izrazimo v relacijski algebri, lahko izrazimo tudi v n-teričnem ali domenskem relacijskem računu.
  - Vsak varen izraz v relacijskem računu lahko zapišemo tudi z relacijsko algebro.

- Vsak jezik, s katerim lahko pridobimo relacije, ki jih je moč pridobiti z relacijskim računom, je relacijsko popoln (*relationally complete*).

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Predmet:
Osnove podatkovnih baz

Modul:
Poizvedovalni jezik SQL

Gradivo:
v.2017

Vsebina

- Splošno o SQL ter zgodovina razvoja
- Sintaksa SQL
  - Ukazi iz skupine DML: `SELECT, INSERT, UPDATE, DELETE`
  - Ukazi iz skupine DDL: `CREATE, ALTER, SET TRANSACTION,...`
  - Sprožilci, funkcije in shranjeni programi
- Dostopanje do MySQL iz programskih jezikov

Uvod v SQL…

- SQL sestavljata dve skupini ukazov:
  - Skupina ukazov za DDL (*Data Definition Language*) za opredelitev strukture podatkovne baze in
  - Skupina ukazov za DML (*Data Manipulation Language*) za poizvedovanje in ažuriranje podatkov.

```
create table oseba (
    EMŠO number (13),
    ime char(20),
    priimek char(20)
)
```

```
select ime, priimek
from oseba
where ime = 'Tine'
order by priimek
```

- Do 1999 SQL brez ukazov za kontrolni tok - potrebno obvladati s programskim jezikom ali interaktivno.

Uvod v SQL…

- Lastnosti SQL:
  - Enostaven;
  - Nepostopkoven (kaj in ne kako);
  - Uporaben v okviru številnih vlog: skrbniki PB, vodstvo, razvijalci informacijskih rešitev, končni uporabniki…;
  - SQL de-facto in tudi uradno standardni jezik za delo z relacijskimi podatkovnimi bazami.

Zgodovina SQL

- V 1970h IBM razvija sistem *System R*, ki bo temeljil na relacijskem modelu.
- 1974 – D. Chamberlin in F. Boyce (*IBM San Jose Laboratory*) definirata jezik 'Structured English Query Language' (SEQUEL).
  - SEQUEL se kasneje preimenuje v SQL
- V poznih 1970h – *Relational Software* (danes *Oracle*) razvije svoj SUPB, ki temelji na relacijskem modelu in implementira SQL.
- Poleti 1979 – Oracle izda prvo komercialno različico SQL; nekaj tednov pred IBM-ovo implementacijo *System/38*

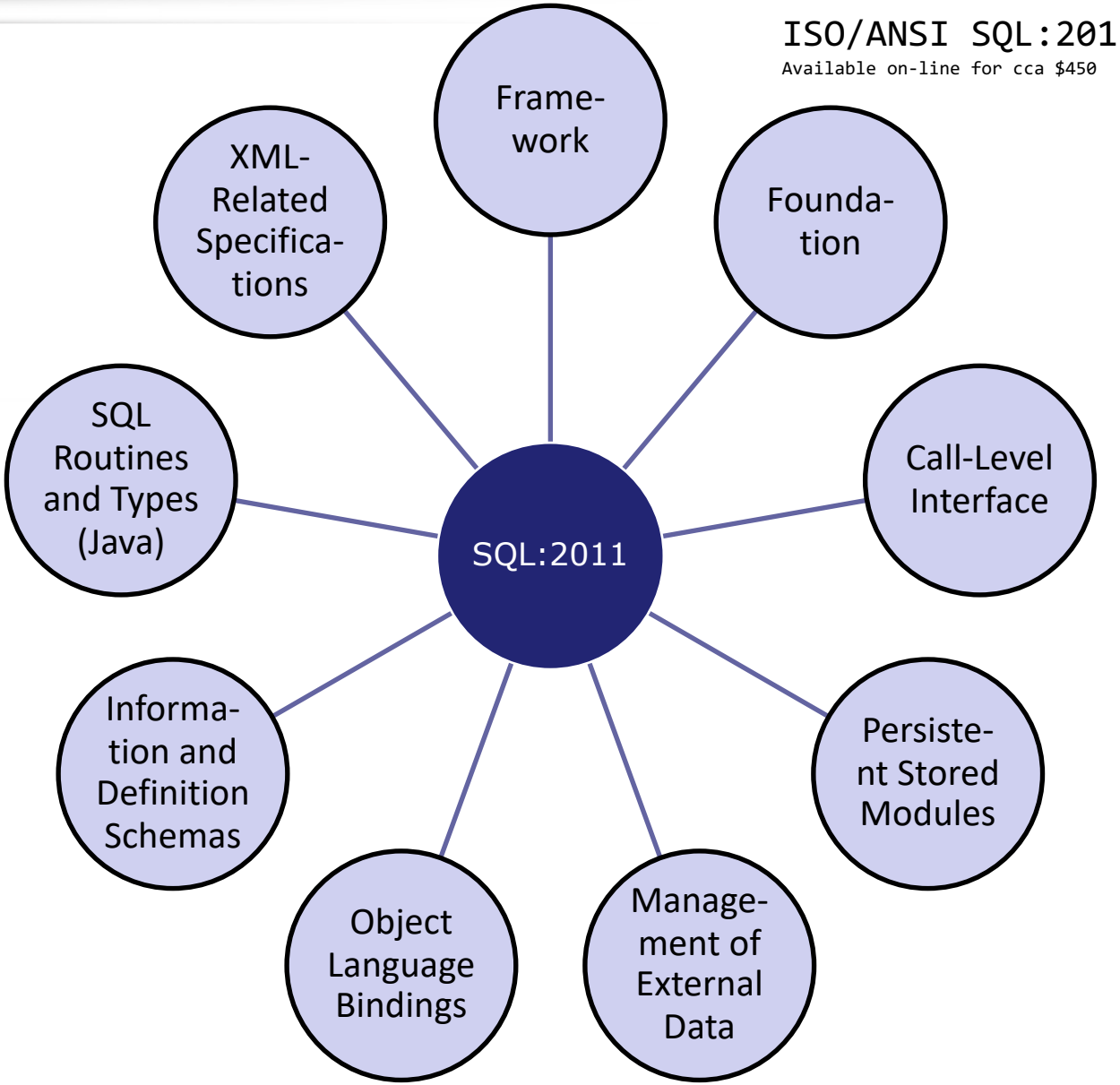
Standardizacija SQL

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision, in which the major addition were integrity constraints. Adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice-versa (SQL/JRT).
2003	SQL:2003	SQL 2003	Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it enables applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[33]
2008	SQL:2008	SQL 2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.[34]
2011	SQL:2011		

Vir: Wikipedia EN

SQL:2011

ISO/ANSI SQL:2011 standard
Available on-line for cca $450



Implementacije SQL

- Med standardi SQL-92, SQL:1999; SQL:2008,… razlike
- Implementacije ponudnikov SUPB različne (dialekti)

- Primerjava implementacij SQL
  http://troels.arvin.dk/db/rdbms/

Stavki skupine SQL DML

- DML skupina zajema SQL stavke za manipulacijo s podatki
  - SELECT → Izbira
  - INSERT → Dodajanje
  - DELETE → Brisanje
  - UPDATE → Spreminjanje

- Sintaksa SELECT stavka najbolj kompleksna

SELECT stavek…

```
SELECT [DISTINCT | ALL]
    {* | [columnExpression [AS newName]] [,...] }
FROM TableName [alias] [, ...]
[WHERE condition]
[GROUP BY columnList]
[HAVING condition]
[ORDER BY columnList]
```

SELECT stavek…

- SELECT Določa stolpce, ki naj se pojavijo v izhodni relaciji
- FROM Določa tabele za poizvedbo
- WHERE Filtrira vrstice
- GROUP BY Združuje vrstice po vrednostih izbranih stolpcev
- HAVING Filtrira skupine glede na določene pogoje
- ORDER BY Določa vrstni red vrstic na izhodu

Vrstnega reda sklopov ni možno spreminjati!
Obvezna sta samo SELECT in FROM sklopa!

Primeri

- SUPB: MySQL
- Testni podatki:
  - Employees.db
  - https://launchpad.net/test-db/+download
  - Tabele:
    - Employees
    - Salaries
    - Titles
    - Dept_emp
    - Departments
    - Dept_manager
  - Skupaj 3.919.015 zapisov

Primeri

Ali je shema popolna? Ji kaj manjka?

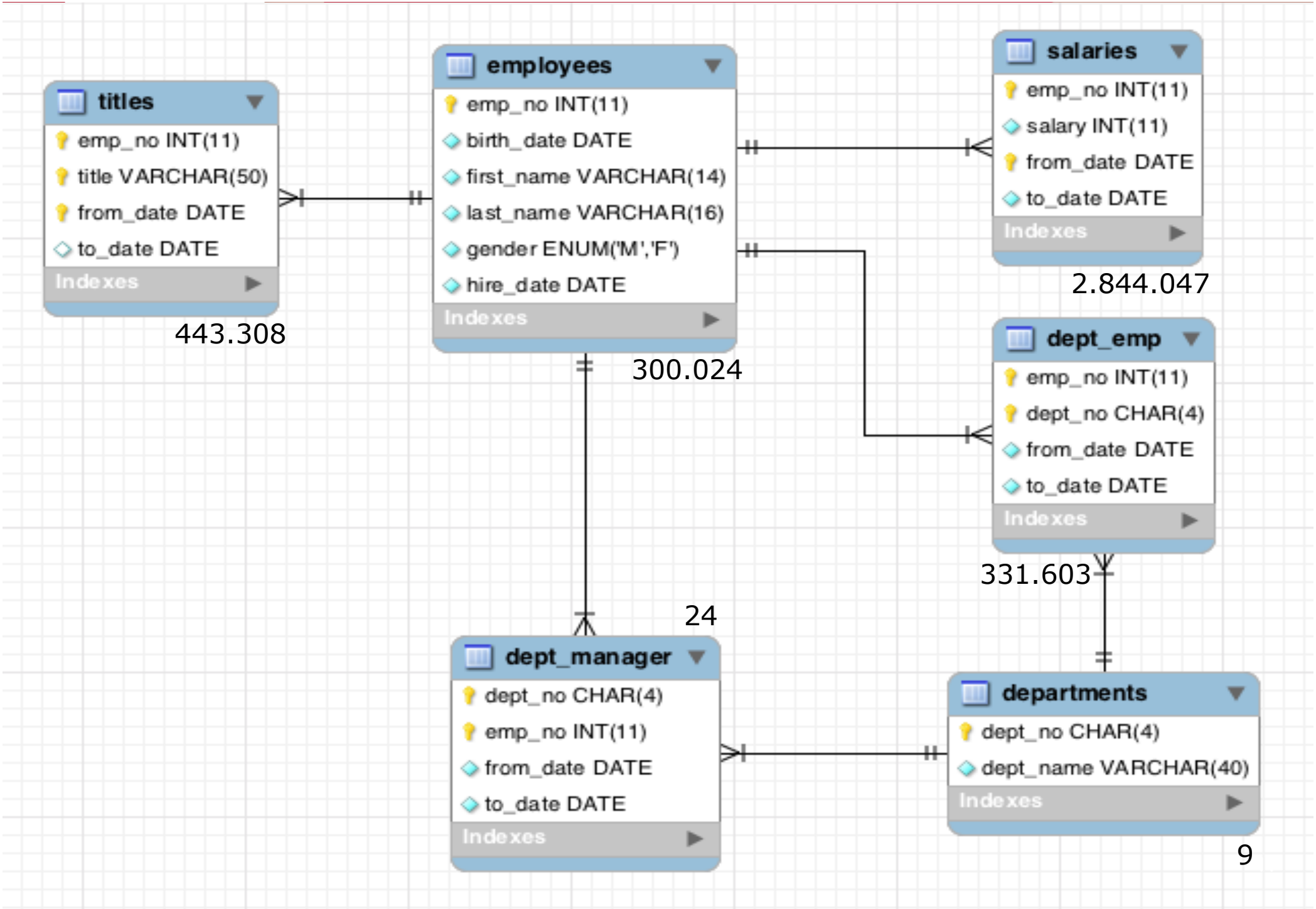
- Za primere bomo uporabljali shemo `employees`

```
employees(emp_no, birth_date, first_name, last_name,   gender,
  hire_date)
titles(emp_no, title, from_date, to_date)
salaries(emp_no, salary, from_date, to_date)
dept_emp(emp_no, dept_no, from_date, to_date)
departments(dept_no, dept_name)
dept_manager(dept_no, emp_no, from_date, to_date)
```

Primeri

- Za primere bomo uporabljali shemo employees

```
employees(emp_no, birth_date, first_name, last_name,   gender,
  hire_date)
titles(#emp_no, title, from_date, to_date)
salaries(#emp_no, salary, from_date, to_date)
dept_emp(#emp_no, #dept_no, from_date, to_date)
departments(dept_no, dept_name)
dept_manager(#dept_no, #emp_no, from_date, to_date)
```


titles
emp_no INT(11)
title VARCHAR(50)
from_date DATE
to_date DATE
Indexes
443.308
employees
emp_no INT(11)
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
gender ENUM('M','F')
hire_date DATE
Indexes
300.024
salaries
emp_no INT(11)
salary INT(11)
from_date DATE
to_date DATE
Indexes
2.844.047
dept_emp
emp_no INT(11)
dept_no CHAR(4)
from_date DATE
to_date DATE
Indexes
331.603
24
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
9

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
			Piveteau	F	1989-08-24
			Sluis	F	1990-01-22
			Bridgland	M	1992-12-18

Salaries

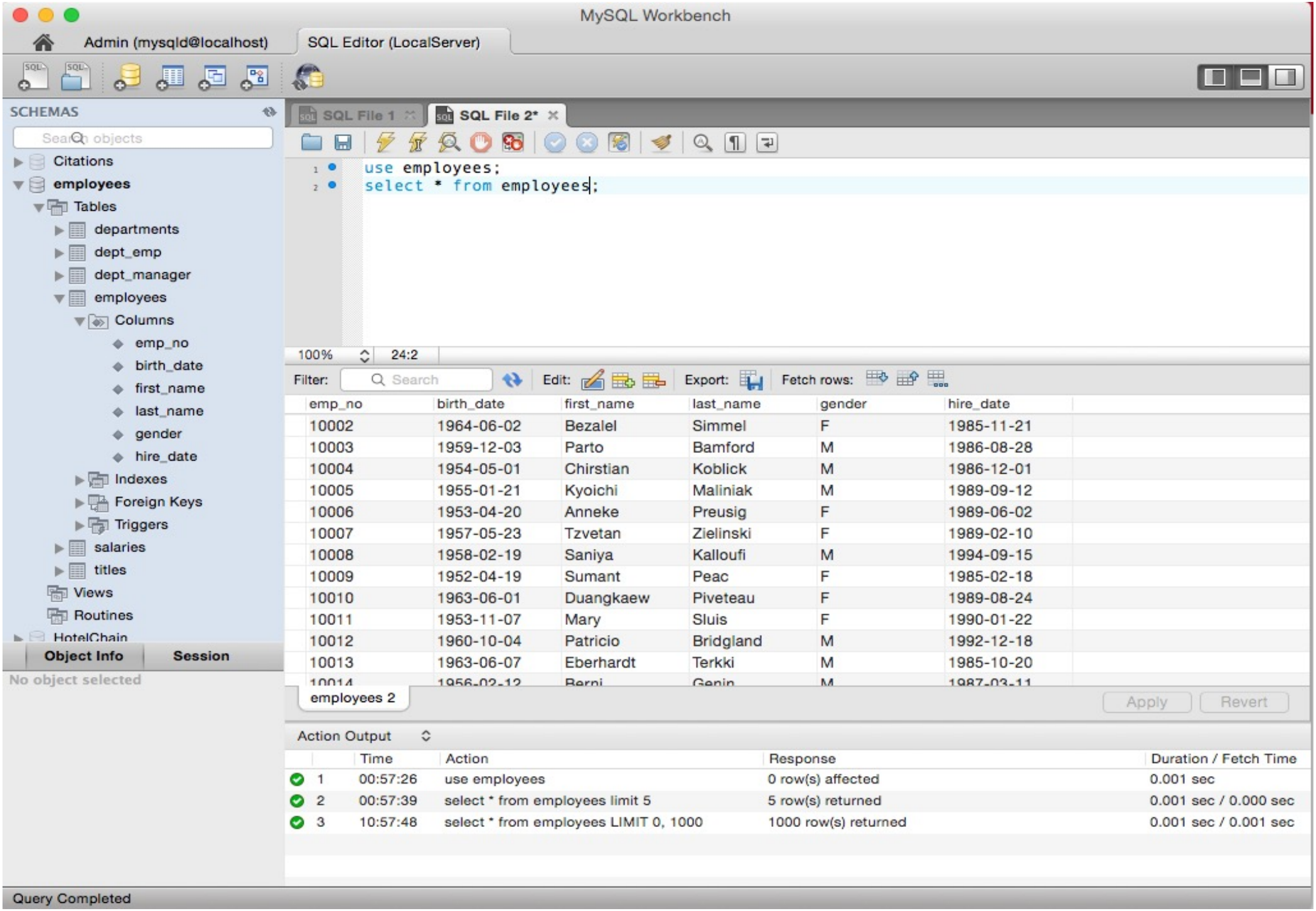
emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24
	74333	1992-06-24	1993-06-24

titles

emp_no	title	from_date	to_date
10001	Senior Engineer	1986-06-26	9999-01-01
10002	Staff	1996-08-03	9999-01-01
10003	Senior Engineer	1995-12-03	9999-01-01
10004	Engineer	1986-12-01	1995-12-01
10004	Senior Engineer	1995-12-01	9999-01-01
10005	Senior Staff	1996-09-12	9999-01-01
10005	Staff	1989-09-12	1996-09-12
10006	Senior Engineer	1990-08-05	9999-01-01

departments

dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales


MySQL Workbench
Admin (mysqld@localhost)
SQL Editor (LocalServer)
SCHEMAS
Search objects
Citations
employees
Tables
departments
dept_emp
dept_manager
employees
Columns
emp_no
birth_date
first_name
last_name
gender
hire_date
Indexes
Foreign Keys
Triggers
salaries
titles
Views
Routines
HotelChain
Object Info
Session
No object selected
SQL File 1
SQL File 2*
use employees;
select * from employees;
100%
24:2
Filter:
Search
Edit:
Export:
Fetch rows:
emp_no birth_date first_name last_name gender hire_date
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
10013 1963-06-07 Eberhardt Terkki M 1985-10-20
employees 2
Apply
Revert
Action Output
Time Action Response Duration / Fetch Time
1 00:57:26 use employees 0 row(s) affected 0.001 sec
2 00:57:39 select * from employees limit 5 5 row(s) returned 0.001 sec / 0.000 sec
3 10:57:48 select * from employees LIMIT 0, 1000 1000 row(s) returned 0.001 sec / 0.001 sec
Query Completed

Pravila formatiranja SQL primerov

- SQL izrazi so neobčutljivi na velikost črk.
- Barve niso predpisane.
- Rezervirane besede SQL ne smemo uporabljati za definicije podatkovnih objektov (tabel, stolpcev itn.)
- Sintaksa, ki jo bomo uporabljali za primere:



Enostaven izpis...

- Izpiši vse podatke o zaposlenih

```
select * from 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
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02

Enostaven izpis...

- Izpiši imena in priimke zaposlenih

```
select first_name, last_name from employees;
```

first_name	last_name
Georgi	Facello
Bezalel	Simmel
Parto	Bamford
Chirstian	Koblick
Kyoichi	Maliniak
Anneke	Preusig
Tzvetan	Zielinski
Saniya	Kalloufi
Sumant	Peac
Duangkaew	Piveteau
Mary	Sluis
Patricio	Bridgland
Eberhardt	Terkki
Berni	Genin
Guoxiang	Nooteboom

Uporaba DISTINCT

- Izpiši vse različne vrednosti, ki nastopajo v stolpcu

```
select distinct gender
from employees;
```

gender
M
F

Izračunana polja in funkcije...

- Izpiši starost delavcev ob zaposlitivi

```
select first_name, last_name, birth_date, hire_date,
DATEDIFF(hire_date, birth_date)/365
as 'Starost ob zaposlitvi'
from employees;
```

first_name	last_name	birth_date	hire_date	Starost ob zaposlitvi
Georgi	Facello	1953-09-02	1986-06-26	32.8356
Bezalel	Simmel	1964-06-02	1985-11-21	21.4849
Parto	Bamford	1959-12-03	1986-08-28	26.7534
Chirstian	Koblick	1954-05-01	1986-12-01	32.6082
Kyoichi	Maliniak	1955-01-21	1989-09-12	34.6658

Izračunana polja in funkcije...

- Druga možnost...

```
select first_name, last_name,
YEAR(hire_date)-YEAR(birth_date)
as 'Starost ob zaposlitvi'
from employees;
```

first_name	last_name	Starost ob zaposlitvi
Georgi	Facello	33
Bezalel	Simmel	21
Parto	Bamford	27
Chirstian	Koblick	32
Kyoichi	Maliniak	34

Funkcije

- Primer: mySQL, Date&Time functions

Name	Description
ADDDATE()	Add time values (intervals) to a date value
ADDTIME()	Add time
CONVERT_TZ()	Convert from one timezone to another
CURDATE()	Return the current date
CURRENT_DATE(), CURRENT_DATE	Synonyms for CURDATE()
CURRENT_TIME(), CURRENT_TIME	Synonyms for CURTIME()
CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP	Synonyms for NOW()
CURTIME()	Return the current time
DATE_ADD()	Add time values (intervals) to a date value
DATE_FORMAT()	Format date as specified
DATE_SUB()	Subtract a time value (interval) from a date
DATE()	Extract the date part of a date or datetime expression
DATEDIFF()	Subtract two dates
DAY()	Synonym for DAYOFMONTH()
DAYNAME()	Return the name of the weekday

Iskalni kriteriji

- Izpiši vse zaposlene, ki so moškega spola in so rojeni pred 1953.

```
select * from employees
```

?

```
YEAR(birth_date)<1953 and
gender = 'M';
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10020	1952-12-24	Mayuko	Warwick	M	1991-01-26
10022	1952-07-08	Shahaf	Famili	M	1995-08-22
10047	1952-06-29	Zvonko	Nyanchama	M	1989-03-31
10066	1952-11-13	Kwee	Schusler	M	1986-02-26
10097	1952-02-27	Remzi	Waschkowski	M	1990-09-15
10106	1952-08-29	Eben	Aingworth	M	1990-12-19
10108	1952-04-07	Lunjin	Civeon	M	1986-10-03

Iskanje z uporabo BETWEEN

- Izpiši zaposlene rojene med 30. in 31. decembrom 1955.

```
select * from employees where birth_date
between '1955-12-30' and '1955-12-31';
```

emp_no	birth_date	first_name	last_name	gender	hire_date
12715	1955-12-31	Honglan	Showalter	M	1991-07-31
19386	1955-12-30	Leandro	Peltason	F	1987-04-01
20437	1955-12-30	Sanjit	Versino	F	1989-09-06
21401	1955-12-31	Uzi	Vakili	F	1992-09-21
25393	1955-12-30	Claudi	Frijda	F	1987-07-12
25562	1955-12-30	Bernardo	Menhardt	F	1986-11-16
26599	1955-12-30	Katsuo	Armand	M	1987-07-01

Iskanje po članstvu množice

- Izpiši podatke o delovnih mestih, kjer gre za delovno mesto staff ali senior staff.

```
select * from titles
where title in ('Staff', 'Senior Staff');
```

emp_no	title	from_date	to_date
10002	Staff	1996-08-03	9999-01-01
10005	Senior Staff	1996-09-12	9999-01-01
10005	Staff	1989-09-12	1996-09-12
10007	Senior Staff	1996-02-11	9999-01-01
10007	Staff	1989-02-10	1996-02-11
10011	Staff	1990-01-22	1996-11-09
10013	Senior Staff	1985-10-20	9999-01-01

Iskanje z vzorcem

- Izpiši vse zaposlene, ki se pišejo na 'B' in so se rodili prvega v mesecu v šestdesetih letih.

```
select * from employees
where last_name like 'B%' and
  birth_date like '__6_____01';
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10056	1961-09-01	Brendon	Bernini	F	1990-02-01
13014	1962-06-01	Gianluca	Budinsky	M	1986-01-21
13199	1962-03-01	Gunilla	Bergere	M	1986-04-17
14499	1963-09-01	Vasilis	Bernardinello	F	1992-02-12
15882	1960-05-01	Gao	Bazelow	F	1990-11-03
15890	1962-09-01	Feiyu	Bratsberg	M	1991-07-26
16936	1964-10-01	Vitaly	Baar	F	1989-12-08
17180	1963-05-01	Salvador	Boguraev	M	1987-05-15
19862	1962-04-01	Marla	Bouloucos	M	1991-11-04
20379	1962-06-01	Arve	Bain	M	1989-11-08

Iskanje z NULL vrednostjo v pogoju

- Izpiši vse zaposlene, ki nimajo podatka o datumu zaposlitve!

```
select * from employees
where hire_date is null;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
	NULL	NULL	NULL	NULL	NULL

Sortiranje vrstic v izhodni relaciji

- Izpiši podatke o zaposlenih sortirane po priimku naraščujoče in po datumu rojstva padajoče.

```
select emp_no, last_name, birth_date
from employees
order by last_name, birth_date desc;
```

emp_no	last_name	birth_date
234739	Aamodt	1965-01-31
100916	Aamodt	1965-01-18
22105	Aamodt	1964-12-10
450068	Aamodt	1964-12-07
448061	Aamodt	1964-12-01
227725	Aamodt	1964-11-26
463875	Aamodt	1964-10-24
11761	Aamodt	1964-07-17
295537	Aamodt	1964-06-30

Agregiranje podatkov…

- ISO standard definira pet agregarnih operacij
  - COUNT vrne število vrednosti v določenem stolpcu
  - SUM vrne seštevek vrednosti v določenem stolpcu
  - AVG vrne povprečje vrednosti v določenem stolpcu
  - MIN vrne najmanjšo vrednost v določenem stolpcu
  - MAX vrne največjo vrednost v določenem stolpcu

- Vse operacije delujejo na enem stolpcu in vračajo eno samo vrednost.

Agregiranje podatkov…

- `COUNT`, `MIN` in `MAX` se uporabljajo za numerične in ne-numerične vrednosti, `SUM` in `AVG` zahtevata numerične vrednosti.

- Vse operacije razen `COUNT(*)` najprej odstranijo vrstice z NULL vrednostjo v stolpcu, po katerem agregiramo.

- `COUNT(*)` prešteje vse vrstice, ne glede na NULL vrednosti ali duplikate.

Agregiranje podatkov…

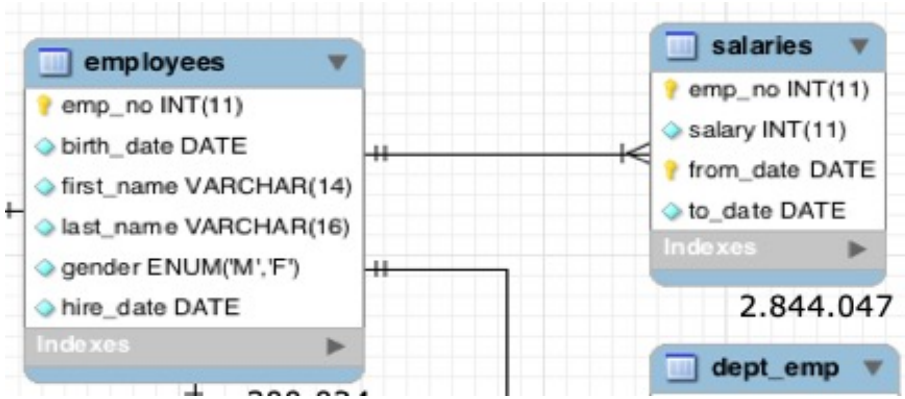
- Če se želimo znebiti duplikatov, uporabimo `DISTINCT` pred imenom stolpca.
- `DISTINCT` nima učinka na `MIN/MAX`, lahko pa vpliva na `SUM/AVG`.
- Agregarne operacije lahko uporabimo le v `SELECT` ali `HAVING` sklopu

Agregiranje podatkov

- Če `SELECT` sklop vsebuje agregarno operacijo, mora obstajati tudi `GROUP BY` sklop, sicer ni moč dodeliti agregirane vrednosti.

```
select emp_no, avg(salary)
from salaries;
```

emp_no	avg(salary)
10001	63810.7448



Agregiranje podatkov

- …

```
select emp_no, avg(salary)
from salaries
group by emp_no;
```

emp_no	avg(salary)
10001	75388.9412
10002	68854.5000
10003	43030.2857
10004	56512.2500
10005	87275.7692
10006	50514.9167
10007	70826.7143
10008	49307.6667
10009	78284.5556

Če uporabimo še
order by avg(salary)

emp_no	avg(salary)
15830	39299.5000
253406	39332.7500
473390	39372.5000
281546	39417.2500
466226	39442.7500
401786	39453.7500
245832	39515.2500
230890	39520.0000
496197	39573.3333

Uporaba COUNT in DISTINCT

- Koliko različnih delovnih mest obstaja v tabeli `titles`?

```
select count(distinct title)
from titles;
```

count(distinct
7

Uporaba več agregatov istočasno

- Izpiši minimalno, maksimalno in povprečno plačo.

```
select min(salary), max(salary), avg(salary)
from salaries;
```

min(salary)	max(salary)	avg(salary)
38623	158220	63810.7448

Združevanje podatkov…

- `GROUP BY` : združevanje podatkov v skupine.
- Ukaza `SELECT` in `GROUP BY` tesno povezana

```
select emp_no, avg(salary)
from salaries group by emp_no;
```

emp_no	salary
10001	60117
10001	62102
10001	66074
10001	66596
10001	66961
10001	71046
10001	74333
10001	75286
10001	75994
10001	76884

emp_no	avg(salary)
10001	75388.9412
10002	68854.5000
10003	43030.2857
10004	56512.2500
10005	87275.7692
10006	50514.9167
10007	70826.7143
10008	49307.6667
10009	78384.5556

Združevanje podatkov…

- Pravila:
  - Stolpci v `SELECT` tudi v `GROUP BY` (izjema stolpci, ki nastopajo samo v agregarnih operacijah).
  - Najprej se izvede `WHERE`, združevanje na preostalih vrsticah.
  - Pri združevanju `NULL` vrednosti tretirane kot enake.

Kaj vrne ta poizvedba?

```
Select YEAR(birth_date) as 'Rojeni leta',
   count(*) as 'Število'
from employees
where gender = 'F'
group by YEAR(birth_date);
```

Rojeni leta	Število
1952	8502
1953	9104
1954	9385
1955	9267
1956	9229
1957	9069
1958	9312
1959	9286
1960	9383
1961	9269

Omejitev skupin

- HAVING sklop postavlja pogoje, katerim morajo zadoščati skupine v rezultatu.
  - ! Stolpci v HAVING sklopu morajo biti tudi v SELECT sklopu ali agregatih.

```
select YEAR(birth_date), count(*)
from employees
where gender = 'F'
group by YEAR(birth_date)
having count(*) > 9200;
```

YEAR(birth_date)	count(*)
1954	9385
1955	9267
1956	9229
1958	9312
1959	9286
1960	9383
1961	9269
1962	9216

Gnezdenje poizvedb…

- Vgnezdeni `SELECT` stavki se lahko uporabijo v `WHERE` ali `HAVING` sklopih drugega `SELECT` stavka (*subselect*).

```
select first_name, last_name
from employees
where emp_no in (
    select emp_no
    from titles
    where title = 'manager'
);
```

first_name	last_name
Margareta	Markovitch
Vishwani	Minakawa
Ebru	Alpin
Isamu	Legleitner
Shirish	Ossenbruggen
Karsten	Sigstam
Krassimir	Wegerle
Rosine	Cools
Shem	Kieras
Oscar	Ghazalie
DeForest	Hagimont
Leon	DasSarma
Peternela	Onuegbe

Primer vgnezdenega stavka...WHERE

- Izpiši imena in priimke zaposlenih, ki so delali na vsaj treh delovnih mestih.

```
select first_name, last_name
from employees
where emp_no in (
 select emp_no
 from titles
 group by emp_no
 ?
);
```

Primer vgnezdenega stavka ... FROM

- Izpiši najvišjo povprečno plačo.

```
select max(dE.avgSalary)
from (
 select avg(salary) as avgSalary
 from salaries
 group by emp_no
) dE;
```

max(dE.avgSalary)
141835.3333

Pravila gnezdenja SELECT stavkov…

- Pravila vgnezdenih `SELECT` stavkov…:
  - Uporaba `ORDER BY` v vgnezdenem stavku nesmiselna.
  - `SELECT` vgnezdenega stavka lahko zajema samo en stolpec (razen v primeru uporabe ukaza `EXISTS`.
    - Mnoge implementacije tega ne upoštevajo. Tudi mySQL.

```
select... where (emp_no, title) in (
  select emp_no, title from...
);
```

Pravila gnezdenja SELECT stavkov…

- Pravila vgnezdenih SELECT stavkov…:
  - Imena stolpcev v vgnezdenem stavku se nanašajo na tabele iz vgnezdenega ali zunanjega stavka (uporaba alias-ov)

```
select E.emp_no, E.first_name, E.last_name
from employees E
where E.emp_no in (
  select S.emp_no
  from salaries S
  where E.birth_date >= S.from_date
);
```

Pravila gnezdenja SELECT stavkov…

- Pravila vgnezdenih `SELECT` stavkov…:
  - Ko je vgnezden `SELECT` stavek operand v primerjavi, se mora nahajati na desni strani enačbe.
    - Ne velja povsod. Tudi v MySQL ne.

```
select first_name, last_name
from employees
where emp_no in (
  select emp_no
  from salaries
  where (select max(salary)
         from salaries) = salary);
```

first_name	last_name
Tokuyasu	Pesch

Pravila gnezdenja SELECT stavkov…

- Pravila vgnezdenih `SELECT` stavkov…:
  - Vgnezdeni `SELECT` stavek ne more biti operand v izrazu.
    - Ne velja povsod. Tudi v MySQL ne.

```
select emp_no,
    avg(salary) - (select avg(salary) from salaries)
from salaries
group by emp_no;
```

emp_no	avg(salary) - (s…
10001	11578.1963
10002	5043.7552
10003	-20780.4591
10004	-7298.4948
10005	23465.0244
10006	-13295.8282
10007	7015.9694
10008	-14503.0782
10009	14473.8107
10010	12912.2552
10011	-14028.7448

Uporaba ANY/SOME, ALL

- V vgnezedenih stavkih, ki vračajo en sam stolpec, lahko uporabljamo operatorja `ANY/SOME` in `ALL`.
  - `ANY/SOME`: pogoj izpolnjen, če valja za vsaj eno vrednosti poizvedbe.
  - `ALL`: pogoj izpolnjen, če valja za vse vrednosti poizvedbe.

- Če rezultat poizvedbe prazen:
  - `ALL` vrne true,
  - `ANY/SOME` vrne false.

Primer uporabe ANY

- Izpiši podatke o zaposlenih, za katere velja, da se pred njimi ni rodil nobeden od ostalih zaposlenih.

```
select emp_no, first_name, last_name, birth_date
from employees
where birth_date <=
   ALL(select birth_date from employees);
```

emp_no	first_name	last_name	birth_date
65308	Jouni	Pocchiola	1952-02-01
87461	Moni	Decaestecker	1952-02-01
91374	Eishiro	Kuzuoka	1952-02-01
207658	Kiyokazu	Whitcomb	1952-02-01
237571	Ronghao	Schaad	1952-02-01
406121	Supot	Remmele	1952-02-01

Uporaba EXISTS

- `EXISTS` lahko uporabljamo le v vgnezdenih poizvedbah.
- Vrača logičen rezultat true/false.
  - True: če obstaja vsaj ena vrstica v tabeli, ki je rezultat vgnezdene poizvedbe.
  - False: če vgnezdena poizvedba vrača prazno množico.

Primer uporabe EXISTS

```
select E.emp_no, E.first_name, E.last_name, E.hire_date
from employees E where not exists (
    select E1.emp_no
    from employees E1
    where E1.hire_date < E.hire_date
);
```

emp_no	first_name	last_name	hire_date
110022	Margareta	Markovitch	1985-01-01
110085	Ebru	Alpin	1985-01-01
110183	Shirish	Ossenbruggen	1985-01-01
110303	Krassimir	Wegerle	1985-01-01
110511	DeForest	Hagimont	1985-01-01
110725	Peternela	Onuegbe	1985-01-01
111035	Przemyslawa	Kaelbling	1985-01-01
111400	Arie	Staelin	1985-01-01
111692	Tonny	Butterworth	1985-01-01

Poizvedbe po več tabelah...

- Poizvedbe po več tabelah lahko izvajamo z uporabo vgnezdenih `SELECT` stavkov
- Omejitev: stolpci v rezultatu so lahko le iz ene tabele

- V poizvedbah, ki vračajo stolpce različnih tabel, moramo uporabljati stik.
  - v sklopu `FROM` navedemo tabele, v sklopu `WHERE` določimo stolpce za stik.

Poizvedbe po več tabelah...

- Za ločevanje med istoimenskimi stolpci uporabljamo sinonime (*alias*).

```
select E.emp_no, E.first_name, E.last_name
from employees E, dept_manager DM, departments D
where
    E.emp_no = DM.emp_no AND
    DM.dept_no = D.dept_no AND
    D.dept_name = 'marketing';
```

emp_no	first_name	last_name
110022	Margareta	Markovitch
110039	Vishwani	Minakawa

Alternativni načini stika več tabel

- Alternativni načini stika med več tabelami:
  - `FROM employees E JOIN titles T ON E.emp_no = T.emp_no`
  - `FROM employees JOIN titles USING emp_no`
  - `FROM employees NATURAL JOIN titles`

- Zgornji zapisi nadomestijo sklopa `FROM` in `WHERE`
- V prvem primeru rezultat vsebuje dva identična stolpca .

Primer naravnega stika

```
select * from employees natural join titles;
```

emp_no	birth_date	first_name	last_name	gender	hire_date	title	from_date	to_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26	Senior Engineer	1986-06-26	9999-01-01
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	Staff	1996-08-03	9999-01-01
10003	1959-12-03	Parto	Bamford	M	1986-08-28	Senior Engineer	1995-12-03	9999-01-01
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	Engineer	1986-12-01	1995-12-01
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	Senior Engineer	1995-12-01	9999-01-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	Senior Staff	1996-09-12	9999-01-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	Staff	1989-09-12	1996-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02	Senior Engineer	1990-08-05	9999-01-01
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10	Senior Staff	1996-02-11	9999-01-01
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10	Staff	1989-02-10	1996-02-11
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15	Assistant Engi...	1998-03-11	2000-07-31
10009	1952-04-19	Sumant	Peac	F	1985-02-18	Assistant Engi...	1985-02-18	1990-02-18
10009	1952-04-19	Sumant	Peac	F	1985-02-18	Engineer	1990-02-18	1995-02-18
10009	1952-04-19	Sumant	Peac	F	1985-02-18	Senior Engineer	1995-02-18	9999-01-01
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24	Engineer	1996-11-24	9999-01-01

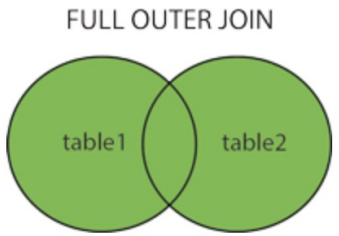
Zunanji stik

- Primer `LEFT JOIN`

```
select E.emp_no, E.first_name, E.last_name
from employees E left join dept_manager DM
on E.emp_no = DM.emp_no;
```

emp_no	first_name	last_name
10001	Georgi	Facello
10002	Bezalel	Simmel
10003	Parto	Bamford
10004	Chirstian	Koblick
10005	Kyoichi	Maliniak
10006	Anneke	Preusig
10007	Tzvetan	Zielinski
10008	Saniya	Kalloufi
10009	Sumant	Peac
10010	Duangkaew	Piveteau

Dodatne možnosti:
`RIGHT JOIN`
`FULL OUTER JOIN`



Uporaba operacij nad množicami...

- Rezultate dveh ali več poizvedb lahko združujemo z ukazi:
  - `UNION` (unija),
  - `INTERSECTION` (Presek)
  - `DIFFERENCE` (`EXCEPT`) (Razlika)

- Da lahko izvajamo naštete operacije, morata tabeli A in B biti skladni (domene atributov morajo biti enake).
  - Različne implementacije imajo glede tega zelo različna pravila!

Uporaba operacij nad množicami

- Sintaksa:

```
op [ALL] [CORRESPONDING [BY
{column1 [, ...]}]]
```

- Če uporabimo `CORRESPONDING BY`, se operacija izvede samo nad poimenovanimi stolpci
- Če uporabimo samo `CORRESPONDING` brez `BY` člena, se operacija izvede nad skupnimi stolpci.
- Če uporabimo `ALL`, lahko rezultat vključuje tudi dvojnike

MySQL

V MYSQL opcija `CORRESPONDING BY` ni na voljo.

Primer unije

- Izpiši prve tri zaposlene moške in prve tri zaposlene ženske.

```
select EM.first_name, EM.last_name, 'Moški'
from
    (select * from employees where gender = 'M'
    order by hire_date desc limit 3) EM
union
select EF.first_name, EF.last_name, 'Ženske'
from
    (select * from employees where gender = 'F'
    order by hire_date desc limit 3) EF;
```

INSERT stavek...

```
INSERT INTO TableName [ (columnList) ]
VALUES (dataValueList)
```

- Seznam columnList ni obvezen;
- Pri vnosu moramo vpisati najmanj vse obvezne vrednosti (not null), razen za stolpce s privzeto vrednostjo (DEFAULT).
- Seznam dataValueList mora ustrezati seznamu columnList.

Primeri INSERT stavkov...

- Vnos nove vrstice v tabelo departments

neobvezno

```
insert into departments (dept_no, dept_name)
values ('d010', 'Education');
```

Shema relacije:
departments (dept_no, dept_name)

- Posebnosti:
  - Obravnava neobveznih polj
  - Privzete vrednosti

Primeri INSERT stavkov...

- Vnos več vrstic iz ene ali več drugih tabel...

```
INSERT INTO TableName [ (columnList) ]
SELECT ...
```

UPDATE stavek...

```
UPDATE TableName
SET columnName1 = dataValue1
   [, columnName2 = dataValue2...]
[WHERE searchCondition]
```

- TableName se lahko nanaša na ime osnovne tabele ali ime pogleda.
- Sklop SET določa nazive enega ali več stolpcev ter nove vrednosti teh stolpcev (morajo ustrezati po podatkovnem tipu).
- WHERE sklop neobvezen.

Primeri UPDATE stavkov

- Vsem zaposlenim dvigni plačo za 10%.

```
update salaries set salary = salary * 1.1
 where to_date = '9999-01-01'
```

DELETE stavek

```
DELETE FROM TableName
[WHERE searchCondition]
```

- TableName se lahko nanaša na ime osnovne tabele ali ime pogleda.
- WHERE sklop ni obvezen. Če ga spustimo, zbrišemo vse vrstice v tabeli. Tabela ostane.

Primeri DELETE stavkov

- Izbriši podatke o plačah delavcev, ki niso več zaposleni.

```
delete from salaries
where emp_no not in (
  select emp_no
  from salaries
  where to_date = '9999-01-01'
);
```

Drop vs Truncate

Stavki skupine SQL DDL...

- DDL skupina zajema SQL stavke za manipulacijo s strukturo podatkovne baze.

- Kaj si bomo pogledali:
  - Podatkovni tipi, ki jih podpira SQL standard.
  - Mehanizmi za zagotavljanje kakovosti podatkov.
  - Uporaba mehanizmov za zagotavljanje kakovosti v `CREATE` in `ALTER TABLE` stavkih.
  - Način delovanja ISO transakcijskega modela.
  - Sprožilci, shranjene procedure in funkcije.

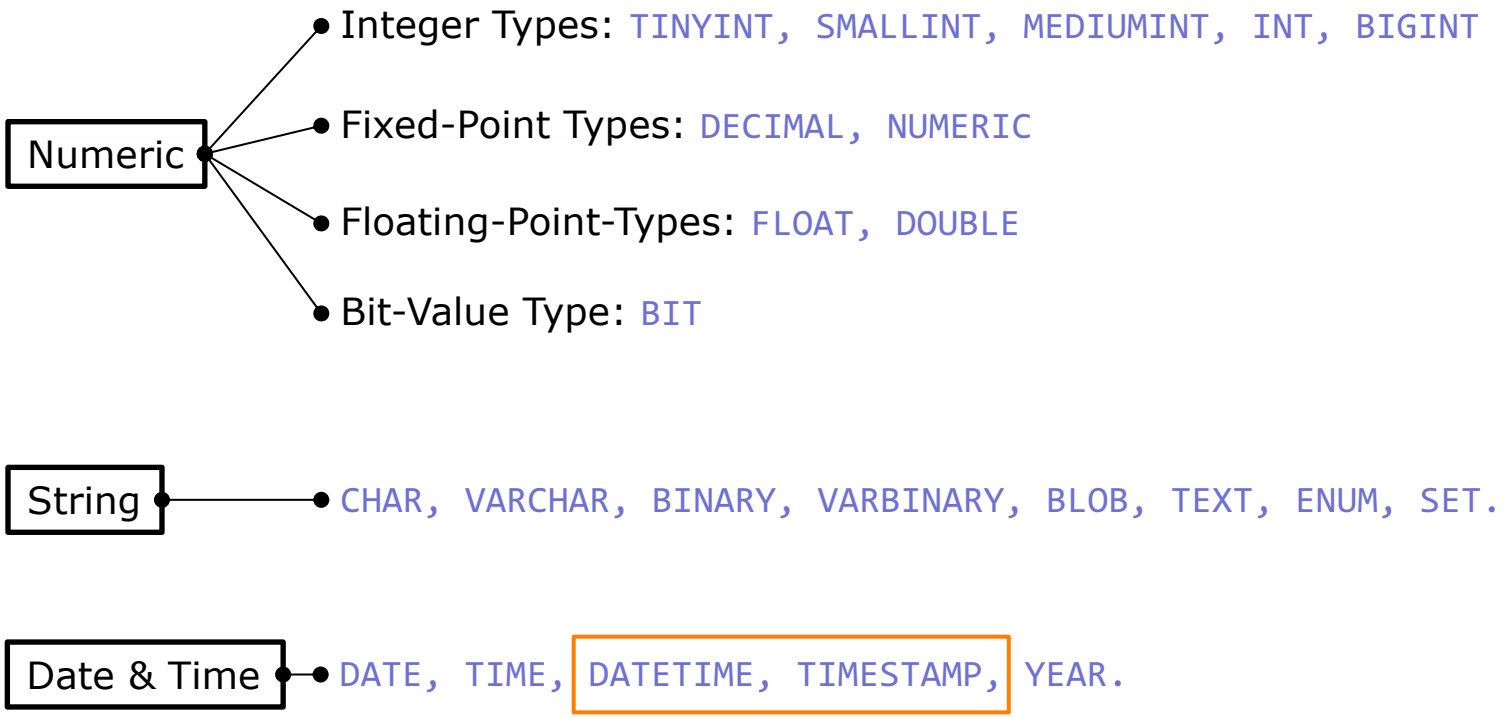
Podatkovni tipi v SQL standardu

Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

Vir: [2,159]

Podatkovni tipi v MySQL



Zagotavljanje kakovosti podatkov...

- SQL standard ponuja več vrst omejitev (*Integrity Enhacement Fetures*):
  - Obveznost podatkov
  - Omejitve domene (*Domain constraints*)
  - Pravila za celovitost podatkov (*Integrity constraints*)
    - Celovitost entitet (*Entity Integrity*)
    - Celovitost povezav (*Referential Integrity*)
  - Številnost (*Multyplicity*)
  - Splošne omejitve (*General constraints*)

- Omejitve so lahko definirane v `CREATE` in `ALTER TABLE` stavkih.

Zagotavljanje kakovosti podatkov...

- Obveznost podatkov

```
emp_no numeric(5) NOT NULL
```

- Omejitve domene

```
gender CHAR NOT NULL
    CHECK (gender in ('M', 'F'))
```

Zagotavljanje kakovosti podatkov...

- Kreiranje domene

```
CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]
```

Primer:

```
CREATE DOMAIN Tgender AS CHAR
CHECK (VALUE IN ('M', 'Ž'));

gender Tgender NOT NULL
```

MySQL

MySQL ne podpira uporabniških domen.

Zagotavljanje kakovosti podatkov...

- searchCondition lahko vsebuje iskalno tabelo:

```
CREATE DOMAIN TempID AS numeric(5)
CHECK (VALUE IN (SELECT emp_no FROM employees));
```

- Domeno lahko ukinemo z uporabo stavka DROP DOMAIN:

```
DROP DOMAIN DomainName
[RESTRICT | CASCADE]
```

Pove, kako ravnati, če je domena trenutno v uporabi

Celovitost entitet

- ISO standard podpira kreirnje primarnih in tujih ključev v okviru `CREATE` in `ALTER TABLE` stavkov.

```
PRIMARY KEY(dept_no, emp_no)
FOREIGN KEY(emp_no) REFERENCES employees
```

- Določimo lahko enolične stolpce ali kombinacije stolpcev.

```
UNIQUE(last_name)
UNIQUE(last_name, first_name)
```

Celovitost povezav...

- Celovitost povezav: če ima FK neko vrednost, potem se mora ta vrednost nahajati v primarnem ključu povezane tabele!

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

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

Celovitost povezav...

- Vsak `INSERT/UPDATE` stavek, ki skuša kreirati FK vrednost v tabeli, brez da bi ta vrednost obstajala kot PK v povezani tabeli, je zavrnjen.

- Ob zavrnitvi so možne naslednje akcije
  - `CASCADE`
  - `SET NULL`
  - `SET DEFAULT`
  - `NO ACTION`

MySQL

MySQL opcije: `RESTRICT, CASCADE, SET NULL, NO ACTION`

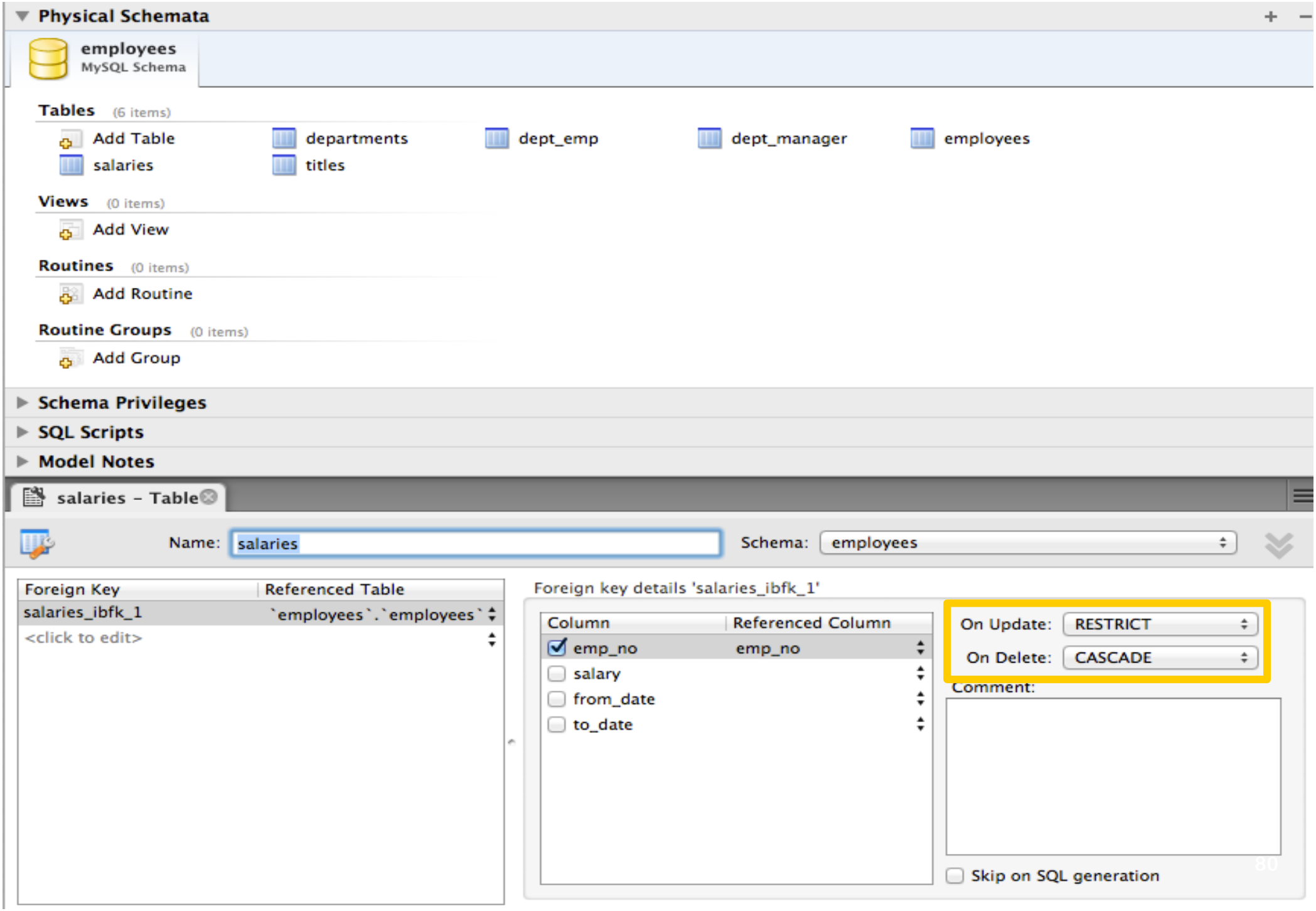
Celovitost povezav...

- Določimo z uporabo `ON UPDATE`, `ON DELETE`, `ON UPDATE SET NULL`

- Primeri:

```
FOREIGN KEY (emp_no) REFERENCES employees
ON DELETE CASCADE
```

```
show create table salaries
```


Physical Schemata
employees
MySQL Schema
Tables (6 items)
Add Table
departments
dept_emp
dept_manager
employees
salaries
titles
Views (0 items)
Add View
Routines (0 items)
Add Routine
Routine Groups (0 items)
Add Group
Schema Privileges
SQL Scripts
Model Notes
salaries – Table
Name: salaries
Schema: employees
Foreign Key
Referenced Table
salaries_ibfk_1
`employees`.`employees`
<click to edit>
Foreign key details 'salaries_ibfk_1'
Column
Referenced Column
emp_no
emp_no
salary
from_date
to_date
On Update: RESTRICT
On Delete: CASCADE
Comment:
Skip on SQL generation

IEF – Splošne omejitve

- Splošne omejitve določimo z `CHECK/UNIQUE` opcijami v `CREATE` in `ALTER TABLE` stavkih.

```
CREATE ASSERTION AssertionName
CHECK (searchCondition)
```

MySQL

MySQL ne podpira splošnih omejitev. V večini primerov zadoščajo sprožilci (trigger)

Primer splošne omejitve

- Poslovno pravilo: nobenemu zaposlenemu se plača ne sme v istem letu povečati več kot trikrat.

```
CREATE ASSERTION PrevecSprememb
 CHECK not exists (
    select count(*)
    from salaries
    group by emp_id, YEAR(date_from)
    having count(*) > 3
 );
```

Ali ta SQL stavek ustreza našim zahtevam?

Kreiranje podatkovnih objektov

- SQL DDL omogoča kreiranje in brisanje podatkovnih objektov, kot so: shema, domena, tabela, pogled in indeks.

- Glavni SQL DDL stavki:

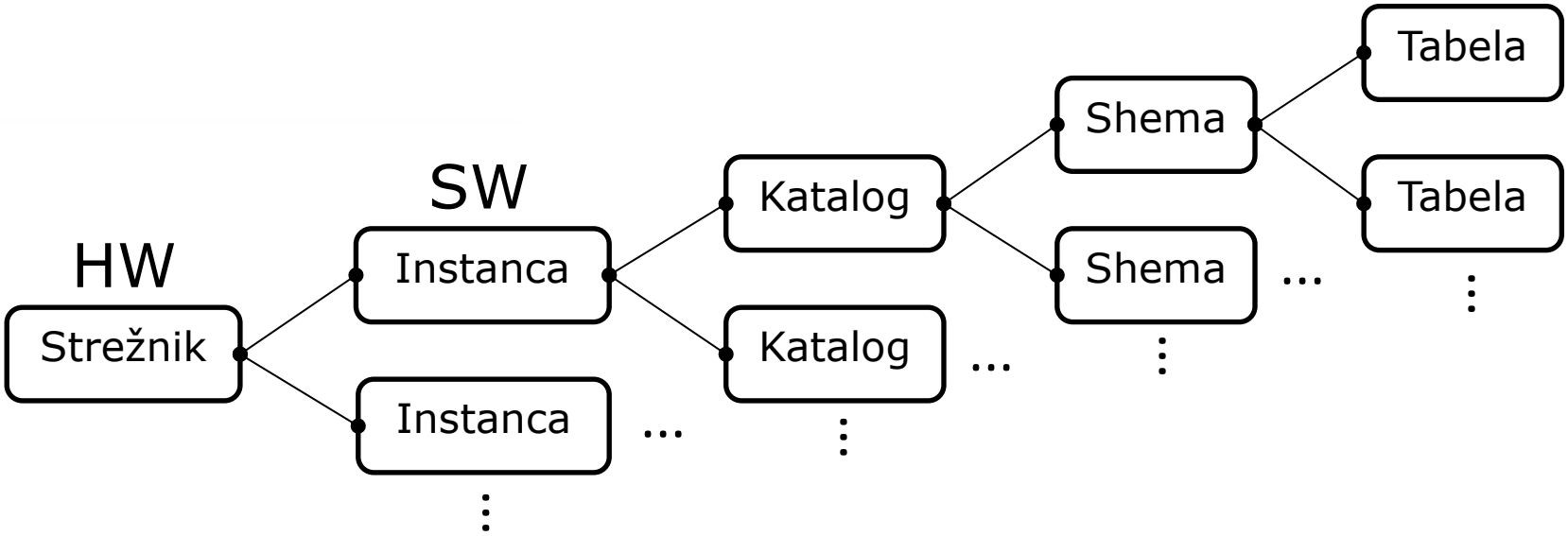
```
CREATE SCHEMA          DROP SCHEMA
CREATE/ALTER DOMAIN    DROP DOMAIN
CREATE/ALTER TABLE     DROP TABLE
CREATE/ALTER VIEW      DROP VIEW
CREATE INDEX           DROP INDEX
```

Kreiranje podatkovnih objektov...

- Relacije in drugi podatkovni objekti obstajajo v nekem okolju.
  - Vsako okolje vsebuje enega ali več katalogov, vsak katalog pa množico shem.
  - Shema je poimenovana kolekcija povezanih podatkovnih objektov.
  - Objekti v shemi so lahko tabele, pogledi, domene, trditve, dodelitve, pretvorbe in znakovni nizi. Vsi objekti imajo istega lastnika.

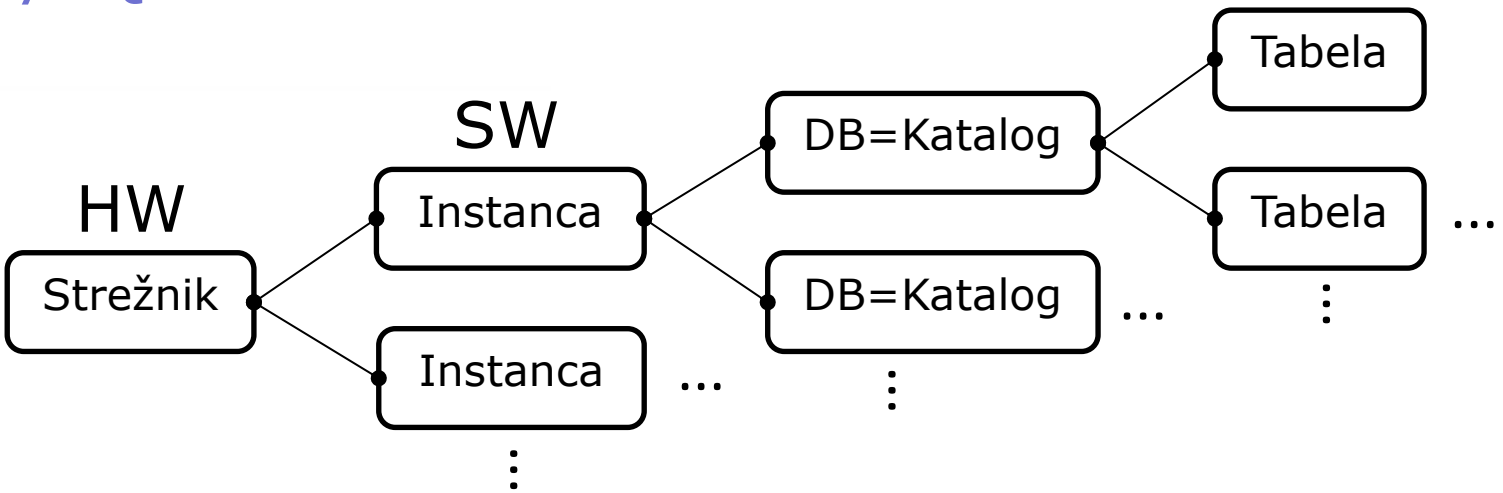
- Med implemantacijami razlike v poimenovanju.

Kreiranje podatkovnih objektov...



Kreiranje podatkovnih objektov...

MySQL



Kreiranje sheme

```
CREATE SCHEMA [Name |
    AUTHORIZATION CreatorId ]

DROP SCHEMA Name [RESTRICT | CASCADE]
```

- `RESTRICT` (privzeto): shema mora biti prazna, sicer brisanje ni možno.
- `CASCADE`: kaskadno se brišejo vsi objekti, povezani s shemo. Če katerokoli brisanje ne uspe, se zavrne celotna operacija.

Kreiranje tabele...

```
CREATE TABLE TableName
({colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] […,]}
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES ParentTableName [(listOfCKColumns)],
  [ON UPDATE referentialAction]
  [ON DELETE referentialAction ]] [,…]}
{[CHECK (searchCondition)
```

Primer kreiranja tabele

Primer CREATE stavka v MySQL

```
CREATE TABLE dept_manager (
    dept_no CHAR(4) NOT NULL,
    emp_no INT(11) NOT NULL,
    from_date DATE NOT NULL,
    to_date DATE NOT NULL,
    PRIMARY KEY (emp_no, dept_no),
    CONSTRAINT dept_manager_ibfk_1
          FOREIGN KEY (emp_no)
          REFERENCES employees(emp_no)
          ON DELETE CASCADE,
    CONSTRAINT dept_manager_ibfk_2
          FOREIGN KEY (dept_no)
          REFERENCES departments (dept_no)
          ON DELETE RESTRICT)
```

Vaja

- Kreirati želimo podatkovno bazo za hranjenje filmov – Filmoteka.
- Tabela FILM
  - Naslov
  - Avtorji
  - Žanr
  - Ocena IMDB
  - Moja ocena
  - Tip
  - Jezik
  - Videno

Kreiramo in izberemo podatkovno bazo

```
create database filmoteka;
use filmoteka;
```

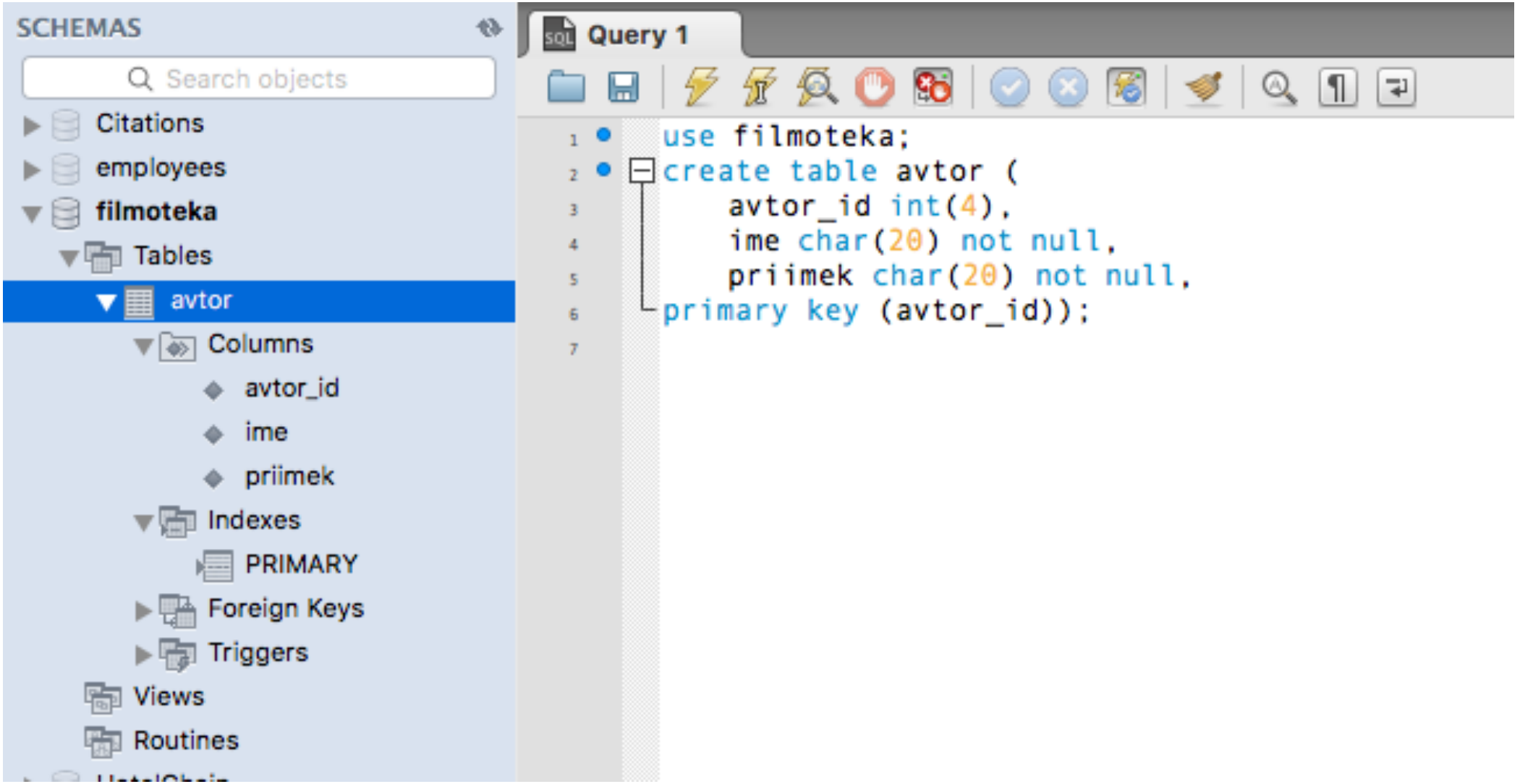
Tabele

- Ali poleg tabele filmoteka potrebujemo še kakšno tabelo? So atributi vsi enostavni?

Naslov
Avtorji
Žanr
Ocena IMDB
Moja ocena
Tip
Jezik
Videno

Kreiramo tabelo avtor

```
create table avtor (
      avtor_id int(4),
      ime char(20) not null,
      priimek char(20) not null,
primary key (avtor_id));
```

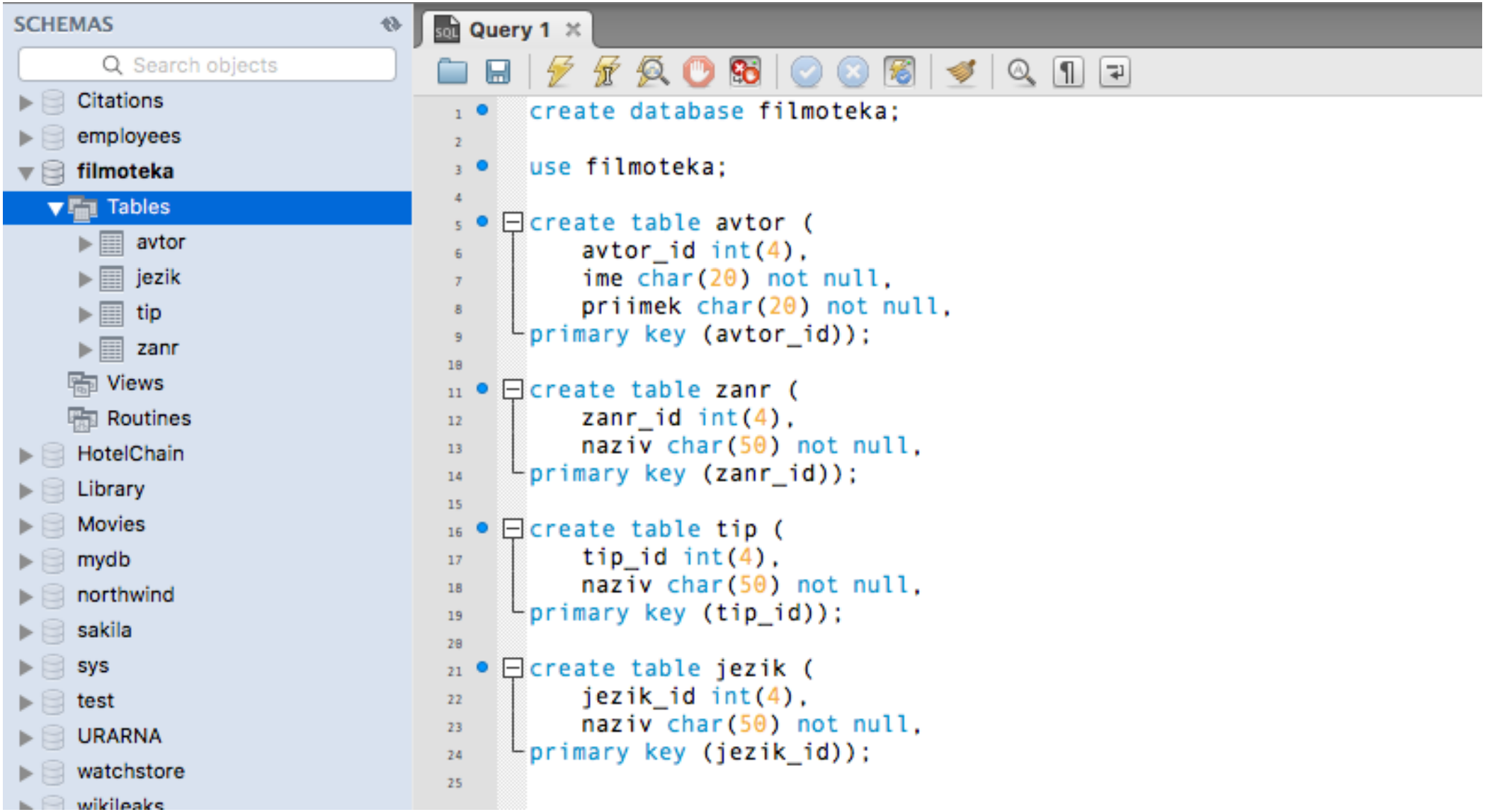

SCHEMAS
Search objects
Citations
employees
filmoteka
Tables
avtor
Columns
avtor_id
ime
priimek
Indexes
PRIMARY
Foreign Keys
Triggers
Views
Routines
Query 1
use filmoteka;
create table avtor (
avtor_id int(4),
ime char(20) not null,
priimek char(20) not null,
primary key (avtor_id));

Kreiramo ostale povezane tabele

```
create table zanr (
      zanr_id int(4),
      naziv char(50) not null,
primary key (zanr_id));

create table tip (
      tip_id int(4),
      naziv char(50) not null,
primary key (tip_id));

create table jezik (
      jezik_id int(4),
      naziv char(50) not null,
primary key (jezik_id));
```

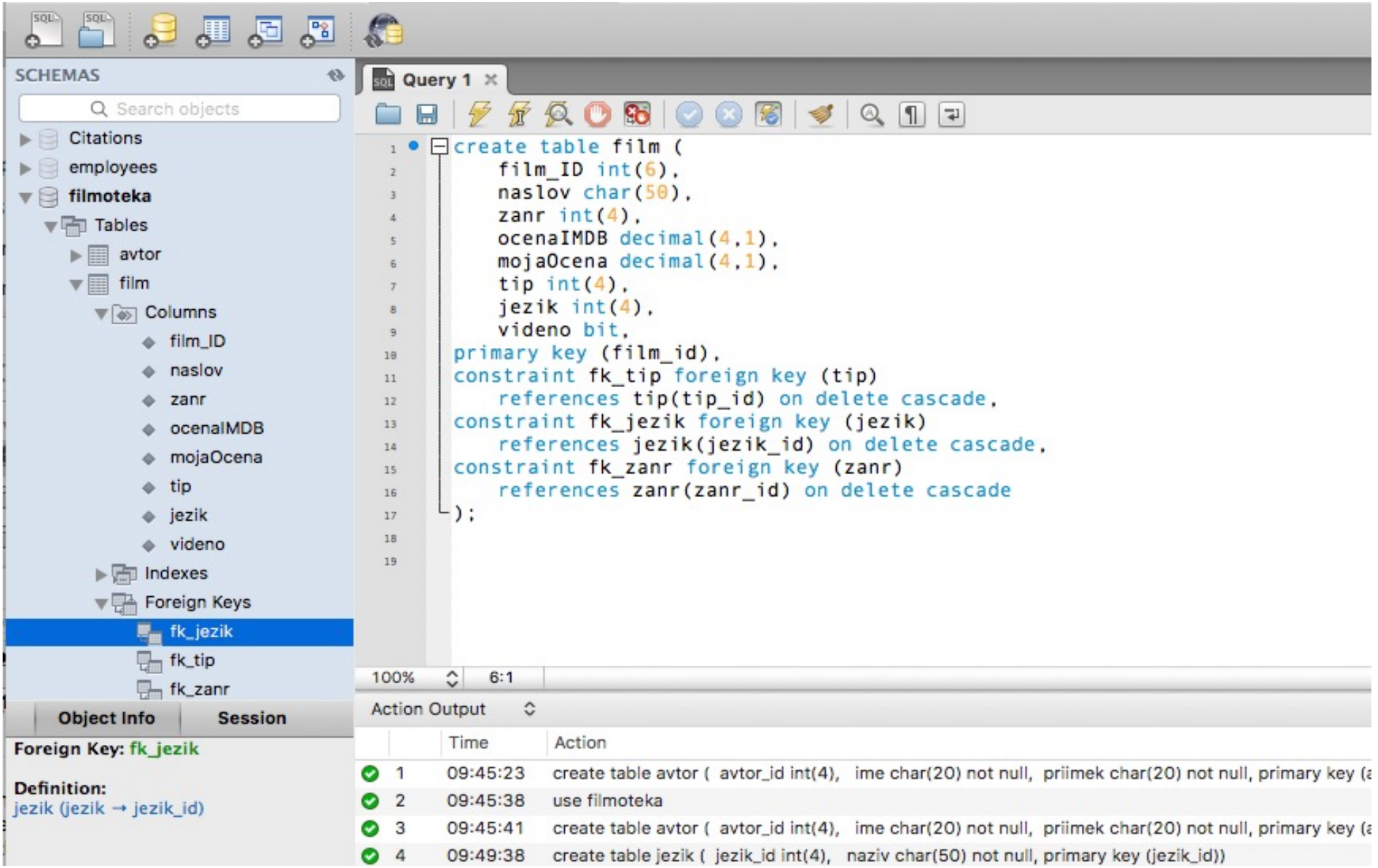

SCHEMAS
Search objects
Citations
employees
filmoteka
Tables
avtor
jezik
tip
zanr
Views
Routines
HotelChain
Library
Movies
mydb
northwind
sakila
sys
test
URARNA
watchstore
Query 1
create database filmoteka;
use filmoteka;
create table avtor (
avtor_id int(4),
ime char(20) not null,
priimek char(20) not null,
primary key (avtor_id));
create table zanr (
zanr_id int(4),
naziv char(50) not null,
primary key (zanr_id));
create table tip (
tip_id int(4),
naziv char(50) not null,
primary key (tip_id));
create table jezik (
jezik_id int(4),
naziv char(50) not null,
primary key (jezik_id));

Kreiramo tabelo film

```
create table film (
      film_id int(6),
      naslov char(50),
      avtor int(6),
      zanr int(4),
      ocenaIMDB decimal(4,1),
      mojaOcena decimal(4,1),
      tip int(4),
      jezik int(4),
      videno bit,
primary key (film_id),
...
```

...povezave na druge tabele

```
...
   constraint fk_avtor foreign key (avtor)
      references avtor(avtor_id) on delete restrict
   constraint fk_tip foreign key (tip)
      references tip(tip_id) on delete cascade,
   constraint fk_jezik foreign key (jezik)
      references jezik(jezik_id) on delete cascade
      on update cascade,
   constraint fk_zanr foreign key (zanr)
      references zanr(zanr_id) on delete cascade on
      update restrict
   );
```


SCHEMAS
Search objects
Citations
employees
filmoteka
Tables
avtor
film
Columns
film_ID
naslov
zanr
ocenaIMDB
mojaOcena
tip
jezik
videno
Indexes
Foreign Keys
fk_jezik
fk_tip
fk_zanr
Object Info
Session
Foreign Key: fk_jezik
Definition:
jezik (jezik → jezik_id)
Query 1
create table film (
film_ID int(6),
naslov char(50),
zanr int(4),
ocenaIMDB decimal(4,1),
mojaOcena decimal(4,1),
tip int(4),
jezik int(4),
videno bit,
primary key (film_id),
constraint fk_tip foreign key (tip)
references tip(tip_id) on delete cascade,
constraint fk_jezik foreign key (jezik)
references jezik(jezik_id) on delete cascade,
constraint fk_zanr foreign key (zanr)
references zanr(zanr_id) on delete cascade
);
100%
6:1
Action Output
Time
Action
1 09:45:23 create table avtor (avtor_id int(4), ime char(20) not null, priimek char(20) not null, primary key (a
2 09:45:38 use filmoteka
3 09:45:41 create table avtor (avtor_id int(4), ime char(20) not null, priimek char(20) not null, primary key (a
4 09:49:38 create table jezik (jezik_id int(4), naziv char(50) not null, primary key (jezik_id))

Obveznost povezav

- Vnesimo nekaj podatkov:

```
insert into avtor (avtor_id, ime, priimek) values (…)
insert into zanr (zanr_id, naziv) values (…)
insert into tip (tip_id, naziv) values (…)
insert into jezik (jezik_id, naziv) values (…)
insert into film (film_id, naslov, avtor, zanr,
   ocenaIMDB, mojaOcena, tip, jezik, videno) values (…)
```

```
insert into avtor (avtor_id, ime, priimek) values (1, 'David', 'Fincher'), (2, 'Matevž', 'Luzar');
insert into zanr (zanr_id, naziv) values (1, 'Kriminalka'), (2, 'Akcija'), (3, 'Drama');
insert into tip (tip_id, naziv) values (1, 'Film'), (2, 'TV serija'), (3, 'Risani film');
insert into jezik (jezik_id, naziv) values (1, 'Angleški'), (2, 'Nemški'), (3, 'Slovenski');
insert into film (film_id, naslov, avtor, zanr, ocenaIMDB, mojaOcena, tip, jezik, videno)
values (1, 'Se7en', 1, 1, 8.6, 9, 1, 1, 1), (1, 'Jezero', 2, 1, 7.9, 8.2, 2, 3, 1);
```

Kaj pa podatki o igralcih v filmu?

ALTER TABLE stavek...

- S stavkom `ALTER TABLE` lahko:
  - Dodajamo ali ukinjamo stolpce v tabeli;
  - Dodajamo ali ukinemo omejitve tabele;
  - Za stolpce v tabeli določamo ali ukinjamo privzete vrednosti;
  - Spreminjamo podatkovne tipe stolpcev v tabeli;

Primeri ALTER TABLE stavkov...

- Spremeni tabelo `dept_manager` tako, da ukineš privzeto vrednost stolpca `from_date`.

```
ALTER TABLE dept_manager
    ALTER from_date DROP DEFAULT;
```

Primeri ALTER TABLE stavkov...

- Spremeni tabelo `salaries` tako, da ukineš omejitev `PrevecSprememb`. Tabeli `employees` dodaj stolpec `retired` s privzeto vrednost 'N'.

```
ALTER TABLE salaries
 DROP CONSTRAINT PrevecSprememb;

ALTER TABLE employees
 ADD retired CHAR(1) NOT NULL DEFAULT 'N';
```

Primeri ALTER TABLE

```
ALTER TABLE employees ADD nickname AFTER first_name;

ALTER TABLE employees CHANGE first_name fn;

ALTER TABLE employees MODIFY birth_date date NULL;

ALTER TABLE employees ALTER gender SET DEFAULT 'M';

ALTER TABLE employees DROP gender;

DESCRIBE employees;
```

Stavek DROP TABLE

- S pomočjo stavka `DROP TABLE` ukinemo tabelo. Obenem se zbrišejo vsi zapisi tabele.

```
DROP TABLE TableName [RESTRICT | CASCADE]
```

- Primer:

```
DROP TABLE employees RESTRICT;
```

Glej še: `TRUNCATE`, `CREATE LIKE` + `DROP`

Indeksi

- Omogočajo urejanje tabel po različnih stolpcih. So ključnega pomena za hitro poizvedovanje.

- Splošna sintaksa:

```
CREATE [UNIQUE] INDEX index_name
ON table_name
   (column1 [ASC|DESC],
    column2 [ASC|DESC],
   ...);
```

Indeksi

- Sintaksa v MySQL:

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
[USING {BTREE|HASH}] ON tbl_name (index_col_name,...)
[index_option][algorithm_option|lock_option]

index_col_name: col_name [(length)] [ASC|DESC]
index_option:
    KEY_BLOCK_SIZE[=]value|
    USING {BTREE|HASH}|
    WITH PARSER parser_name|
    COMMENT 'string'
algorithm_option: ALGORITHM[=]{DEFAULT|INPLACE|COPY}
lock_option: LOCK[=]{DEFAULT|NONE|SHARED|EXCLUSIVE}
```

Primer kreiranja indeksa

- Kreiraj indeks na tabeli `Employees`, in sicer nad stolpci ime, priimek in spol. Indeks naj bo urejen naraščajoče po imenu in padajoče po priimku.

```
CREATE INDEX idx_emp_fullname_gender
ON employees
(first_name ASC, last_name DESC, gender);
```

```
SHOW INDEX FROM employees;
```

Pogledi

- Predstavljajo navidezne tabele. Vsakokrat, ko dostopamo do pogleda, se v ozadju izvede SELECT stavek.

- Splošna sintaksa

```
CREATE VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Primer kreiranja pogleda

- Kreiraj pogled, ki prikazoval podatke o tem, kdo so trenutno vodje posameznih oddelkov ter od kdaj so na funkciji.

```
CREATE VIEW Managers AS
    SELECT e.emp_no, e.first_name, e.last_name,
           d.dept_name, dm.from_date
    FROM employees e
    JOIN dept_manager dm ON e.emp_no = dm.emp_no
    JOIN departments d ON d.dept_no = dm.dept_no
    WHERE dm.to_date = '9999-01-01';
```

Transakcije…

- SQL definira transakcijski model z ukazoma `COMMIT` in `ROLLBACK`.
- Transakcija: logična enota dela z enim ali več SQL ukazi. Je atomarna.
- Spremembe znotraj transakcije praviloma drugim transakcijam skrite, dokler transakcija ni končana.

Transakcije...

- Transakcija se lahko zaključi eksplicitno (`COMMIT/ROLLBACK`) ali implicitno (skupaj s programom, v katerem klicana).
- Nova transakcija se začne z novim SQL stavkom, ki transakcijo inicira.
- SQL transakcij ne moremo gnezditi.
  - Pravilo ne velja pri vseh implementacijah.

Transakcije

- Transakcijo nastavimo s pomočjo ukaza `SET TRANSACTION`

```
SET TRANSACTION
[READ ONLY | READ WRITE] |
[ISOLATION LEVEL READ UNCOMMITTED |
READ COMMITTED|REPEATABLE READ |SERIALIZABLE ]
```

Transakcije…

- `READ ONLY` – pove, da transakcija vključuje samo operacije, ki iz baze berejo.
  - SUPB bo dovolil `INSERT`, `UPDATE` in `DELETE` samo nad začasnimi tabelami.
- `ISOLATION LEVEL` – pove stopnjo interakcije, ki jo SUPB dovoli med to in drugimi transakcijami.
  - `READ UNCOMMITTED`
  - `READ COMMITTED`
  - `REPEATABLE READ`
  - `SERIALIZABLE`
- Varen je samo način `SERIALIZABLE` (vrača serializabilne urnike)

Izolacijske ravni transakcij

Dirty Read

users

id	name	age
1	Joe	20
2	Jill	25

Transaction 1

```
/* Query 1 */
SELECT age FROM users WHERE id = 1;
/* will read 20 */
```

Transaction 2

```
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
/* No commit here */
```

```
/* Query 1 */
SELECT age FROM users WHERE id = 1;
/* will read 21 */
```

```
ROLLBACK; /* lock-based DIRTY READ */
```

Dirty Read se lahko zgodi pri izolaciji Read Uncommited

Izolacijske ravni transakcij

Non-repeatable read

Transaction 1

```
/* Query 1 */
SELECT * FROM users WHERE id = 1;
```

Transaction 2

```
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
COMMIT; /* in multiversion concurrency
   control, or lock-based READ COMMITTED */
```

```
/* Query 1 */
SELECT * FROM users WHERE id = 1;
COMMIT; /* lock-based REPEATABLE READ */
```

Non-repeatable read se lahko zgodi pri izolacijskih ravneh Read Committed, Read Uncommitted

Izolacijske ravni transakcij

Phantom read

Transaction 1

```
/* Query 1 */
SELECT * FROM users
WHERE age BETWEEN 10 AND 30;
```

Transaction 2

```
/* Query 2 */
INSERT INTO users VALUES ( 3, 'Bob', 27 );
COMMIT;
```

```
/* Query 1 */
SELECT * FROM users
WHERE age BETWEEN 10 AND 30;
```

Phantom read se lahko zgodi pri vseh izolacijskih ravneh razen pri Serializable

Izolacijske ravni transakcij

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommited	●	●	●
Read Commited		●	●
Repeatable Read			●
Serializable			

Delo s transakcijami v mySQL

- Sintaksa:

```
SET TRANSACTION [GLOBAL|SESSION]
   [READ ONLY | READ WRITE] |
   [ISOLATION LEVEL READ UNCOMMITTED |
   READ COMMITTED|REPEATABLE READ |SERIALIZABLE ]
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.02
```

```
mysql> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
ERROR 1568 (25001): Transaction characteristics can't be changed
while a transaction is in progress
```

Delo s transakcijami v mySQL

- Inicializacija transakcije

```
START TRANSACTION [ READ WRITE | READ ONLY]
SQL stavki
Eksplicitno zaključevanje transakcije
```

- Eksplicitno zaključevanje transakcije:

```
COMMIT | ROLLBACK
```

- Samodejno zaključevanje transakcije:

```
SET autocommit = {0 | 1}
```

Transakcije v mySQL Workbench

- Seja v MySQL Workbench

- Primer:

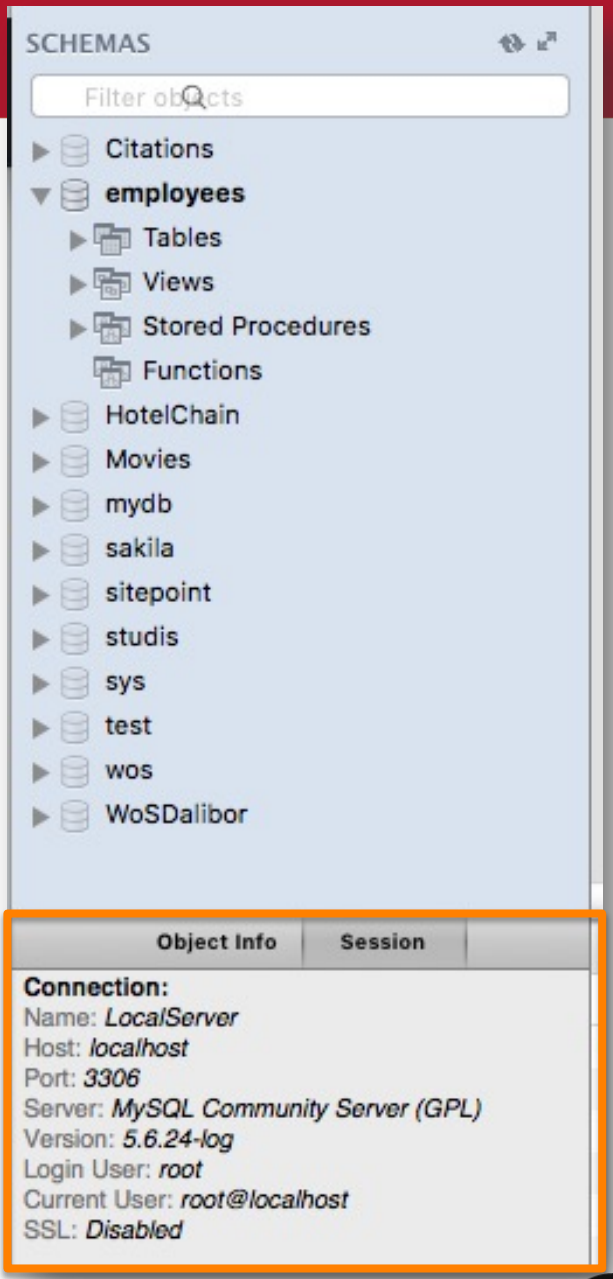
```
use employees;

select * from departments order by dept_no;

insert into departments (dept_no, dept_name)
values ('d011', 'Public relationships');

delete from departments where dept_no = 'd011';

start transaction;
insert into departments (dept_no, dept_name)
values ('d011', 'Public relationships');
rollback;
```



Takojšnje in zapoznele omejitve...

- Upoštevanje omejitev: ob začetku/po zaključku transakcije.
  - `INITIALLY IMMEDIATE` – ob začetku transakcije;
  - `INITIALLY DEFERRED` – ob zaključku transakcije.

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

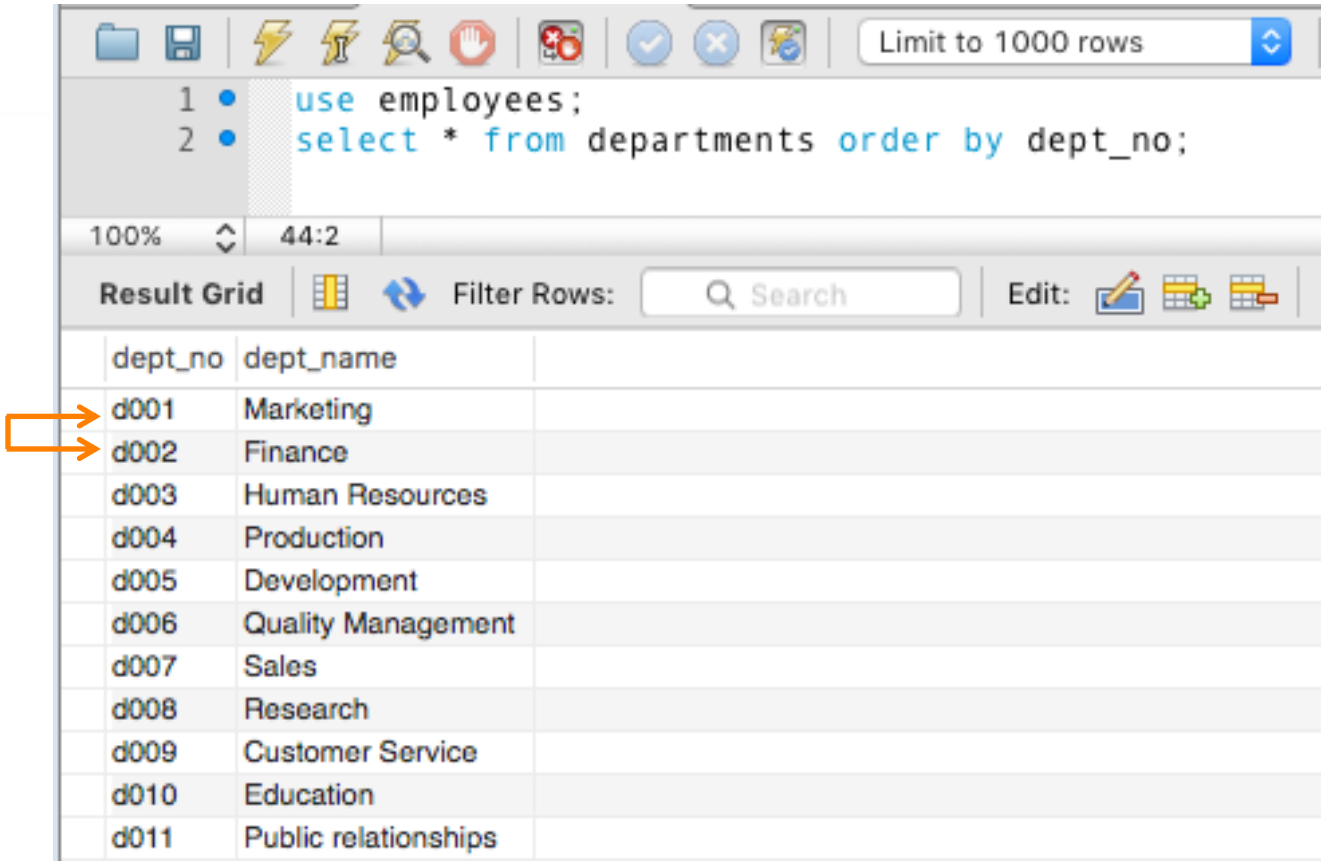
Omejitev: za vsakega delavca mora obstajati vsaj en zapis o plači.

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

Takojšnje in zapoznele omejitve...

- Primer: zamenjava ključa



dept_no	dept_name
d001	Marketing
d002	Finance
d003	Human Resources
d004	Production
d005	Development
d006	Quality Management
d007	Sales
d008	Research
d009	Customer Service
d010	Education
d011	Public relationships

Takojšnje in zapoznele omejitve

- Način upoštevanja omejitev za trenutno transakcijo nastavimo z ukazom `SET CONSTRAINTS`.

```
SET CONSTRAINTS
 {ALL | constraintName [, . . . ]}
 {DEFERRED | IMMEDIATE}
```

Takojšnje in zapoznele omejitve...

- Če izberemo `INITIALLY IMMEDIATE` (privzeta možnost), lahko določimo tudi, ali je zakasnitev moč določiti kasneje. Uporabimo `[NOT] DEFERRABLE`.

```
ALTER TABLE tab1 ADD CONSTRAINT fk_tab1_tab2
  FOREIGN KEY (tab2_id)
  REFERENCES tab2(id)
  DEFERRABLE
  INITIALLY IMMEDIATE;

ALTER SESSION SET CONSTRAINTS = DEFERRED;
ALTER SESSION SET CONSTRAINTS = IMMEDIATE;
```

MySQL

MySQL ne podpira zapozneli omejitev...

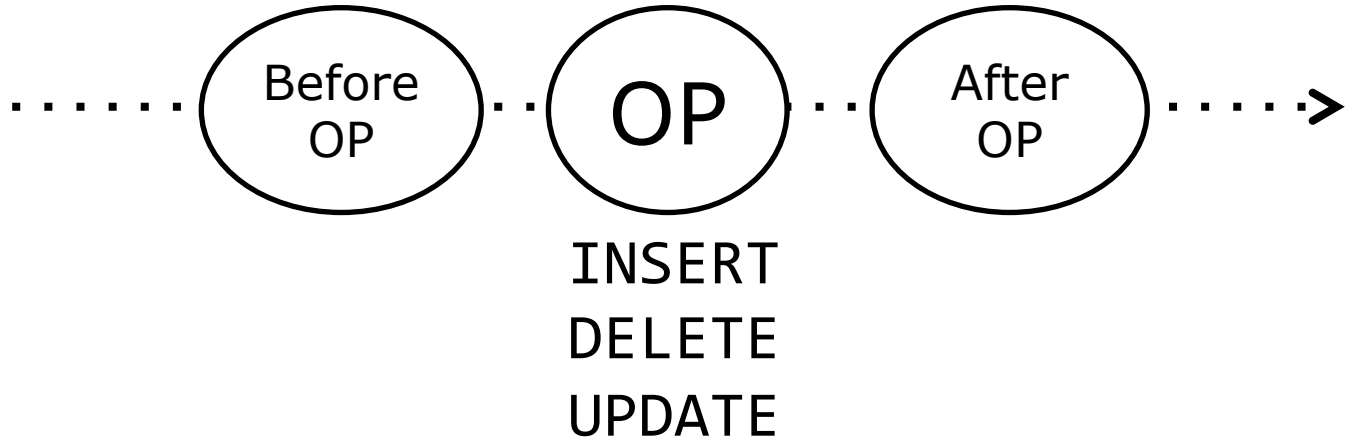
Bazni izvedbeni objekti

- Večina implementaciji SUPB podpira:
  - Sprožilce (*Trigger*)
  - V bazi shranjene podprograme (*Stored procedure*)
  - Funkcije (*Functions*)

- PROS:
  - Hitro izvajanje, izvedbeni objekti že prevedeni!
- CONS:
  - vključevanje poslovne logike na podatkovno raven.

Sprožilci...

- Se prožijo pred ali po opearciji INS, DEL, UPD



Sprožilci...

- Sintaksa v MySql:

```
CREATE TRIGGER trigger_name
trigger_time
trigger_event
ON tbl_name FOR EACH ROW
[trigger_order] trigger_body;

trigger_time = {BEFORE, AFTER}
trigger_event = {INSERT, UPDATE, DELETE}
Trigger_order = {FOLLOWS | PRECEDES} other_trigger_name
```

Sprožilci

- Primer sprožilca:
  - V tabelo `employees` dodamo polje `avgSalary`, ki ga spremenimo vsakokrat, ko se delavcu spremeni plača.

```
alter table employees
add column avgSalary decimal(10,2) null;

create trigger upd_avgSalary
after insert on salaries
for each row update employees set avgSalary = (
  select avg(salary) from salaries
  where emp_no = NEW.emp_no)
  where emp_no = NEW.emp_no;
```

Shranjeni podprogrami...

- Sintaksa v MySql

```
CREATE PROCEDURE sp_name
    ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE FUNCTION sp_name
    ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

Func_parameter, proc_parameter:
   [ IN | OUT | INOUT ] param_name type
```

Shranjeni podprogrami...

- Sintaksa v MySql

```
characteristic:
    COMMENT 'string'  |
    LANGUAGE SQL  |
    [NOT] DETERMINISTIC  |
    {CONTAINS SQL|NO SQL|READS SQL DATA|MODIFIES SQL DATA}|
    SQL SECURITY { DEFINER | INVOKER }
```

Shranjeni programi

- Primer shranjene funkcije in njenega klica v sintaksi mySQL:

```
create function hello (empno INT(5))
    returns char(50) deterministic
    return concat('Pozdravljen ',
        (select first_name
        from employees
        where emp_no = empno));

mysql> SELECT hello(10001);
mysql> Pozdravljen Georgi
```

Shranjeni programi

- Primer shranjene procedure in njenega klica v sintaksi mySQL:

```
create procedure DoubleIt (INOUT p1 INT)
begin
    select p1*p1 into p1;
end;
```

```
mysql> set @a = 10;
mysql> call DoubleIt(@a);
mysql> select @a;
mysql> 100
```

```
select DoubleIt(@a);
ERROR 1305 (42000):
FUNCTION employees.DoubleIt
does not exist
```

Vaja

- Izpiši naslednje podatke o zaposlenih:
  - Ime
  - Priimek
  - Starost
- Če gre za žensko, ne izpiši dejanske starosti, temveč starost v obliki:
  - Če starost manjša od 60 let ⇒ '`Pod 60`', sicer '`Nad 60`';

first_name	last_name	Starost
Georgi	Facello	64
Bezalel	Simmel	Pod 60
Parto	Bamford	58
Chirstian	Koblick	63
Kyoichi	Maliniak	62
Anneke	Preusig	Nad 60
Tzvetan	Zielinski	Pod 60
Saniya	Kalloufi	59
Sumant	Peac	Nad 60
Duangkaew	Piveteau	Pod 60
Mary	Sluis	Nad 60
Patricio	Bridgland	57

Vaja

```
create function empAge (empno int(5))
returns int(3) not deterministic reads sql data
begin
  return (select YEAR(now()) - YEAR(birth_date)
  from employees where emp_no = empno);
end
//delimiter ;
```

#Primer uporabe funkcije

```
select e.first_name, e.last_name,
if(gender='F',(if (empAge(e.emp_no)>60, 'Nad 60', 'Pod
60')), empAge(e.emp_no)) as 'Starost' from employees e;
```

```
IF(condition, A, B);
```

Če je pogoj izpolnjen, vrni A sicer B.

Dostopanje do MySQL iz programskih jezikov

- Primer v jeziku Java
- Primer v jeziku Ruby
- Primer v jeziku Python







Primer v jeziku Java




```java
package dbtest;

import java.io.*;
import java.sql.*;
import java.util.*;

/**
 *
 * @author MarkoB
 */
public class DBTest {

    private static final String PROP_FILE = "db.properties";

    //supporting method for making a connection
    private static Connection getConnection() throws
            IOException,
            ClassNotFoundException,
            InstantiationException,
            IllegalAccessException,
            SQLException {

    //settings are read from a file
    Properties p = new Properties();
    p.load(new FileInputStream(PROP_FILE));
    //dynamicaly loding the driver
    String driver = p.getProperty("jdbc.driver");
    Class.forName(driver).newInstance();

    //opening connection to datasource
    return DriverManager.getConnection(
            p.getProperty("jdbc.conn"),
            p.getProperty("jdbc.uid"),
            p.getProperty("jdbc.pwd")
            );
    }
```

Vsebina datoteke db.properties

```
//gonilnik
jdbc.driver=com.mysql.jdbc.Driver
//baza, do katere želimo dostopati
jdbc.conn=jdbc:mysql://localhost/citations
//uporabniško ime
jdbc.uid=root
//geslo – če pustimo prazno, nas vpraša
jdbc.pwd=
```

```
public static void main(String[] args) {
    try {
        //open connection
        System.out.println("odpiram povezavo");
        Connection dbconn = getConnection();
        System.out.println("povezava odprta");

        Statement s = dbconn.createStatement();
        System.out.println("INSERT stavek kreiran");

        //inserting a new row in table Author
        s.executeUpdate("insert into Author (AuthorID, AuthorName, "
                + "AuthorSurname) values (null, 'Marko', 'Bajec')");
        System.out.println("INSERT izveden");
        s.close();
        dbconn.close();
        System.out.println("povezava zaprta");

    } catch (SQLException e) {
        while (e != null) {
            e.printStackTrace();
            e = e.getNextException();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }


}
```

Primer v jeziku Ruby



```ruby
#!/usr/bin/ruby

require "mysql"

begin

    con = Mysql.new 'localhost', 'user12', '34klq*', 'mydb'

    rs = con.query("SELECT * FROM Writers")
    n_rows = rs.num_rows

    puts "There are #{n_rows} rows in the result set"

    n_rows.times do
        puts rs.fetch_row.join("\s")
    end

rescue Mysql::Error => e
    puts e.errno
    puts e.error

ensure
    con.close if con
end
```



Primer v jeziku Python

```
import datetime
import mysql.connector

cnx = mysql.connector.connect(user='scott', database='employees')
cursor = cnx.cursor()

query = ("SELECT first_name, last_name, hire_date FROM employees "
         "WHERE hire_date BETWEEN %s AND %s")

hire_start = datetime.date(1999, 1, 1)
hire_end = datetime.date(1999, 12, 31)

cursor.execute(query, (hire_start, hire_end))

for (first_name, last_name, hire_date) in cursor:
  print("{}, {} was hired on {:%d %b %Y}".format(
    last_name, first_name, hire_date))

cursor.close()
cnx.close()
```

Koristne povezave

- MySQL SUPB
  - http://www.mysql.com
- employees DB
  - https://launchpad.net/test-db/+download
- SQL checker
  - http://developer.mimer.com/validator/parser92/index.tml
- Nekatera MySQL orodja
  - SQLeo: http://sourceforge.net/projects/sqleo/
  - Sequel Pro: http://www.sequelpro.com
  - Mac SQL studio: http://macsqlstudio.com

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Predmet:
Osnove podatkovnih baz

Modul:
Poizvedovanje s QBE

Gradivo:
v.2015

Vsebina

- Kaj je QBE
- Poizvedovanje s QBE v MS Access
- QBE orodja za MySQL

QBE – Query-By-Example…

- Vizualen način dostopa do podatkov s podajanjem primerov…povemo, kaj nas zanima.
- QBE originalno razvil IBM v 70' letih. Včasih zelo popularen. Ponujajo mnogi SUPB.
- Možno prehajanje iz vizualne v tekstovno poizvedbo in obratno.

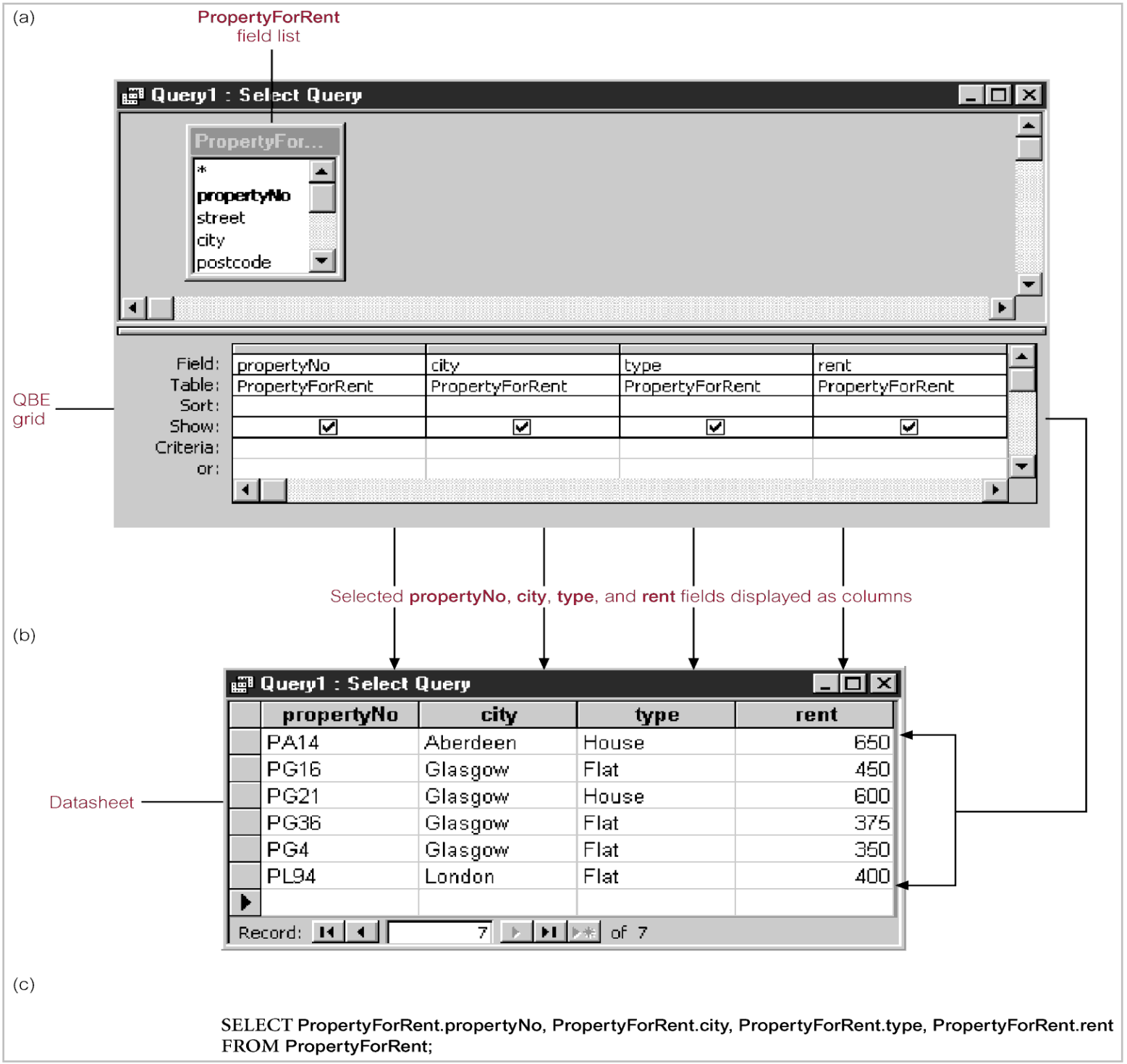
Primer QBE v MS ACCESS...

- QBE v MS Access uporabnikom omogoča:
  - Poizvedovati po podatkih ene ali več tabel.
  - Določiti stolpce, ki jih želimo imeti v odgovoru (projekcija).
  - Določiti kriterije za izbiro vrstic (selekcija).
  - Izvajati izračune nad podatki v tabelah.
  - Dodajati in brisati zapise.
  - Spreminjati vrednosti v poljih.
  - Kreirati nove tabele in stolpce.
  - Izvajati posebne poizvedbe.

Primer QBE v MS ACCESS...

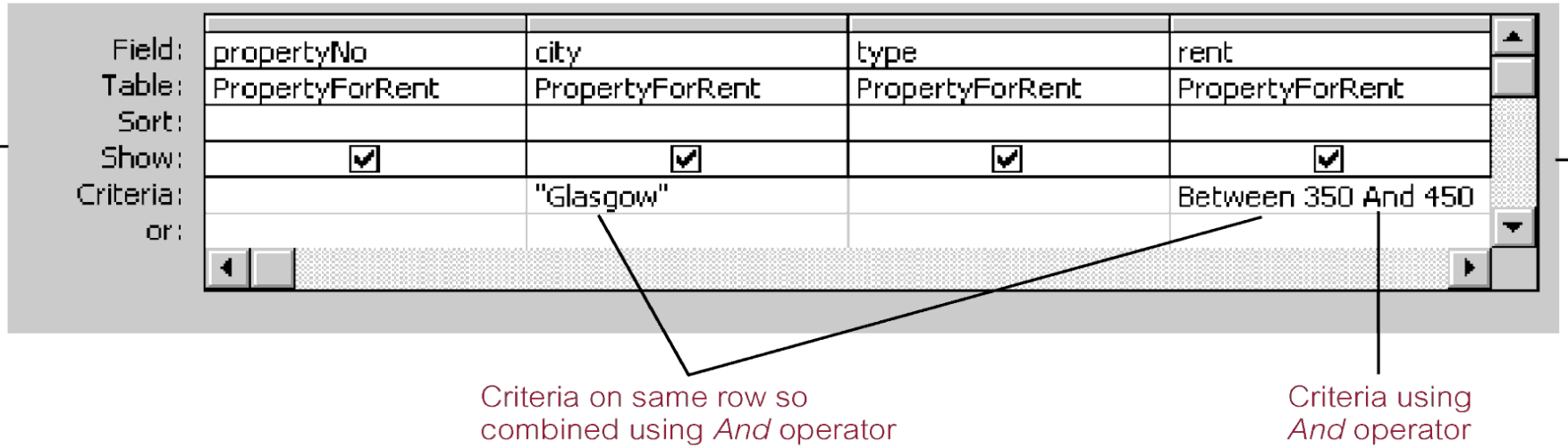
- Primeri iz Accessa

Poizvedba SELECT

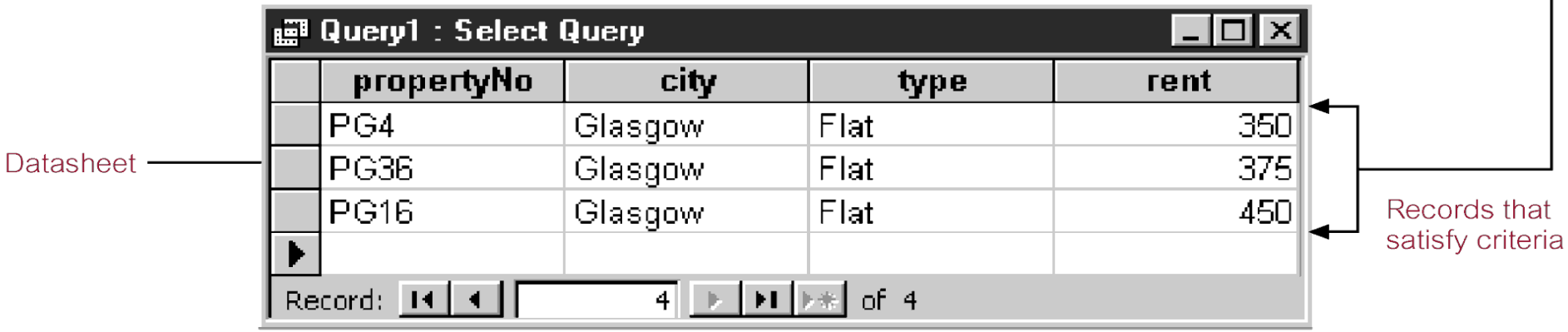


Poizvedba SELECT, določitev kriterija

(a)



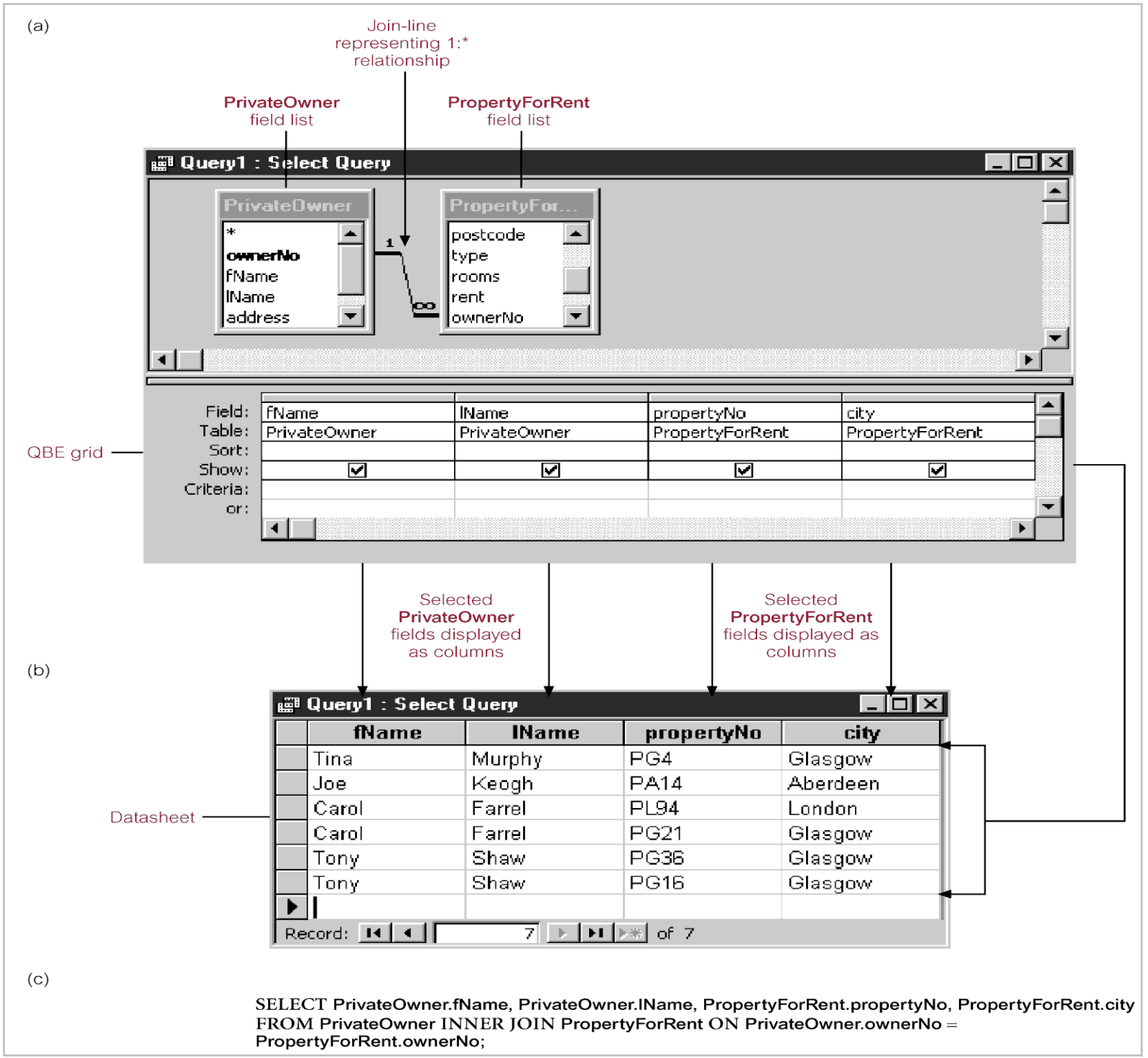
(b)



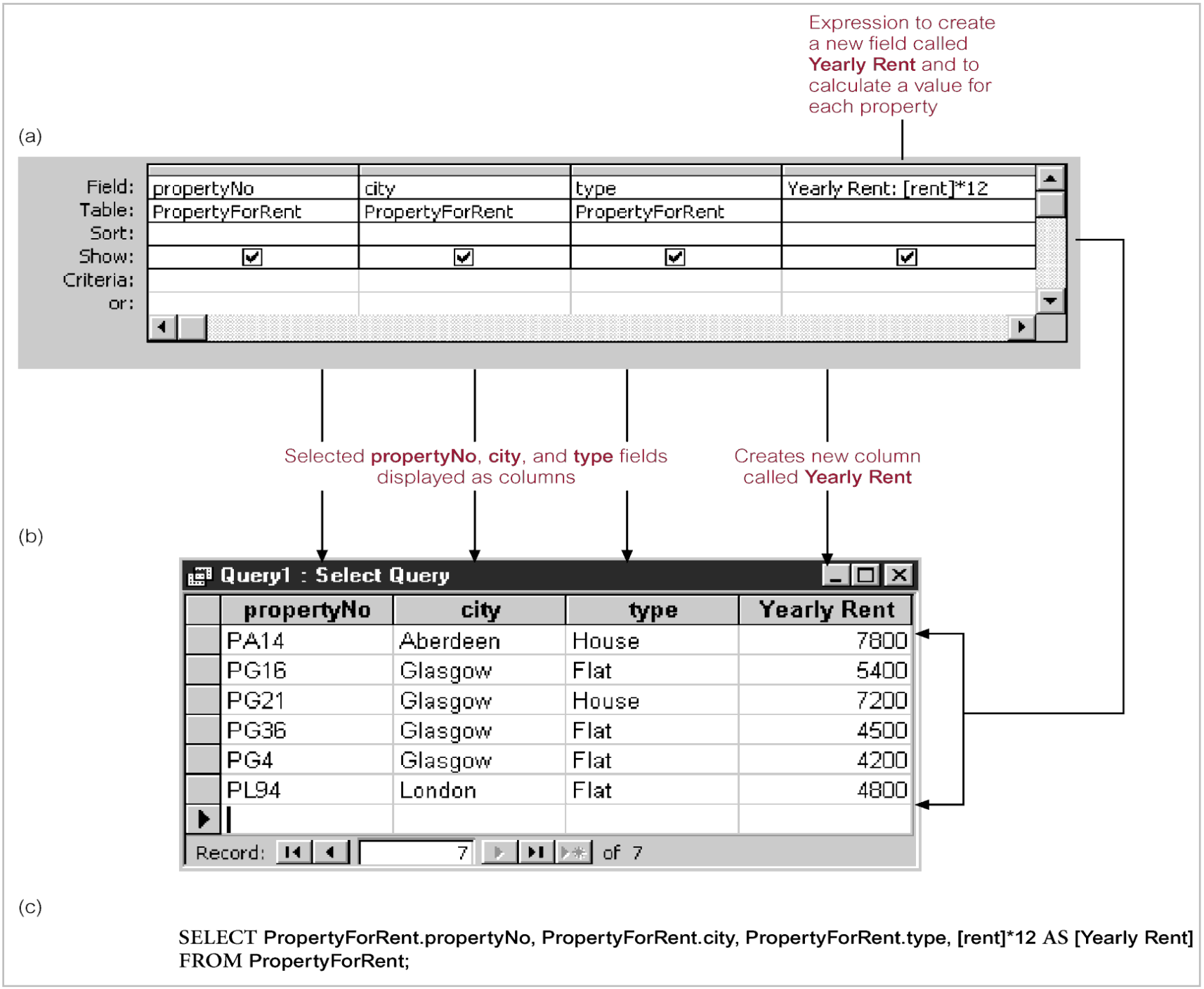
(c)

```
SELECT PropertyForRent.propertyNo, PropertyForRent.city, PropertyForRent.type, PropertyForRent.rent
FROM PropertyForRent
WHERE (((PropertyForRent.city)="Glasgow") AND ((PropertyForRent.rent) Between 350 And 450));
```

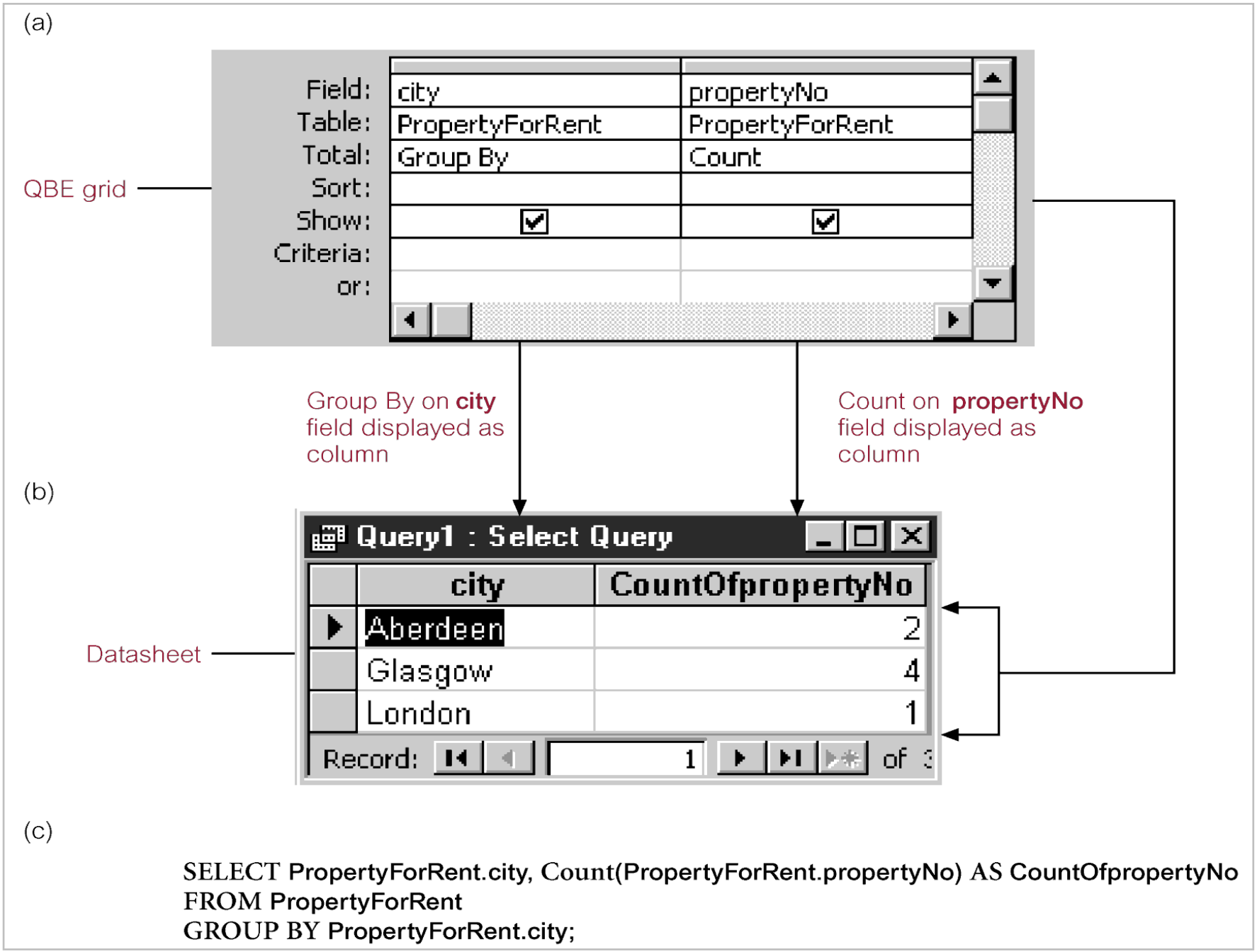
Poizvedba SELECT, stik tabel



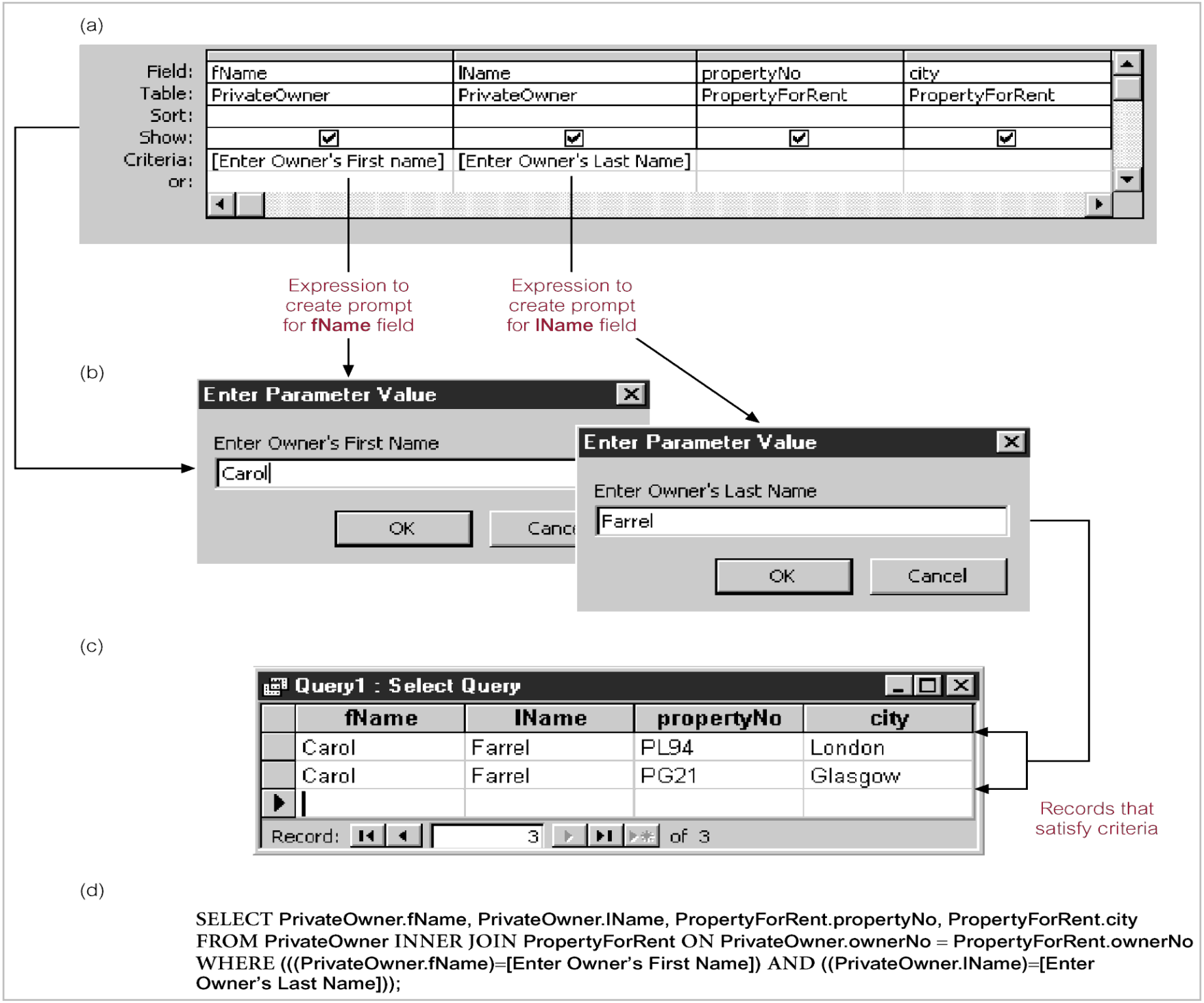
Poizvedba SELECT, izračunana polja



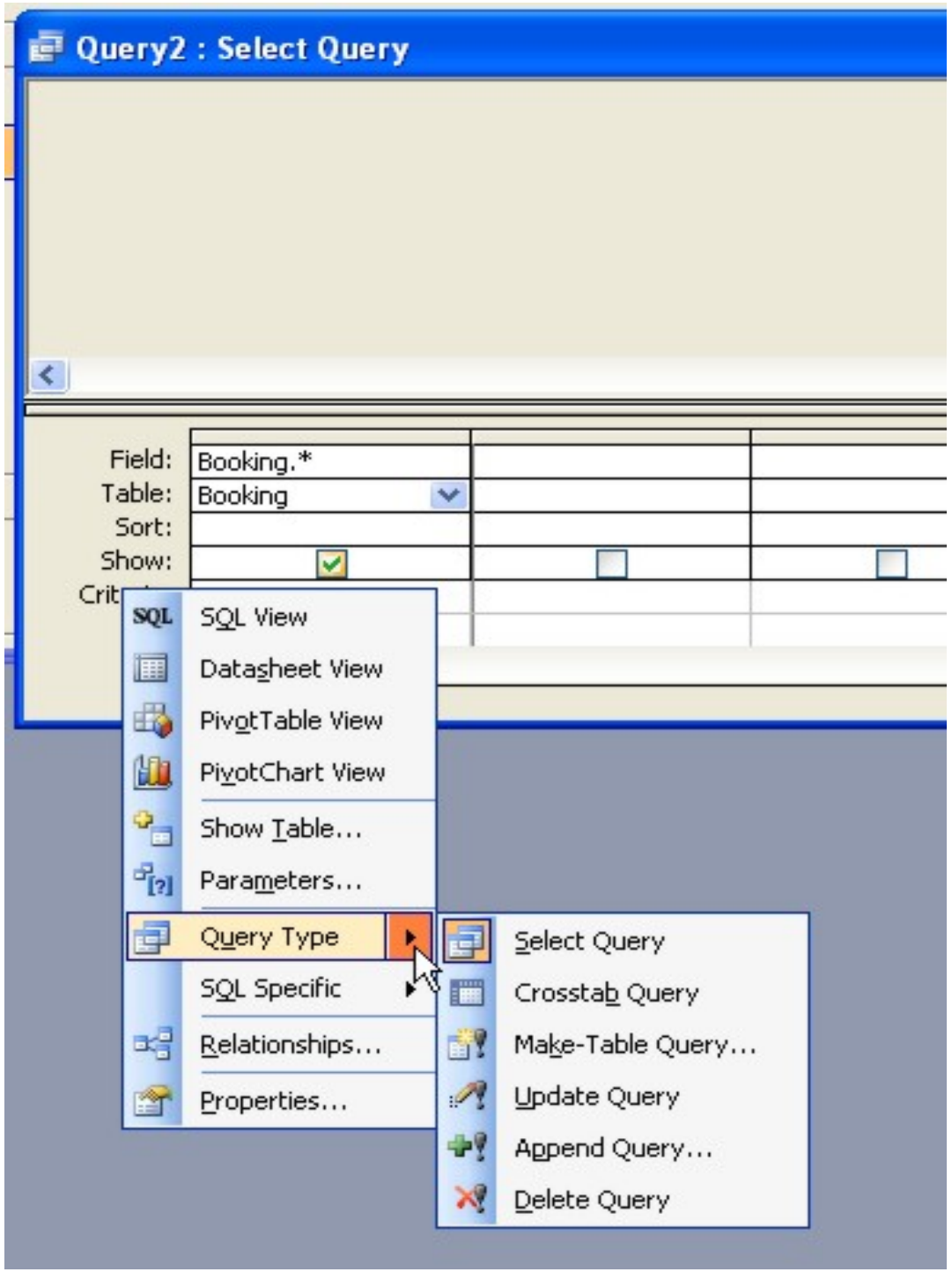
Poizvedba SELECT, uporaba agregatov



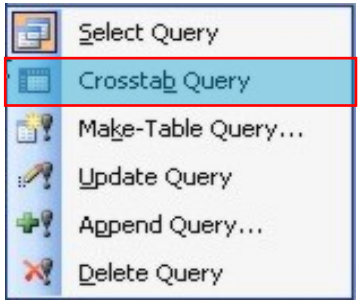
Poizvedba SELECT, uporaba parametrov



Posebne vrste SELECT poizvedb



Poizvedba Crosstab



(a)

Field:	fName	lName	type	propertyNo
Table:	Staff	Staff	PropertyForRent	PropertyForRent
Total:	Group By	Group By	Group By	Count
Sort:				
Show:	☑	☑	☑	☑
Criteria:				
or:				

Group By on **fName**, **lName**, and **type** fields displayed as columns

Count on **propertyNo** field displayed as column

(b)

Query1 : Select Query

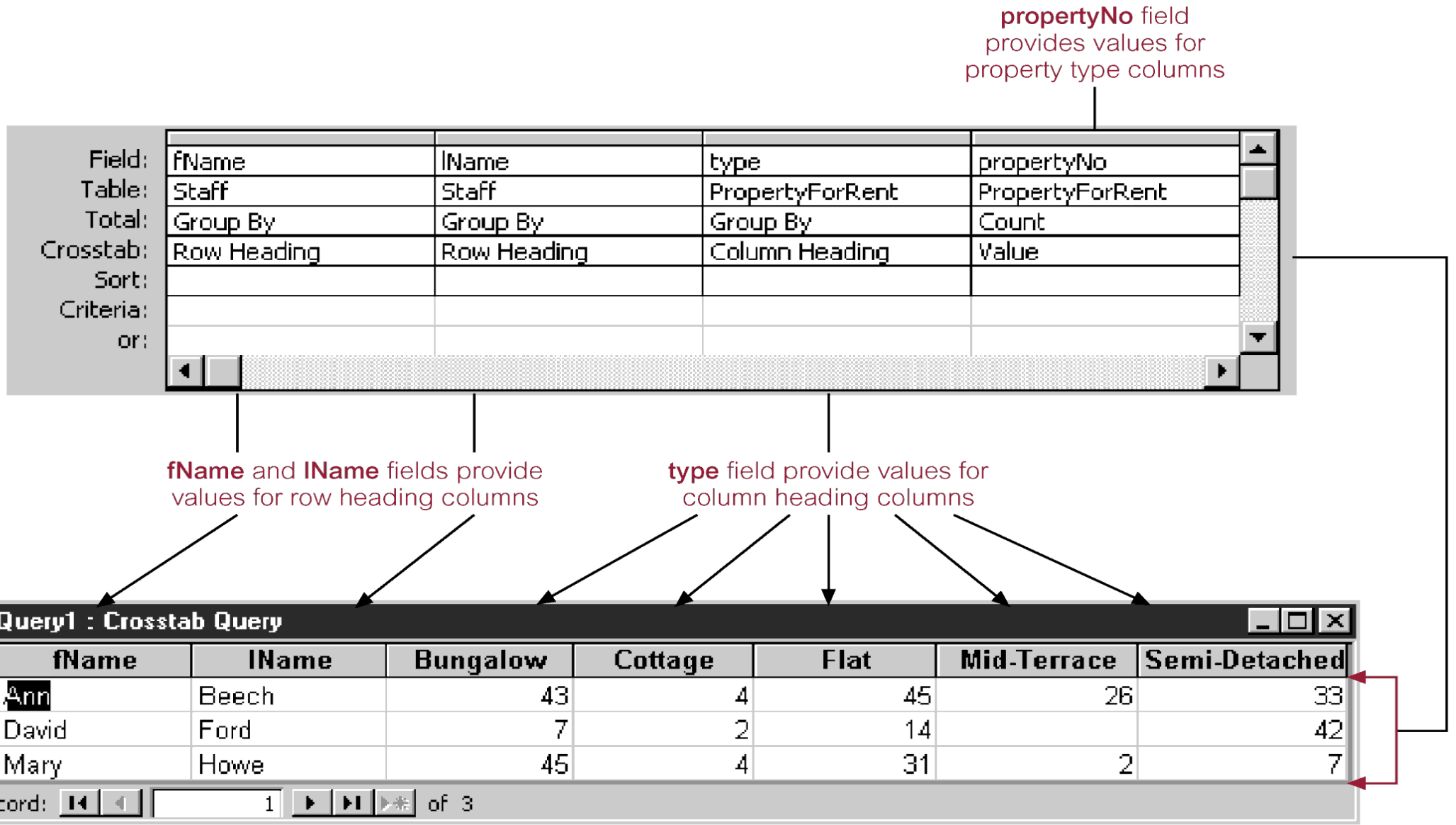
	fName	lName	type	CountOfpropertyNo
▶	Ann	Beech	Bungalow	43
	Ann	Beech	Cottage	4
	Ann	Beech	Flat	45
	Ann	Beech	Mid-Terrace	26
	Ann	Beech	Semi-Detached	33
	David	Ford	Bungalow	7
	David	Ford	Cottage	2
	David	Ford	Flat	14
	David	Ford	Semi-Detached	42
	Mary	Howe	Bungalow	45
	Mary	Howe	Cottage	4
	Mary	Howe	Flat	31
	Mary	Howe	Mid-Terrace	2
	Mary	Howe	Semi-Detached	7

Record: 1 of 14

(c)

```
SELECT Staff.fName, Staff.lName, PropertyForRent.type, Count(PropertyForRent.propertyNo) AS
CountOfpropertyNo
FROM Staff INNER JOIN PropertyForRent ON Staff.staffNo = PropertyForRent.staffNo
GROUP BY Staff.fName, Staff.lName, PropertyForRent.type;
```

(a)



(b)

Query1 : Crosstab Query

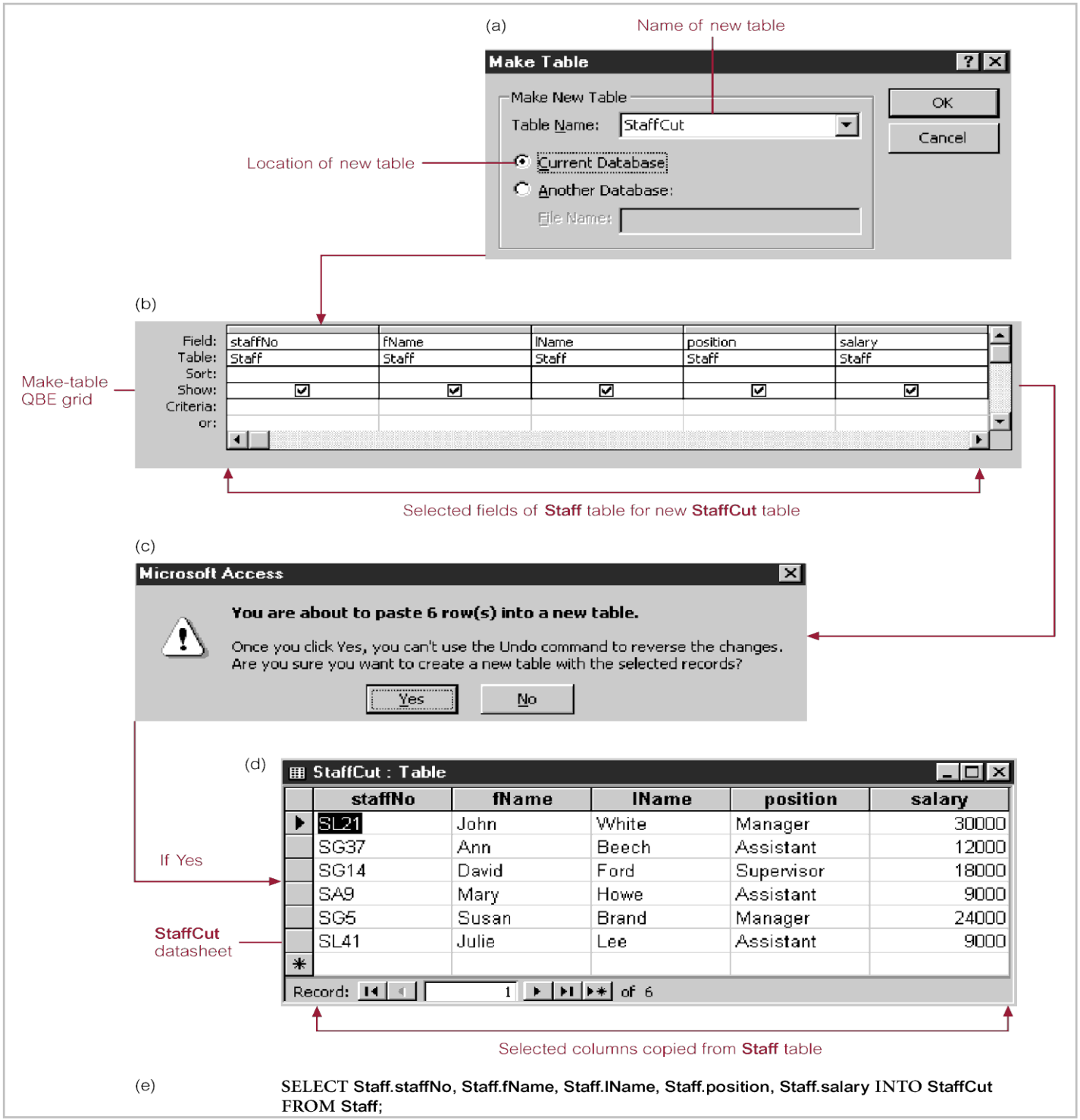
fName	lName	Bungalow	Cottage	Flat	Mid-Terrace	Semi-Detached
Ann	Beech	43	4	45	26	33
David	Ford	7	2	14		42
Mary	Howe	45	4	31	2	7

Record: 1 of 3

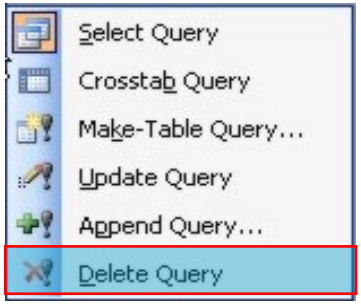
(c)

```
TRANSFORM Count(PropertyForRent.propertyNo) AS CountOfpropertyNo
SELECT Staff.fName, Staff.lName
FROM Staff INNER JOIN PropertyForRent ON Staff.staffNo = PropertyForRent.staffNo
GROUP BY Staff.fName, Staff.lName
PIVOT PropertyForRent.type;
```

Kreiranje novih tabel



Brisanje zapisov



(a)

Target table for deletion

Delete QBE grid

Field:	PropertyForRent.*	city
Table:	PropertyForRent	PropertyForRent
Delete:	From	Where
Criteria:		"Glasgow"
or:		

Criterion for properties in Glasgow

(b)

Microsoft Access

You are about to delete 4 row(s) from the specified table.

Once you click Yes, you can't use the Undo command to reverse the changes.
Are you sure you want to delete the selected records?

Yes No

If Yes

(c)

PropertyForRent : Table

propertyNo	street	city	postcode
PA14	16 Holhead	Aberdeen	AB7 5SU
PL94	6 Argyll Street	London	NW2

Record: 1 of 2

PropertyForRent datasheet minus deleted records (properties in Glasgow)

Deletion cascades to related table if referential integrity is set and cascade deletes are allowed

Viewing : Table

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-01	too small
CR62	PA14	14-May-01	no dining room

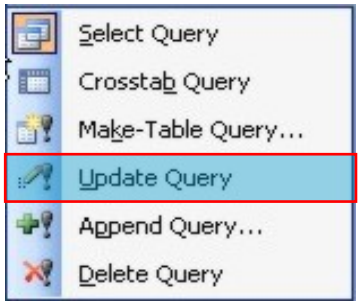
Record: 1 of 2

Viewing datasheet minus deleted records (viewings of properties) in Glasgow)

(d)

```
DELETE PropertyForRent.*, PropertyForRent.city
FROM PropertyForRent
WHERE (((PropertyForRent.city)="Glasgow"));
```

Spreminjanje zapisov



(a)

Target table for update

Field: rent
Table: PropertyForRent
Update To: [rent]*1.1
Criteria:
or:

Update To row

Expression to update values in **rent** field by 10%

Update QBE grid

(b)

Microsoft Access

You are about to update 6 row(s).

Once you click Yes, you can't use the Undo command to reverse the changes.
Are you sure you want to update these records?

Yes | No

If Yes

(c)

Query1 : Select Query

propertyNo	street	city	rent
PA14	16 Holhead	Aberdeen	715
PG16	5 Novar Drive	Glasgow	495
PG21	18 Dale Road	Glasgow	660
PG36	2 Manor Road	Glasgow	412
PG4	6 Lawrence Street	Glasgow	385
PL94	6 Argyll Street	London	440

Record: 1 of 6

Values updated in **rent** column by 10%

(d)

```
UPDATE PropertyForRent SET PropertyForRent.rent = [rent]*1.1;
```

QBE orodja za MySQL

- phpMyAdmin 

- dbForge Studio for MySQL
  http://www.devart.com/dbforge/mysql/ 

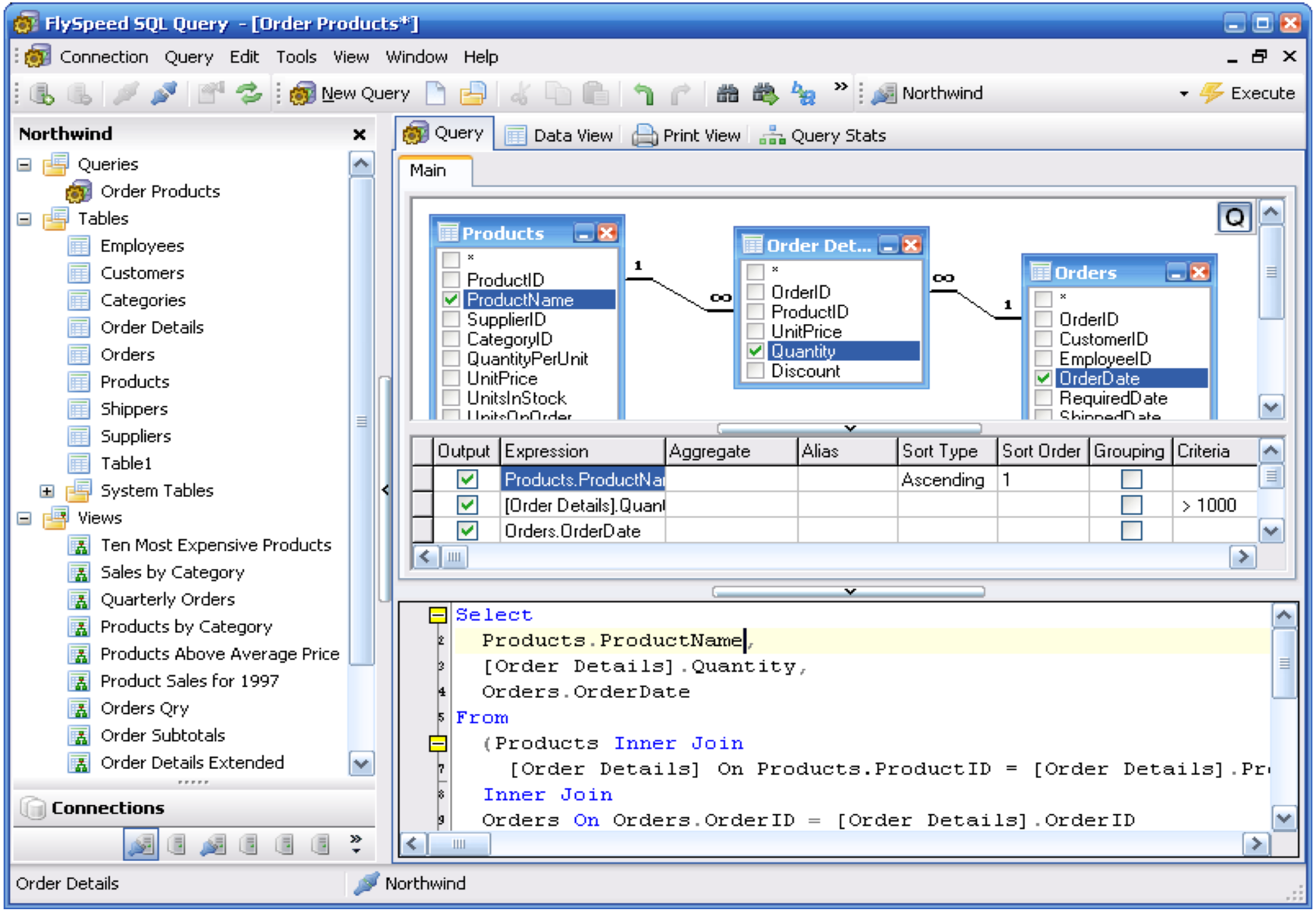
- FlySpeed SQL Query
  http://www.activedbsoft.com/index.html 

- SQLyog MySQL
  https://www.webyog.com/ 

- …

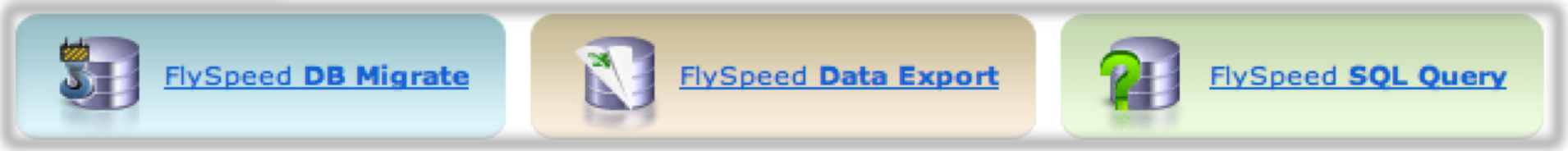
dbForge Studio for MySQL

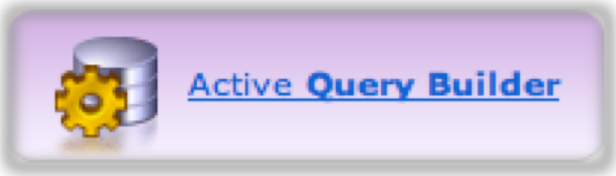



FlySpeed SQL Query - [Order Products*]
Connection Query Edit Tools View Window Help
New Query
Northwind
Execute
Northwind
Queries
Order Products
Tables
Employees
Customers
Categories
Order Details
Orders
Products
Shippers
Suppliers
Table1
System Tables
Views
Ten Most Expensive Products
Sales by Category
Quarterly Orders
Products by Category
Products Above Average Price
Product Sales for 1997
Orders Qry
Order Subtotals
Order Details Extended
Connections
Query
Data View
Print View
Query Stats
Main
Products
ProductID
ProductName
SupplierID
CategoryID
QuantityPerUnit
UnitPrice
UnitsInStock
Order Det...
OrderID
ProductID
UnitPrice
Quantity
Discount
Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
Output
Expression
Aggregate
Alias
Sort Type
Sort Order
Grouping
Criteria
Products.ProductNa
Ascending
1
[Order Details].Quant
> 1000
Orders.OrderDate
Select
Products.ProductName,
[Order Details].Quantity,
Orders.OrderDate
From
(Products Inner Join
[Order Details] On Products.ProductID = [Order Details].Pr
Inner Join
Orders On Orders.OrderID = [Order Details].OrderID
Order Details
Northwind

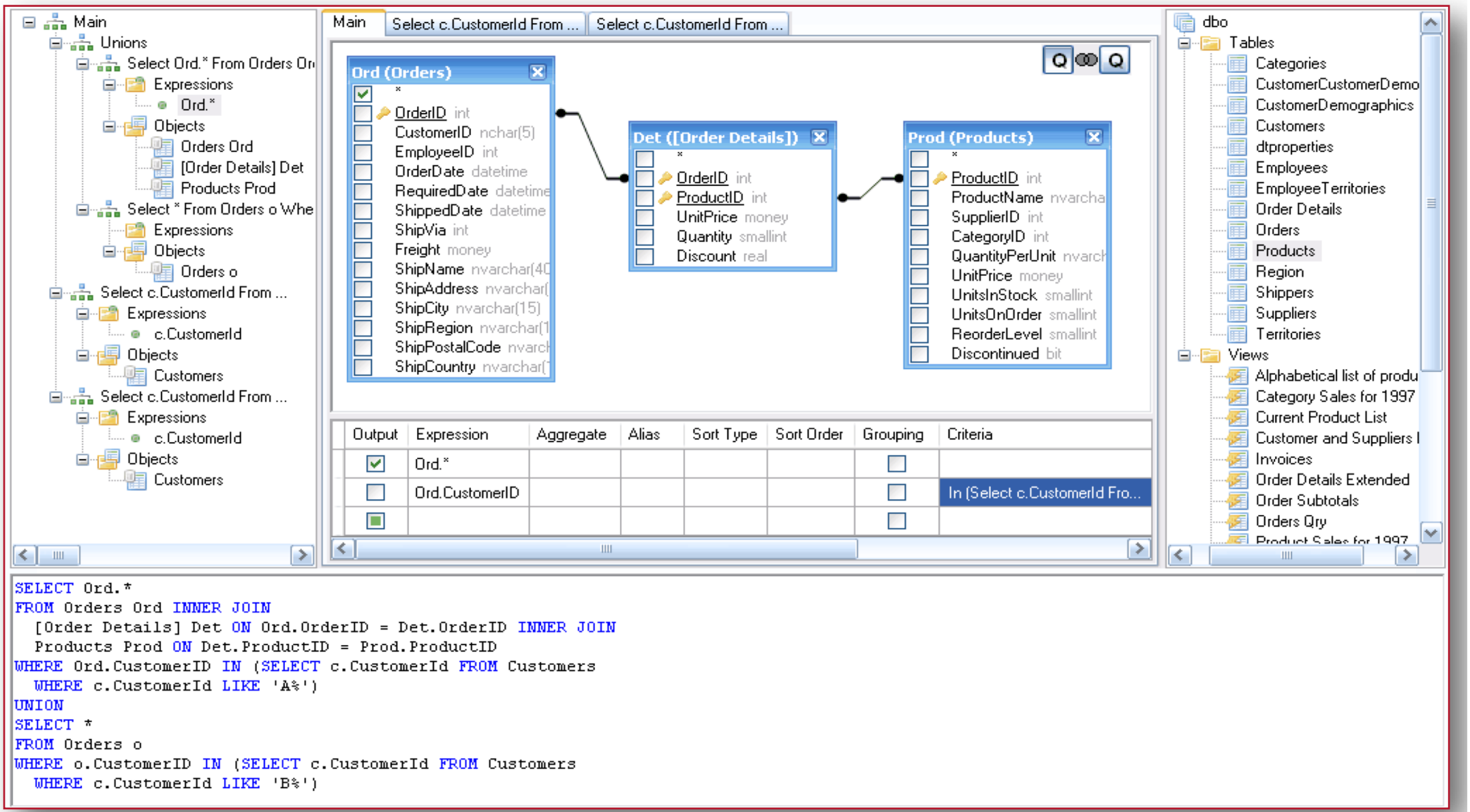
FlySpeed SQL Query







FlySpeed SQL Query




```
SELECT Ord.*
FROM Orders Ord INNER JOIN
  [Order Details] Det ON Ord.OrderID = Det.OrderID INNER JOIN
  Products Prod ON Det.ProductID = Prod.ProductID
WHERE Ord.CustomerID IN (SELECT c.CustomerId FROM Customers
  WHERE c.CustomerId LIKE 'A%')
UNION
SELECT *
FROM Orders o
WHERE o.CustomerID IN (SELECT c.CustomerId FROM Customers
  WHERE c.CustomerId LIKE 'B%')
```

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



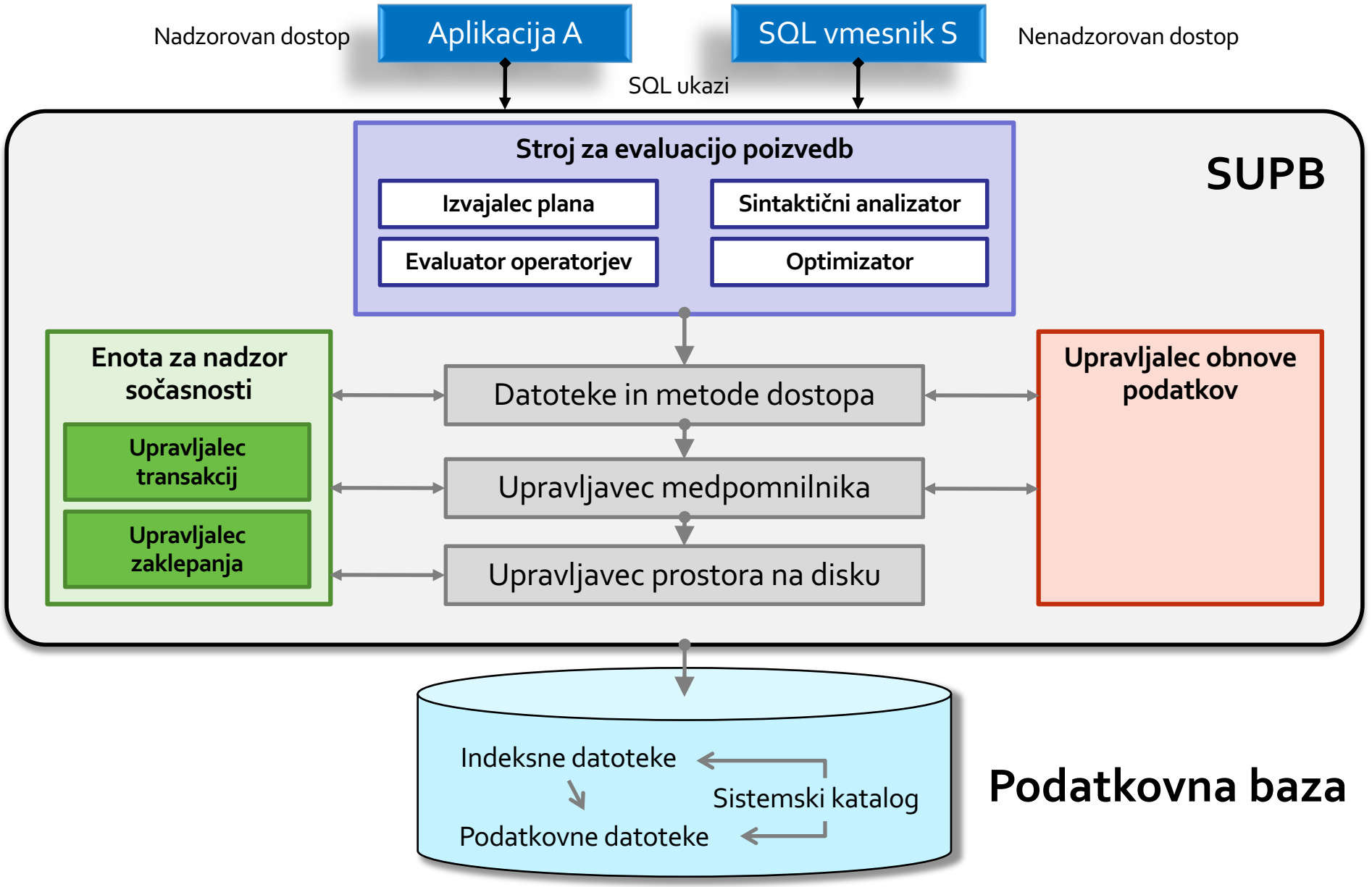
Predmet:
Osnove podatkovnih baz

Modul:
Diski in diskovna polja

Gradivo:
v.2015

Vsebina

- Zgradba SUPB, komponente SUPB za delo s podatki
- Primarni, sekundarni in terciarni pomnilnik
- Magnetni disk
- Polje diskov


Nadzorovan dostop
Aplikacija A
SQL vmesnik S
Nenadzorovan dostop
SQL ukazi
SUPB
Stroj za evaluacijo poizvedb
Izvajalec plana
Sintaktični analizator
Evaluator operatorjev
Optimizator
Enota za nadzor sočasnosti
Upravljalec transakcij
Upravljalec zaklepanja
Datoteke in metode dostopa
Upravljavec medpomnilnika
Upravljavec prostora na disku
Upravljalec obnove podatkov
Indeksne datoteke
Sistemski katalog
Podatkovne datoteke
Podatkovna baza

Komponente SUPB za delo s podatki…

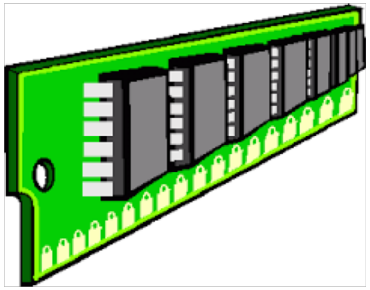
- Podatki iz PB se hranijo na diskih (in trakovih).

- Upravljalec prostora na disku (*Disk Space Manager*):
  - upravlja s prostorom na disku,
  - ukaze v zvezi z zaseganjem/sproščanjem prostora prejema od upravljavca z datotekami.
- Upravljalec z datotekami (*File Manager*):
  - Posreduje zahteve za zaseganje/sproščanje prostora na disku v enotah – straneh.
  - odgovoren za upravljanje strani znotraj datoteke, za urejanje zapisov znotraj strani.
  - Velikost strani eden od parametrov SUPB (tipično 4 - 8 KB).

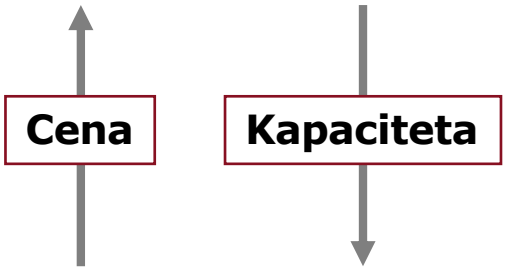
Komponente SUPB za delo s podatki

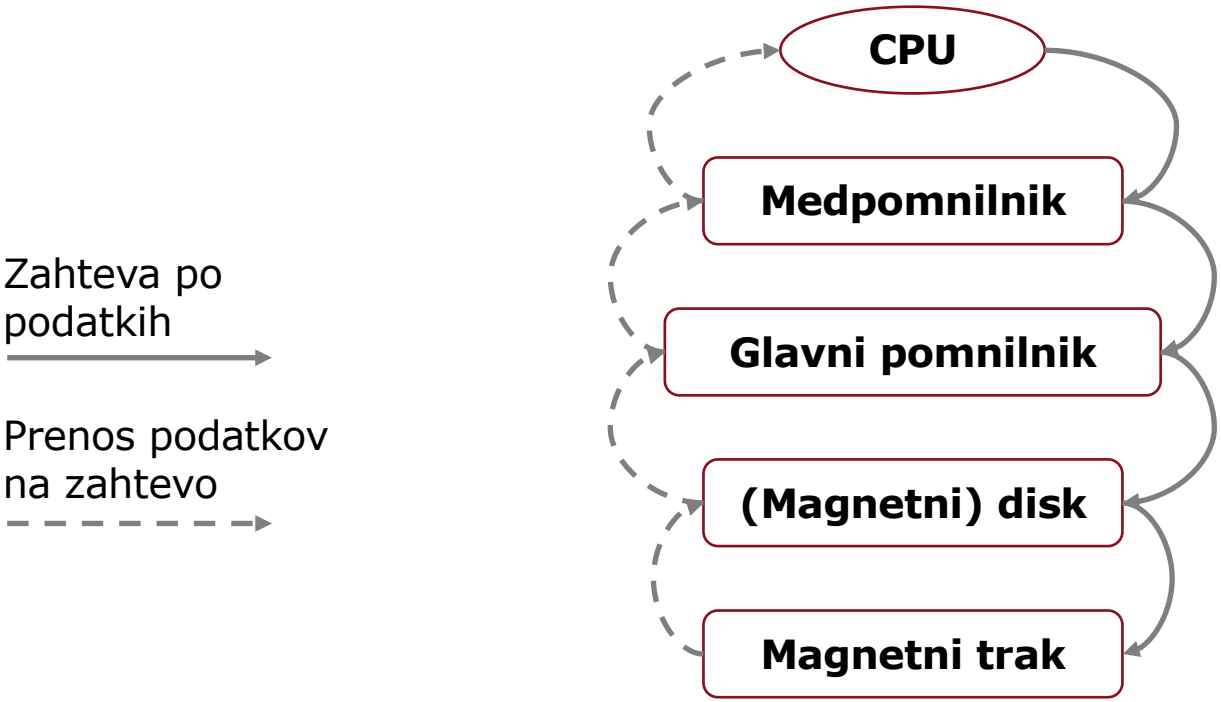
- Upravljalec medpomnilnika (*Buffer manager*):
  - prenos strani iz diska v medpomnilnik (*buffer pool*).
  - stran, kjer je zapis, poišče upravljalec z datotekami.
  - prenos v medpomnilnik izvede upravljalec medpomnilnika.





Hierarhija pomnilnika…



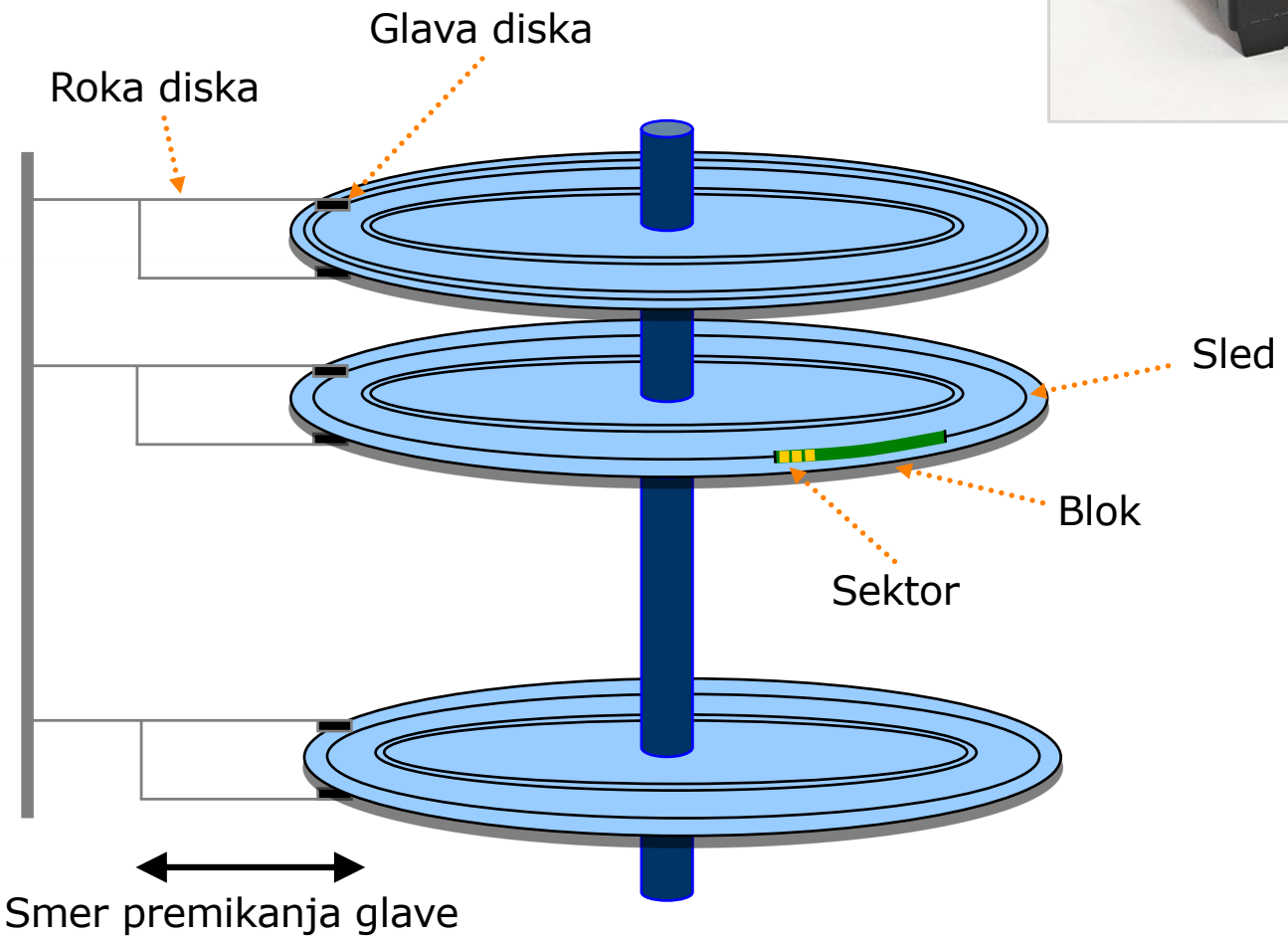


Hierarhija pomnilnika

- Razlogi za shranjevanje podatkov v sekundarnem in terciarnem pomnilniku:
  - Obstojnost podatkov
  - Cena na enoto
  - Omejen naslovni prostor primarnega pomnilnika (2^{32}=4Gb podatkov...)

Magnetni disk





Magnetni disk

- Povprečni dostopni čas:
  - iskalni čas
  - rotacijska zakasnitev
  - čas prenosa

- Organizacija podatkov na disku vpliva na povprečni dostopni čas!

- Čas prenosa običajno večji od časa obdelave → pomembna organizacija strani…

- Dostopni čas RAM : disk ≈ 1 : 1000

HDD in SSD diski…

- HDD – *Hard Disk Drive*
- SSD – *Solid State Drive*



SSD Top Side



SSD Bottom Side



HDD Top Side



HDD Bottom Side

HDD in SSD diski

Attribute	SSD (Solid State Drive)	HDD (Hard Disk Drive)
Power Draw / Battery Life	Less power draw, averages 2 – 3 watts, resulting in 30+ minute battery boost ✓	More power draw, averages 6 – 7 watts and therefore uses more battery
Cost	Expensive, $1.00 per gigabyte (based on buying a 240GB drive)	Only around $0.075 per gigabyte, very cheap (buying a 4TB model) ✓
Capacity	Typically not larger than 512GB for notebook size drives	Typically 500GB – 2TB for notebook size drives ✓
Operating System Boot Time	Around 22 seconds average bootup time ✓	Around 40 seconds average bootup time
Noise	There are no moving parts and as such no sound ✓	Audible clicks and spinning can be heard
Vibration	No vibration as there are no moving parts ✓	The spinning of the platters can sometimes result in vibration
Heat Produced	Lower power draw and no moving parts so little heat is produced ✓	HDD doesn't produce much heat, but it will have a measurable amount more heat than an SSD due to moving parts and higher power draw
Failure Rate	Mean time between failure rate of 2.0 million hours ✓	Mean time between failure rate of 1.5 million hours
File Copy / Write Speed	Generally above 200 MB/s and up to 500 MB/s for cutting edge drives ✓	The range can be anywhere from 50 – 120MB / s
Encryption	Full Disk Encryption (FDE) Supported on some models ✓	Full Disk Encryption (FDE) Supported on some models ✓
File Opening Speed	Up to 30% faster than HDD ✓	Slower than SSD
Magnetism Affected?	An SSD is safe from any effects of magnetism ✓	Magnets can erase data

Polje diskov…

- Disk potencialno ozko grlo za učinkovitost SUPB … vpliva na zanesljivost delovanja sistema.

- Učinkovitost CPU/disk:
  - CPU: 50% na leto
  - Diski: 10% na leto
- Diski mehanske naprave → verjetnost za napake večja kot pri notranjem pomnilniku.
- Odpoved diska … katastrofa.
- Možna rešitev: več diskov.

Polje diskov

- Polje diskov: povezava več diskov z namenom:
  - povečanja učinkovitosti in/ali
  - izboljšanja zanesljivosti.

- Učinkovitost … porazdelitev podatkov (*data striping*)
- Zanesljivost … podvajanje podatkov - redundanca

RAID

- RAID – *Redundant Arrays of Independent Disks* – diskovna polja, ki implementirajo porazdelitev/ podvajanje podatkov.

- Več vrst RAID … razlika v kompromisu med učinkovitostjo in zanesljivostjo.



RAID s porazdelitvijo podatkov

- Uporabniku se kaže kot velik disk.
- Podatki se razdelijo na enake enote (*striping units*), ki se zapišejo na več diskov. Vsaka enota na en disk.
- Enote se po diskih porazdelijo po "*round robin*" algoritmu: če polje vključuje `D` diskov, se enota `i` zapiše na "`i mod D`" disk.

Primer 1

- RAID z `D` diski v polju.
- RAID enota = `1` bit.

Primer 2

- RAID z D diski v polju.
- RAID enota = 1 blok.

RAID z redundanco podatkov…

- Več diskov → večja učinkovitost (shranjevanja) → manjša zanesljivost.

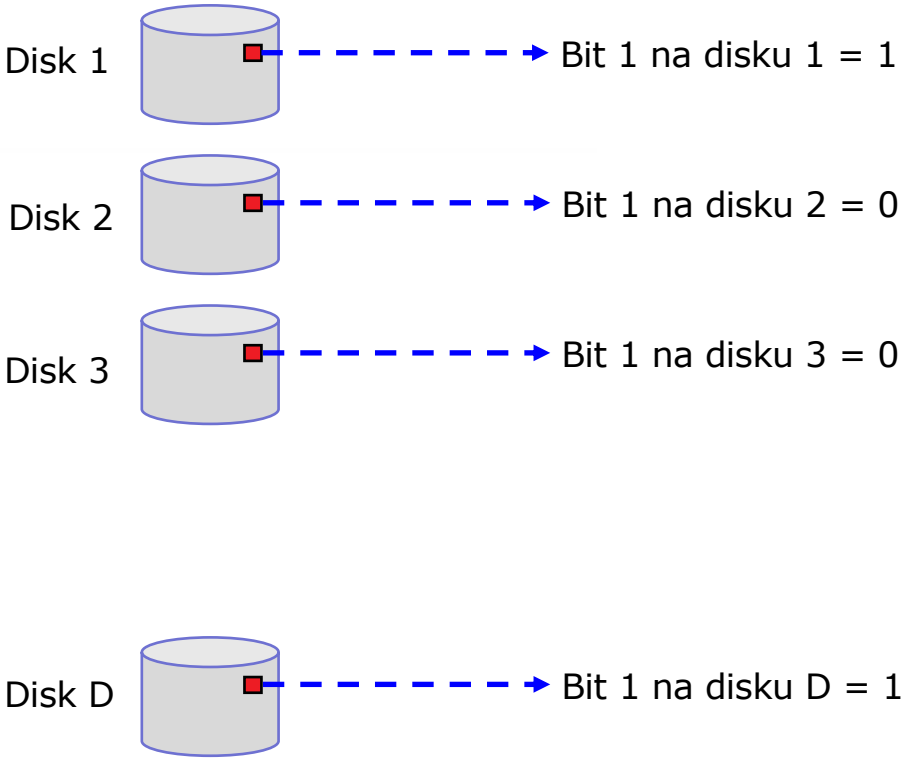
- Primer:
  - MTTF (*mean-time-to-failure*) enega diska ≈ 50.000 ur (5,7 let). Pri 100 diskih v polju MTTF 50.000/100 ≈ 500 ur (21 dni).
- Za večjo zanesljivost (večji MTTF) potrebna redundanca.
- Primer:
  - Če polju 100-ih diskov dodamo 10 diskov z redundantnimi podatki → MTTF > 250 let!!!

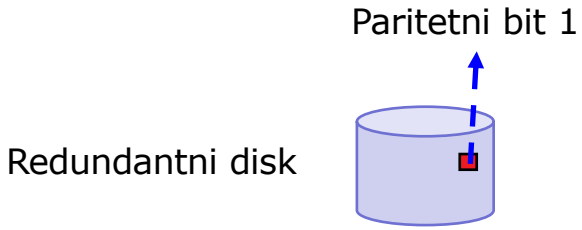
RAID z redundanco podatkov…

- Vprašanje:
  - Kje hraniti redundantne podatke (na določenih/vseh diskih)?
- Kaj podvajati?

- Večinoma redundanten disk za paritetni bit…

Paritetni bit



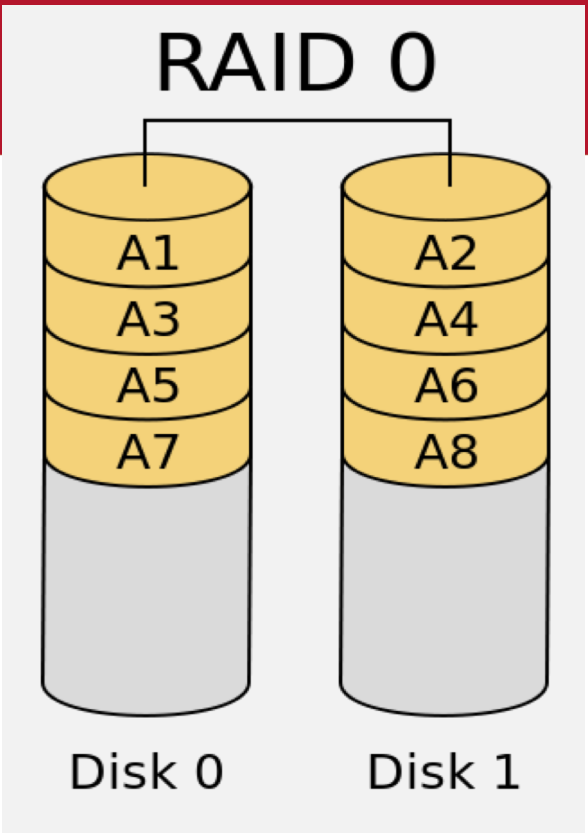


$$Paritetni_bit_i = Odd(\sum_{n=1}^{D} disk_n(bit_i))$$

Stopnje redundance…

- V RAID sistemu diskovno polje sestavljeno iz:
  - množice podatkovnih diskov in
  - množice kontrolnih diskov.
- Število kontrolnih diskov odvisno od stopnje redundance.
- Parametri za primere v nadaljevanju:
  - Količina podatkov za 4 diske
  - Ena sama kontrolna skupina

RAID 0

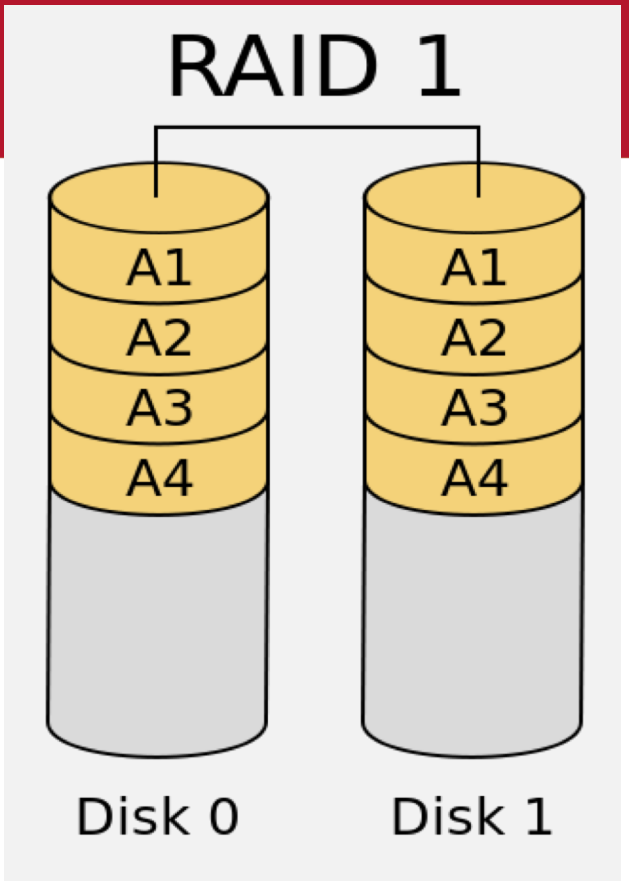


- Porazdeljen, brez redundance (*nonredundant*)
- Uporablja porazdeljevanje podatkov za povečanje pasovne širine.
- Ne vzdržuje nobene redundantne informacije.

- PROBLEM: MTTF pada linearno s številom diskov v polju.
- PREDNOSTI:
  - najvišja učinkovitost → ni potrebno vzdrževati nobenih redundantnih podatkov.
  - 100% izraba prostora na disku. V našem primeru rabimo za svoje podatke 4 diske.

RAID 1

- Zrcaljen (*mirrored*)
- Najdražja rešitev → vzdržujeta se dve kopiji podatkov na dveh diskih.
- Vsako pisanje bloka na disk pomeni pisanje na dva diska.



- Pisanje eno za drugim (možnost nesreče med pisanjem).
- Branje lahko paralelno (branje dveh različnih blokov iz dveh diskov; možno branje iz diska z minimalnim dostopnim časom.
- Izraba prostora 50%. V našem primeru 8 diskov (4 + 4).

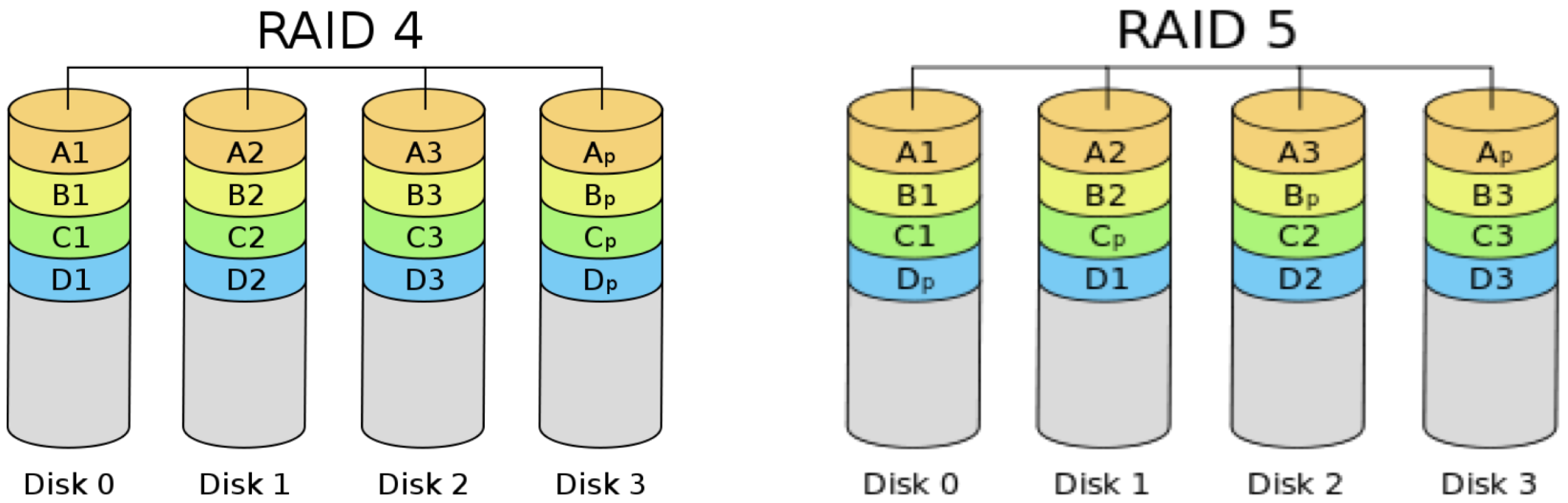
Druge stopnje redundance

- RAID 2: *Error-Correction Codes*
- RAID 3: *Byte-Interleaved Parity*
- RAID 4: *Block-Interleaved Parity*
- RAID 5: *Block-Interleaved Distributed Parity*
- RAID 6: *Block-Interleaved Double Distributed Parity*

- Pomembni parametri:
  - *Space Efficiency* - Izkoriščenost prostora
  - *Fault Tolerance* – Št. diskov, ki lahko brezizgubno odpovedo
  - *Array failure rate* – Verjetnost, da odpove polje
  - *Read performance* – učinkovitost branja
  - *Write performance* – učinkovitost pisanja

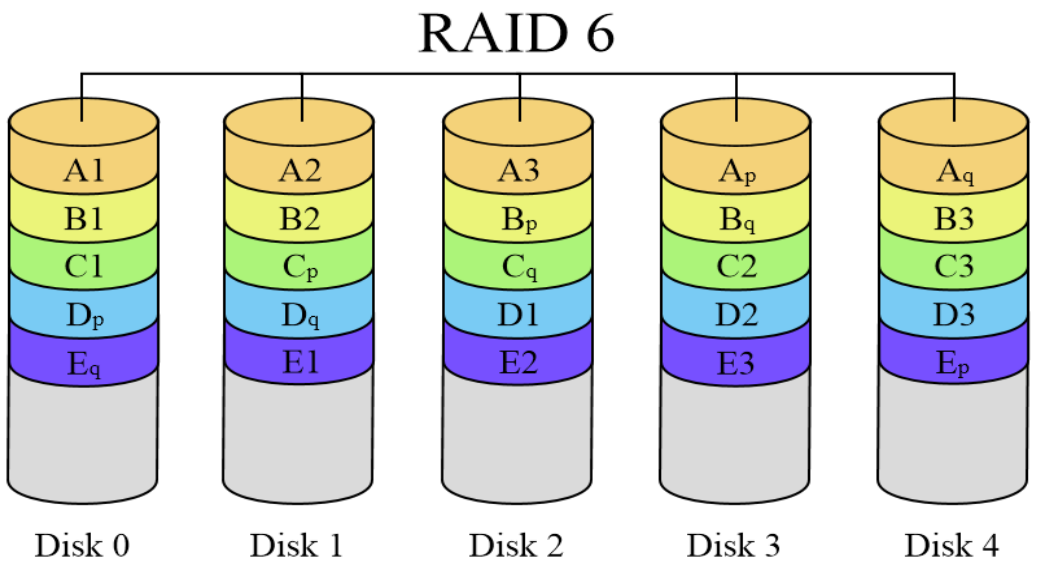
Razlika med RAID 4 in RAID 5

- RAID 4: *Block-Interleaved Parity*
- RAID 5: *Block-Interleaved Distributed Parity*



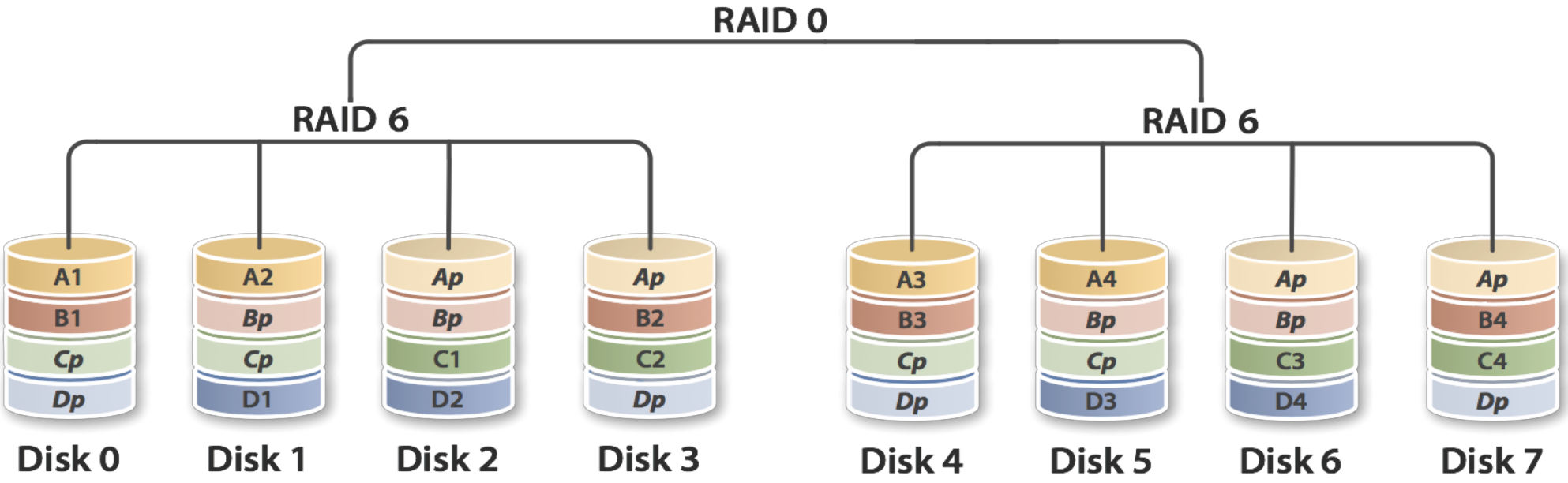
RAID 6

- Porazdeljen s porazdeljeno redundanco
- Enota porazdelitve je blok
- Redundanca je paritetni bit – uporablja dva paritetna bita, oba sta porazdeljena.
- Neobčutljiv na hkratno odpoved do dveh diskov.



Gnezdena polja

- Primer: RAID 6 + 0



Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



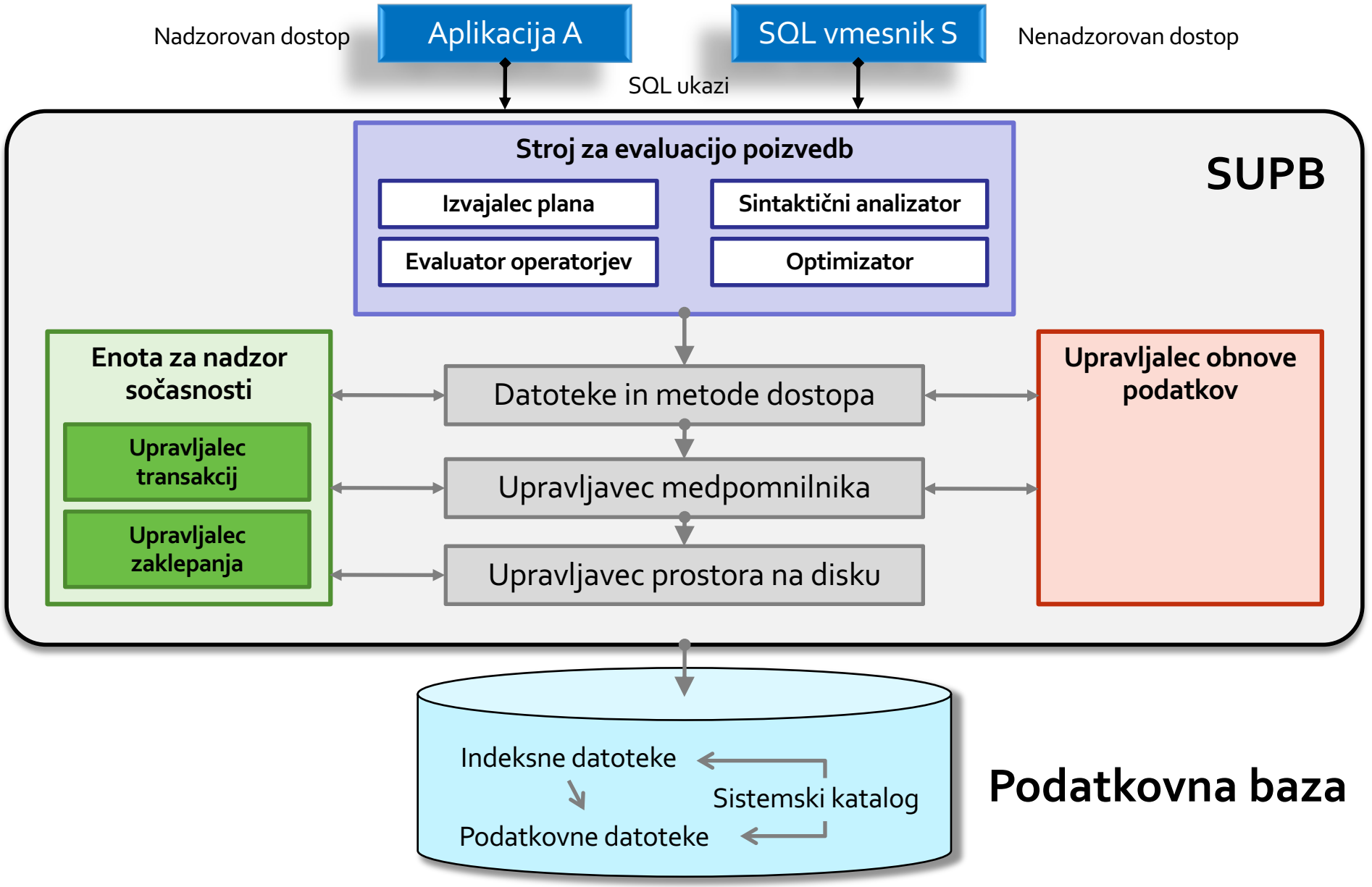
Predmet:
Osnove podatkovnih baz

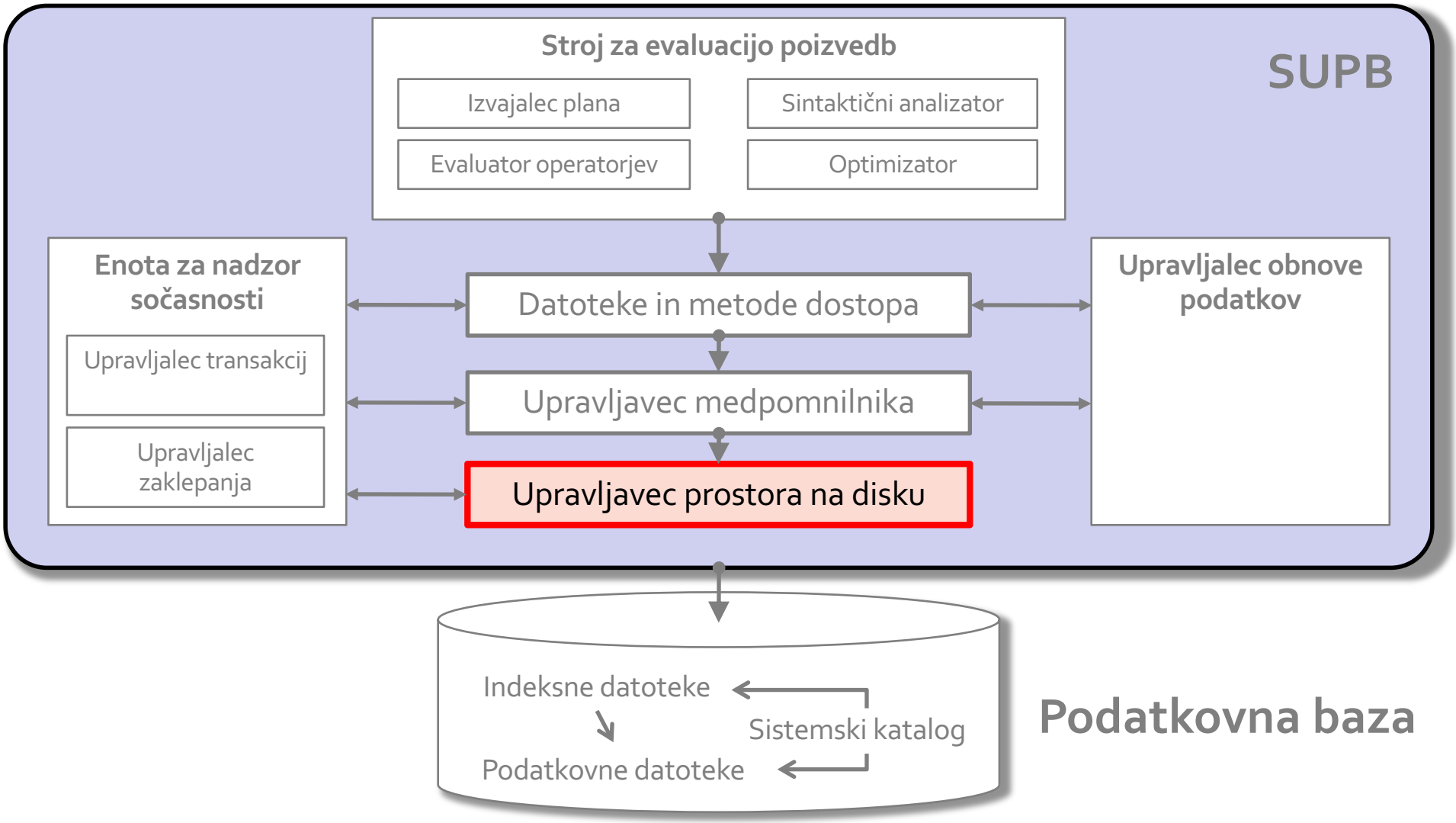
Modul:
Upravljanje z diskom in pomnilnikom

Gradivo:
v.2015

Vsebina

- Zgradba SUPB
- Upravlanje s prostorom na disku
- Upravljanje z medpomnilnikom
- Startegije zamenjave strani v medpomnilniku


Nadzorovan dostop
Aplikacija A
SQL vmesnik S
Nenadzorovan dostop
SQL ukazi
SUPB
Stroj za evaluacijo poizvedb
Izvajalec plana
Sintaktični analizator
Evaluator operatorjev
Optimizator
Enota za nadzor sočasnosti
Upravljalec transakcij
Upravljalec zaklepanja
Datoteke in metode dostopa
Upravljavec medpomnilnika
Upravljavec prostora na disku
Upravljalec obnove podatkov
Indeksne datoteke
Sistemski katalog
Podatkovne datoteke
Podatkovna baza


SUPB
Stroj za evaluacijo poizvedb
Izvajalec plana
Sintaktični analizator
Evaluator operatorjev
Optimizator
Enota za nadzor sočasnosti
Upravljalec transakcij
Upravljalec zaklepanja
Datoteke in metode dostopa
Upravljavec medpomnilnika
Upravljavec prostora na disku
Upravljalec obnove podatkov
Indeksne datoteke
Sistemski katalog
Podatkovne datoteke
Podatkovna baza

Upravljanje s prostorom na disku…

- Upravljanje z diskom - upravljalec z diskom (*Disk Space Manager*).

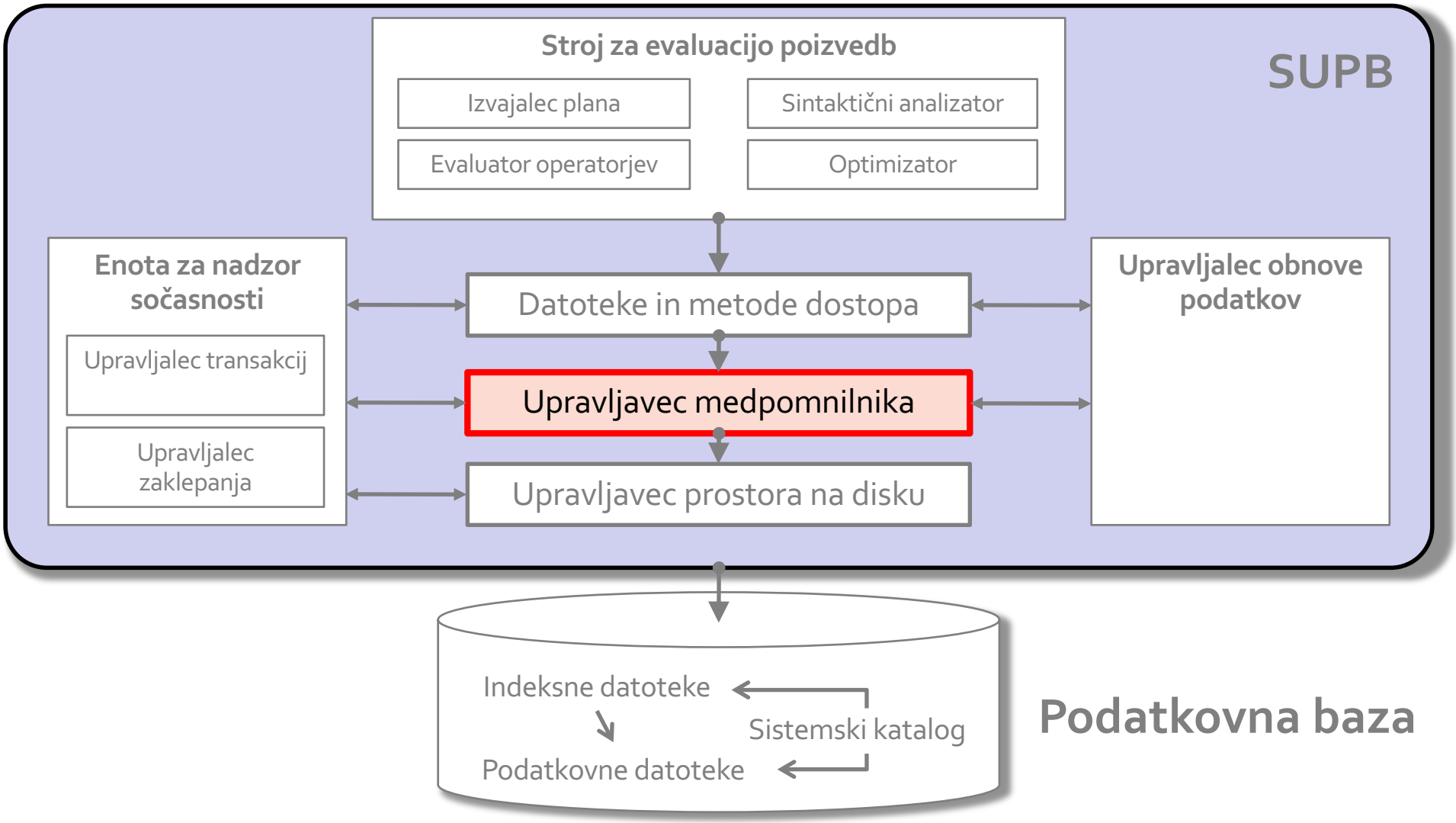
- Lastnosti:
  - Skrije podrobnosti strojne opreme in OS; omogoča, da ostale komponente SUPB vidijo podatke kot zbirko strani.
  - Izvaja ukaze za dodeljevanje in sproščanje prostora na disku ter branje in pisanje;
  - Vse operacije izvaja nad enotami - stran (*page*). Stran = blok na disku. Branje/pisanje strani → ena I/O operacija;

Upravljanje s prostorom na disku…

- Vzdržuje stanje zasedenih in prostih blokov na disku.
- Obstajata dva načina:
  - Vzdrževanje seznama prostih blokov (kazalec na prvi blok seznama se shrani na znano lokacijo na disku),
  - Vzdrževanje bitne mape (za vsak blok je v bitni mapi bit, ki označuje, ali je blok zaseden ali ne).

Upravljanje s prostorom na disku…

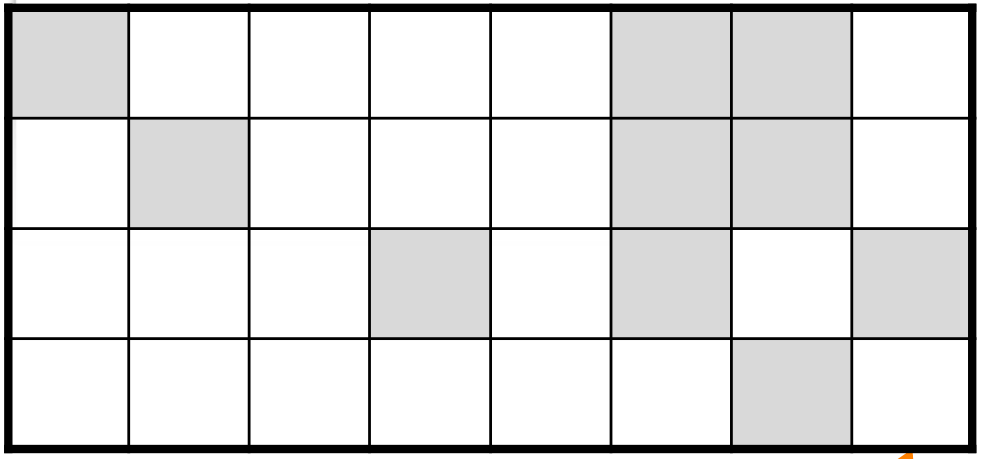
- Uporaba datotečnega sistema za upravljanje s prostorom:
  - Upravljalec z diskom lahko uporablja datoteke operacijskega sistema → celotna PB se nahaja v eni ali več datotekah.
  - V tem primeru je zadolžen za upravljanje prostora v teh datotekah.
  - Veliko PB ne uporablja datotečnega sistema, ampak svoj lastni sistem za upravljanje z diskom (popolnoma svoj ali pa razširja funkcionalnost datotečnega sistema OS). Razlogi:
    - PRAKTIČNI: bazo lahko uporabimo na več platformah,
    - TEHNIČNI: pri 32 bitnem naslavljanju se pojavi omejitev v velikosti datoteke.

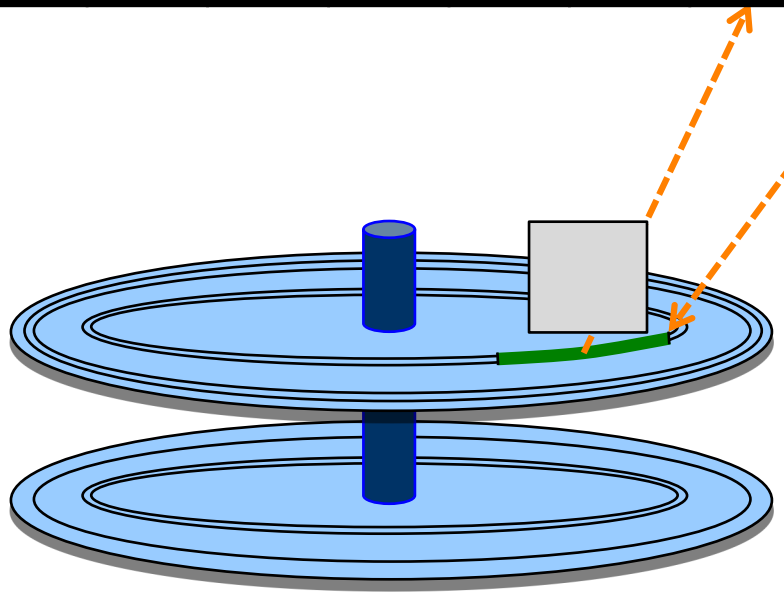

SUPB
Stroj za evaluacijo poizvedb
Izvajalec plana
Sintaktični analizator
Evaluator operatorjev
Optimizator
Enota za nadzor sočasnosti
Upravljalec transakcij
Upravljalec zaklepanja
Datoteke in metode dostopa
Upravljavec medpomnilnika
Upravljavec prostora na disku
Upravljalec obnove podatkov
Indeksne datoteke
Sistemski katalog
Podatkovne datoteke
Podatkovna baza

Upravljanje medpomnilnika…

- Upravljalec medpomnilnika programska plast, skrbi za prenašanje strani v pomnilnik.
  - upravlja z razpoložljivim pomnilnikom (*buffer pool*).
  - višjim plastem zagotavlja strani… prenese kar zahtevajo
  - višje plasti obveščajo o straneh, ki se sprostijo, spremenijo.

Upravljanje medpomnilnika…

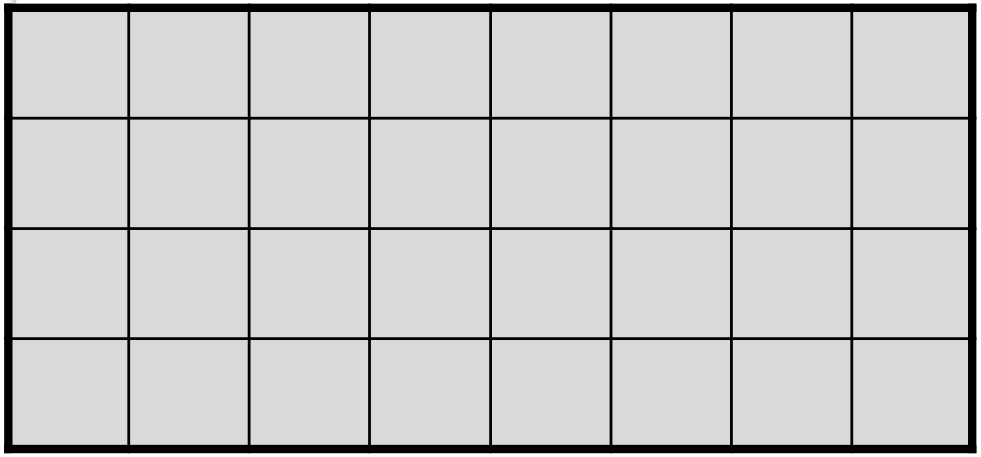




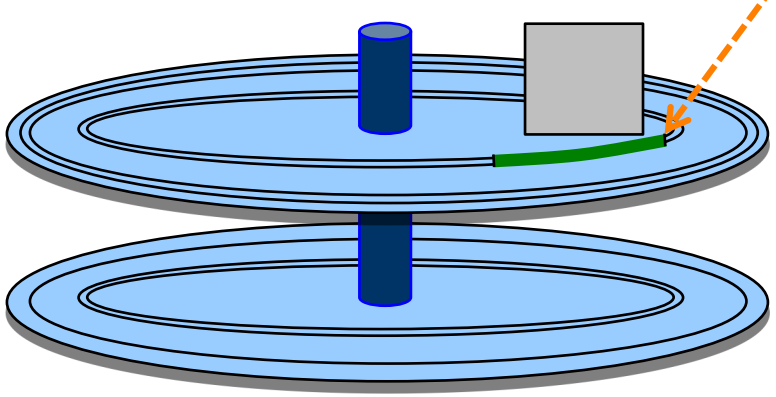
```
select * from employees
where emp_no = 10002;
```

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
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02

Upravljanje medpomnilnika…



```
select * from employees
where emp_no = 10002;
```



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
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02

Upravljanje medpomnilnika…

- Za vsak okvir v medpomnilniku dve spremenljivki:
  - `pin_count`: št. zahtev po neki strani brez sprostitve (št. trenutnih uporabnikov strani).
  - `dirty`: stran spremenjena: true/false.

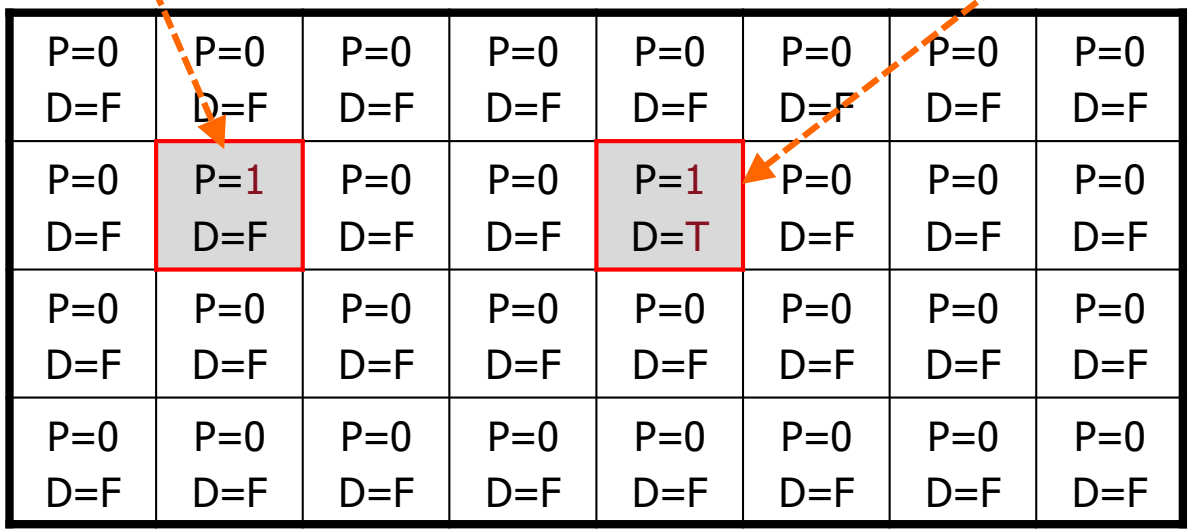
- Začetno stanje okvirja:
  - `pin_count = 0`
  - `dirty = false`

P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F
P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F
P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F
P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F

Primer

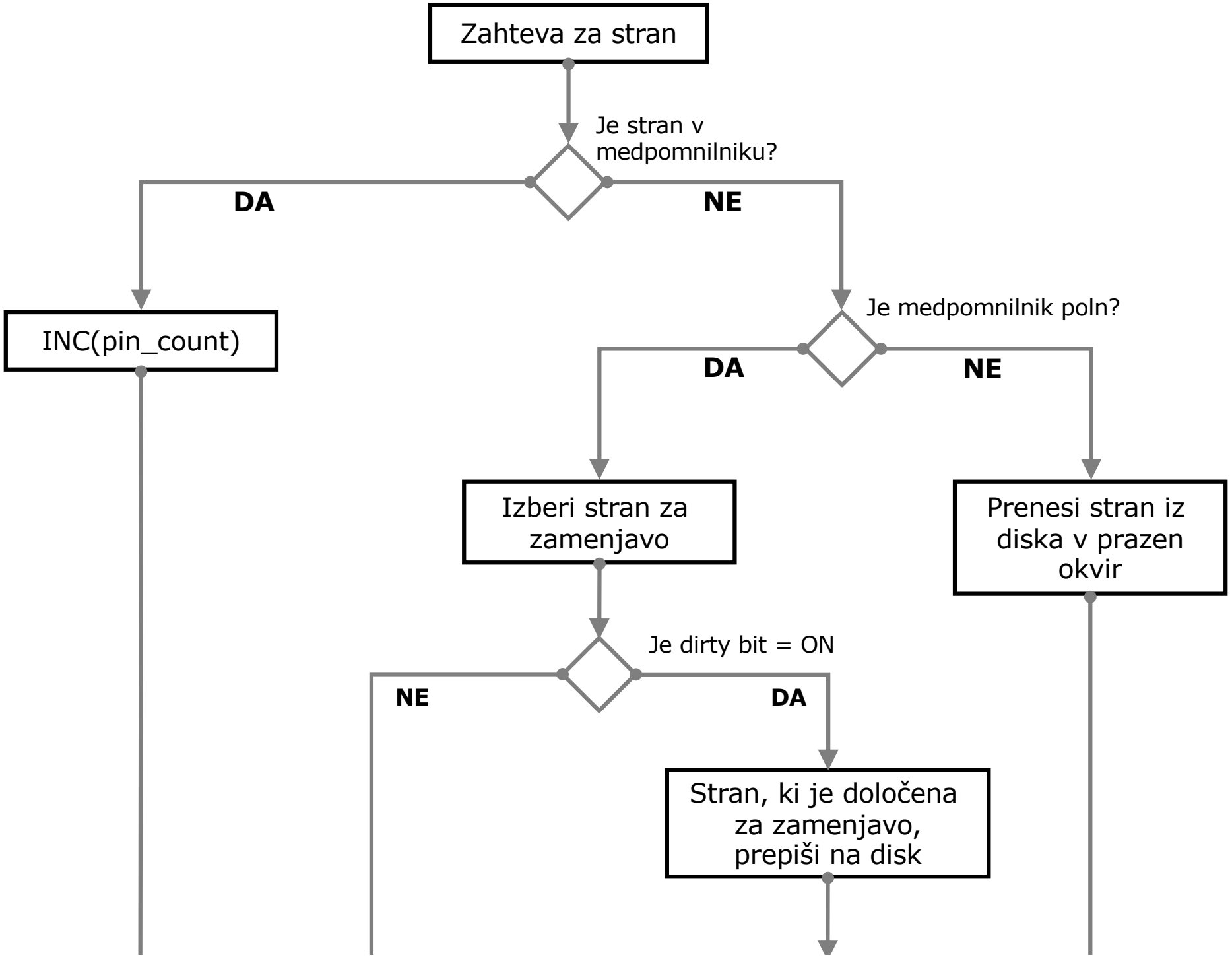
```
T1:
select * from employees
where emp_no = 10002;
```

```
T2:
update employees
set last_name = 'Julius'
where emp_no = 10008;
```



Postopek…

- Ko se pojavi zahteva po določeni strani, upravljalec z medpomnilnikom izvede naslednje:
  - če se stran nahaja v kakšnem od okvirjev, vrne pomnilniški naslov okvirja in poveča `pin_count` za `1`,
  - sicer izvede naslednje:
    - izbere okvir za zamenjavo (z uporabo strategije za zamenjavo).
    - če je `dirty bit` okvirja, ki bo zamenjan, postavljen na "`on`", se stran prepiše na disk.
    - stran se prenese iz diska v okvir, ki je določen za zamenjavo.
    - `pin_count` okvirja se postavi na 1
    - ko sistem sporoči, da se stran sprostila, se `pin_count` zmanjša za `1`. Ko doseže vrednost `0`, sistem prepiše stran nazaj na disk (če `dirty bit ON`)


Zahteva za stran
Je stran v medpomnilniku?
DA
NE
INC(pin_count)
Je medpomnilnik poln?
DA
NE
Izberi stran za zamenjavo
Prenesi stran iz diska v prazen okvir
Je dirty bit = ON
NE
DA
Stran, ki je določena za zamenjavo, prepiši na disk

Postopek

- Dodatna pravila:
  - Zahtevana stran ni v medpomnilniku, vsi okvirji zasedeni → izberi okvir s `pin_count=0`. Če več takih → uporabi strategijo izbire!
  - Ni iskane strani, ni strani s `pin_count=0` → čakaj na sprostitev (transakcija razveljavljena).

Konfliktne spremembe…

- Nevarnost: če neko stran zahteva več neodvisnih transakcij, lahko pride do konfliktnih sprememb…

P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F
P=0 D=F	P=2 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F
P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F
P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=0 D=F

Konfliktne spremembe

- Reševanje z zaklepanjem:
  - Obstaja protokol zaklepanja, za katerega skrbijo višje ravni SUPB (posebej upravljalec transakcij).
  - Vsaka transakcija lahko pridobi deljeno (*shared*) ali ekskluzivno (*exclusive*) zaklepanje preden lahko stran bere ali spreminja.
  - Ekskluzivno zaklepanje iste strani ne sme biti odobreno dvema transakcijama istočasno!

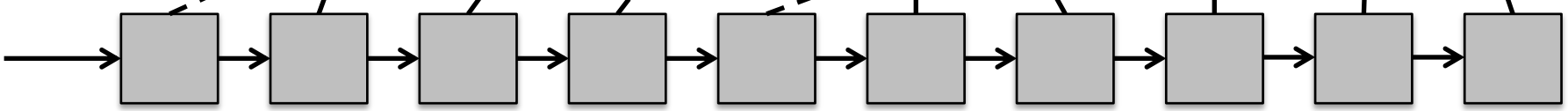
Zamenjava strani v medpomnilniku…

- Strategija zamenjave strani v medpomnilniku močno vpliva na učinkovitost SUPB.
- Različne strategije primerne za različne situacije.

- Strategija LRU – *Least Recently Used*
  - Vrsta kazalcev na okvirje s `pin_count = 0`
  - Ko stran postane kandidat za zamenjavo (`pin_count = 0`), okvir strani dodamo na konec vrste
  - Za zamenjavo izberemo stran iz okvirja, na katerega kaže prvi kazalec v vrsti
  - Problem: časovna kompleksnost

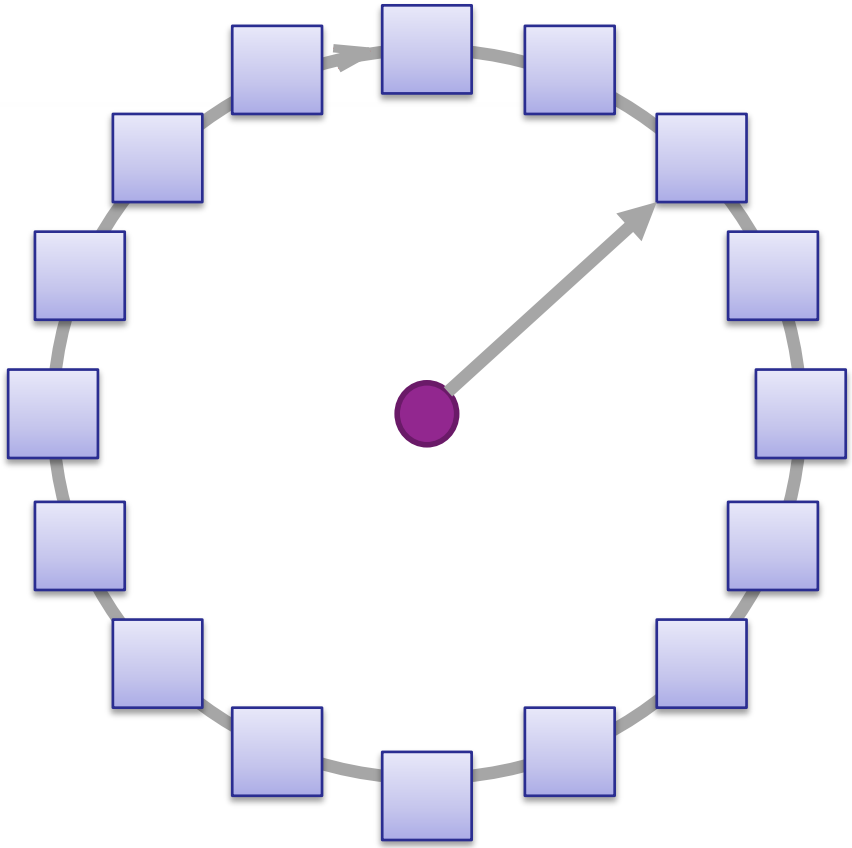
Zamenjava strani v medpomnilniku…

P=1 D=T	P=1 D=F	P=4 D=F	P=1 D=F	P=1 D=F	P=1 D=F	P=0 D=T	P=1 D=F
P=1 D=F	P=0 D=F	P=1 D=T	P=0 D=F	P=0 D=F	P=5 D=F	P=1 D=F	P=0 D=F
P=1 D=F	P=2 D=F	P=1 D=F	P=1 D=F	P=0 D=F	P=1 D=F	P=1 D=F	P=0 D=F
P=1 D=F	P=0 D=F	P=1 D=F	P=7 D=F	P=0 D=F	P=0 D=F	P=0 D=F	P=1 D=F



Zamenjava strani v medpomnilniku…

- Strategija urne zamenjave (*clock replacement*)



Vsaka stran ima:

- `R - reference bit`
- `P - Pin count`

Algoritem

Ko `P` postane `0` postavimo `R=1`;
Če potrebna zamenjava,
preverimo stran, kamor kaže kazalec;
Če velja P>0 ali R=1, pomaknemo kazalec naprej
Sicer stran uporabimo za zamenjavo.

Upravljanje navideznega pomnilnika z OS

- Upravljanje z navideznim pomnilnikom OS in medpomnilnikom pri SUPB podobno.
- Cilj pri SUPB: zagotoviti učinkovit dostop do podatkov, ko jih sistem rabi. Če je podatek v pomnilniku – dobro, sicer slabo.

- SUPB tipično realizira svojega upravljavca z medpomnilnikom:
  - lahko bolje predvidi zaporedje (vzorce dostopanja) kot tipičen OS. Strategija vnaprejšnjega branja (*prefetching*).
  - SUPB rabi več nadzora nad stranmi, ki se zapisujejo na disk, kot ga omogoča tipičen OS.

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



Predmet:
Osnove podatkovnih baz

Modul:
Organizacija datotek in indeksi

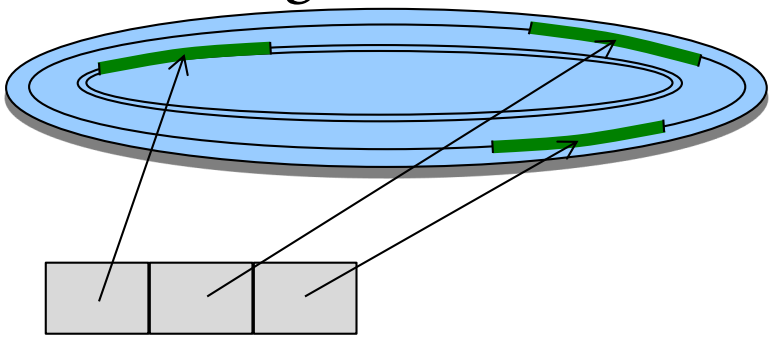
Gradivo:
v.2015

Vsebina

- Datoteke, zapisi in polja
- Datotečne organizacije
  - Neurejene datoteke
  - Urejene datoteke
  - Razpršene datoteke
- Indeksi
- Drevesno indeksiranje
  - ISAM
  - B+drevo
- Bitni in stičn indeksi
- Gruče

Osnovni koncepti…

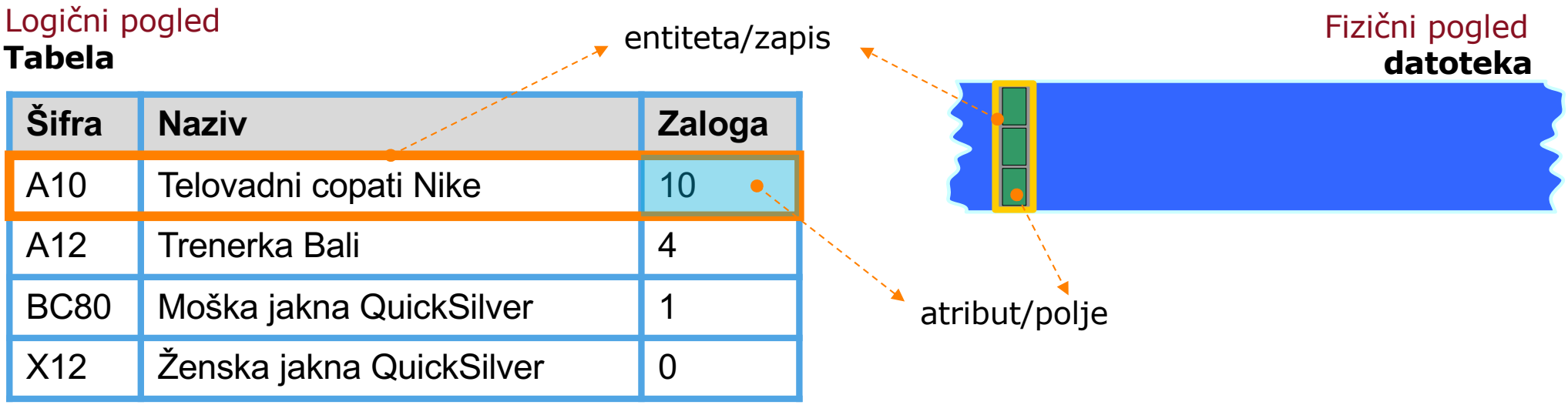
- Podatkovna baza je na sekundarnem pomnilniku organizirana v eno ali več datotek (*file*)



- Vsaka datoteka zajema enega ali več zapisov (*record*).
- Zapis sestavljajo polja (*field*).
- Zapisi običajno označujejo entitete, polja pa njihove atribute.

Osnovni koncepti…

- Uporabnik zahteva zapis `A10` od SUPB:
  - SUPB naredi preslikavo logičnega zapisa v fizični;
  - Poišče fizični zapis in ga prepiše v primarni pomnilnik oziroma v medpomnilnik;



Šifra	Naziv	Zaloga
A10	Telovadni copati Nike	10
A12	Trenerka Bali	4
BC80	Moška jakna QuickSilver	1
X12	Ženska jakna QuickSilver	0

Osnovni koncepti…

- Fizični in logični zapis: ne velja vedno 1:1
  - Fizični zapis enota prenosa med diskom in primarnim pomnilnikom. Lahko zajema več logičnih zapisov.
  - Večji logični zapis lahko zapisan čez več fizičnih zapisov.

- Fizični zapis ustreza konceptu strani.

Osnovni koncepti

ARTIKEL

Šifra	Naziv	Zaloga	Stran
A10	Telovadni copati Nike	10	1
A12	Trenerka Bali	4	
BC80	Moška jakna QuickSilver	1	2
X12	Ženska jakna QuickSilver	0	

Datotečna organizacija…

- Fizična urejenost podatkov v zapise in strani na sekundarnem pomnilniku → datotečna organizacija.

- Osnovne vrste datotečnih organizacij:
  - Kopica ali neurejena datoteka: zapisi so na disku shranjeni v nedefiniranem vrstnem redu.
  - Zaporedno urejena datoteka: zapisi so urejeni po vrednosti določenega polja.
  - Razpršena datoteka: zapisi so razpršeni z uporabo hash funkcije.

Datotečna organizacija…

- Za delo z zapisi v datotekah obstajajo različne tehnike ali metode dostopa (*access methods*):
  - določajo korake, ki jih je potrebno izvesti za shranjevanje ali iskanje nekega zapisa v datoteki.
  - odvisne od datotečne organizacije.

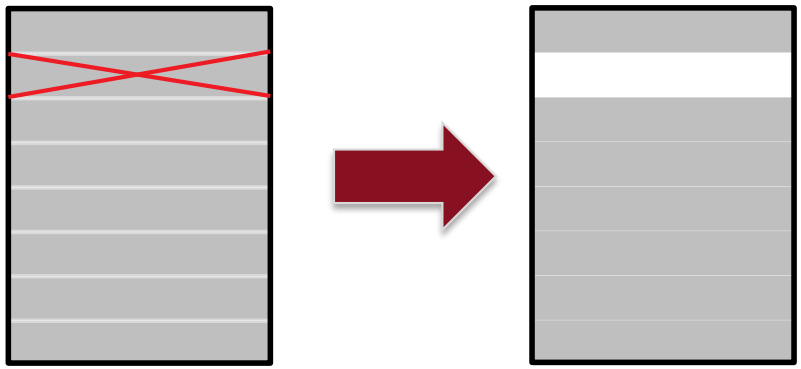
Neurejene datoteke…

- Imenujemo tudi kopica (*heap*).
- Najenostavnejša datotečna organizacija:
  - Zapisi shranjeni v vrstnem redu, kot so bili dodani
  - Nov zapis dodan na zadnjo stran datoteke
  - Če ni dovolj prostora, se doda nova stran
- Dobra lastnost:
  - Zelo učinkovito dodajanje zapisov - ni potrebno računati, na katero stran bomo zapis vstavili.

- Uporabljamo za masovni vnos.

Neurejene datoteke

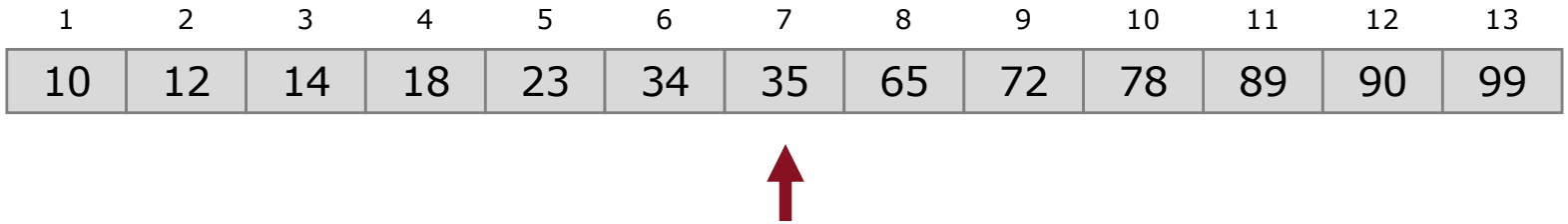
- Slabosti:
  - Neučinkovitost iskanja: linearno iskanje (zapisi so neurejeni)
  - Neizkoriščenost prostora: brisani zapisi puščajo prazen prostor na straneh… neurejene datoteke potrebno občasno reorganizirati!



Urejene datoteke…

- Zapisi v datotekah urejeni po enem ali več poljih → urejena ali zaporedna datoteka.
- Možno binarno iskanje.

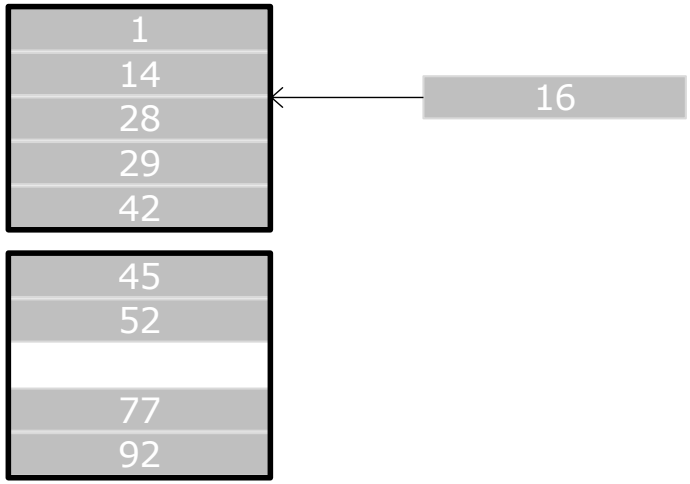
- Primer:
  - Poišči artikel s šifro 14



Urejene datoteke…

- Prednosti:
  - Učinkovitost iskanja

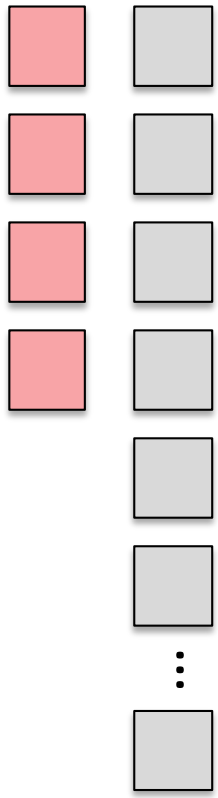
- Problem
  - Dodajanje in brisanje zapisov → potrebno vzdrževati vrstni red
  - Kaj če na strani ni dovolj prostora?



Urejene datoteke

- Dodajanje zapisa na začetek velike datoteke posebej problematično.

- Možna rešitev: uporaba dodatne neurejene datoteke (*overflow file*):
  - Nov zapis dodan v neurejeno datoteko
  - Neurejena datoteka se periodično prepiše v urejeno
  - Pri iskanju zapisa se najprej pogleda urejena datoteka. Če zapisa ne najdemo, se linearno pregleda še neurejena.



Razpršene ali hash datoteke…

- Zapisi razpršeni v skladu s hash funkcijo.
- Hash funkcija za vsak zapis izračuna naslov strani (naslov bloka na disku), kamor zapis sodi glede na vrednost določenega polja (hash polje).

  $f_{hash}(P)$ → naslov strani;
  P: vrednost hash polja.

- Iskanje zapisa na strani se izvede v primarnem pomnilniku. Za večino zapisov moramo prebrati le eno stran.

Razpršene ali hash datoteke…

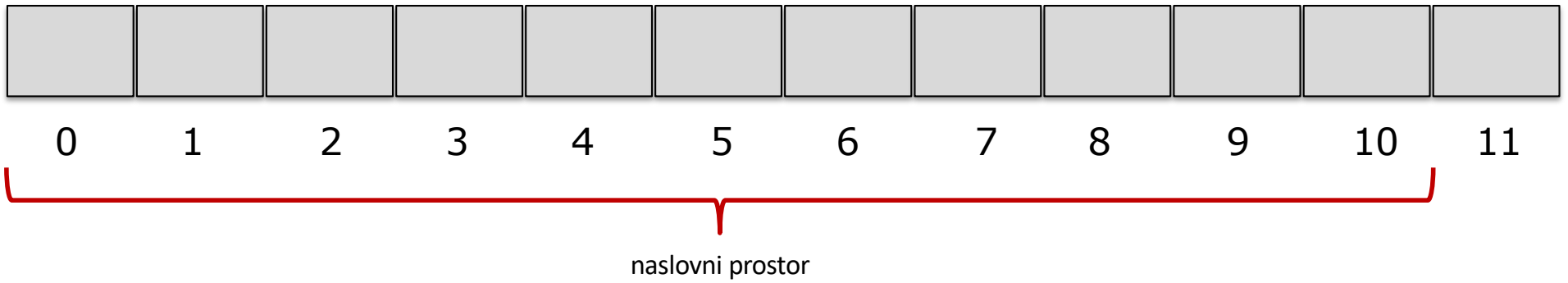
- Operacije za hash funkcijo mora zagotavljati enakomerno porazdeljenost zapisov po datoteki.

- Tipična hash funkcija: ostanek pri deljenju (MOD) z določenim številom (n)

Primer hash funkcije

- MOD 11





Razpršene ali hash datoteke…

- Slabosti razpršenih datotek:
  - Zaloga vrednosti hash polja navadno večja od števila naslovov, ki jih lahko vrne hash funkcija.
  - Vsak naslov ustreza določeni strani (*bucket*), ki ima mesta za več zapisov.
  - Linerano iskanje zapisov po strani: znotraj strani zapisi urejeni po vrsti, kot so bili vstavljeni.
  - Ko hash funkcija za nek zapis vrne naslov strani, ki je polna, pride do kolizije.

Razpršene ali hash datoteke…

- Tehnike za reševanje kolizije:
  - Odprto naslavljanje (*open addressing*)
  - Nepovezane dodatne strani (*unchained overflow*)
  - Povezane dodatne strani (*chained overflow*)
  - Večkratno razprševanje (*multiple hashing*)

Odprto naslavljanje…

- Pravila:
  - Zapisovanje: kolizija → poiščemo prvo stran, ki ima še kakšno prosto mesto. Ko pridemo do zadnje strani, gremo na začetek.
  - Iskanje: enako kot pri zapisovanju. Če naletimo na prazno mesto preden na iskani zapis → zapisa ni v datoteki.

Primer odprtega naslavljanja

- Hash operacija: `MOD 3`
- Dodajamo zapis `SL 41`

?

Stanje prej	Stran
SA 9	0
SG 37	1
SG 5 SG 14	2

Stanje potem	Stran
	0
	1
	2

Nepovezane dodatne strani

- Vzdržujemo seznam dodatnih strani

Primer

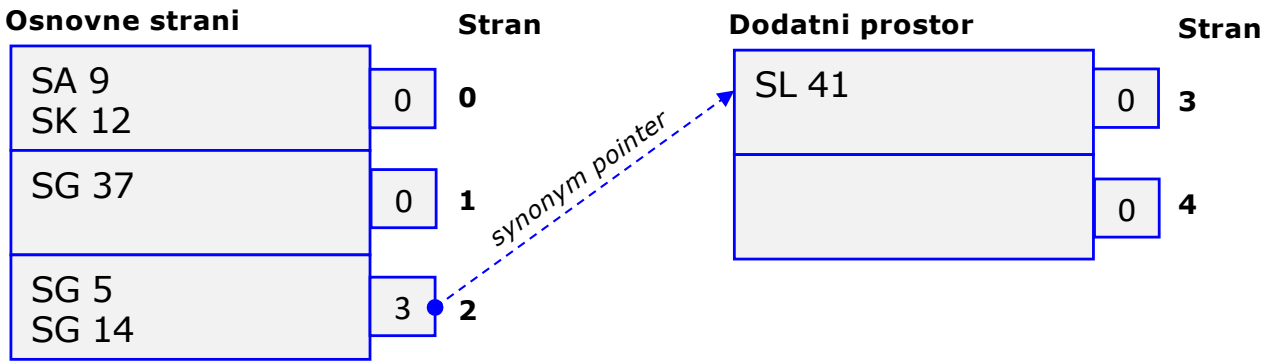
Osnovne strani	Stran
SA 9	0
SG 37	1
SG 5 SG 14	2

Dodatni prostor	Stran
SL 41	3
	4

Povezane dodatne strani

- Dodatne strani povezane z osnovnimi
- Vsaka stran ima dodatno polje, ki pove, ali je prišlo pri tej strani do kolizije ali ne. V polju je naslov dodatnega polja, kamor so razvrščeni zapisi, pri katerih pride do kolizije. Vrednost 0 pomeni, da kolizije ni bilo.

Primer



Večkratno razprševanje

- V primeru kolizije uporaba dodatne hash funkcije (vrača drugačen naslov).
- Dodatna hash funkcija navadno razvršča v dodati prostor.

Primer:

- H_1: MOD 3, H2: 3 + MOD 5; dodajamo SL 11

Osnovne strani	Stran
SA 9	0
SG 37	1
SG 5 SG 14	2

Dodatni prostor	Stran
	3
SL 41	4

Razpršene ali hash datoteke…

- Razprševanje v splošnem
  - Prinaša dobre rezultate pri iskanju (ob uporabi ene izmed tehnik reševanja kolizij).
  - Spreminjanje zapisov enostavno, razen v primerih, ko spremenimo hash polje.

Dinamične razpršene datoteke…

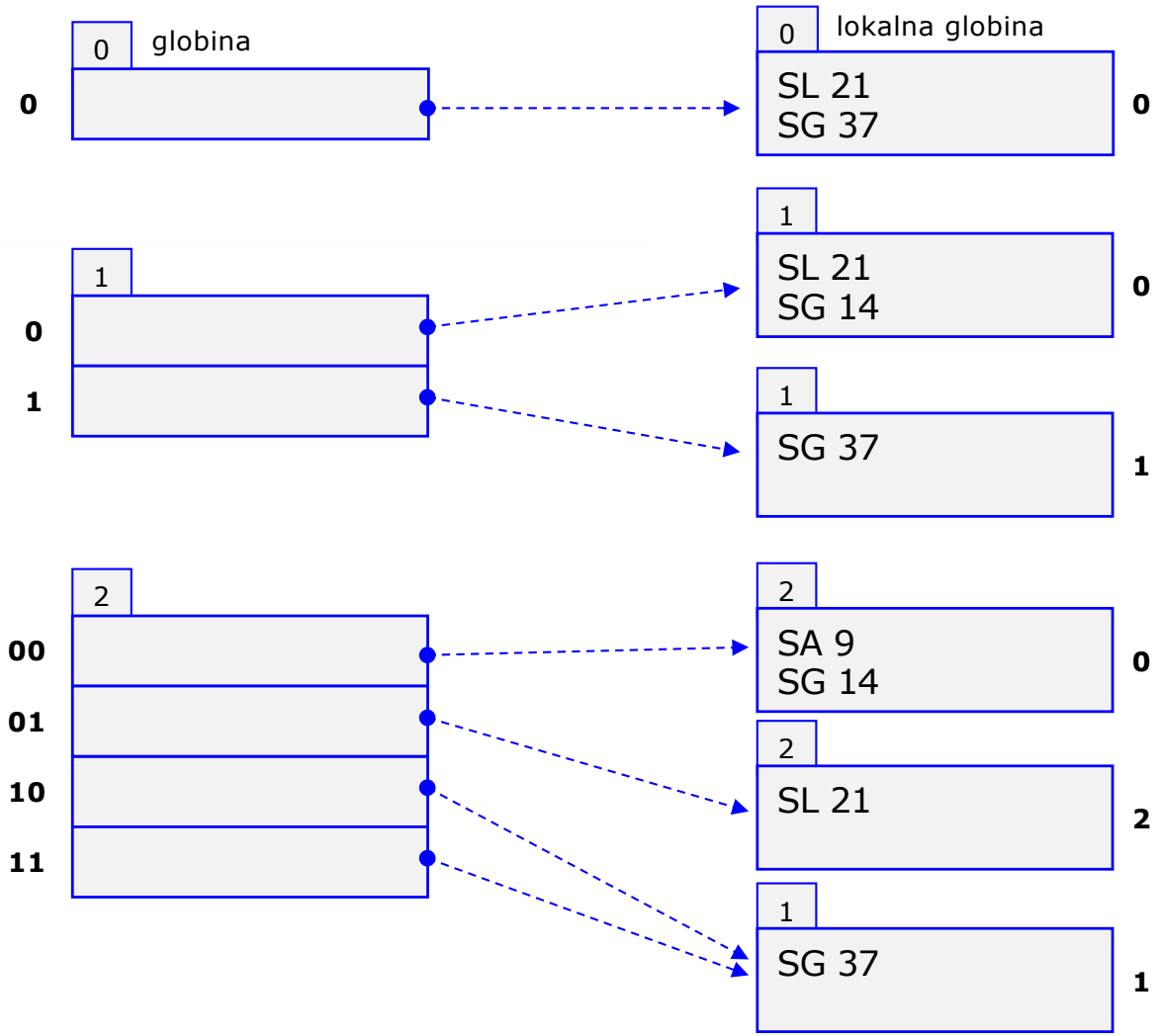
- Obravnavane hash tehnike statične: ko zapisu določimo naslov, se ta ne spremeni, če se ne spremeni vrednost hash polja.
- Problem: datoteka postane premajhna → vzamemo večjo, določimo novo hash funkcijo, vse zapise ponovno razpršimo.
- Alternativa: uporaba dinamičnih razpršenih datotek: velikost datoteke spreminjamo po potrebi.
- Več tehnik realizacije dinamičnih razpršenih datotek.

Razširljivo razprševanje (*extendible hashing*)

- Pravila:
  - Strani kreiramo po potrebi. V začetku gredo zapisi v prvo stran.
  - Ko stran polna, razdelimo glede na prvih `i` bitov, kjer velja `0 ≤ i < b` (`b` označuje dolžino ključa v bitih).
  - Izbranih `i` bitov določa naslov oziroma offset v naslovni tabeli strani (*BAT - Bucket Address Table*). Vrednost `i` se spreminja z velikostjo datoteke.
  - V glavi imenika je zapisana trenutna vrednost `i` (globina) skupaj z `2i` kazalci.
  - Vsaka stran ima tudi lokalno globino, ki pove, pri katerem `i` dobimo naslov te strani.

Primer razširljivega razprševanja

```
21 = 010101
37 = 100101
14 = 001110
09 = 001001
```



Izgled dinamično razpršene datoteke po vpisu zapisov SL 21 in SL 37.

Izgled dinamično razpršene datoteke po vpisu zapisa SG 14

Izgled dinamično razpršene datoteke po vpisu zapisa SA 9.

Razpršene ali hash datoteke

- Razprševanje v splošnem
  - Prinaša dobre rezultate pri iskanju (ob uporabi ene izmed tehnik reševanja kolizij).
  - Spreminjanje zapisov enostavno, razen v primerih, ko spremenimo hash polje.

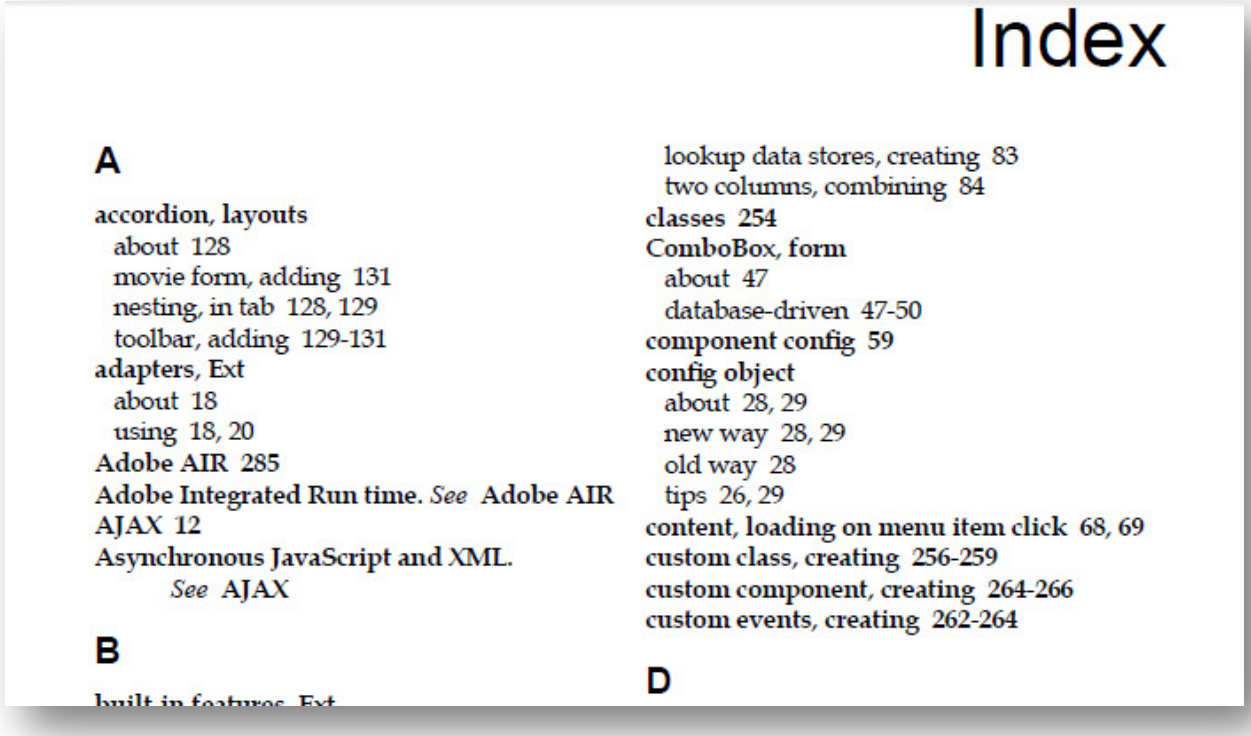
- Omejitve:
  - Učinkovitost razpršenih datotek za iskanje odvisna od hash polja. Uporaba razpršenih datotek ni primerna za:
    - Iskanje po vzorcu
    - Iskanje po nizu vrednosti
    - Iskanje po polju, ki ni hash polje

Ponovitev

- Kakšne datotečne organizacije poznamo?
- Zakaj je datotečna organizacija pomembna?
- V kakšnih primerih bi uporabili:
  - Urejeno datoteko
  - Neurejeno datoteko
  - Razpršeno datoteko
- Datoteka ima n strani. Koliko I/O operacij je potrebnih pri iskanju, če je datoteka:
  - urejena
  - neurejena
  - razpršena

Indeksi in indeksiranje…

- Indeks je podatkovna struktura, ki SUPB-ju omogoča hitrejše lociranje zapisov v datoteki.
- Analogija z indeksom knjige



Index

A

accordion, layouts
  about 128
  movie form, adding 131
  nesting, in tab 128, 129
  toolbar, adding 129-131
adapters, Ext
  about 18
  using 18, 20
Adobe AIR 285
Adobe Integrated Run time. *See* Adobe AIR
AJAX 12
Asynchronous JavaScript and XML.
  See AJAX

B

built in features, Ext
  lookup data stores, creating 83
  two columns, combining 84
classes 254
ComboBox, form
  about 47
  database-driven 47-50
component config 59
config object
  about 28, 29
  new way 28, 29
  old way 28
  tips 26, 29
content, loading on menu item click 68, 69
custom class, creating 256-259
custom component, creating 264-266
custom events, creating 262-264

D

Primer indeksa

Indeks

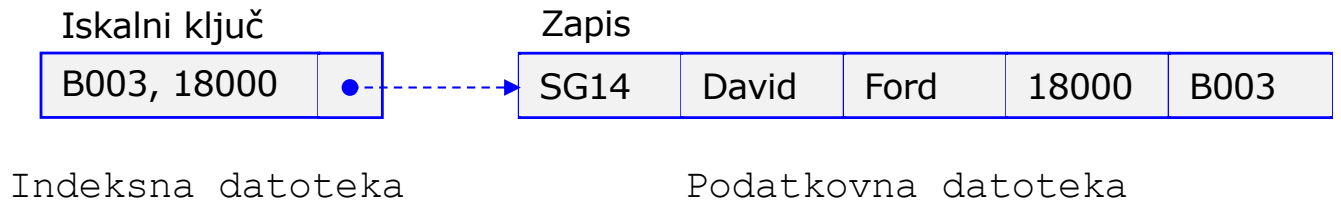
Priimek
Bolčina
Bregar
Bric
Colja
Kavčič
Markovič
Robnik
…

Podatkovna datoteka

Ime	Priimek	Spol	…	Starost
Mateja	Bolčina	Ž		27
Teja	Bregar	Ž		66
Miha	Bric	M		43
Dejan	Bric	M		29
Milka	Colja	Ž		32
Gregor	Kavčič	M		62
Iztok	Kavčič	M		55
Nina	Kavčič	Ž		33
Tadej	Markovič	M		45
Breda	Robnik	Ž		37
…				

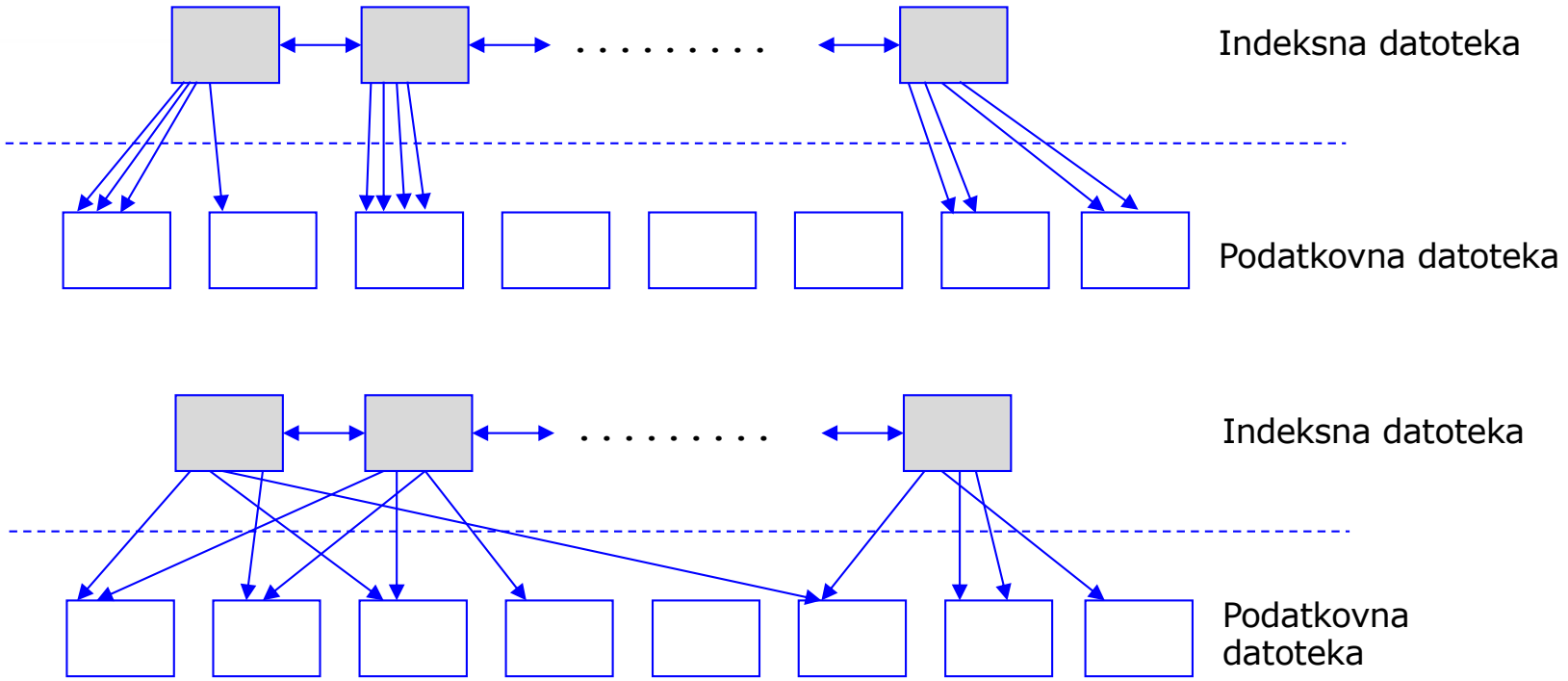
Indeksi in indeksiranje…

- Terminologija:
  - Podatkovna datoteka: datoteka s podatki (imenujemo tudi osnovna datoteka)
  - Indeksna datoteka: datoteka z indeksom. Indeks sestavlja iskalni ključ ter kazalec na zapis v podatkovni datoteki.
  - Iskalni ključ: sestavljen iz vrednosti polj, po katerih je datoteka indeksirana. Imenujemo tudi indeksno polje.



Indeks gruče (*clustered index*)

- Indeks po polju, po katerih je urejena podatkovna datoteka.



Primerni in sekundarni indeksi

- Primarni indeks (*Primary index*): indeks po poljih, ki vsebujejo primarni ključ. Vsak iskalni ključ kaže natanko na en zapis v podatkovni datoteki. Datoteka je po ključu urejena.
- Sekundarni indeks (*Secondary key*): vsak indeks, ki ne temelji na poljih, ki bi vsebovala primarni ključ.

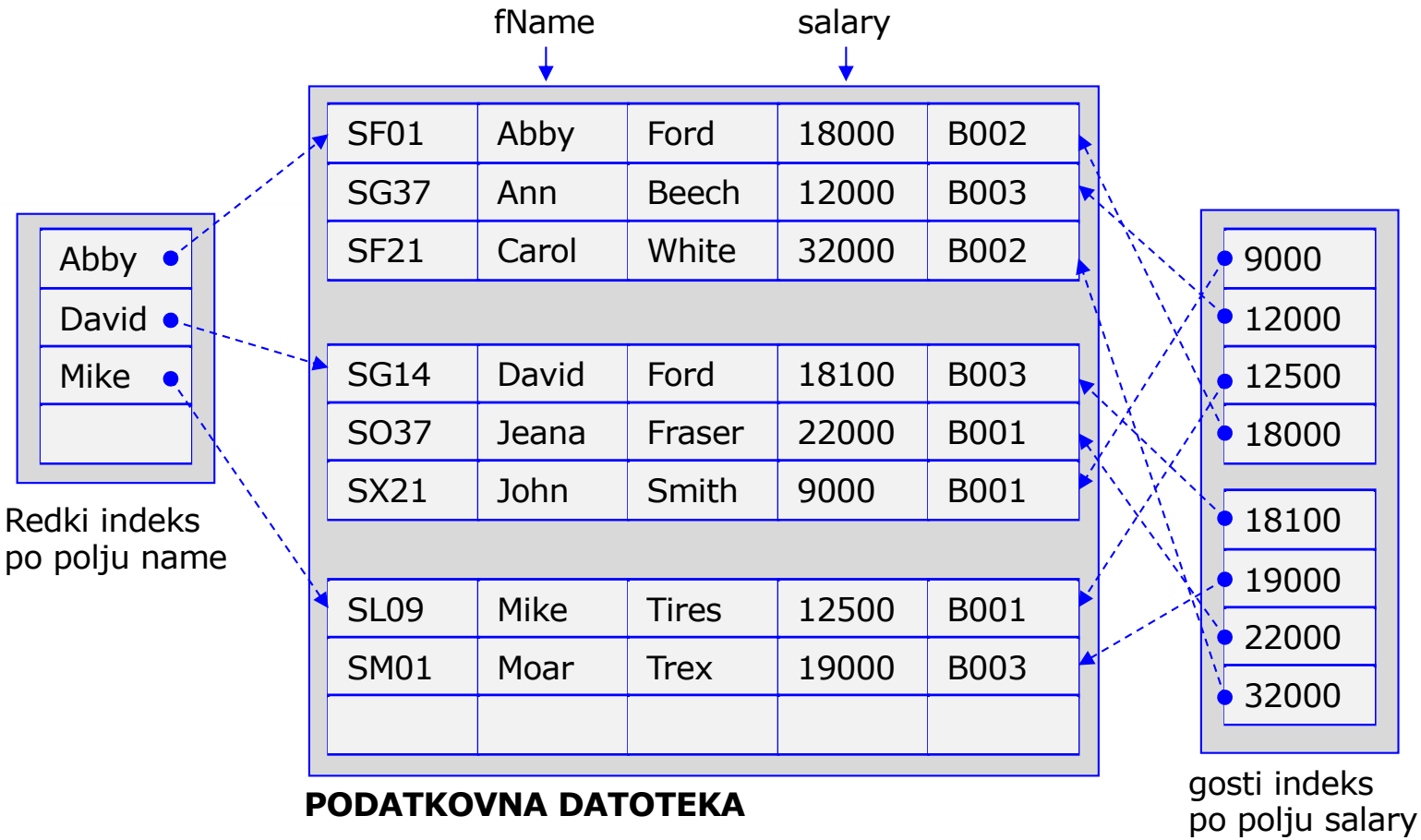
- Vsaka datoteka ima lahko:
  - primarni indeks ali indeks gruče ter
  - več sekundarnih indeksov!

Redki in gosti indeks

- Redki indeks (*sparse index*): indeksna datoteka vsebuje kazalce, ki kažejo le na določene zapise v podatkovni datoteki. Navadno vsebuje po en zapis za vsako stran v podatkovni datoteki;

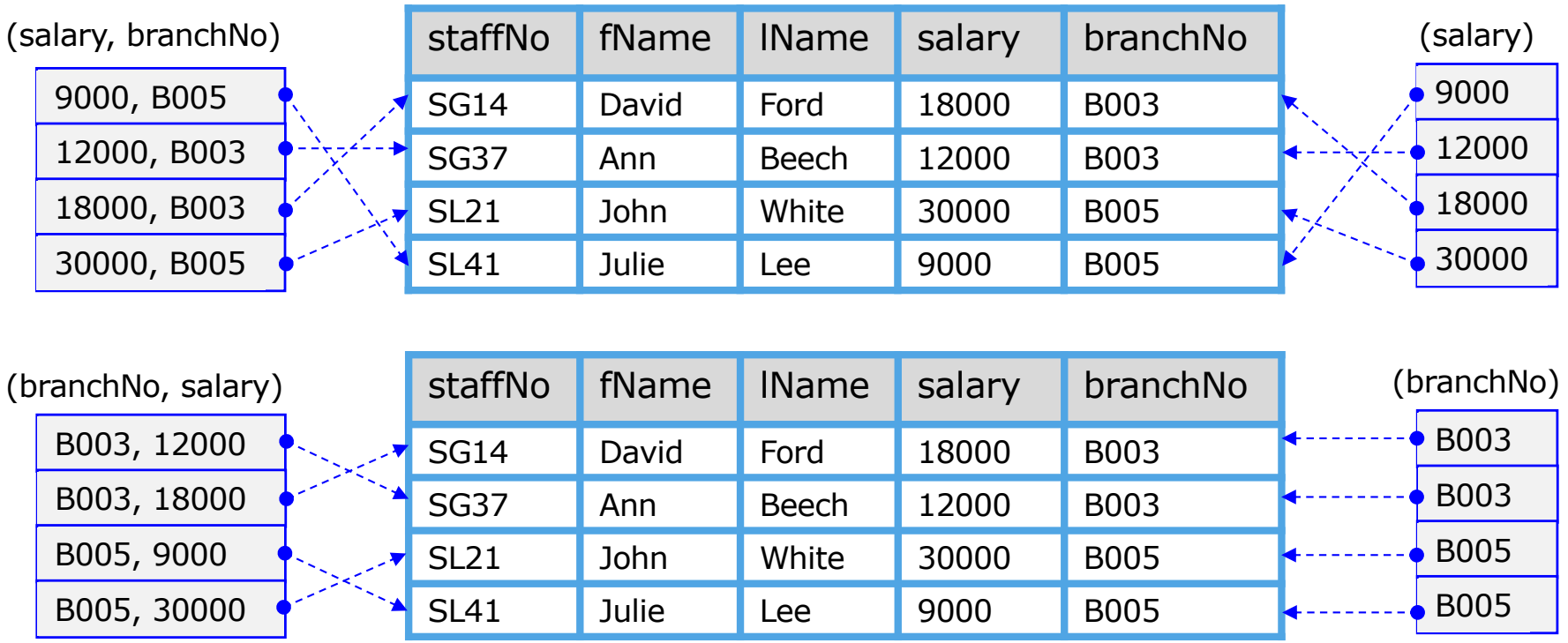
- Gosti indeks (*dense index*): indeksna datoteka vsebuje kazalce na vse zapise v podatkovni datoteki.

Primer redkega in gostega indeksa



Sestavljen indeks

- Indeksi s sestavljenim iskalnim ključem (*composite key*):
  - Indeksi po več kot enem polju
  - Uporabljamo za kombinacije polj, po katerih pogosto iščemo



Vaja

- Imamo tabelo finančnih transakcij z 50 milioni zapisov. Vsak zapis ima naslednje atribute:
  - ID transakcije, datum, znesek, tip transakcije, plačnik, prejemnik, valuta
- Transakcije bomo predvidoma iskali po:
  - Datumu – npr. vse transakcije iz določenega dne ali obdobja,
  - Plačniku – npr. vse transakcije določenega plačnika,
  - Plačniku in prejemniku – npr. vse transakcije določenega plačnika, agregirano po prejemniku.

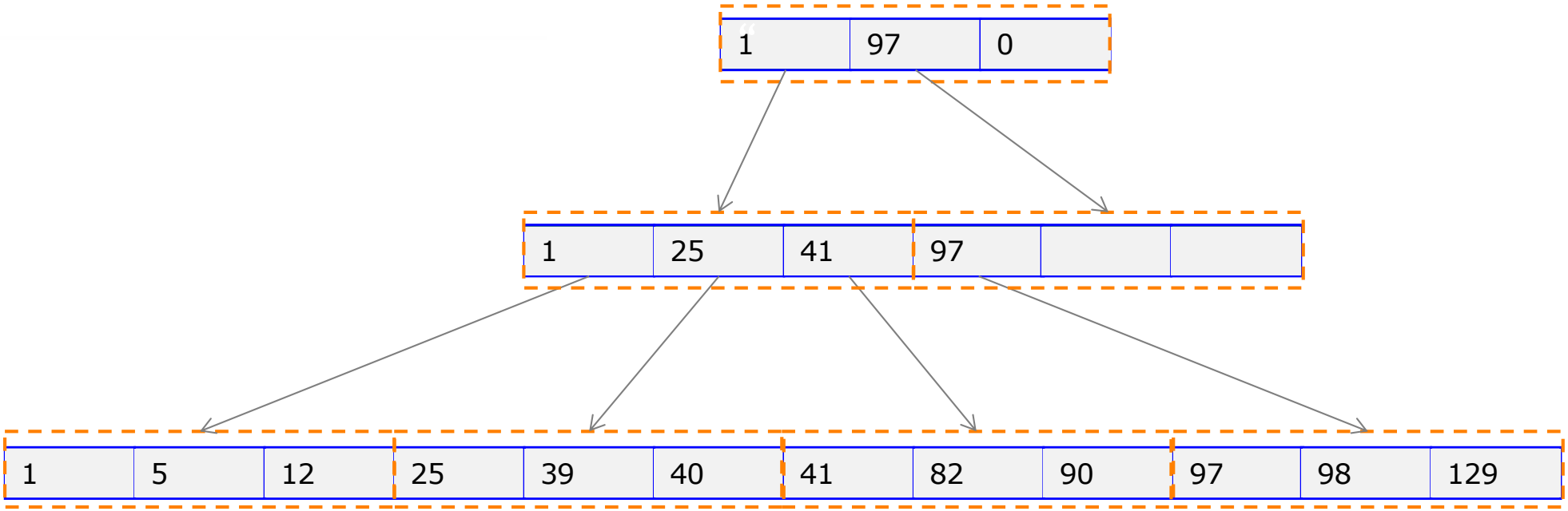
- Kakšno datotečno organizacijo in indekse bi uporabili?

Drevesno indeksiranje…

- Primer:
  - Imamo datoteko oseb, urejeno po polju `starost`.
  - Če želimo najti vse osebe starejše od `30` let, moramo najprej najti prvo (s pomočjo binarnega iskanja), ki je starejša od `30` in od te naprej prebrati preostale zapise datoteke.
- Možnost za pohitritev iskanja:
  - Izdelamo dodatno datoteko, s po enim zapisom za vsako stran v podatkovni datoteki in jo uredimo po polju `starost`.
  - Vsak zapis v dodatni datoteki vsebuje par

    `<ključ prvega zapisa na strani, kazalec na zapis>`

Primer drevesnega indeksa



Drevesno indeksiranje…

- Drevesno indeksiranje učinkovito pri:
  - intervalnem iskanju ter
  - dodajanju in brisanju (za razliko od urejenih datotek).
- Iskanje po enakosti boljše pri hash indeksiranju.

- Dve tipični indeksni strukturi, ki temeljita na drevesni organizaciji:
  - ISAM (*Indexed Sequential Access Method*):
  - B+ drevesna struktura

Drevesno indeksiranje…

- ISAM:
  - statičen indeks,
  - učinkovit v primerih, ko se podatkovna datoteka ne spreminja pogosto,
  - ni učinkovit pri datotekah, ki se hitro povečujejo ali krčijo.
- B+ drevo:
  - dinamična struktura; se zelo dobro prilagaja spremembam v podatkovni datoteki,
  - najbolj uporabljana vrsta indeksa; omogoča hitro iskanje zapisov, intervalno iskanje in se učinkovito prilagaja spremembam v podatkovni datoteki.

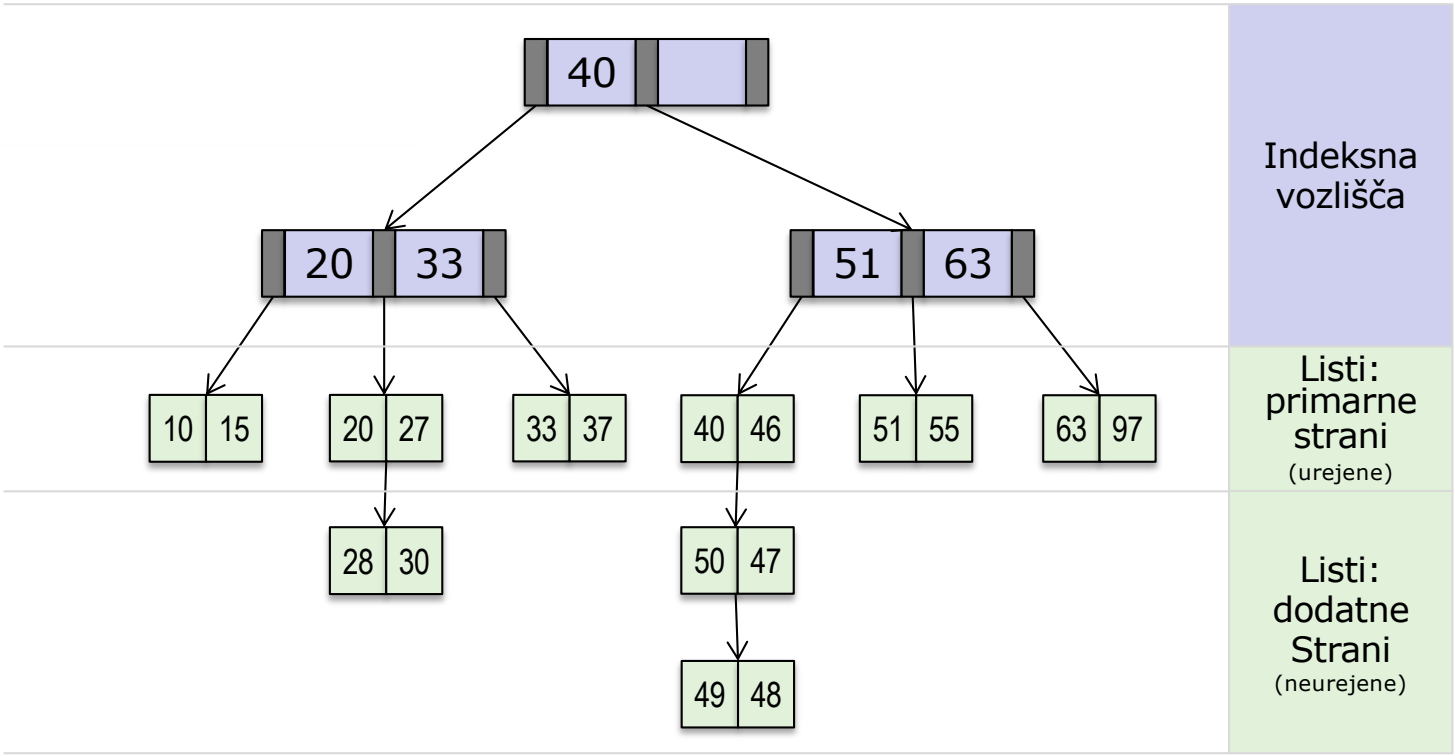
ISAM…

- ISAM statična struktura (izjema dodatne strani)

- Gradnja ISAM indeksa:
  - Ko se indeksna datoteka kreira, so vse strani v listih urejene zaporedno in po iskalnem ključu.
  - Vstavljanje novih podatkov lahko zahteva kreiranje dodatnih strani (če je stran, kamor podatek sodi, polna).

- Podprte operacije: operacije iskanje, brisanje in vstavljanje.

Primer ISAM drevesa

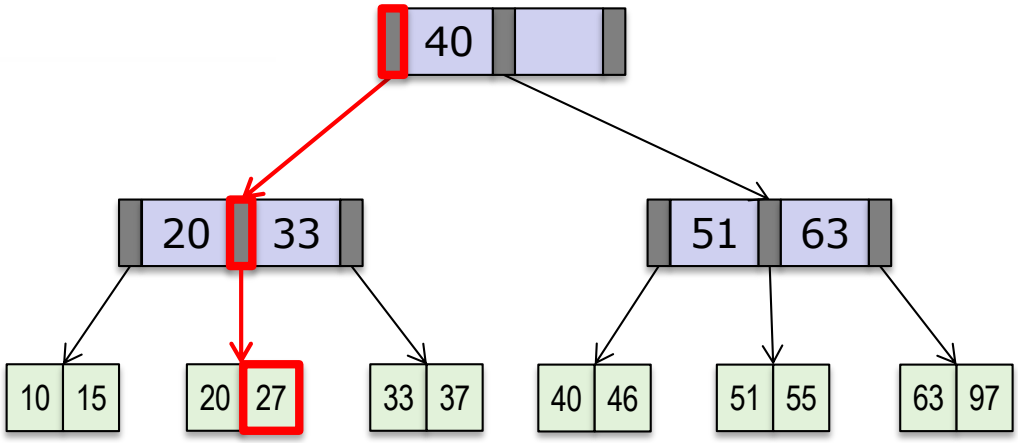


Iskanje po ISAM indeksu

- Postopek:
  - Začnemo v korenu
  - Če iskana vrednost manjša ali enaka ključu korena, sledimo levemu kazalcu, sicer desnemu.
  - Ponavljamo, dokler ne pridemo do listov drevesa. Če iskanega podatka ni v listu, iščemo v dodatnih straneh.

Primer iskanja po ISAM drevesu

- Iščemo element 27

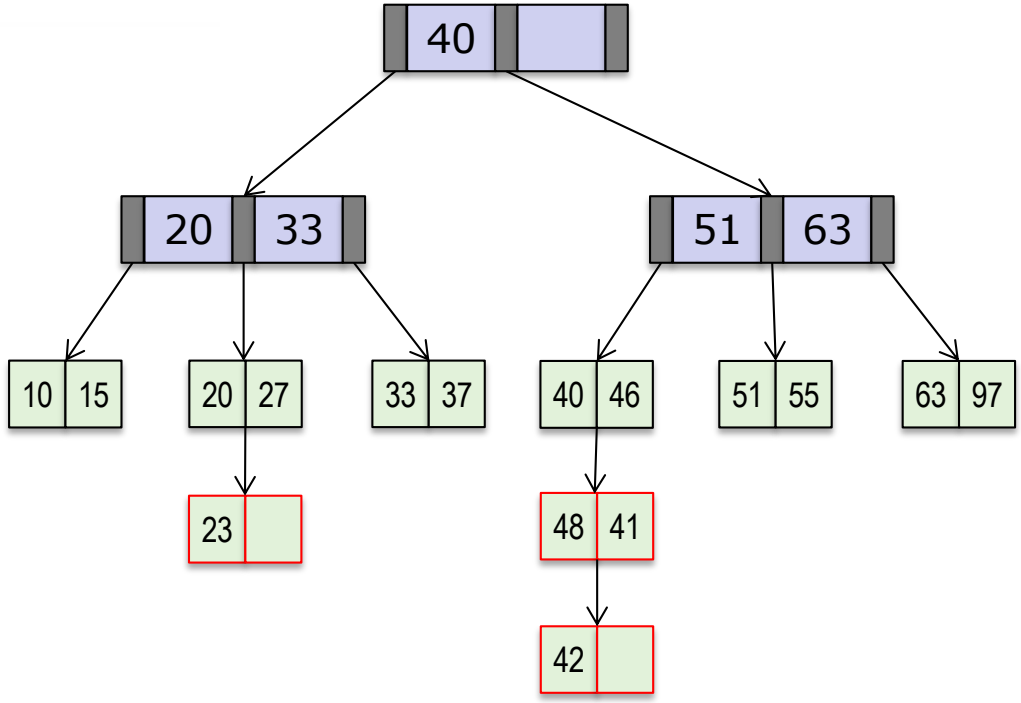


Dodajanje v ISAM indeks

- Postopek:
  - Stran v listih, kamor zapis sodi, poiščemo enako kot pri iskanju
  - Če je prosto mesto v primarnem listu, moramo element dodati na ustrezno mesto (primarni listi so urejeni)
  - Če v primarnih listih ni prostora, dodamo dodamo element na prvo prosto mesto v dodatnih listih. Po potrebi dodamo nov list v seznam dodatnih listov.

Primer dodajanja v ISAM drevo

- Dodajamo elemente 23, 48, 41, 42



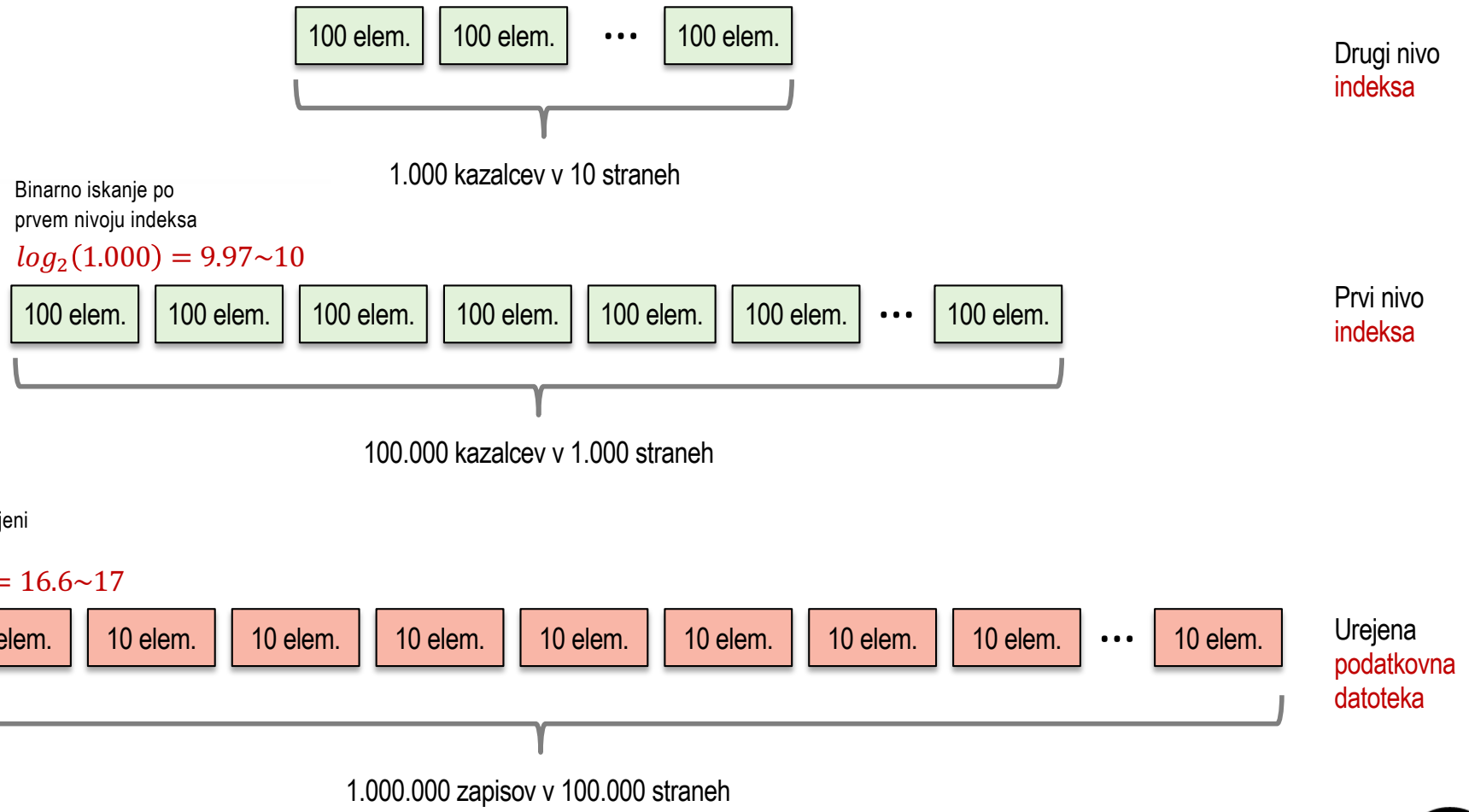
Brisanje iz ISAM indeksa

- Postopek:
  - Zapis, ki ga brišemo, poiščemo enako kot pri iskanju
    - Če je zapis na dodatni strani in gre za zadnji zapis na strani, stran zbrišemo.
    - Če je zapis na primarni strani in gre za zadnji zapis na strani, pustimo stran prazno. Služi kot mesto za kasnejšo rabo.
  - Lahko se zgodi, da se iskalni ključ, ki nastopa v indeksnem delu, ne pojavi v listih.

Stroški iskanja v ISAM strukturi

- Primer izračuna stroškov iskanja zapisa v ISAM strukturi:
  - Število I/O operacij = $\log_F N$, kjer je N število primarnih strani listov, F število otrok vsake indeksne strani.
  - Število I/O operacij pri binarnem iskanju po urejeni datoteki je $\log_2 N$. Pri iskanju po eno-nivojskem indeksu: $\log_2(N/F)$
  - Primer: datoteka z $1.000.000$ zapisi, 10 zapisov na stran v listih in 100 zapisov v indeksnih straneh
    - Strošek branja cele datoteke: 100.000
    - Strošek binarnega iskanja po urejeni datoteki: 17
    - Strošek binarnega iskanja po eno-nivojskem indeksu: 10
    - Strošek iskanja po ISAM strukturi: 3 (brez dodatnih strani)

Primerjava z iskanjem v urejeni datoteki



Omejitve ISAM indeksne strukture

- Problem ISAM strukture se pokaže, ko naraste število dodatnih strani
  - Podatki v dodatnih straneh so načeloma lahko urejene, običajno pa niso (zaradi učinkovitosti dodajanja zapisov)
  - Problem omilimo tako, da v začetku, ko izgradimo indeks, pustimo nekaj praznega prostora v straneh listov.

Kje smo?



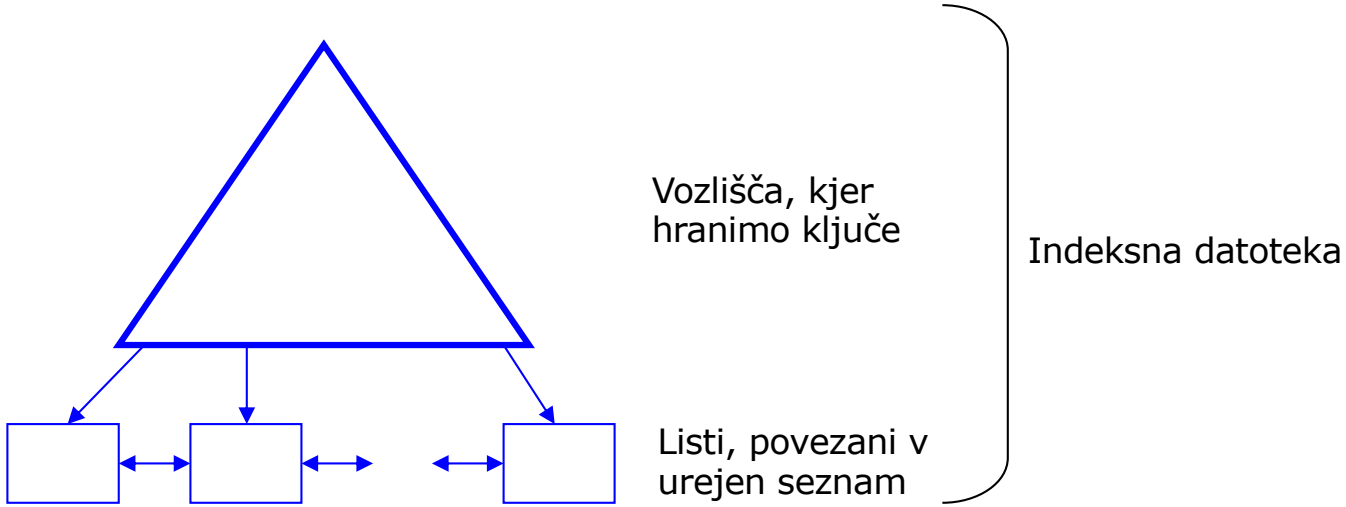
- Datoteke, zapisi in polja
- Datotečne organizacije
  - Neurejene datoteke
  - Urejene datoteke
  - Razpršene datoteke
- Indeksi
- Drevesno indeksiranje
  - ISAM
  - B+drevo
- Bitni in stični indeksi
- Gruče

B+ drevesa…

- B+ indeks je dinamičen → njegova struktura se dinamično prilagaja spremembam v podatkovni datoteki.
- Odpravlja težave ISAM indeksa, npr. s povečevanjem števila dodatnih strani pada učinkovitost…
- B+ je uravnoteženo drevo, katerega vozlišča usmerjajo iskanje, listi pa vsebujejo podatke (ključe).

B+ drevesa

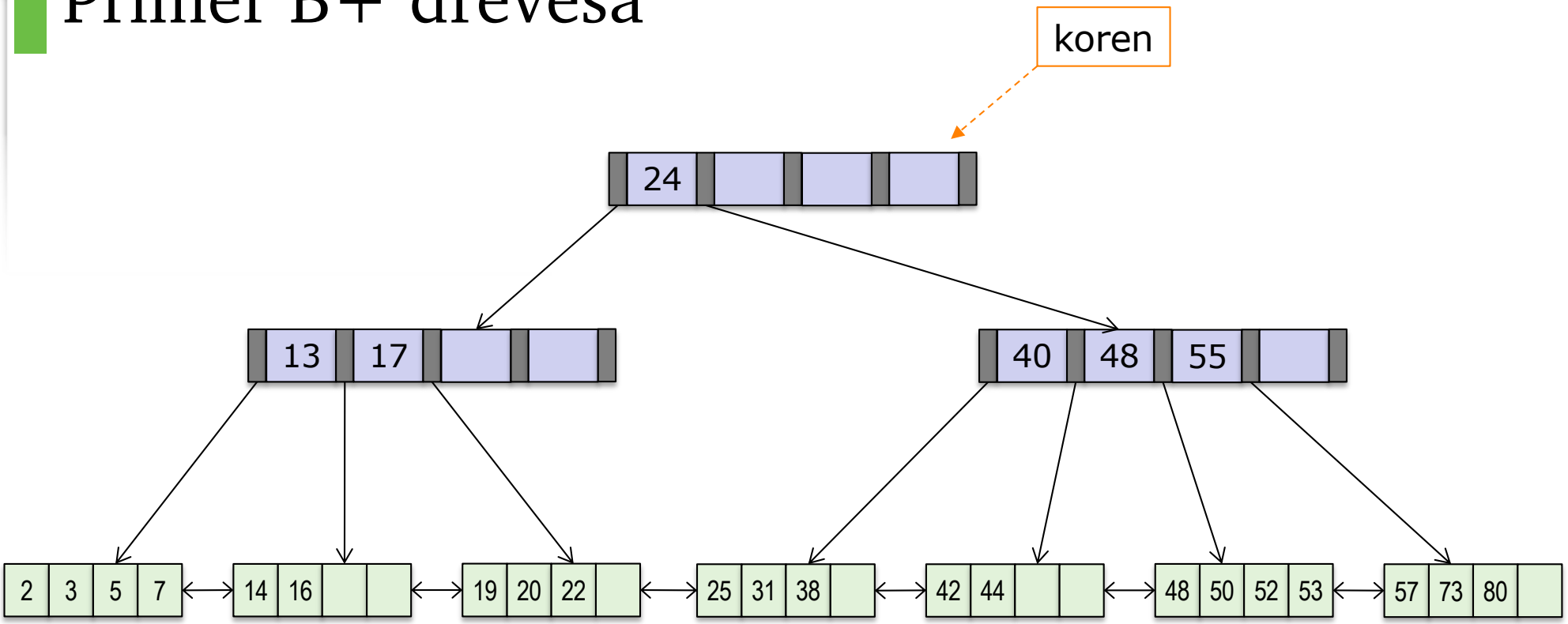
- Zaradi dinamične narave B+ drevesa, strani v listih ni možno alocirati zaporedno. Uporabimo kazalce.
- Liste B+ drevesa uredimo z dvosmernim seznamom.



Lastnosti B+ dreves

- Operacije dodajanja in brisanja ohranjajo drevo uravnoteženo;
- Vozlišča (razen korena) so vsaj 50% zasedena, če uporabimo ustrezen algoritem brisanja;
- Iskanje določene vrednosti zahteva le pot od korena do ustreznega lista.
- Poti do vseh vozlišč so zaradi uravnoteženosti enake in določajo višino drevesa.
- Zaradi velikega razvejanja (parameter F) je višina običajnih B+ dreves majhna (redko več kot 3 ali 4).

Primer B+ drevesa



`d` je parameter B+ drevesa imenovan red drevesa;
`2*d` predstavlja kapaciteto vozlišča. Edina izjema je korensko vozlišče, za katerega velja `1=<m=<2d`.
Vsako vozlišče B+ drevesa vsebuje `m` vpisov, pri čemer velja `d ≤ m≤ 2*d`.

Prednosti B+ dreves

- Uporaba B+ dreves se splača v primerih, ko se podatki velikokrat spreminjajo, obenem pa potrebujemo zaporedno iskanje.

- Prednosti pred uporabo urejenih datotek:
  - Za ceno dodatnega prostora, ki ga porabimo z indeksno datoteko, pridobimo vse prednosti urejene datoteke, obenem pa ne izgubimo na učinkovitosti dodajanja in brisanja.

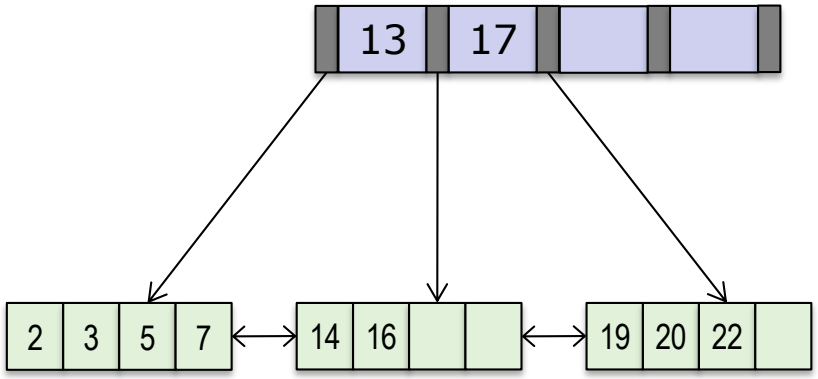
- Prednosti pred uporabo ISAM-a:
  - Zaradi dodajanja ni potrebno ustvarjati dodatnih oziroma presežnih strani.

Operacije z B+ drevesi

- Iskanje - enostavno

- Dodajanje
- Bisanje

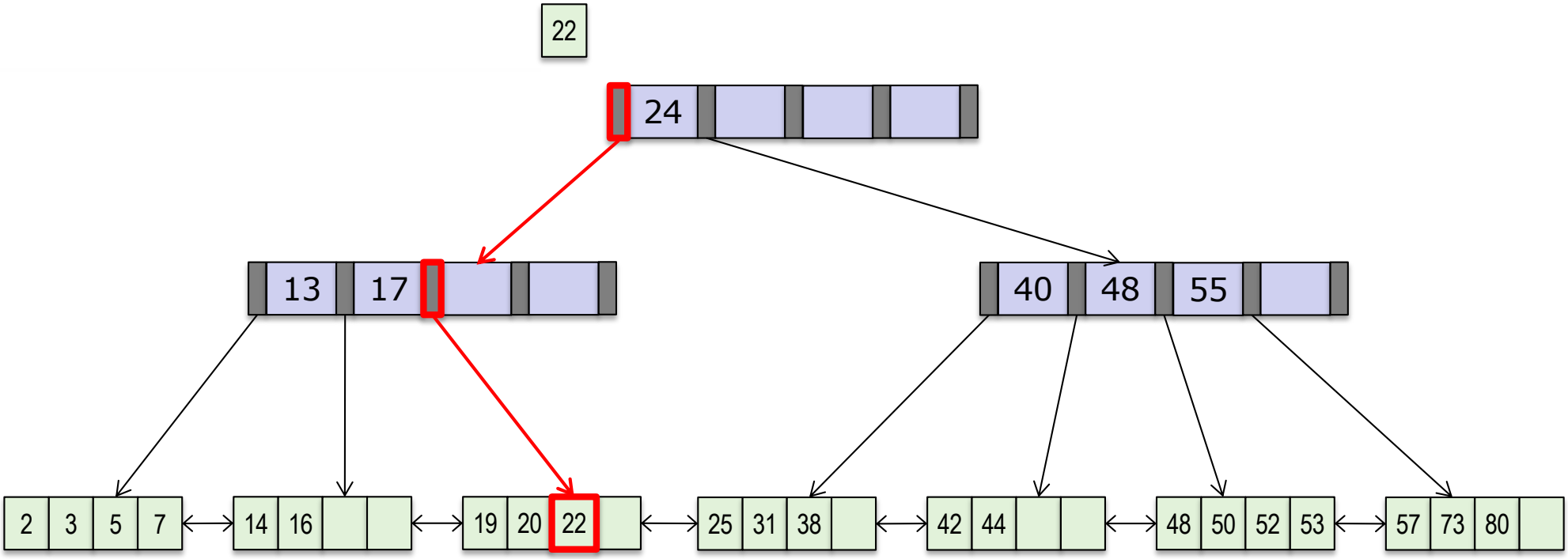
- kompleksno, potrebno zagotavljati uravnoteženost drevesa



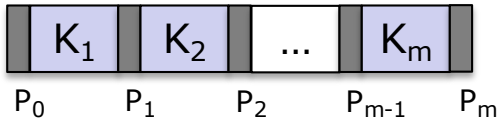
Iskanje v B+ drevesu

Postopek: iščemo element k
1.Premakni se v koren
2.Poišči ustrezen kazalec k_i
3.Premakni se na naslednje vozlišče prek kazalca k_i
4.Vrni se na korak 2

- Poišči element: 22

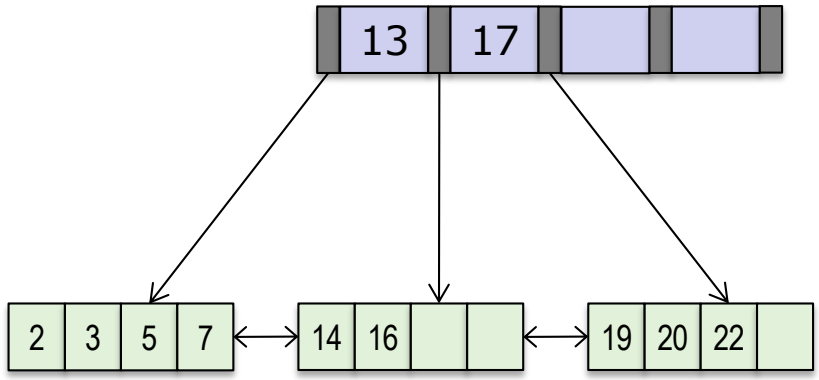


Algoritem za iskanje v B+ drevesu




```
//v drevesu s korenom root poišči vozlišče z vrednostjo K
function find (search key value K): return nodepointer
  return tree_search(root, K);
end;

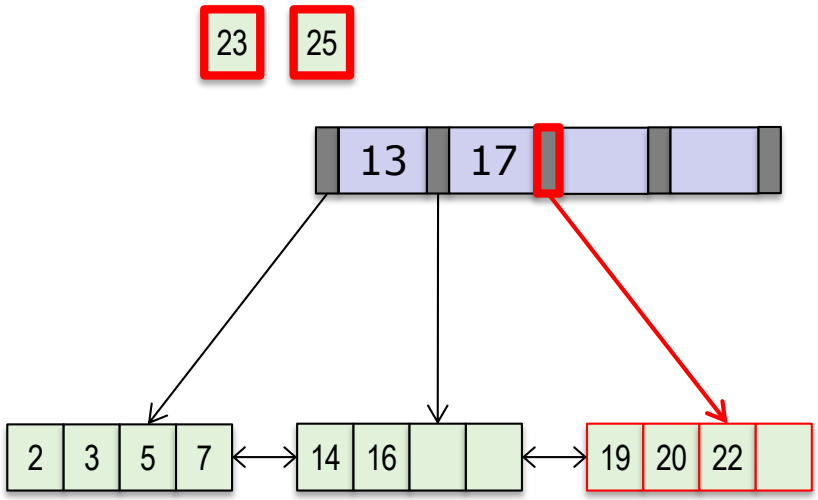
//v drevesu poišči vozlišče z vrednostjo K. Začni pri vozlišču, na katerega kaže nodepointer.
function tree_search (nodepointer, search key value K) return nodepointer
  if *nodepointer is a leaf, return nodepointer
  else
    if K < K1 then return tree_search(P0, K)
    else
      if K > Km then return tree_search(Pm, K)
      find i such that Ki ≤ K < Ki+1;
      return tree_search(Pi, K)
end;
```



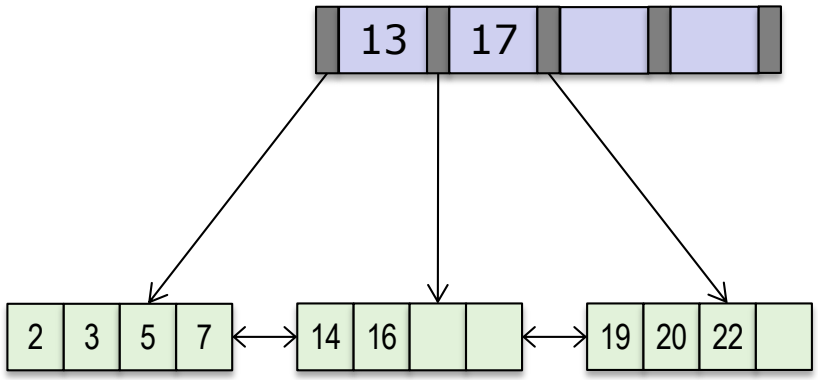
Dodajanje elementov v B+ drevo

- Dodajanje elementov zahteva razdelitev vozlišča, če je to že polno.

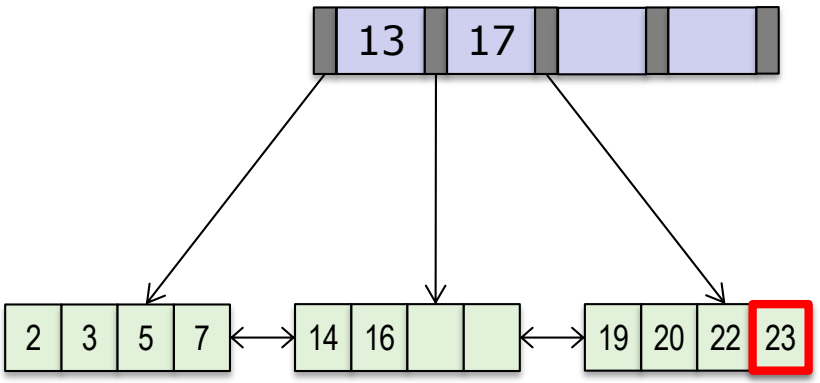
Primer: dodajamo elementa 23 in 25



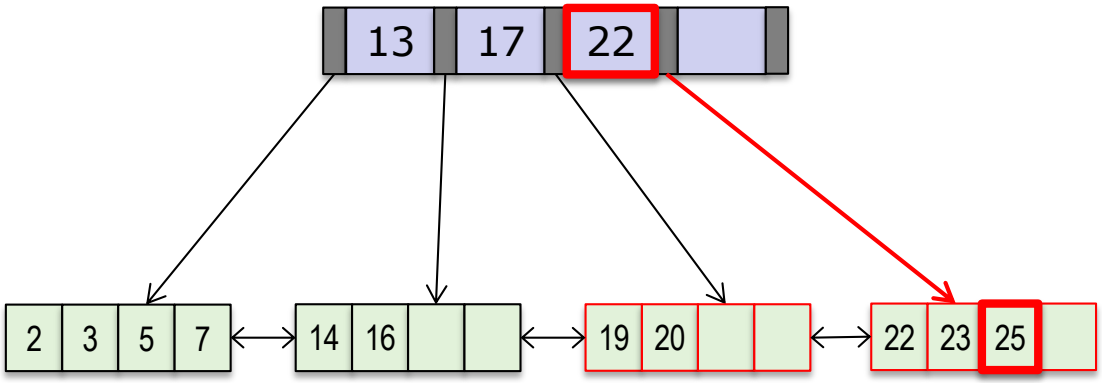
Korak **1**: začetno drevo



Korak **2**: po dodajanju elementa 23

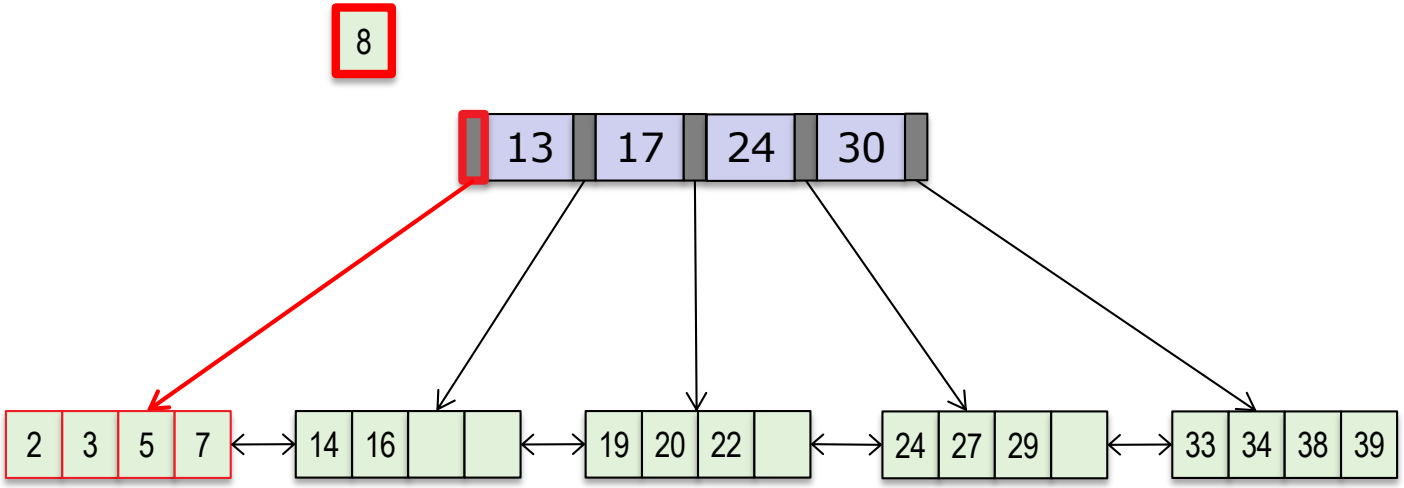


Korak **3**: končno drevo – po dodajanju elementa 25

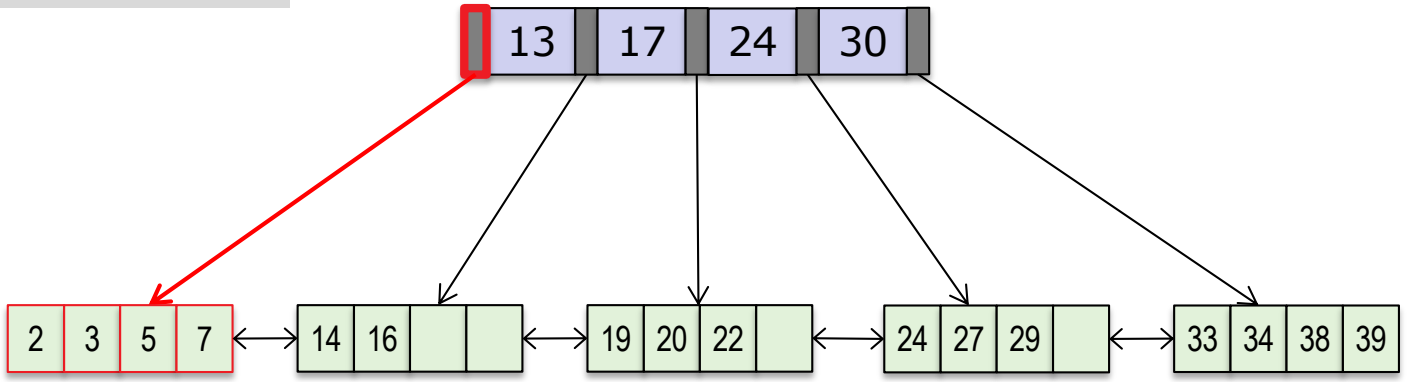


Primer: dodajanje nivoja

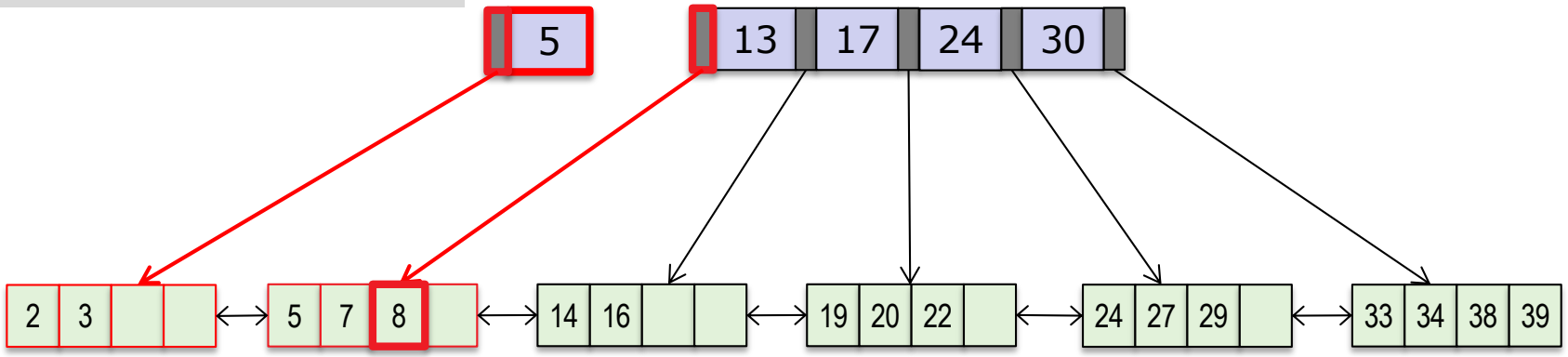
Dodajamo element: 8



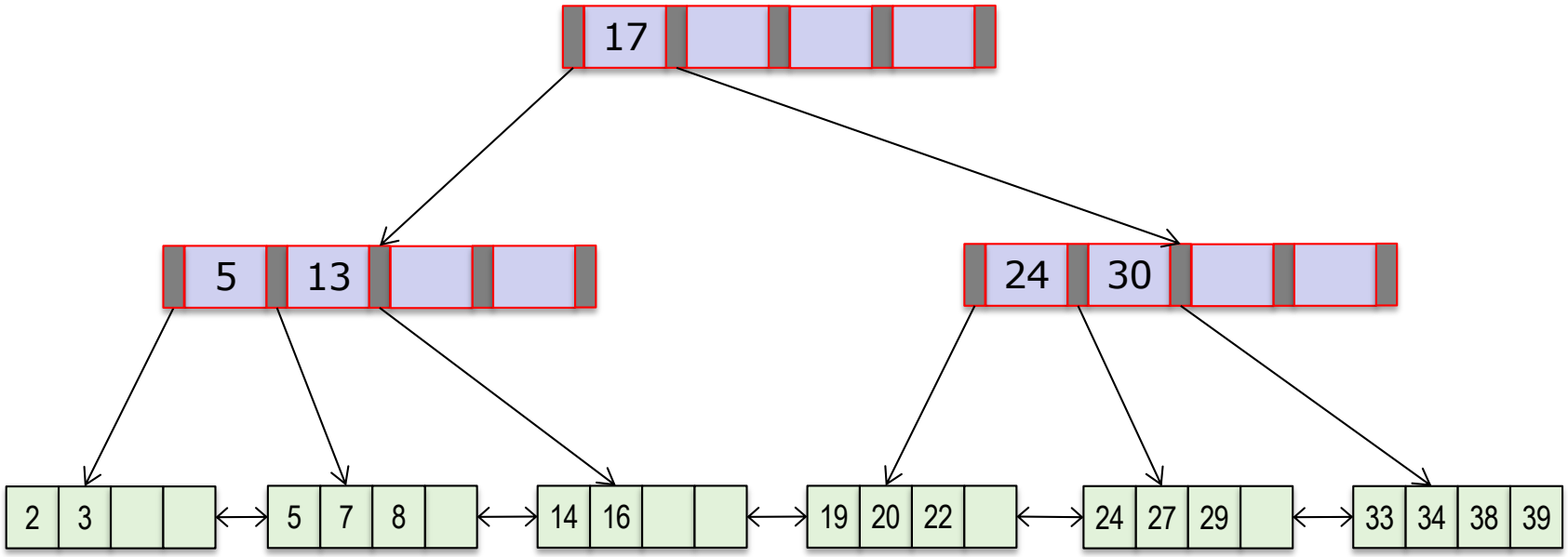
Korak **1**: začetno drevo



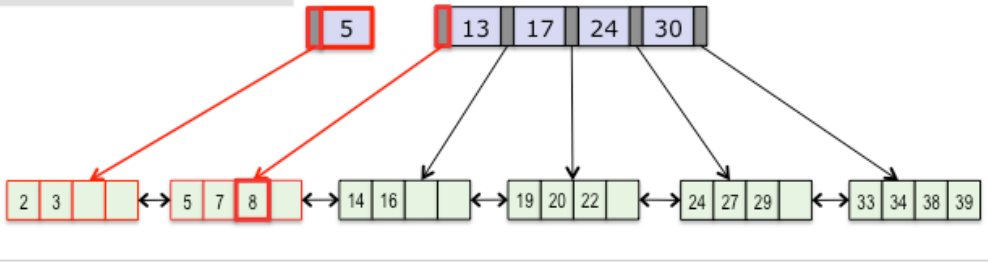
Korak **2**: dodamo element 8



Korak **3**: končno drevo - po delitvi indeksnega vozlišča



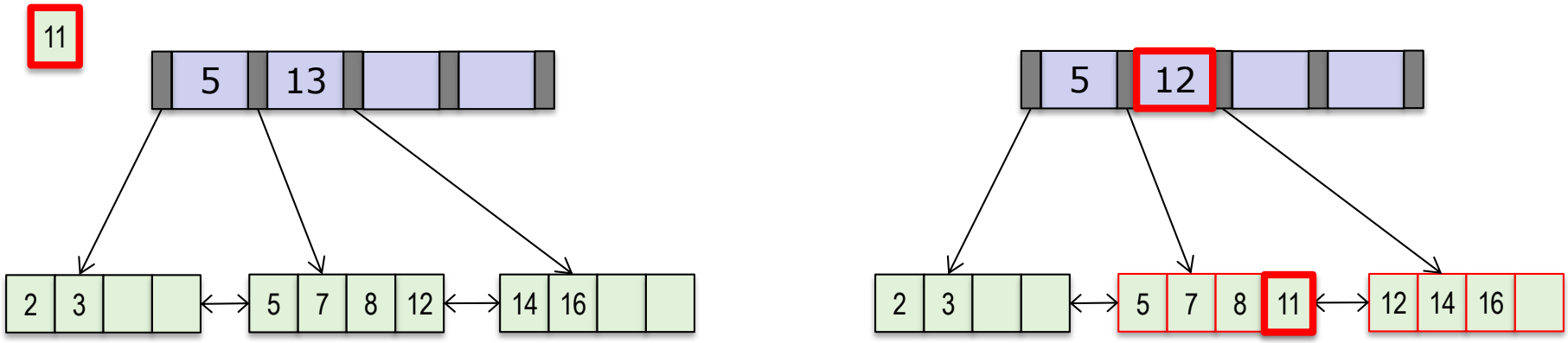
Korak **2**: dodamo element 8



Dodajanje elementov v B+ drevo

- Če je vozlišče polno, lahko namesto delitve uporabimo redistribucijo, če je možna.

Primer: dodajamo element 11

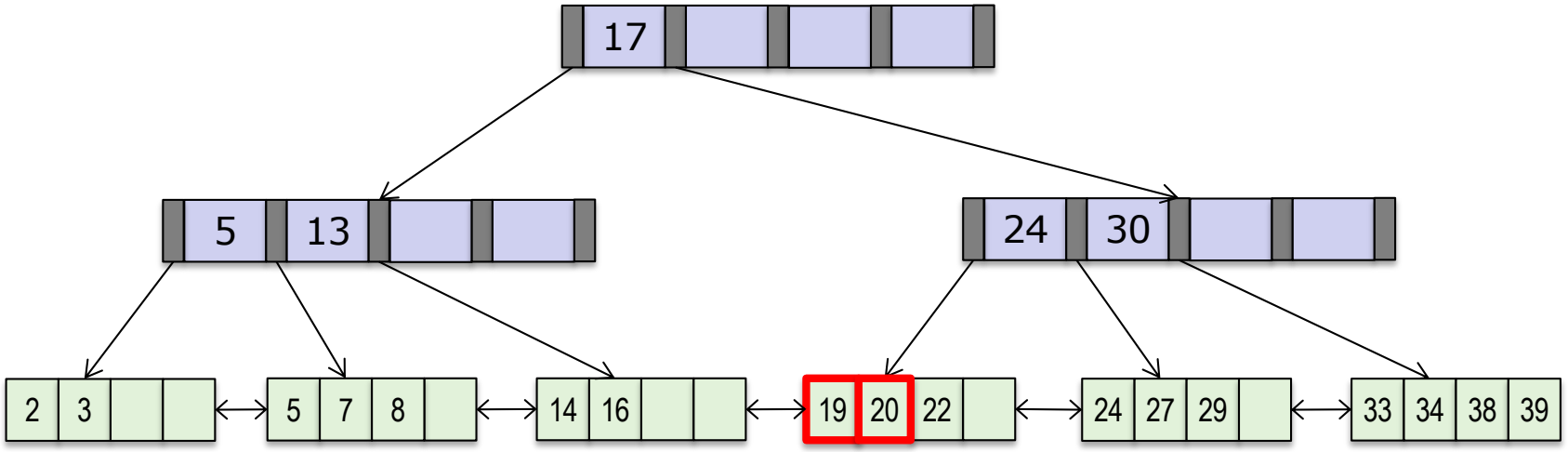


Brisanje elementov iz B+ drevesa

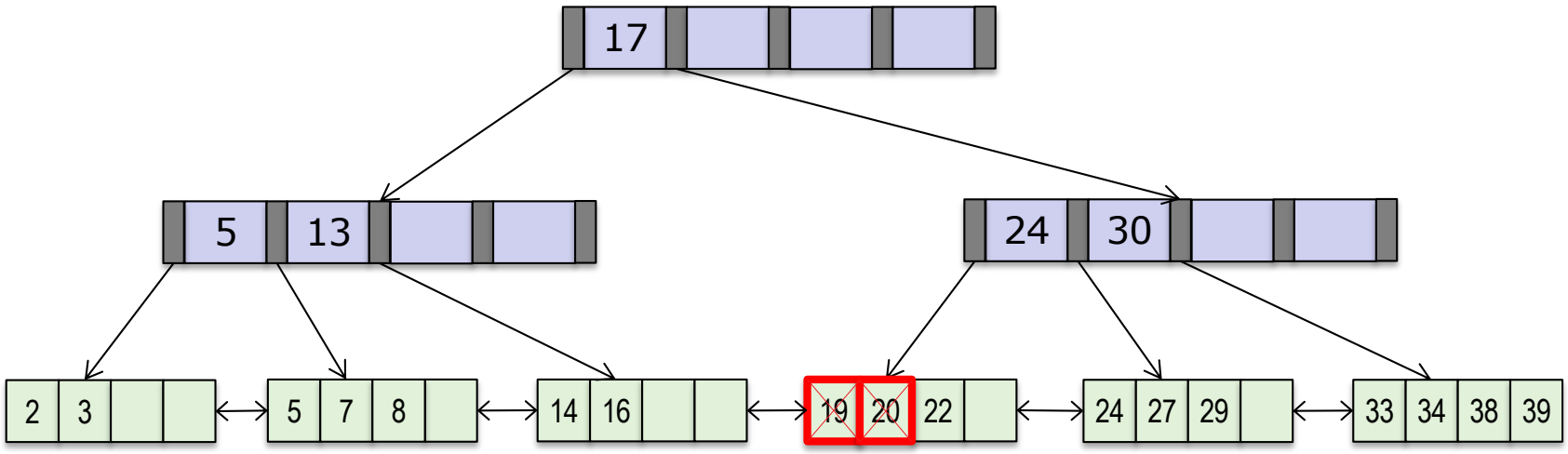
- Postopek:
  - Poiščemo zapis in ga brišemo.
  - Če se zgodi, da zasedenost lista pade pod mejo, moramo redistribuirati podatke s sosednjimi listi ali pa združiti več listov skupaj.
  - Če podatke redistribuiramo med dve vozlišči, moramo ustrezno spremeniti vsebino nad-vozlišča.
  - Če podatke združimo iz dveh vozlišč, moramo vsebino njunih nad-vozlišč spremeniti tako, da brišemo indeksno polje za drugo vozlišče.

Primer: redistribucija pri brisanju

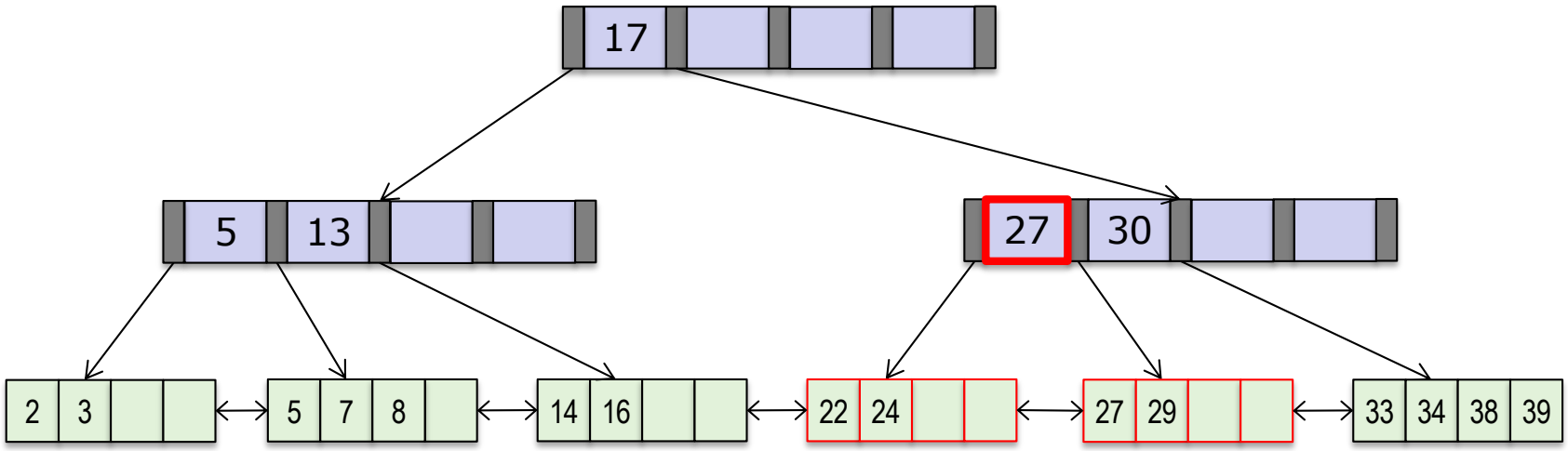
- Brišemo elementa `19` in `20`



Korak **1**: začetno drevo - brišemo elemente 19 in 20

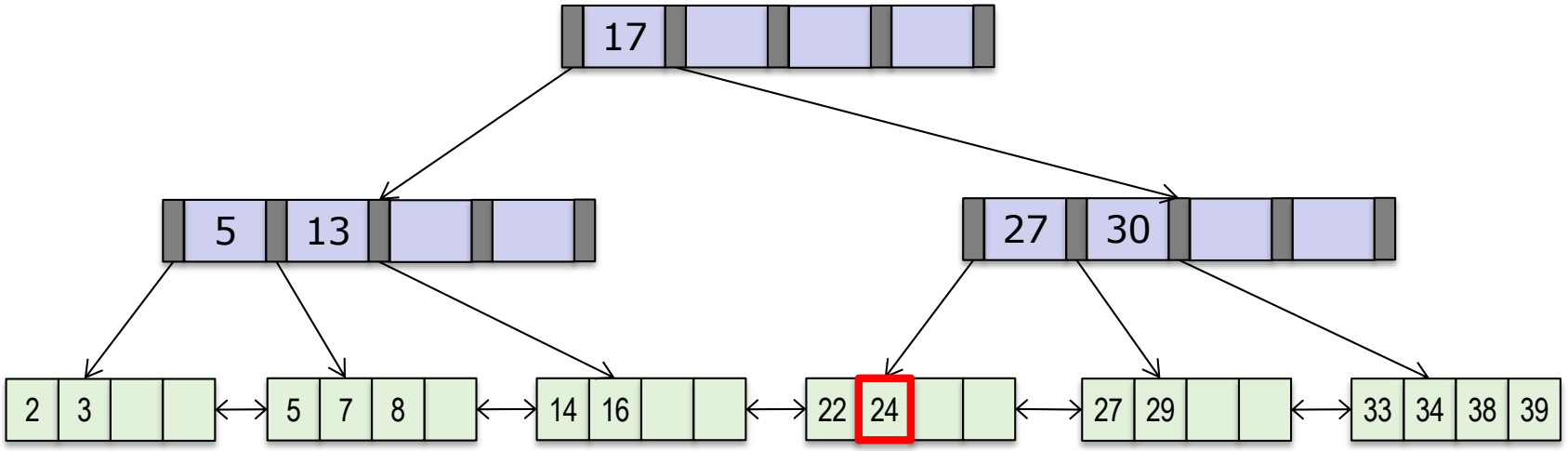


Korak **2**: končno drevo – po redistribuciji

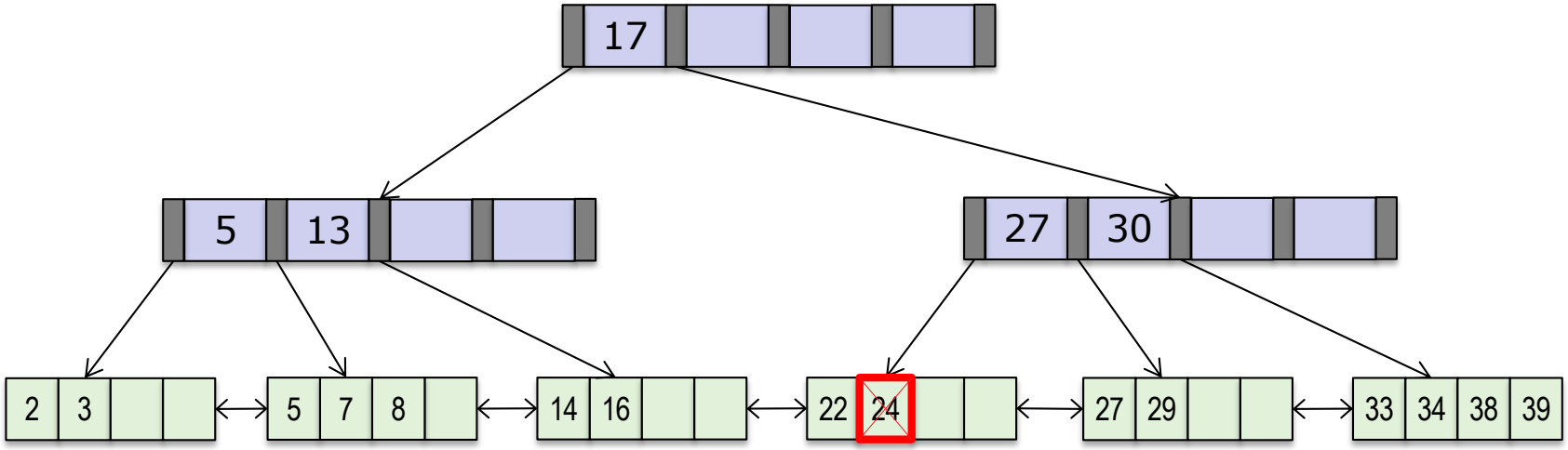


Primer: združevanje pri brisanju

- Brišemo element 24

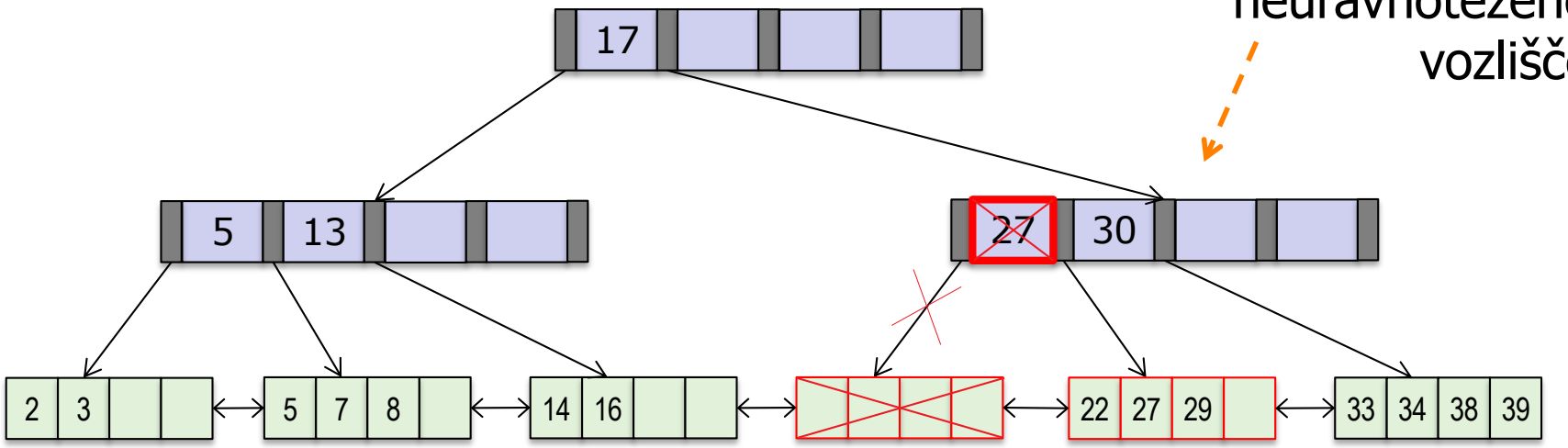


Korak **1**: začetno drevo - brišemo element 24

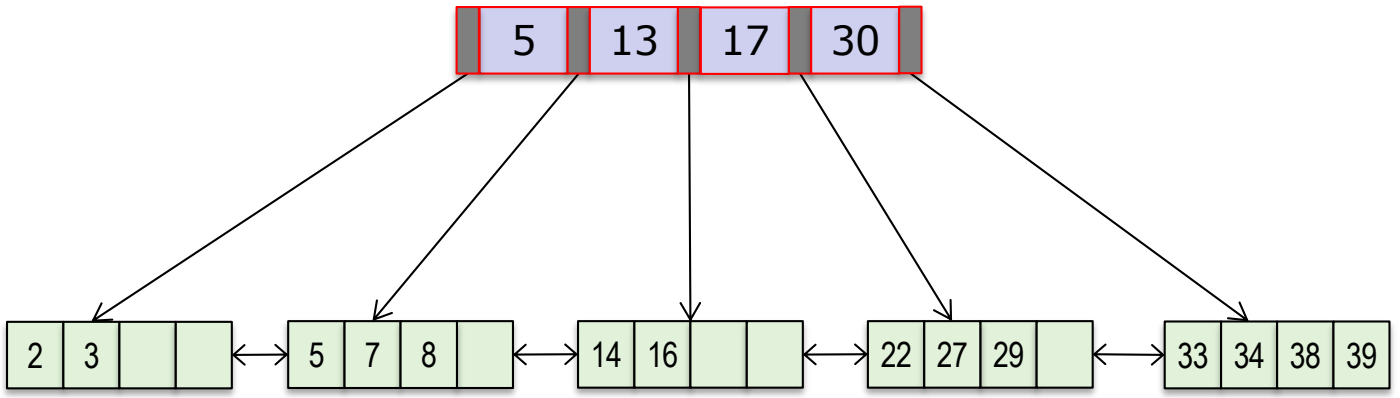


Korak **2**: vmesno drevo – po brisanju in združevanju

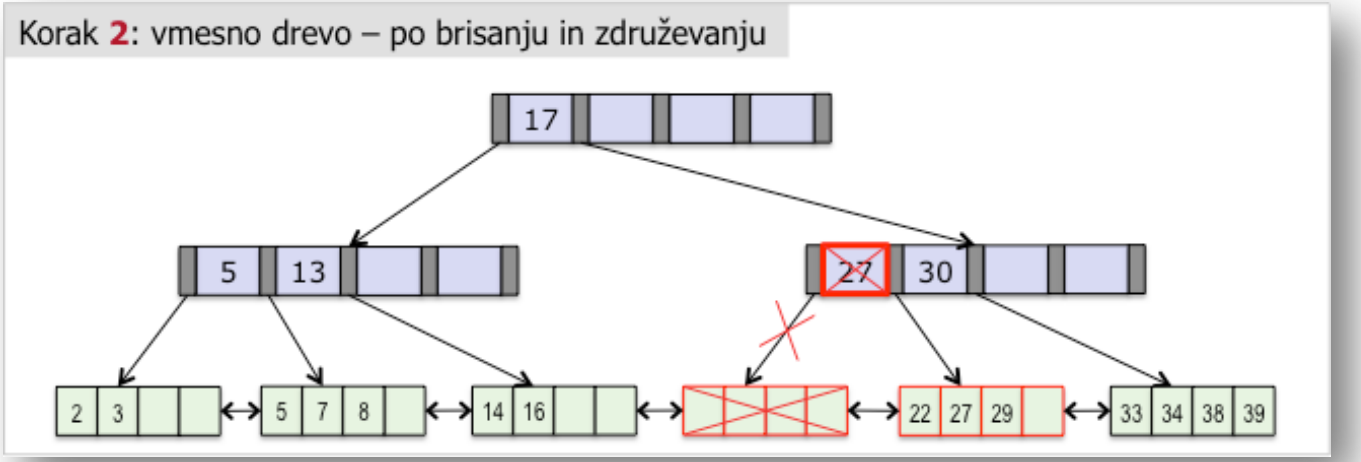
Problem: neuravnoteženo vozlišče



Korak **3**: končno drevo – po združitvi v indeksnem delu



Korak **2**: vmesno drevo – po brisanju in združevanju



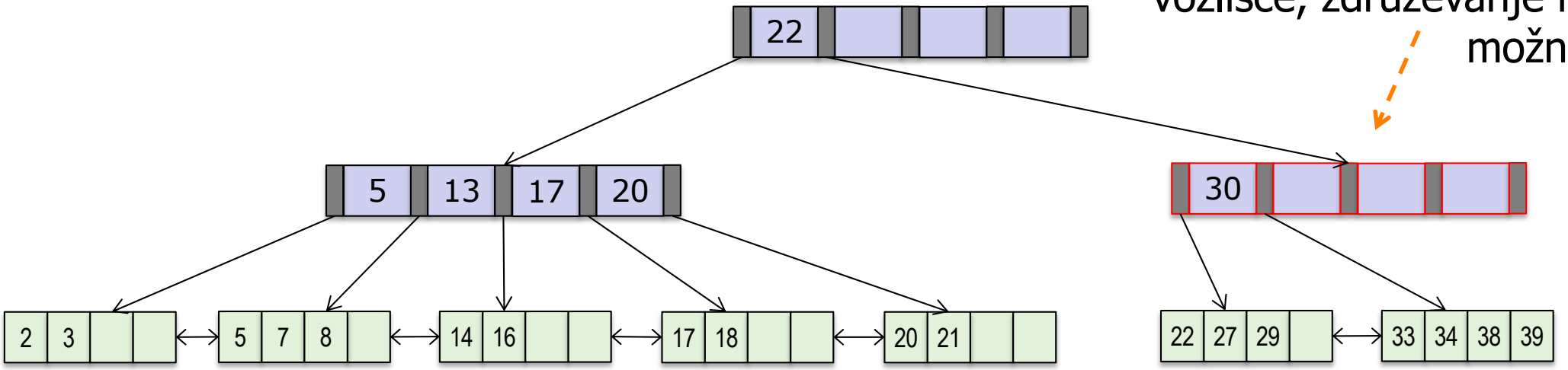
Brisanje elementov iz B+ drevesa

- Pogledali smo, kako deluje:
  - redistribucija v listih,
  - združevanje v listih in
  - združevanje v indeksnem delu
- Ostane:
  - redistribucija v indeksnem delu

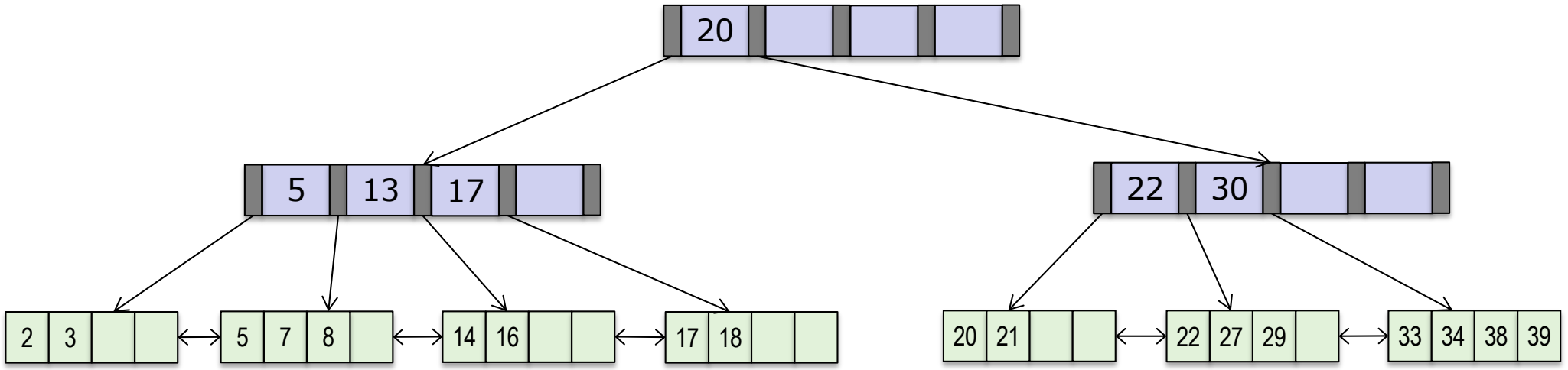
Primer: redistribucija v indeksnem delu

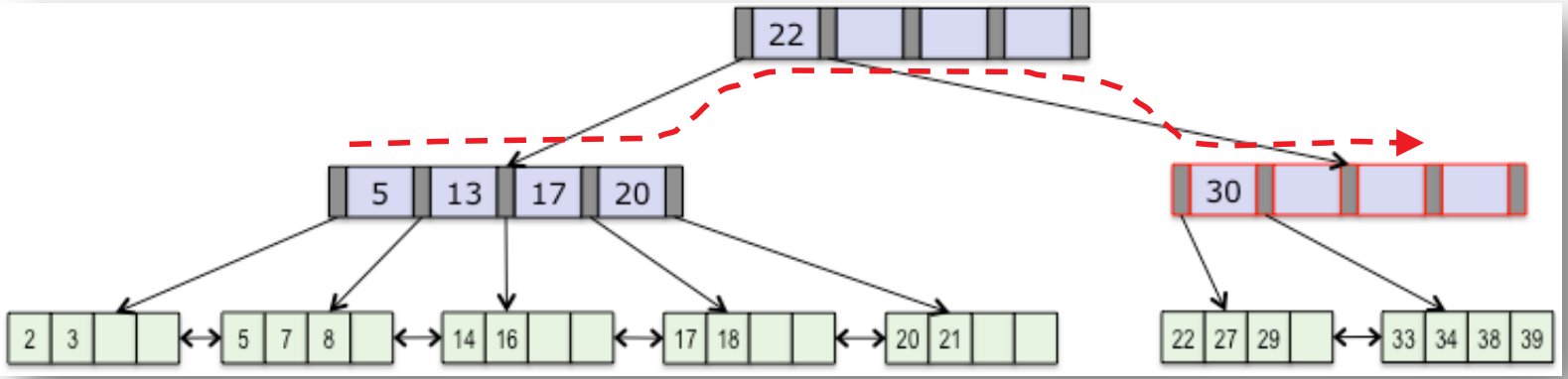
- Primer vmesnega neuravnoteženega drevesa

Problem: neuravnoteženo vozlišče, združevanje ni možno



Korak **3**: končno drevo – po zamiku indeksnih elementov za eno mesto v desno





B+ drevesa, ponovitev

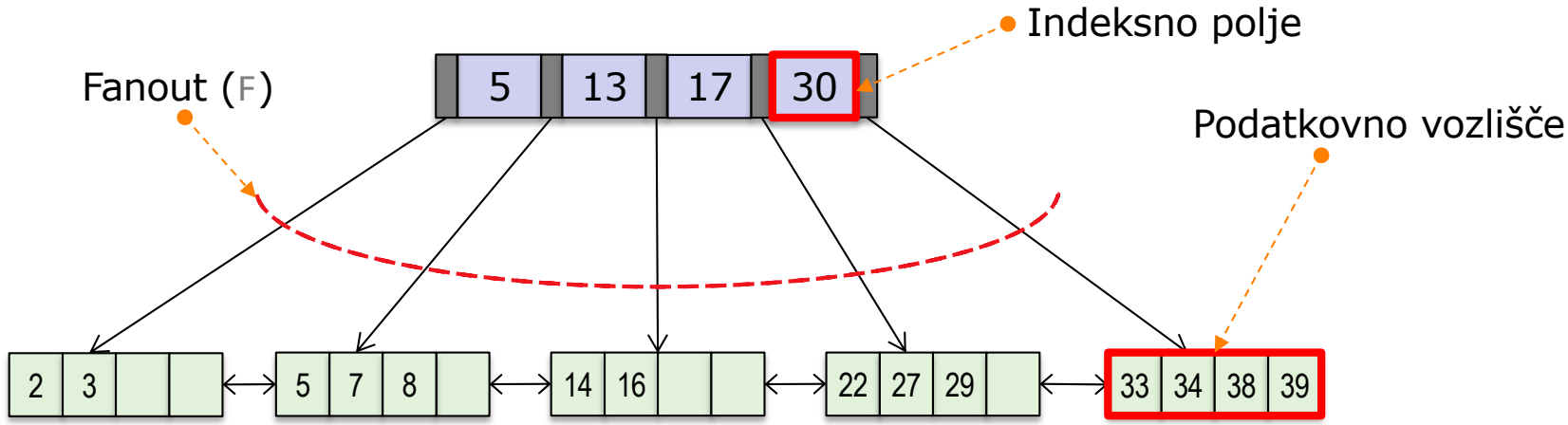
	List	Indeksni del
Delitev	Ključ iz lista kopiraj nivo višje	Delitveni ključ prenesi nivo višje in poveži na novo vozlišče
Združevanje	Briši ključ iz indeksnega dela	Prenesi ključ iz višjega nivoja
Redistribucija	Ključ v indeksnem delu nadomesti s ključem v listu	Ključe v indeksnem delu zamakni za eno mesto v desno

Stiskanje ključev

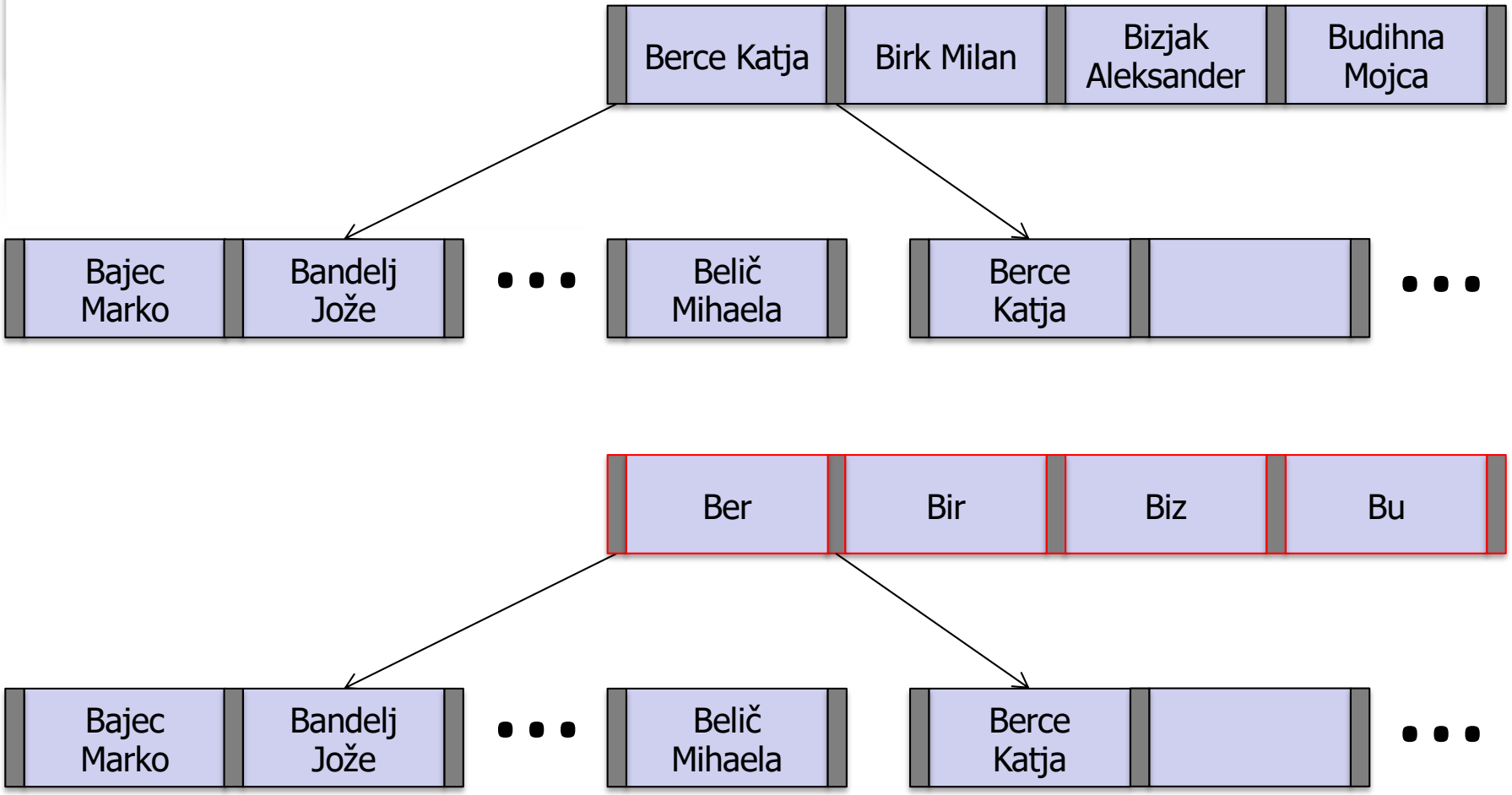
- Višina drevesa je premo sorazmerna številu podatkovnih vozlišč in obratno sorazmerna velikosti indeksnih polj (F).

- Število I/O operacij za branje podatkovnega polja = h.

$$h = \log_F(\#\ \text{podatkovnih polj})$$



Primer


Berce Katja
Birk Milan
Bizjak Aleksander
Budihna Mojca
Bajec Marko
Bandelj Jože
Belič Mihaela
Berce Katja
Ber
Bir
Biz
Bu
Bajec Marko
Bandelj Jože
Belič Mihaela
Berce Katja

Bitni indeks…

staffNo	fName	lName	position	salary	branchNo
SG14	David	Ford	Manager	18000	B003
SG37	Ann	Beech	Assistant	12000	B003
SL21	John	White	Supervisor	30000	B005
SL41	Julie	Lee	Assistant	9000	B005
SF56	Tim	Becker	Manager	32000	B005

Tabela Staff

Manager	Assistent	Supervisor
1	0	0
0	1	0
0	0	1
0	1	0
1	0	0

Bitni indeks po polju position

B003	B005
1	0
1	0
0	1
0	1
0	1

Bitni indeks po polju branchNo

Bitni indeks

- Prednosti bitnega indeksa v primerjavi z B+ drevesi:
  - Kompaktnost
  - Večja učinkovitost pri iskanju po več predikatih (če za njih obstajajo bitni indeksi)

Primer:

SELECT staffNo, salary
FROM Staff
WHERE position = 'Supervisor' **AND** branchNo = 'B003'

Za rezultat pomnožimo bitna vektorja za vrednost Supervisor in B003.

Supervisor
0
0
1
0
0

B003
1
1
0
0
0

Stični indeks…

- Stični indeksi (*Join index*) podobno kot bitni indeksi popularni na področju podatkovnih skladišč.
- Koncept:
  - Uporabljajo se za tabele, ki imajo skupna polja (npr. tuji ključi).
  - Indeks vsebuje za vsako vrednost ključa kazalec na vse tabele, ki so indeksirane z istim stičnim indeksom.
  - Način sortiranja indeksa je poljuben.

Primer stičnega indeksa

Tabela Branch

rowID	branchNo	street	city	postcode
20001	B005	22 Deer Rd	London	SW1 4EH
20002	B007	16 Argyll St	Aberdeen	AB2 3SU
20003	B003	163 Main St	Glasgow	G11 9QX
20004	B004	32 Manse Rd	Bristol	BS99 1NZ
20005	…	…	…	…

branch rowID	property rowID	city
20001	30002	London
20002	30001	Aberdeen
20003	30003	Glasgow
20003	30004	Glasgow
…	…	…

Tabela Property

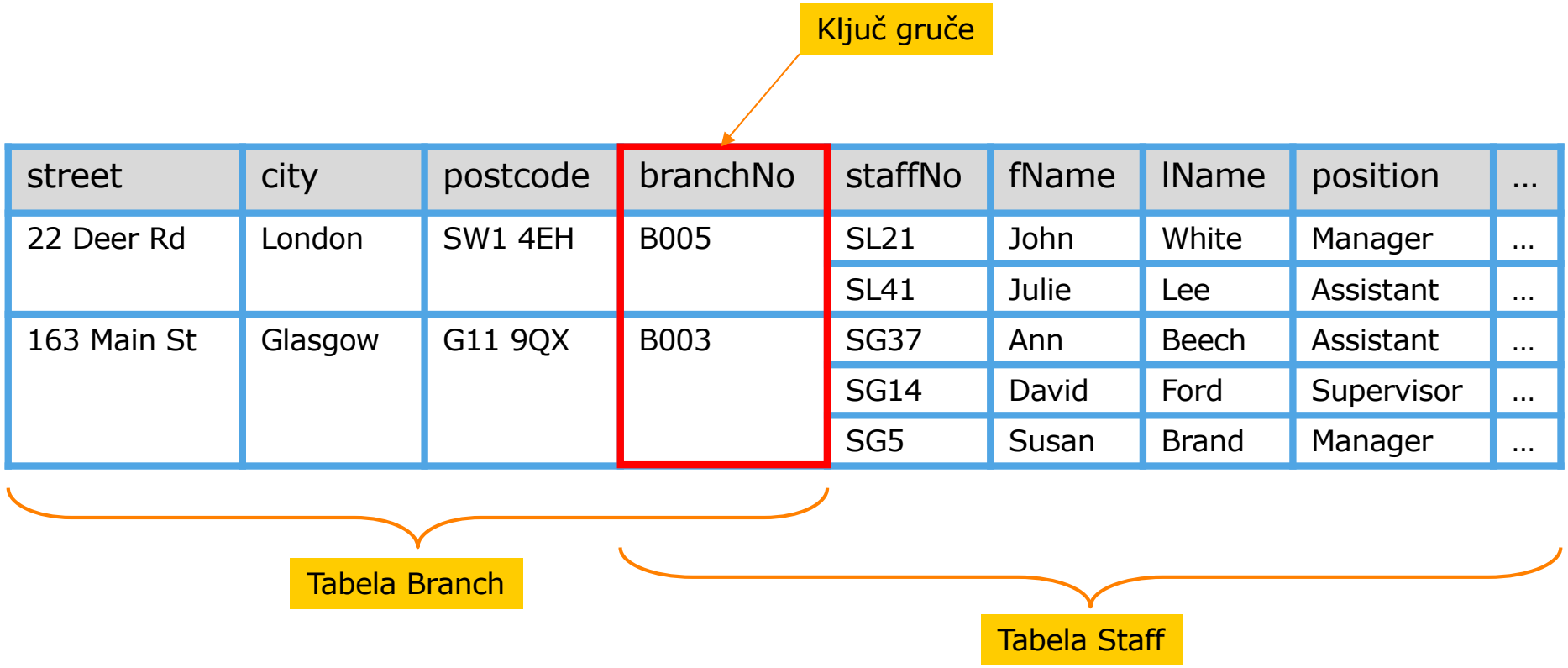
rowID	propertyNo	street	city	postcode	…
30001	PA14	16 Holhead	Aberdeen	AB2 3SU	
30002	PL94	6 Argyll St	London	SW1 4EH	
30003	PG4	6 Lawrence St	Glasgow	G11 9QX	
30004	PG36	2 Manor Rd	Glasgow	G11 9QX	
30005	…	…	…	…	…

Koliko nepremičnin je v mestih, kjer so locirane organizacijske enote nepremičninske agencije?

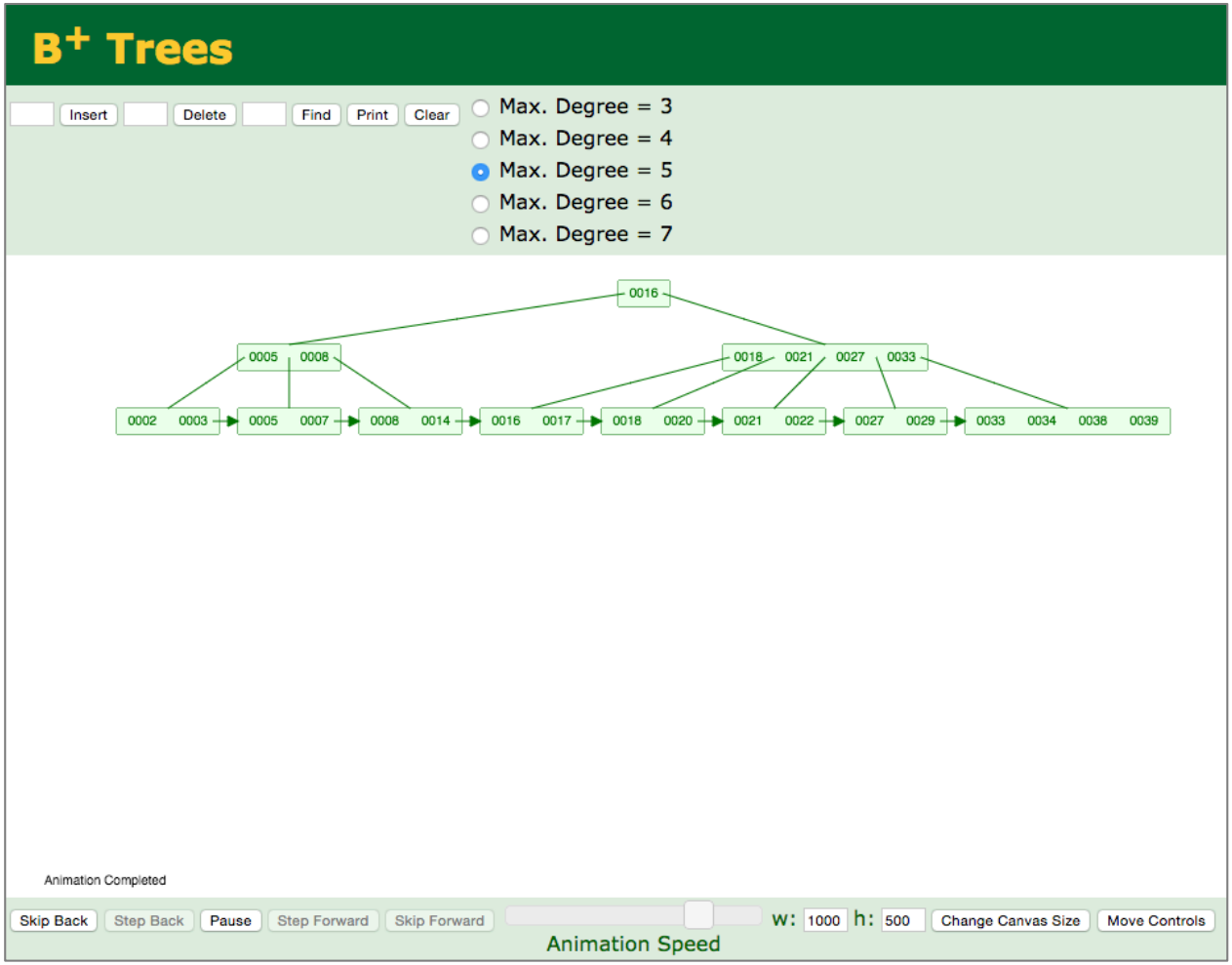
Gruče…

- Nekateri SUPB nudijo združevanje relacij v gruče (*clustered relations)*
- Gruča označuje skupino relacij, ki so fizično shranjene skupaj, ker imajo nekatere stolpce enake in se pogosto uporabljajo skupaj.
- Gruče izboljšajo čas dostopa do podatkov na disku.
- Stolpce, ki so za relacije v gruči skupni, imenujemo ključ gruče (*cluster key*).

Primer gruče



Vizualizacija B+ dreves



https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html

Ponovitev

- Kje so shranjene tabele podatkovne baze?
- Kakšna je najmanjša enota, ki jo preberemo iz diska?
- Koliko zapisov ima ena stran (blok)?
- Kaj vpliva bolj na učinkovitost poizvedovanja, višina ali širina B+ drevesa?
- Ali imamo lahko več primarnih indeksov?
- Kdaj uporabljamo bitne indekse?

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)

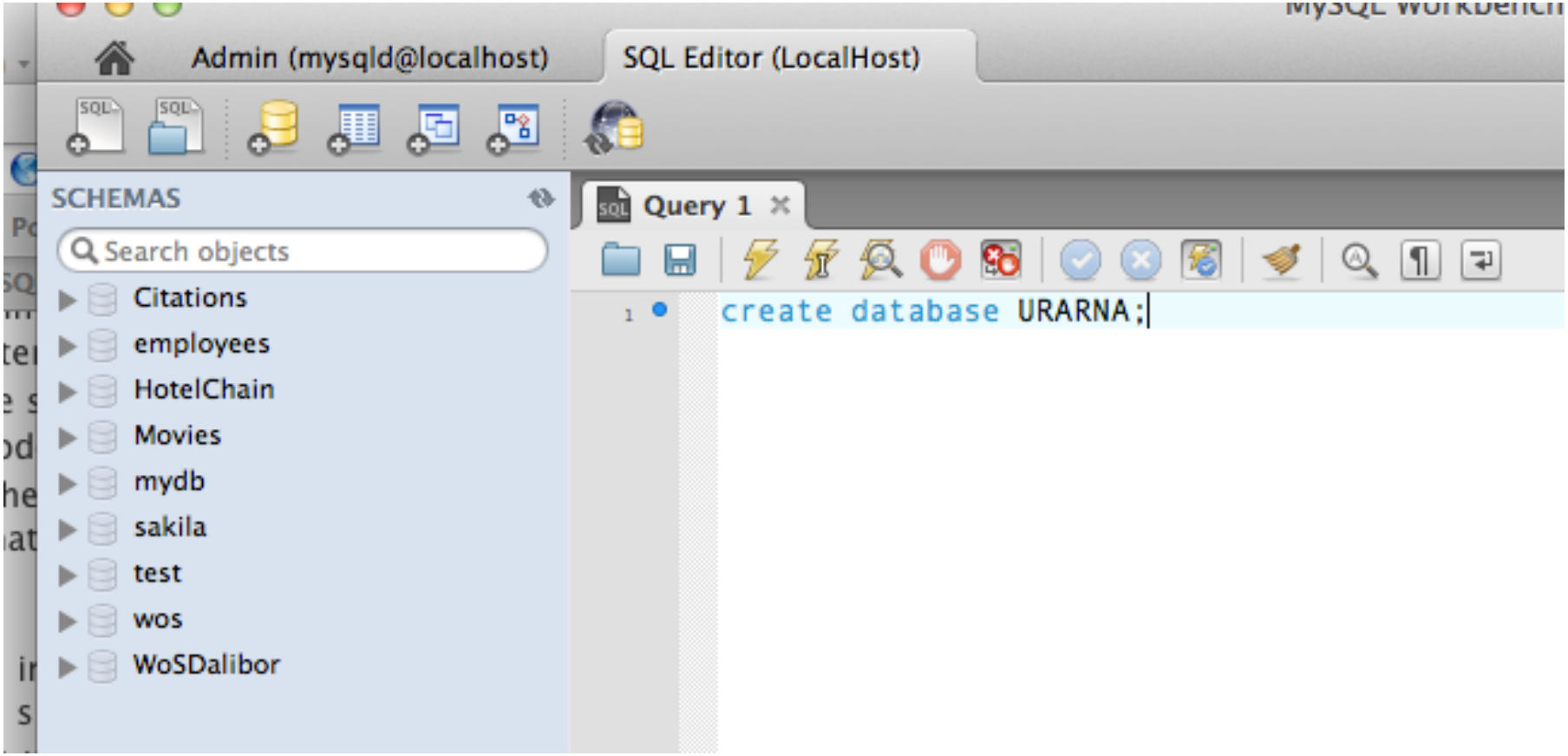
- ARTIKLI IN STORITVE
  - Šifra artikla
  - Naziv artikla
  - Enota mere
  - Zaloga
  - Kritična zaloga
  - Vrsta artikla
  - ...
- ...

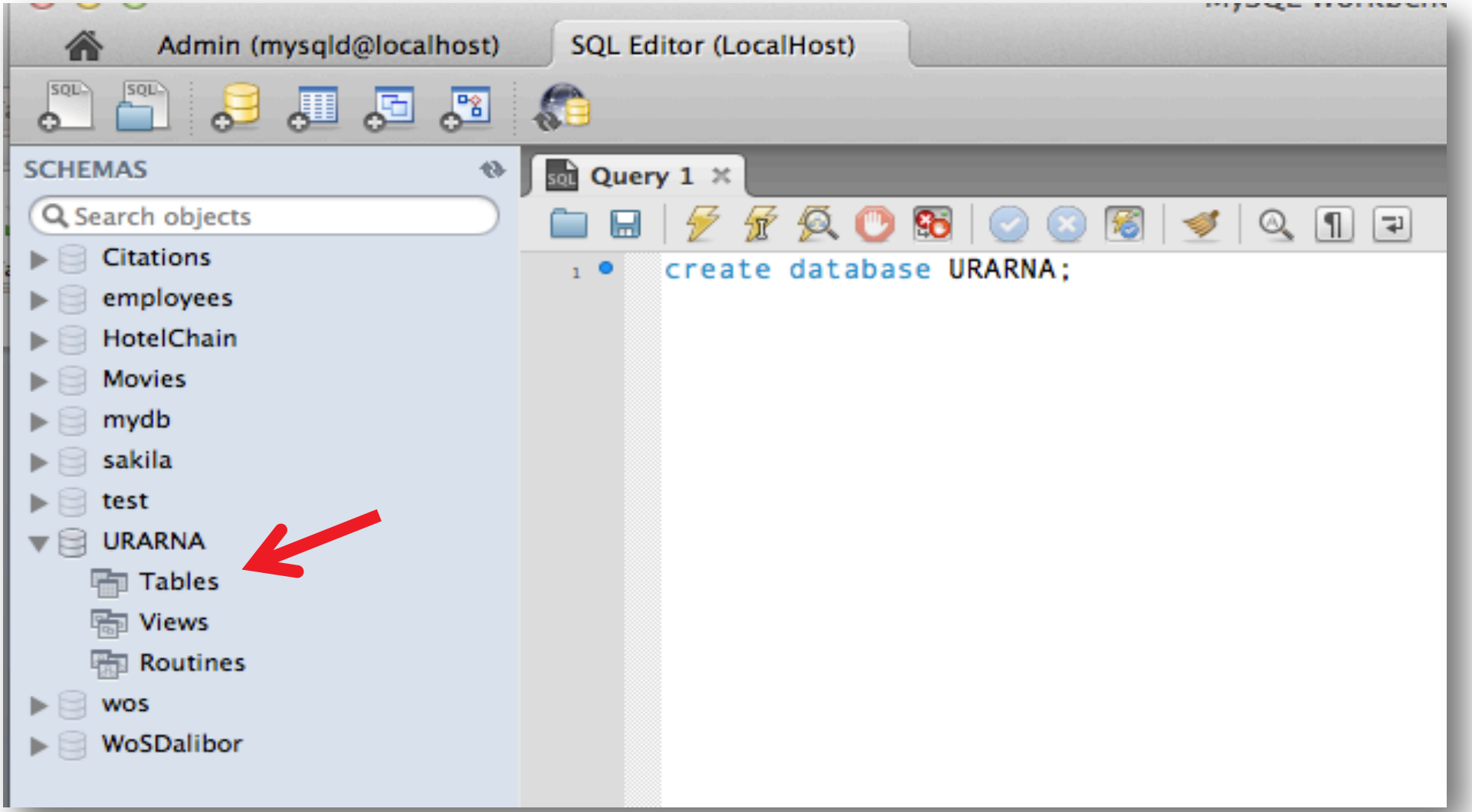
Kreiranje baze z uporabo SQL DDL

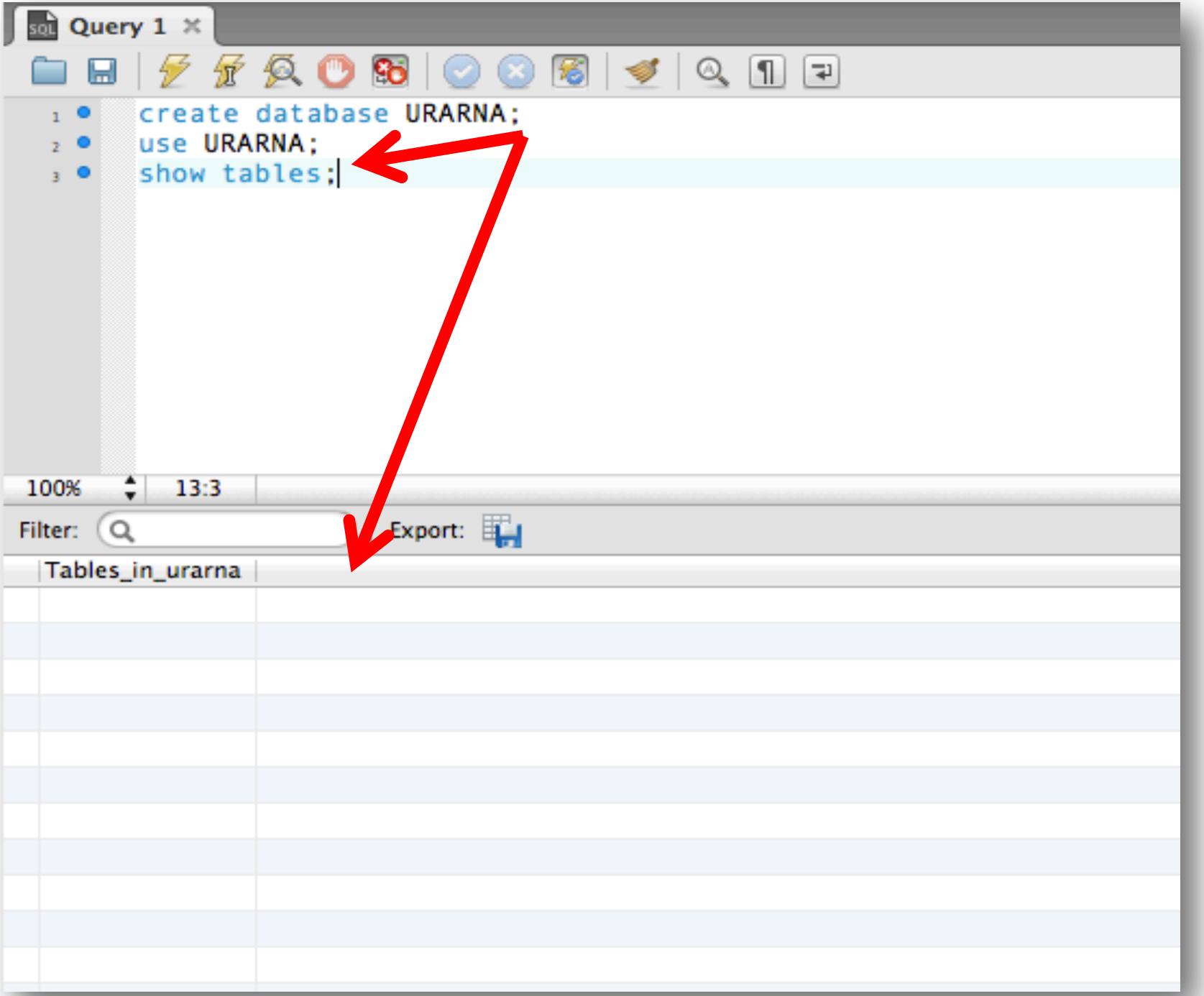
```
CREATE DATABASE urarna;
USE urarna;

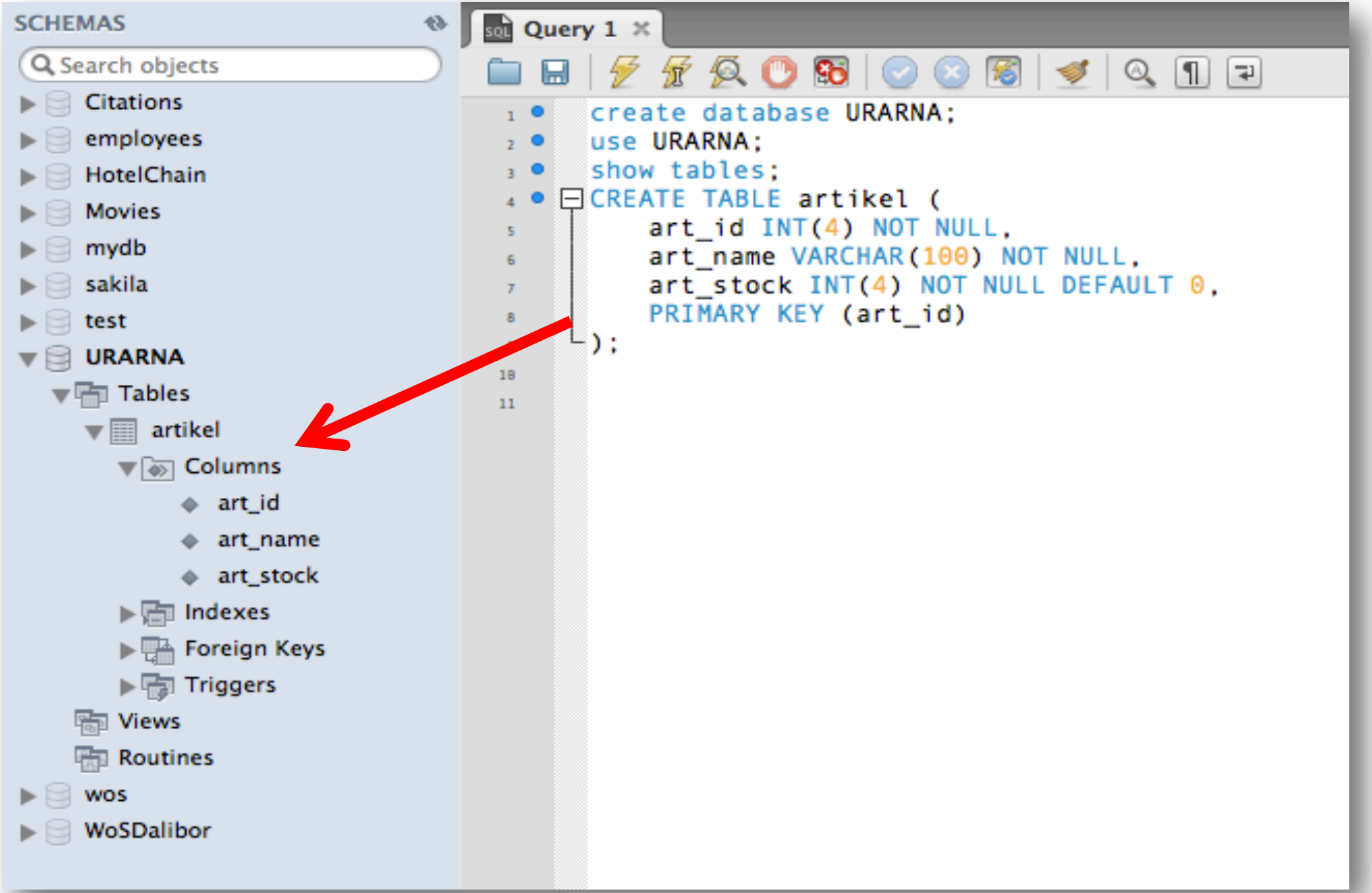
CREATE TABLE artikel (
        art_id INT(4) NOT NULL,
        art_name VARCHAR(100) NOT NULL,
        art_stock INT(4) NOT NULL DEFAULT 0,
        PRIMARY KEY (art_id)
...
```

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

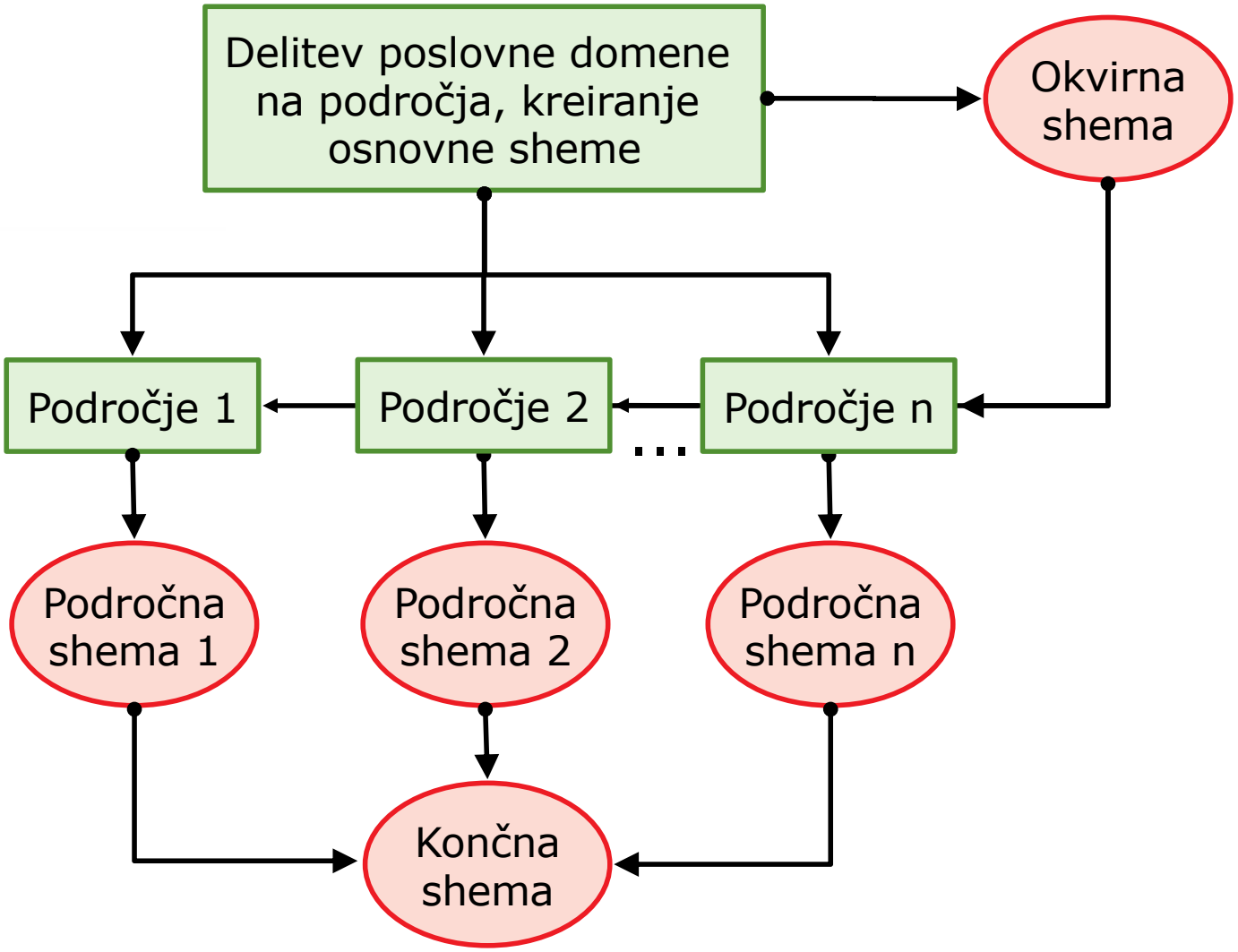
- Dva pristopa k načrtovanju PB:
  - od spodaj navzgor in
  - iz vrha navzdol.
- Pristop od spodaj navzgor:
  - začne z atributi ter jih združuje v skupine.
  - Primeren za enostavne PB (majhno število atributov).
  - Tak pristop predstavlja normalizacija.

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).

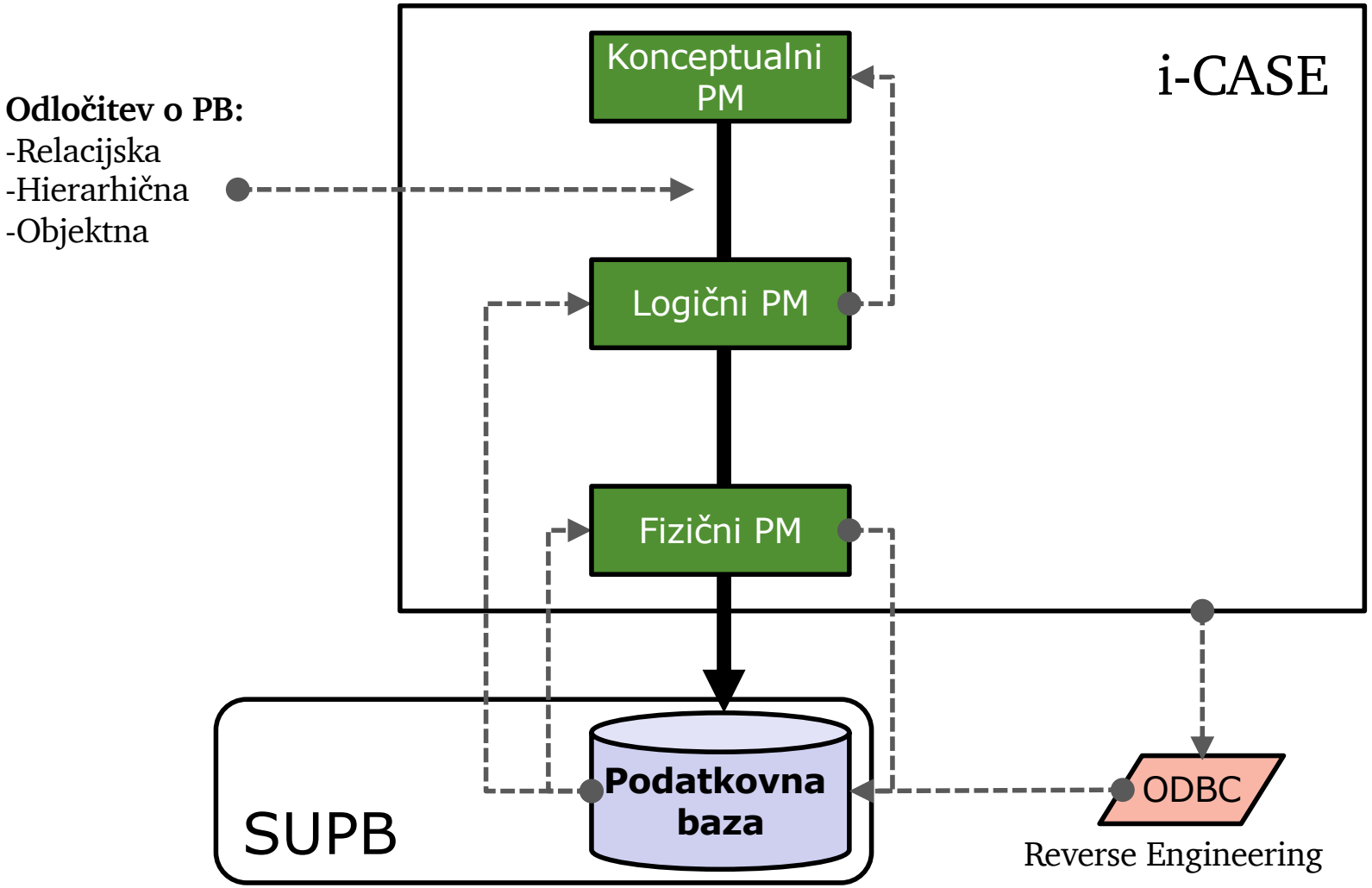
Za kompleksne domene pristop “po delih”



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...)

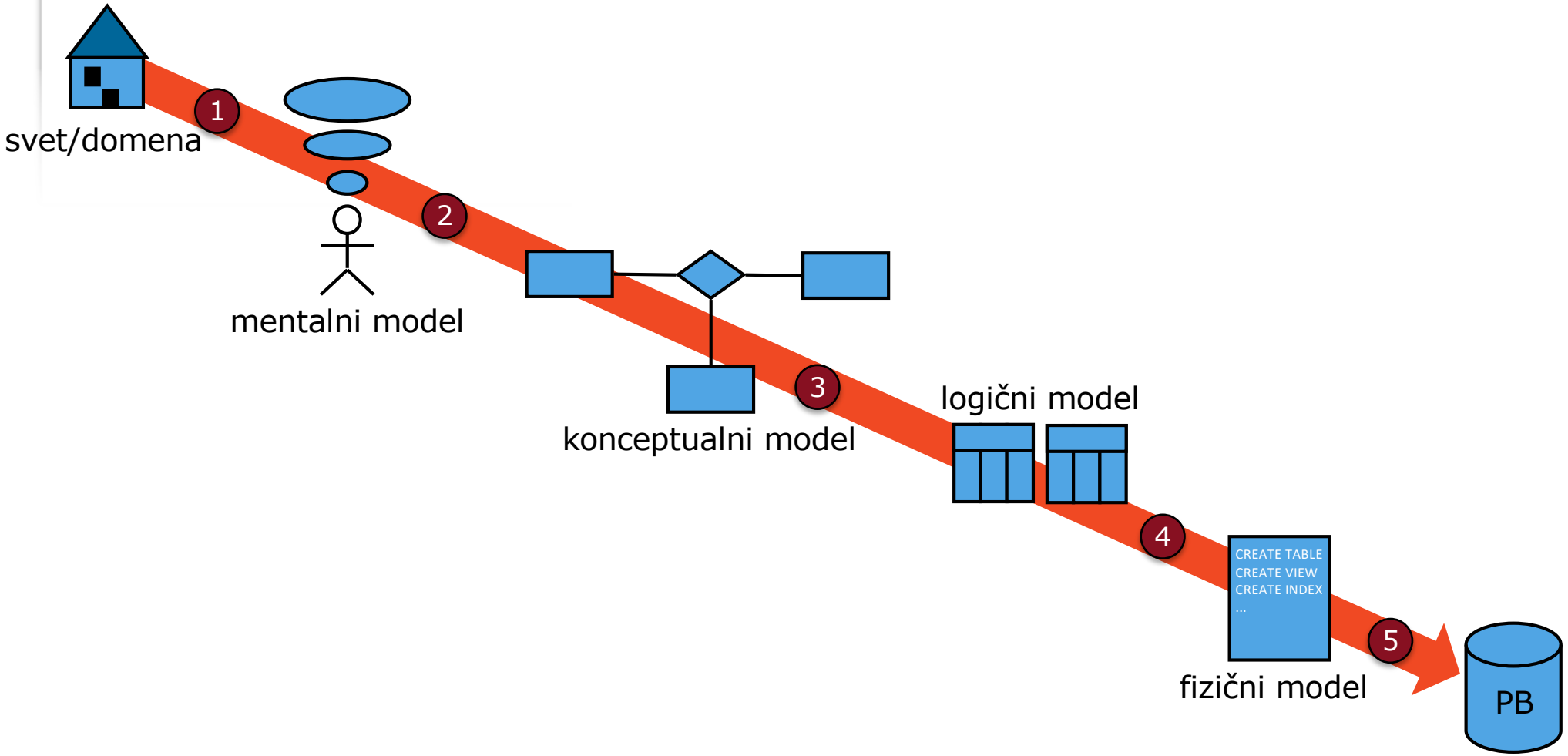
i-CASE orodja



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



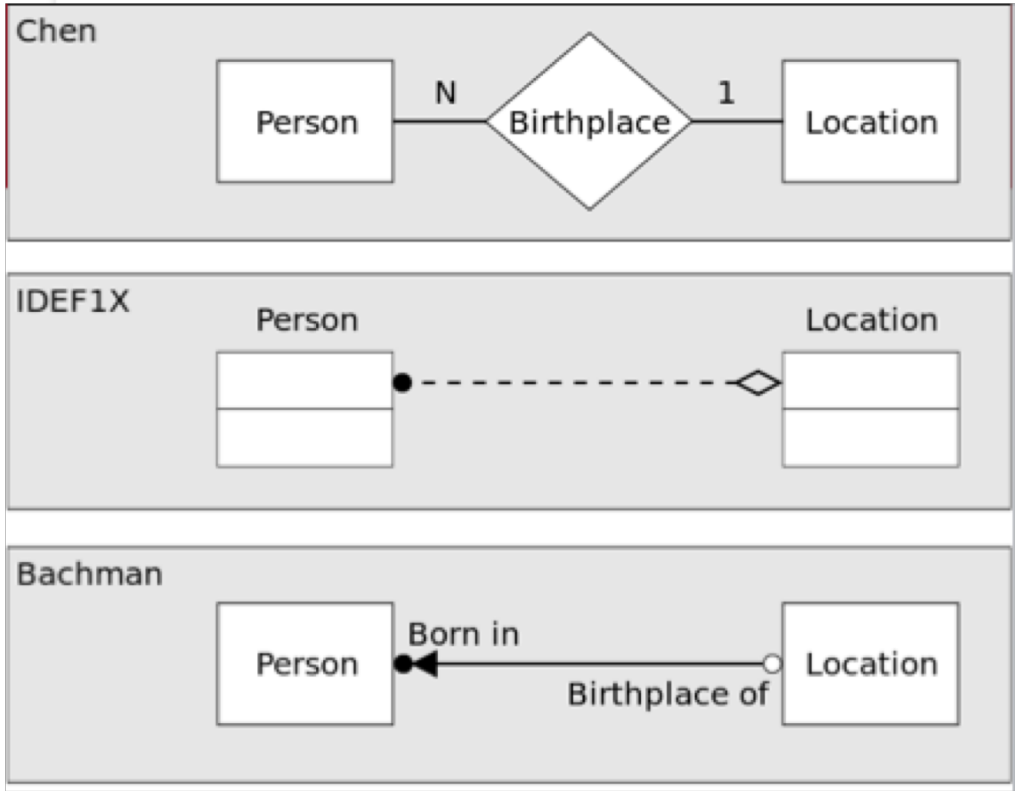
Tehnika modeliranja

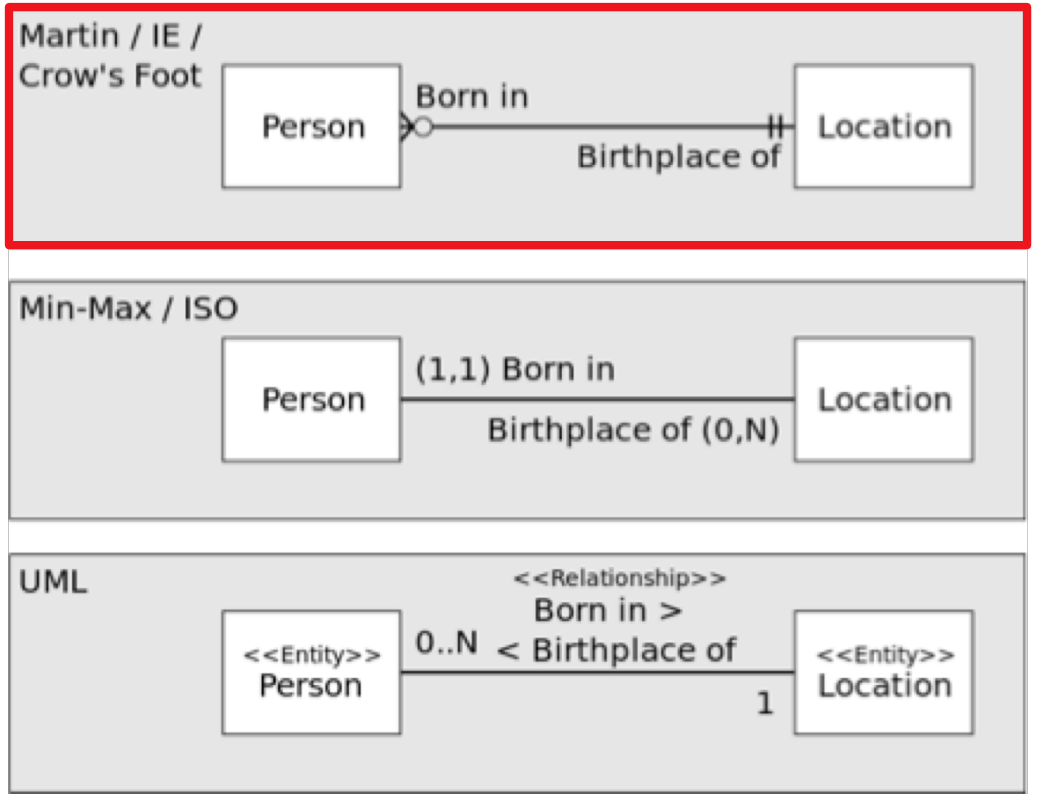
- Za izdelavo konceptualnega načrta/modela lahko uporabimo različne tehnike. Ena najbolj uporabljenih je tehnika entiteta-razmerje (ER).
- Z ER lahko nedvoumno prikažemo koncepte, ki nas zanimajo, njihove lastnosti ter povezave med njimi.

- 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…



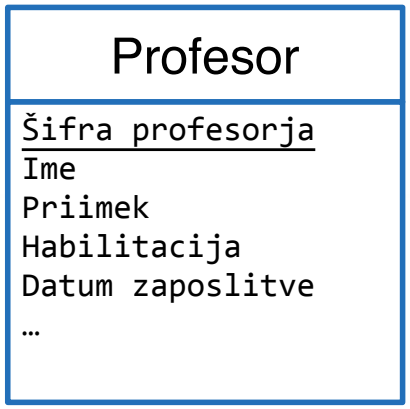


Opisovanje entitet

- 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.

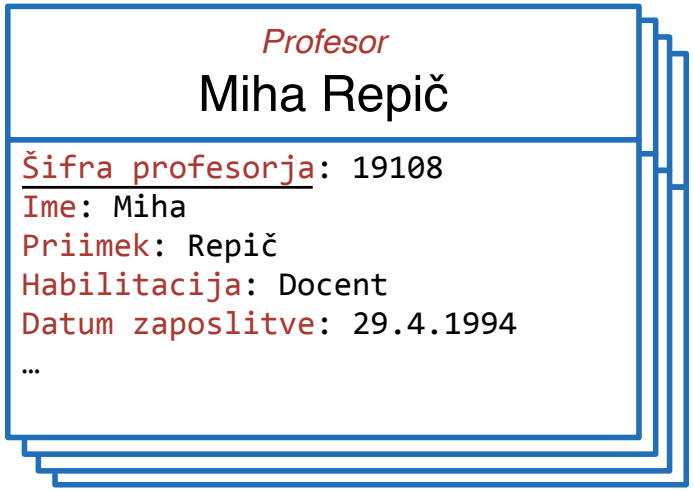
Grafična upodobitev entitetnih tipov im entitet



Naziv entitetnega tipa

Prostor za atribute

Entitete



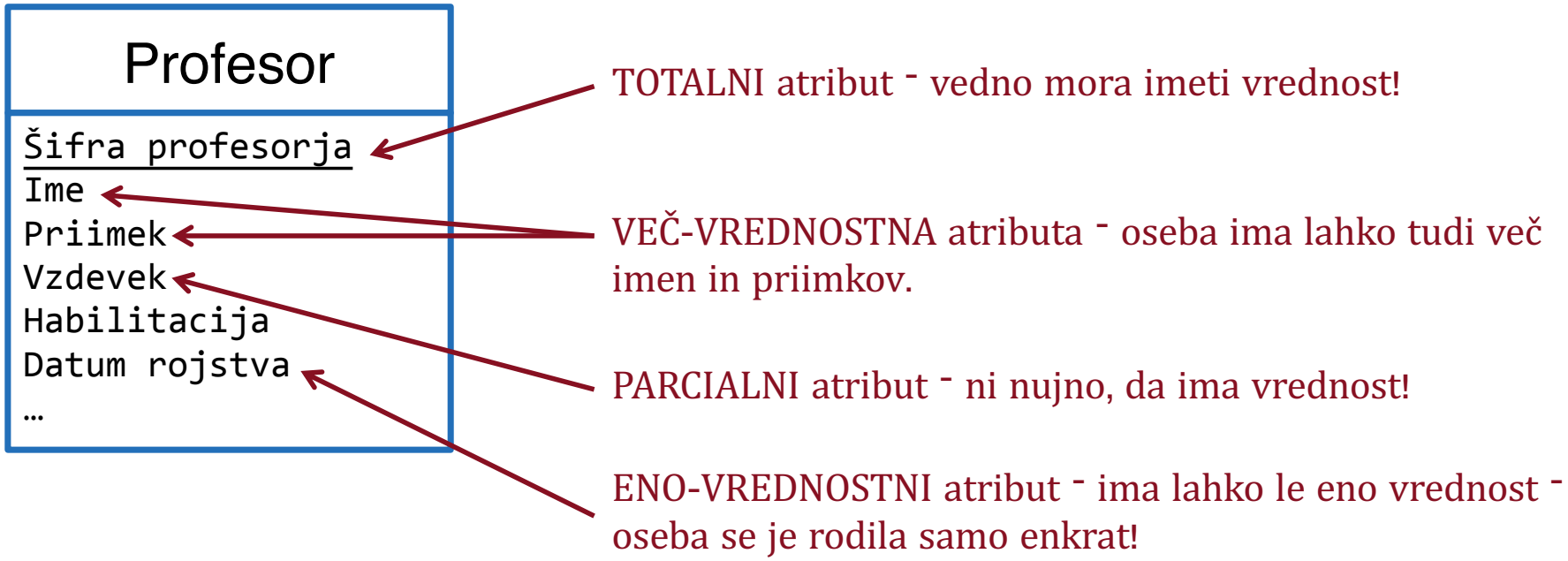
Entitetni tip



Kratka predstavitev brez atributov

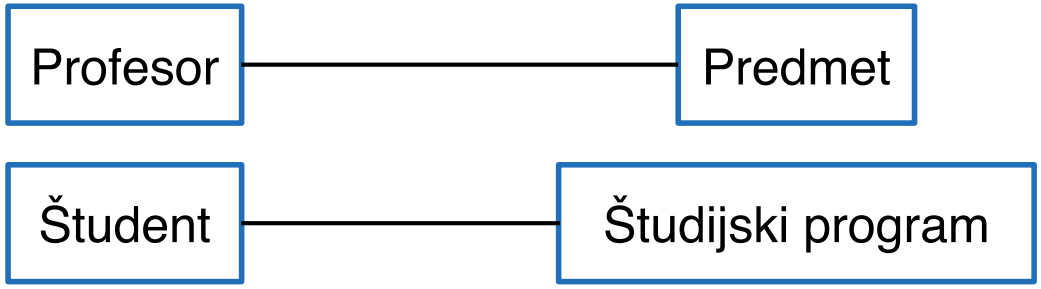
Vrste atributov

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



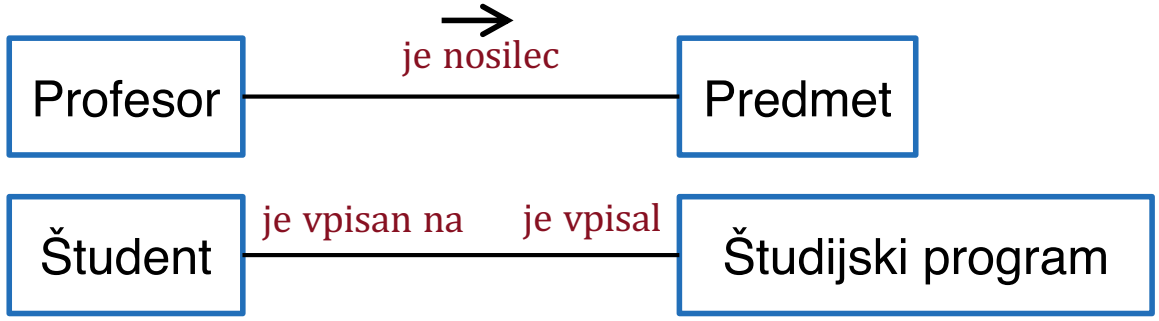
Razmerja med entitetami

- 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.



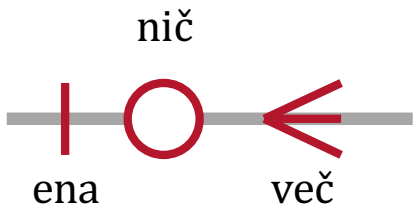
Pomen razmerja

- 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

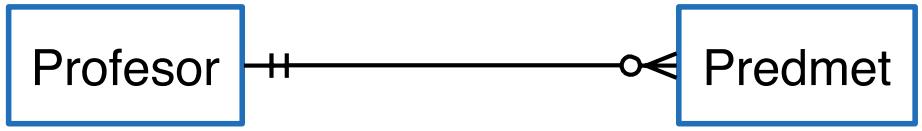


Številnost razmerja

- 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:

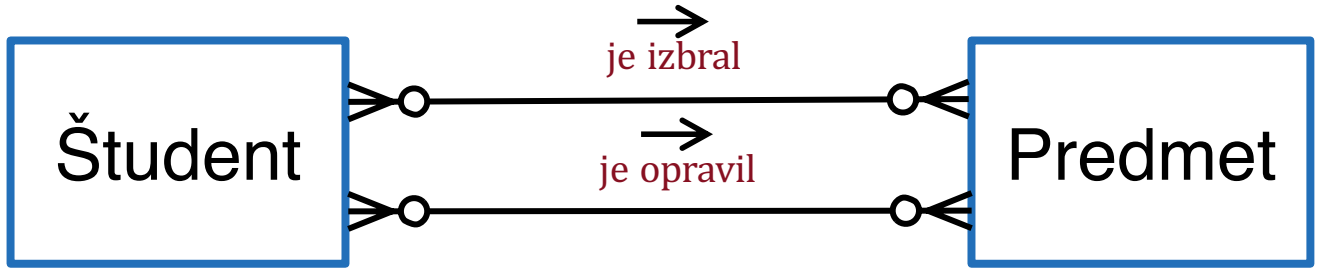


- Na vsaki strani razmerja označimo minimalno in maksimalno številnost!



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



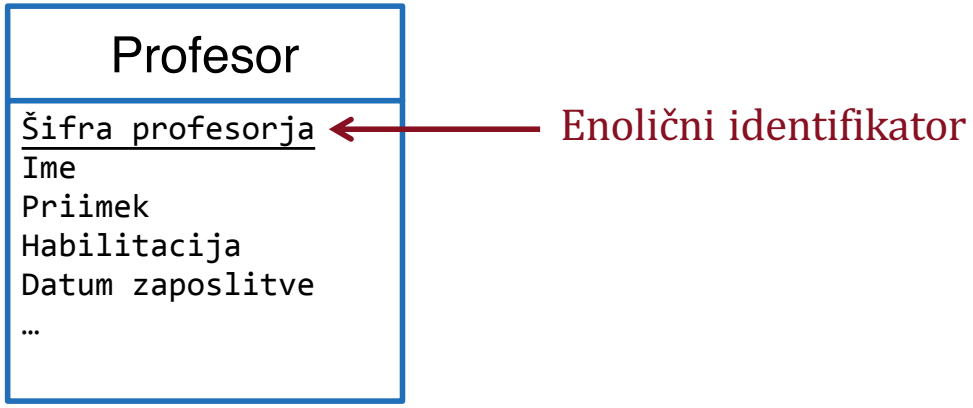
Dodatne lastnosti razmerja

- 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

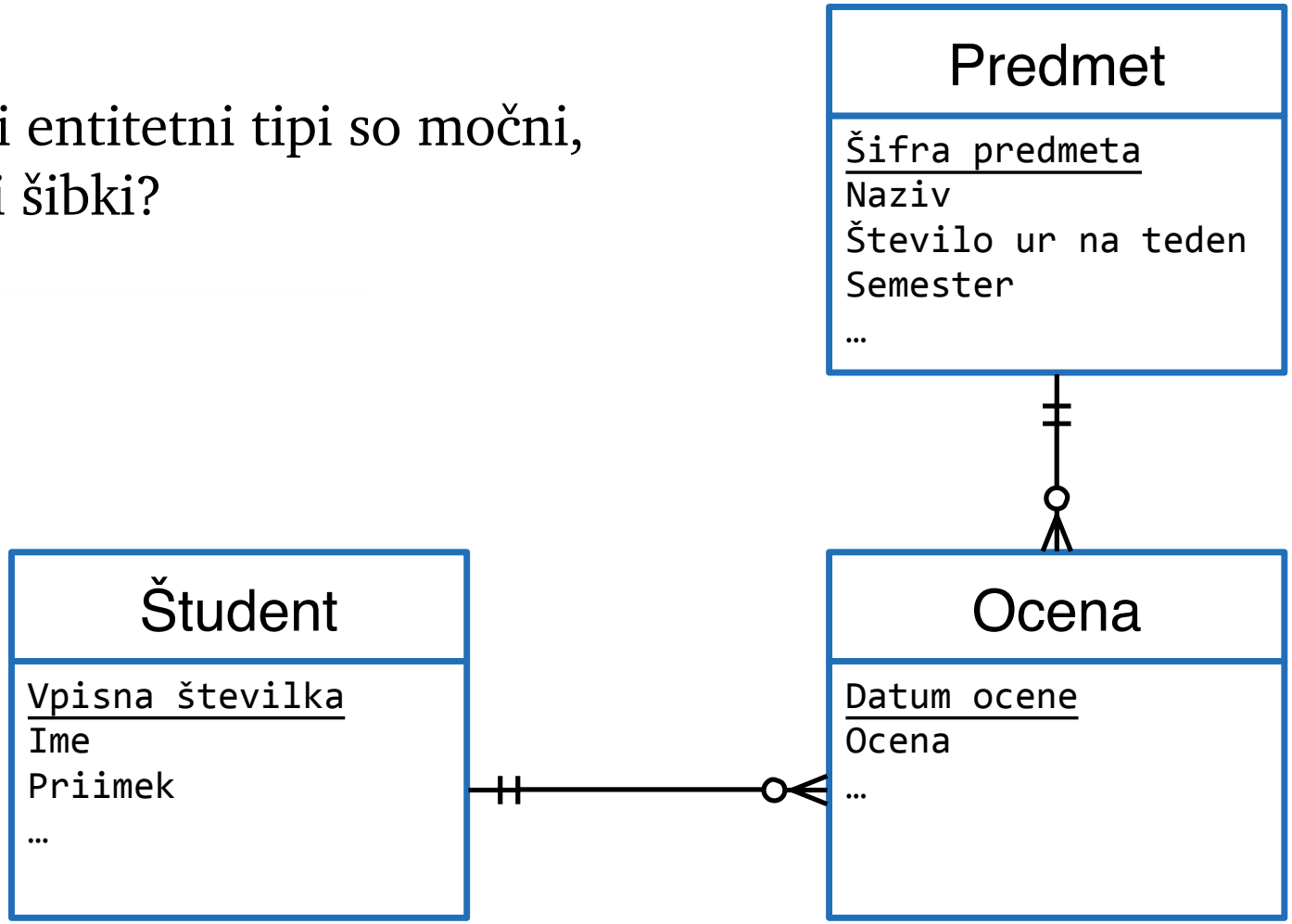
- Kadar enolični identifikator entitetnega tipa ni sestavljen le iz lastnih atributov, temveč tudi iz razmerij oz. drugih entitet v razmerju oz. njenih identifikatorjev, govorimo o šibkem entitetnem tipu.
- $\{a_1, \ldots a_k\} \cup I_{ET1} \cup .. \cup I_{ETn}$ je enolični identifikator entitete A, 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

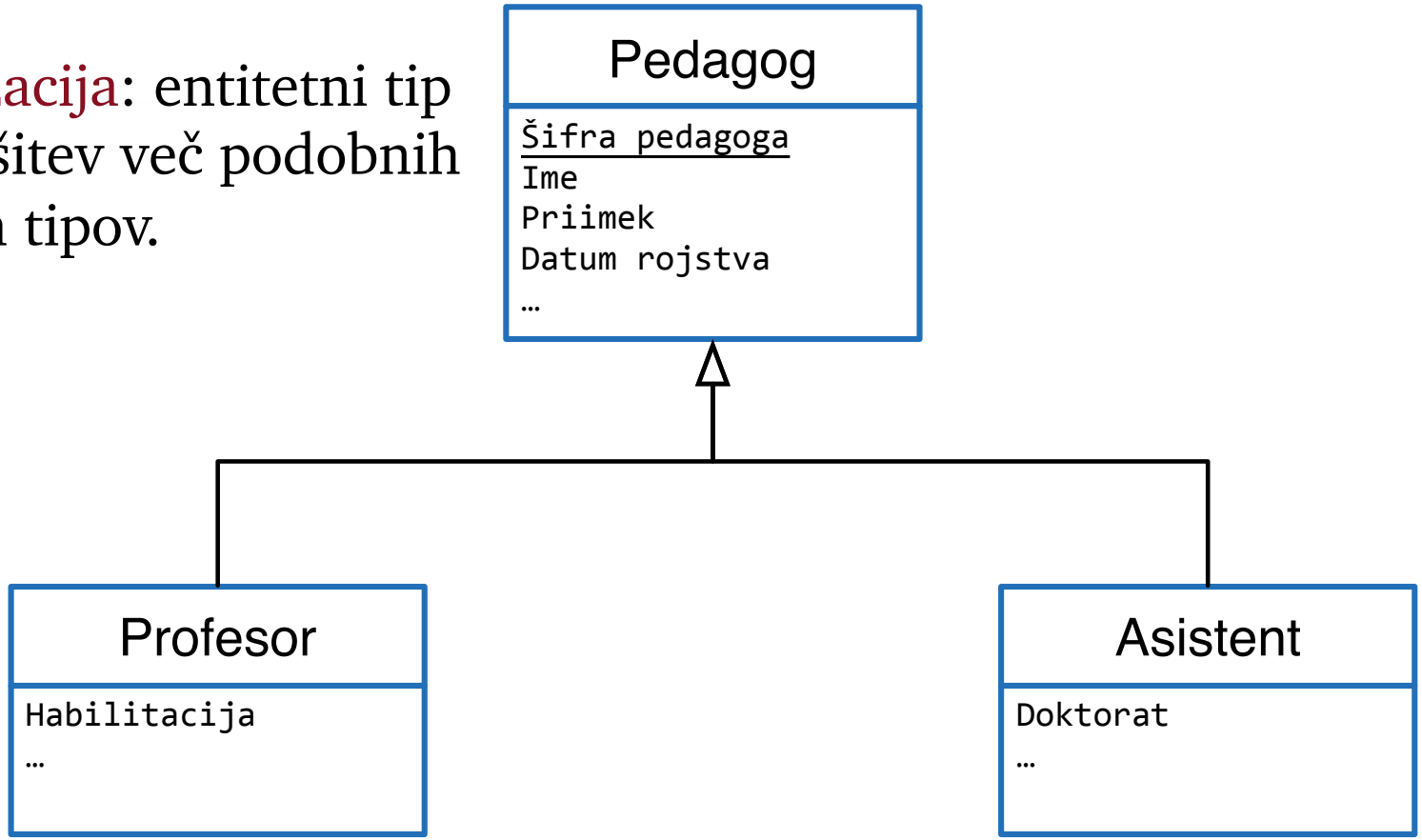
- Narišite podatkovni model urarne, za katero smo identificirali naslednje entitetne tipe:
  - ARTIKEL
  - SKUPINA ARTIKLA
  - STORITEV
  - KUPCI
  - PRODAJALCI
  - PRODAJA
  - PREVZEM
  - DOBAVITELJI
  - ...

Razširjena ER notacija

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

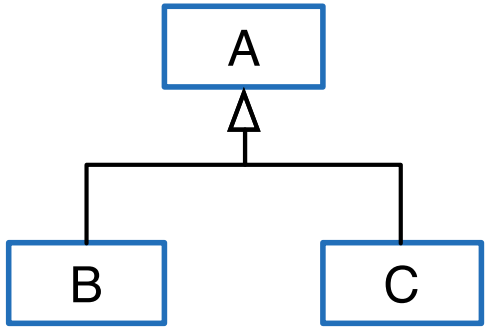
Generalizacija in specializacija

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



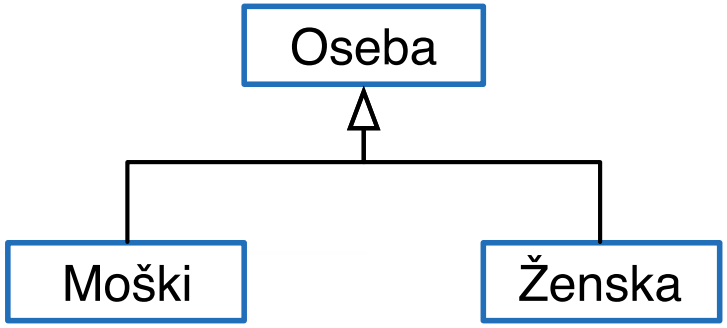
Specifike generalizacije in specializacije

- Imamo entitetni tip A s podtipoma B in C.

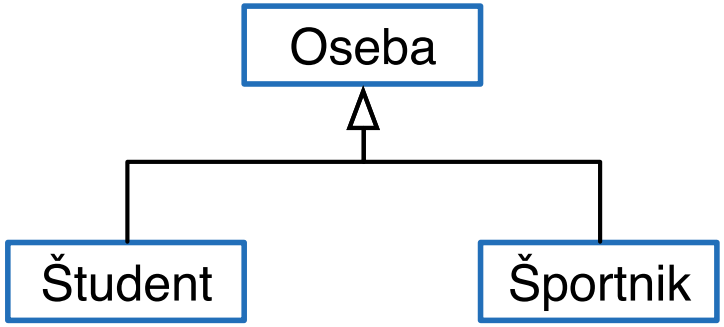


- Možne so naslednje povezave:
  - B in C pokrivata A totalno in ekskluzivno, če velja: $EB \cup EC = EA$ in $EB \cap EC = \{\}$
  - B in C pokrivata A totalno in prekrivno, če velja: $EB \cup EC = EA$ in $EB \cap EC \neq \{\}$
  - B in C pokrivata A delno in ekskluzivno, če velja: $EB \cup EC \subset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



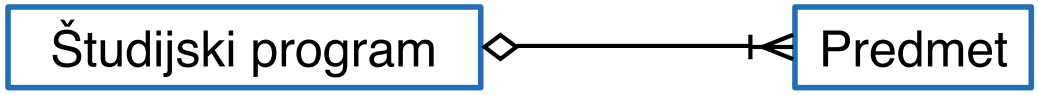
Totalno in ekskluzivno razmerje



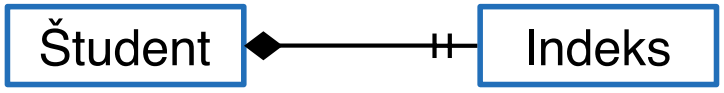
Delno in prekrivno razmerje

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

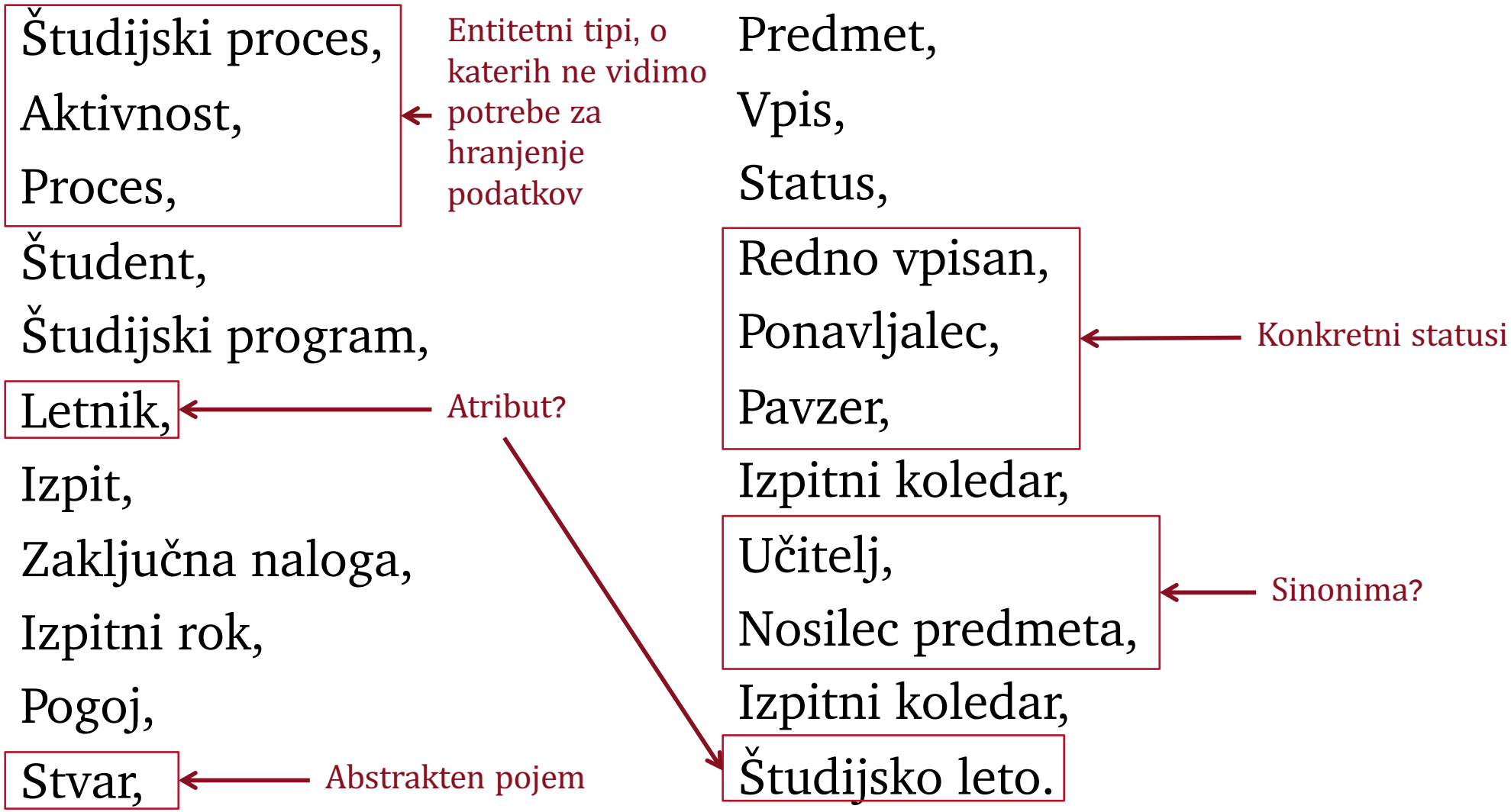
K1.1 – Identificiraj entitetne tipe...

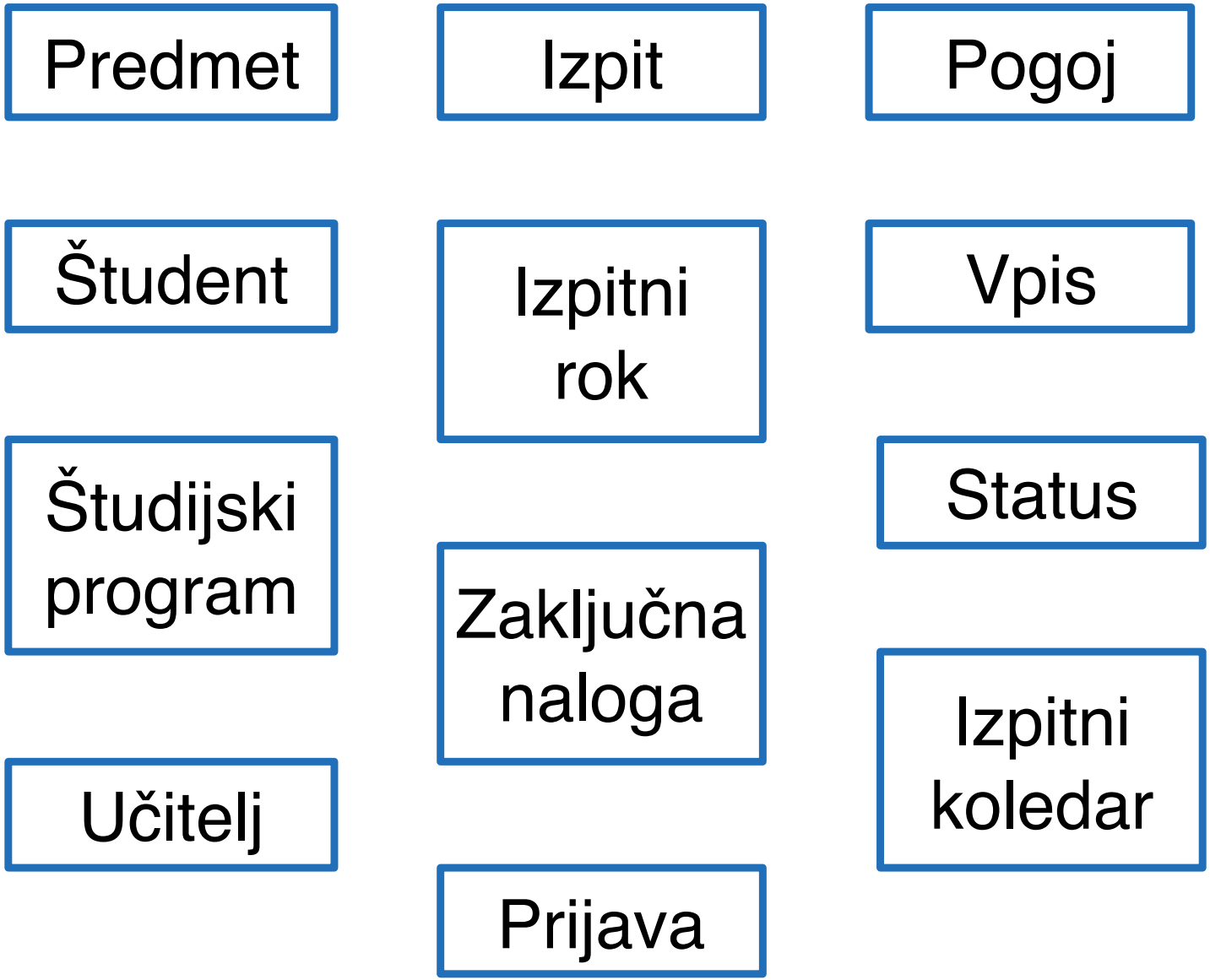
- Na voljo različne tehnike
- Ena izmed tehnik je pregled uporabniških zahtev:
  - Pregledamo vse omenjene samostalnike in fraze (npr. *profesor*, *predmet*, *izpit*, *rok*, *datum izpita*,...)
  - Pozorni smo na pomembne objekte (npr. *ljudje*, *lokacije*...)
  - Skušamo ločiti objekte (npr. *profesor*, *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

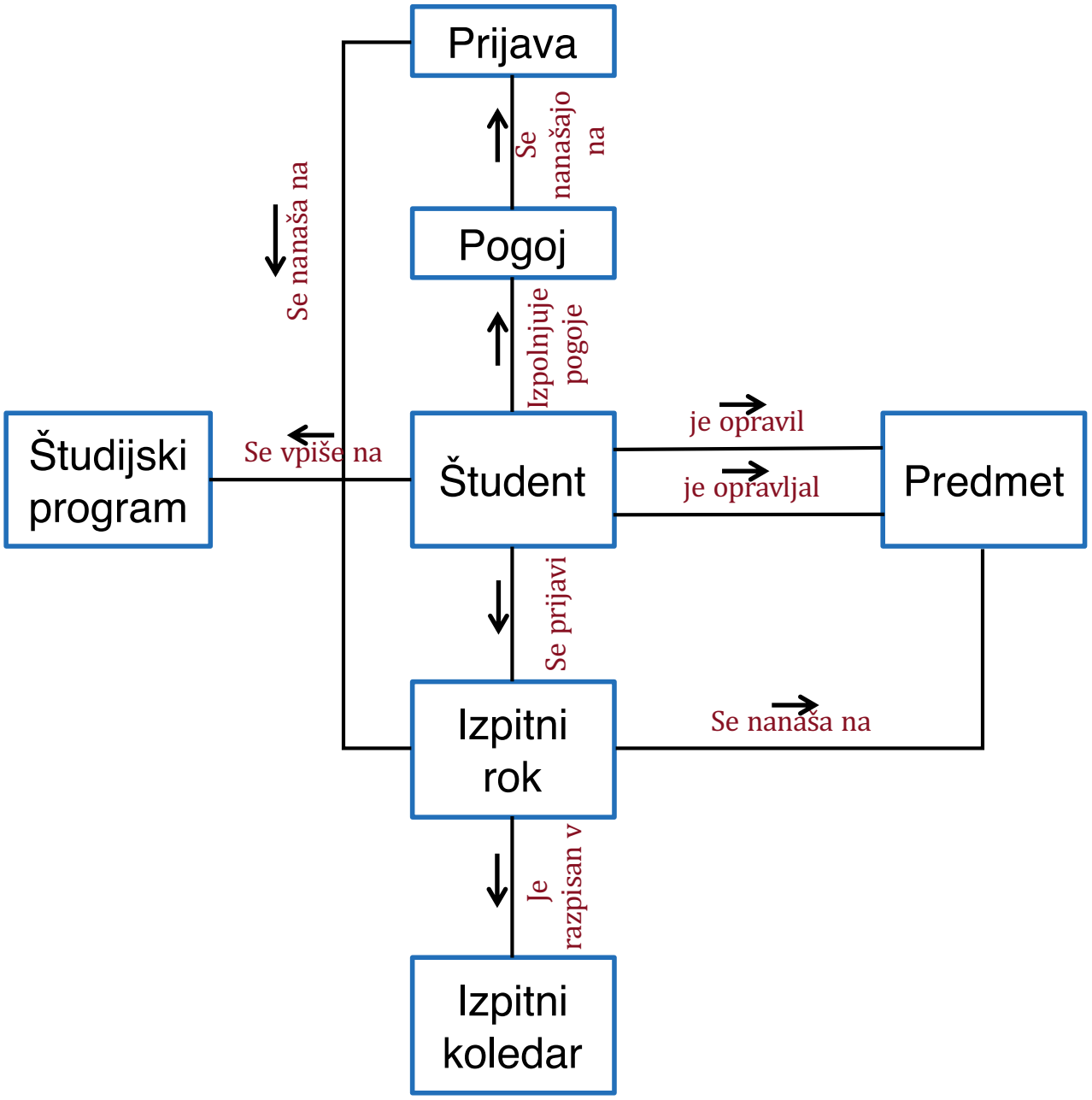
Študent *izpolnjuje* pogoje za prijavo

Študent *je opravljal* predmet

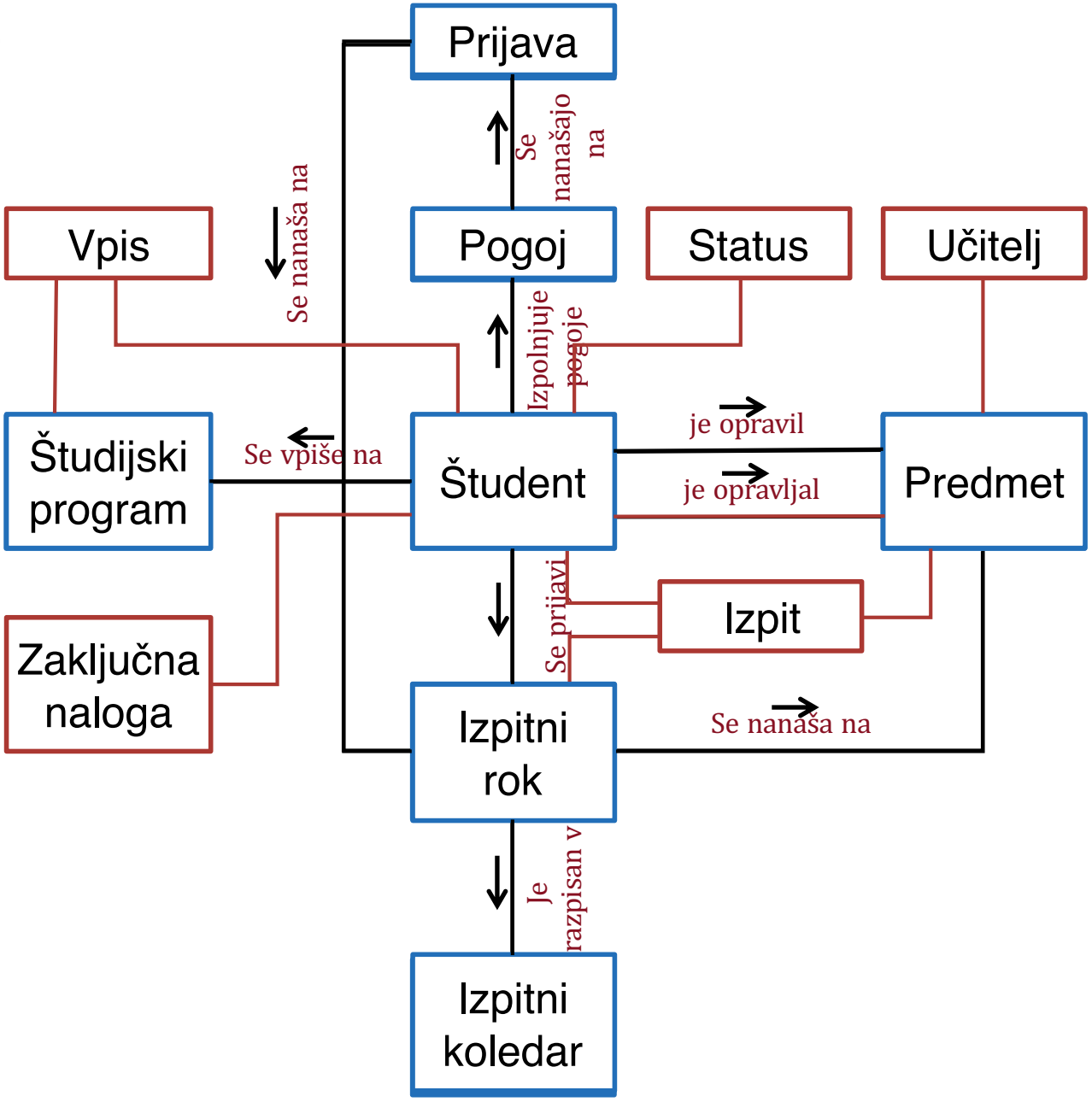
Študent *je opravil* predmet

Izpitni rok *je razpisan* z izpitnim koledarjem

...

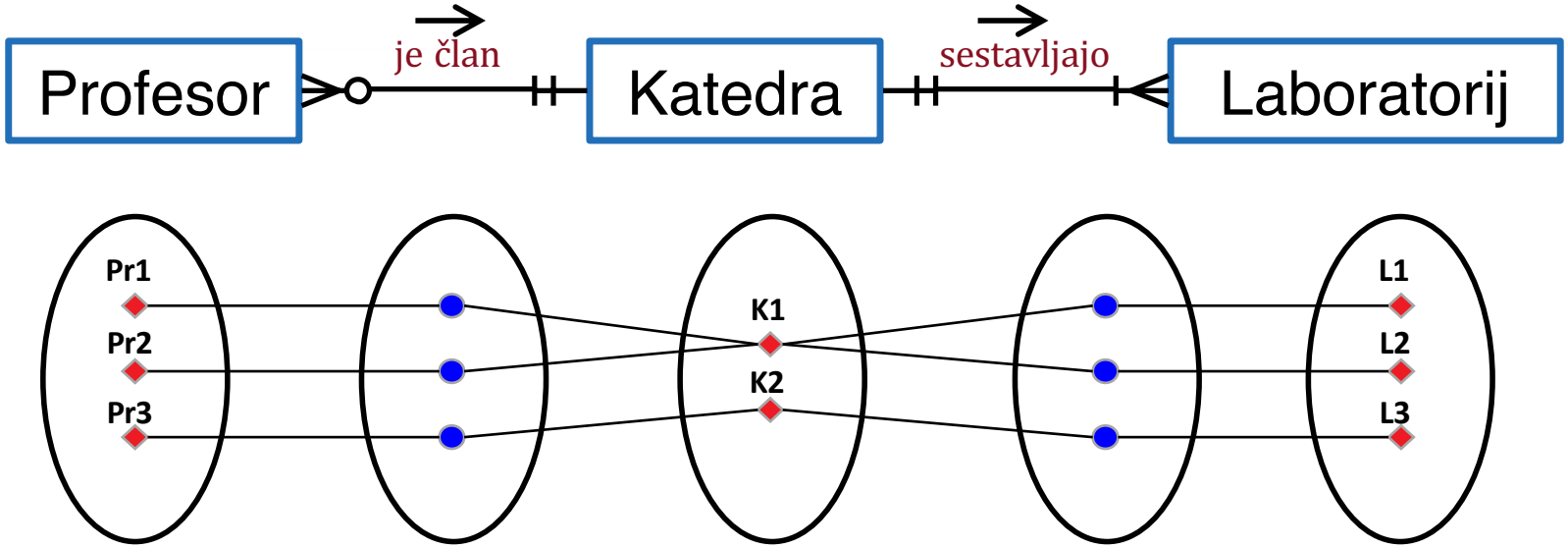

Prijava
Se nanašajo na
Pogoj
Se nanaša na
Izpolnjuje pogoje
Študijski program
Se vpiše na
Študent
je opravil
je opravljal
Predmet
Se prijavi
Izpitni rok
Se nanaša na
Je razpisan v
Izpitni koledar

Kakšne so številnosti povezav?



K1.2 – Identificiraj povezave...

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

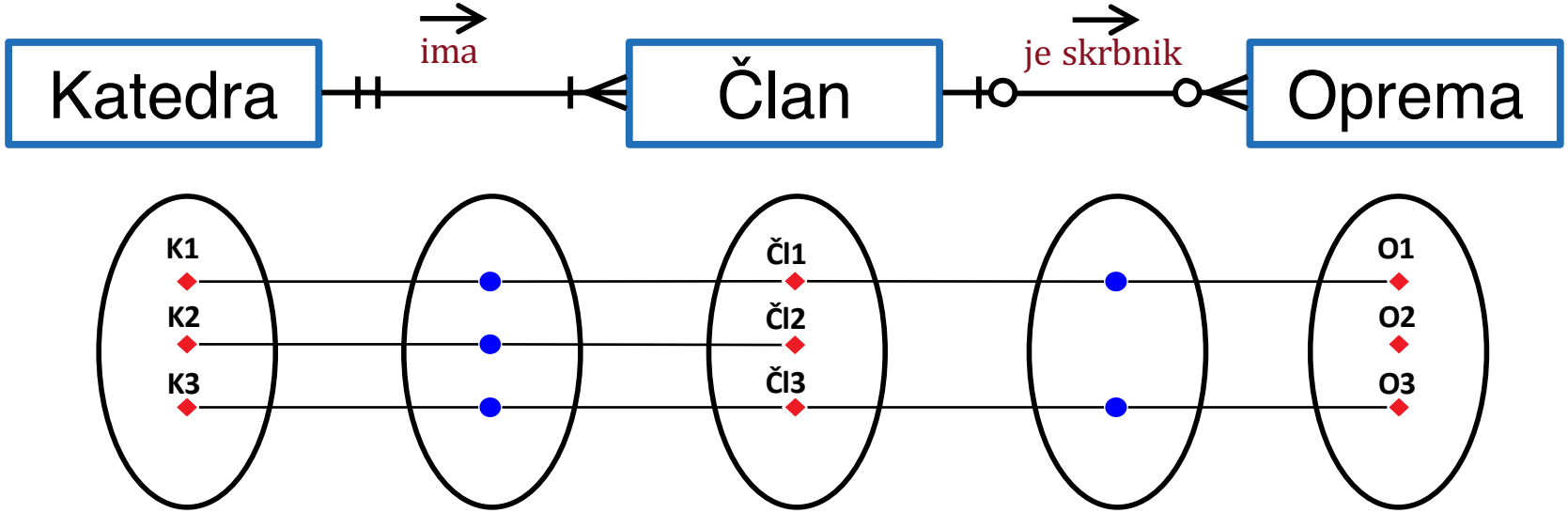


V katerem laboratoriju dela profesor `Pr1`?

K1.2 – Identificiraj povezave...

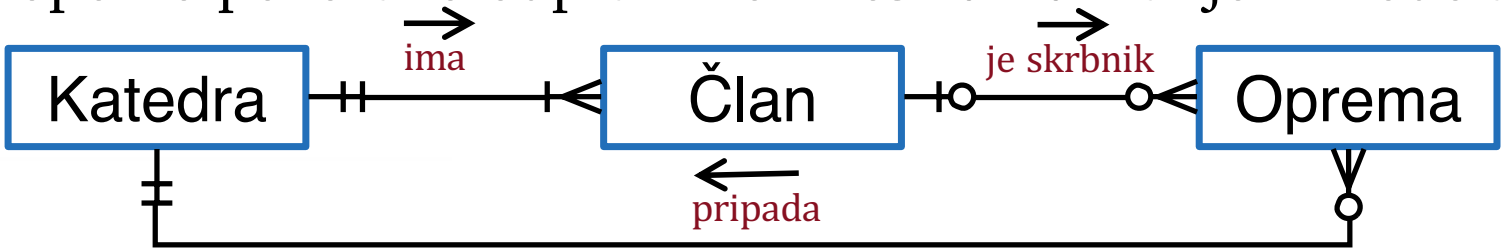
- Primer nepopolne povezave.

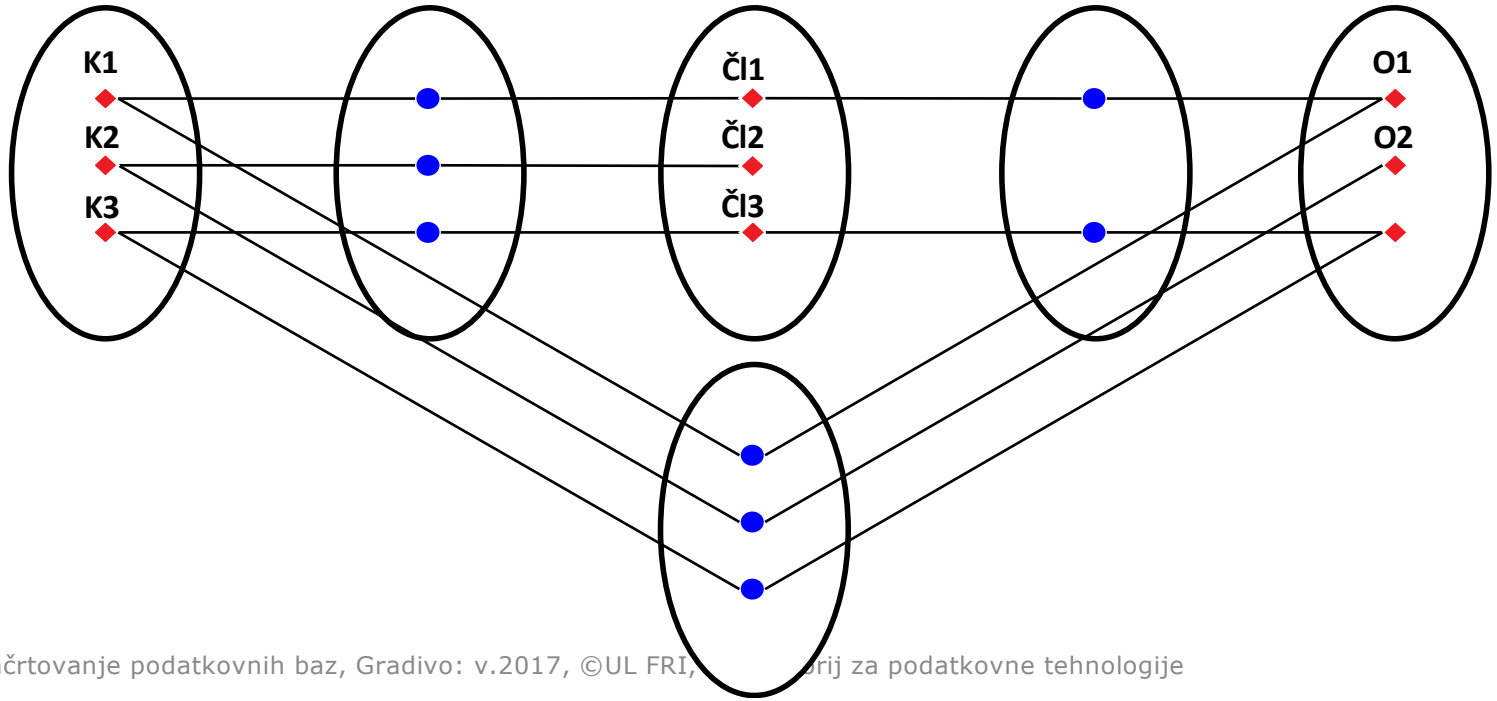
  Kateri katedri pripada oprema o2?



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

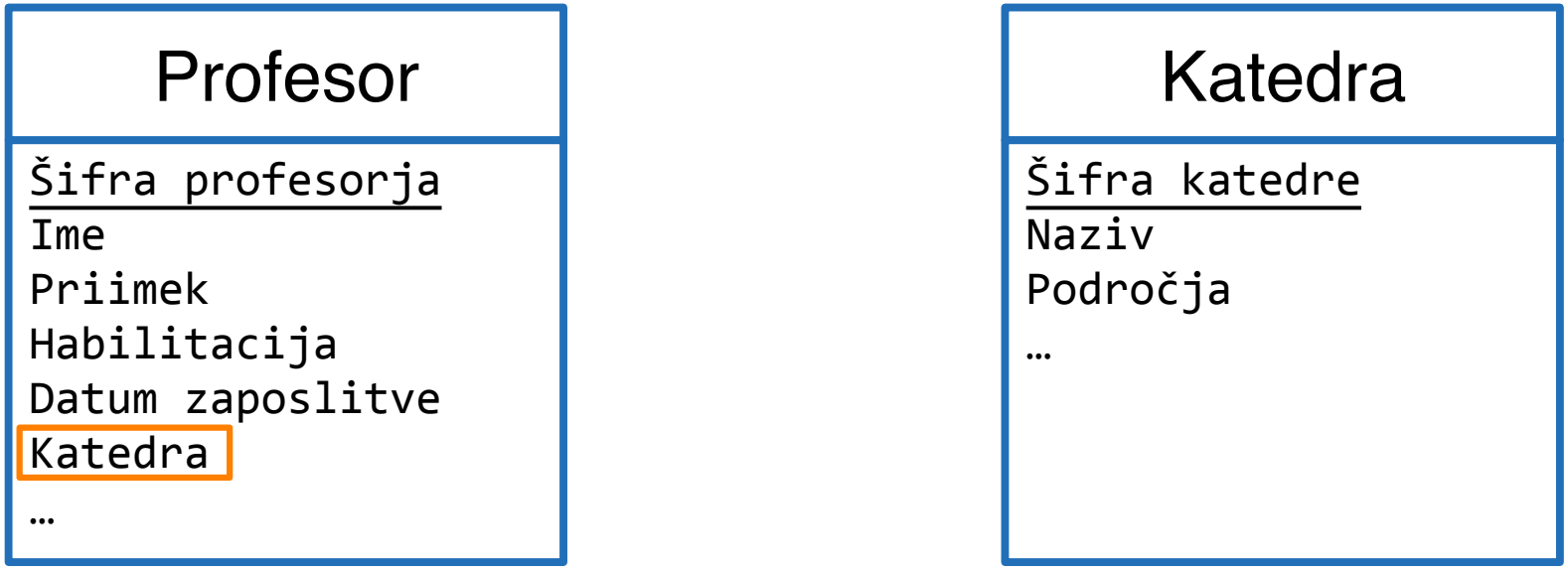
… o študentih in zaposlenih hranimo tudi kontaktne podatke, kot so elektronska pošta, telefonska številka ipd. …

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...

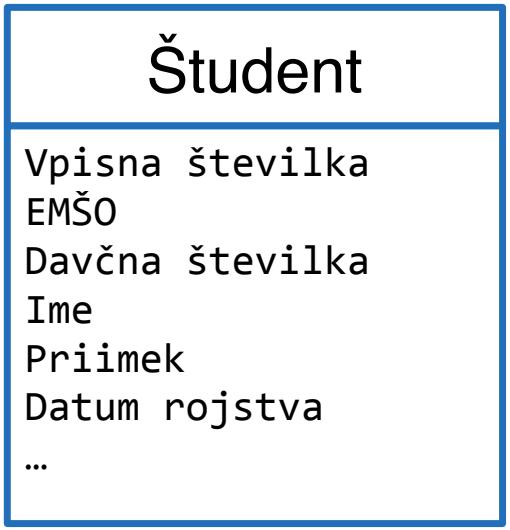
- Domena je množica vrednosti, ki jih lahko zavzamejo atributi, vključeni v to domeno.
- Domeni lahko določimo:
  - Seznam dovoljenih vrednosti
  - Minimalno in maksimalno vrednost
  - Podatkovni tip in dolžino
  - Dovoljene operacije nad atributom (še v raziskavi)
- Primeri domen:
  - `Barva → {bela, 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...

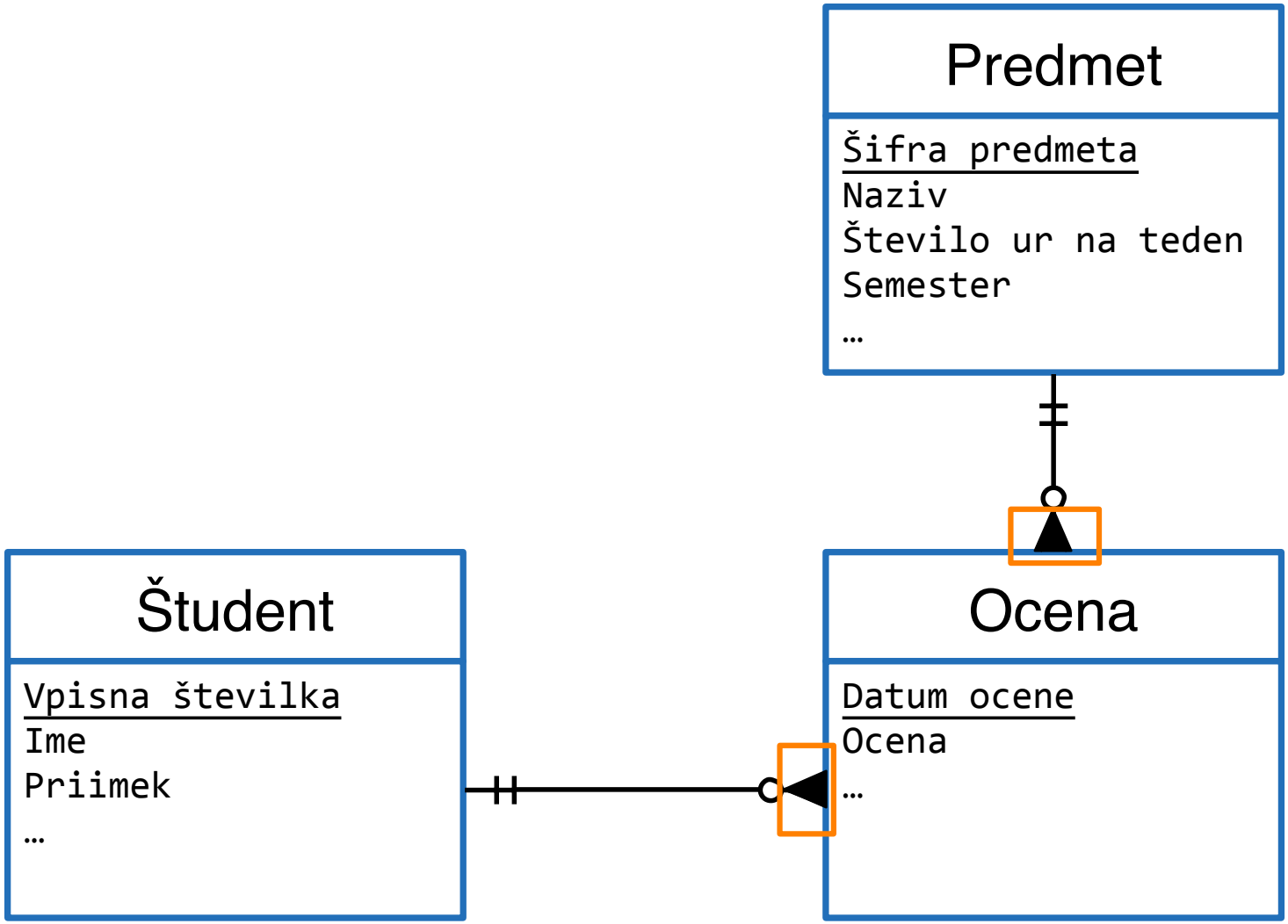
- Za vsak entitetni tip določimo kandidate za ključ ter izberemo enega za primarni ključ.
- Kandidati za ključ so minimalne podmnožice atributov, ki enolično identificirajo vsako entiteto.
- Če je kandidatov več, izberemo enega, ki je primeren za primarni ključ.



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
…

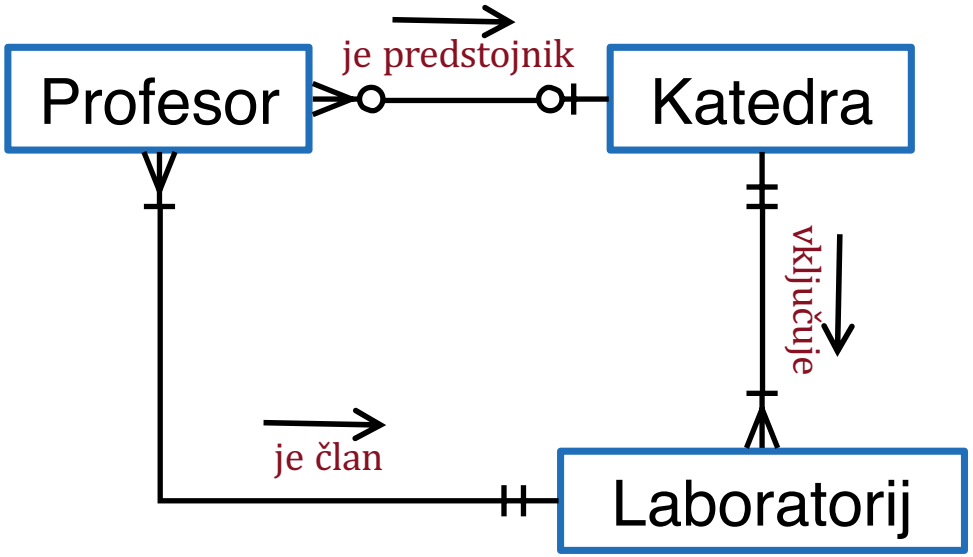
Delavec
Šifra delavca
Ime
Priimek
Vzdevek
Habilitacija
Datum rojstva
…

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)

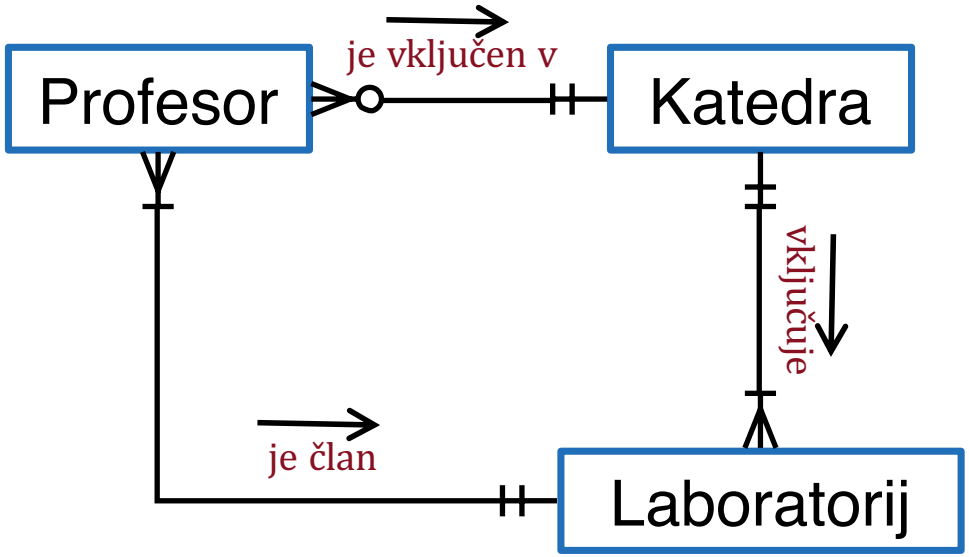
K1.7 - Preveri obstoj odvečnih elem....

- 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)

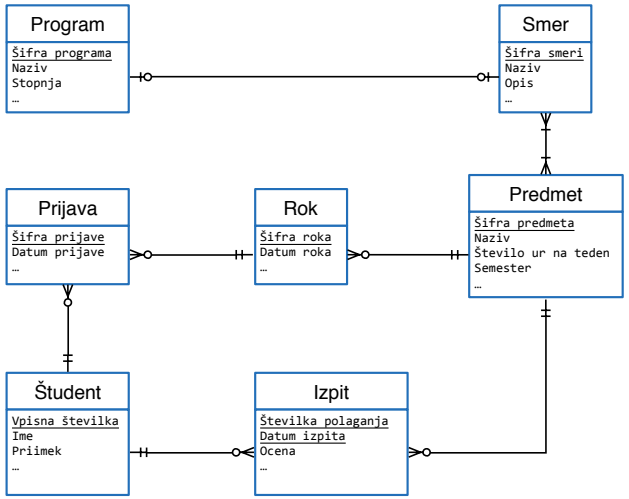
- Preverjanje opisa transakcij
  - Vsako transakcijo opišemo;
  - Preverimo, če model zajema vse entitetne tipe, povezave in atribute, ki jih transakcija potrebuje.

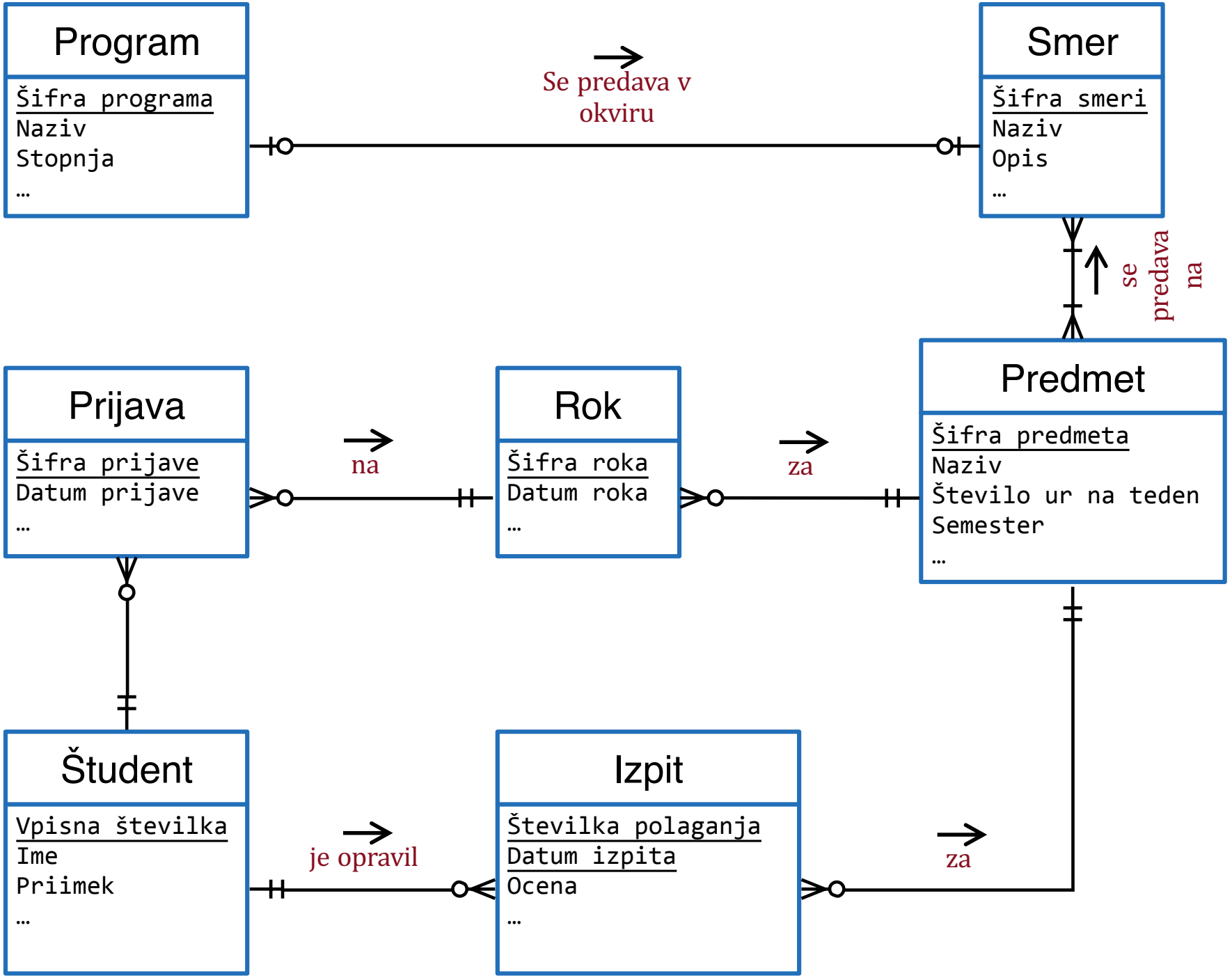
K1.8 - Preveri če model zdrži transkacije...

- 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

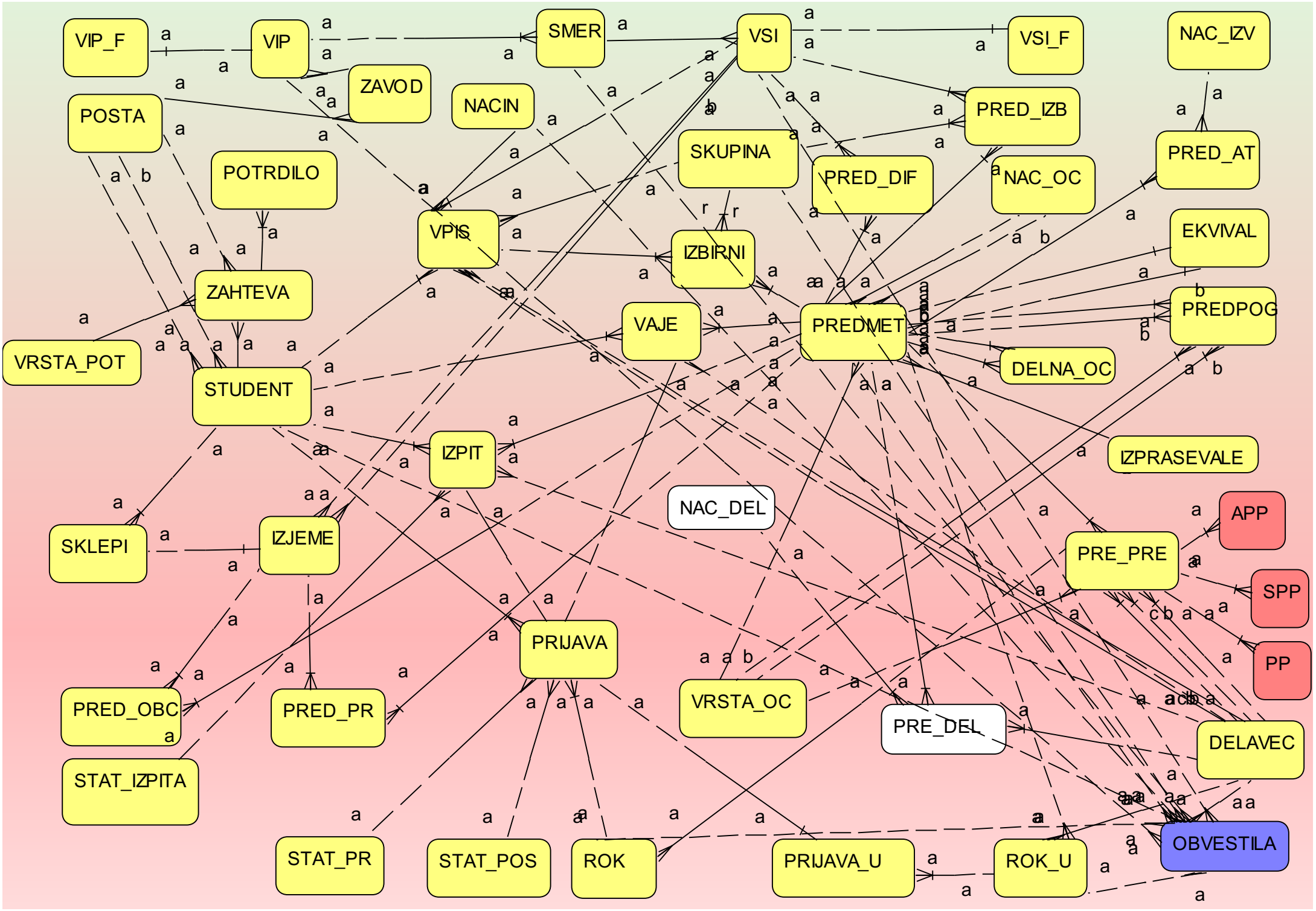
- Preveri naslednji dve transakciji:
  - T1: Izpiši vse predmete, ki jih je opravil določen študent
  - T2: Izpiši vse študente, ki so se vpisali v določen letnik, določene smeri, določenega programa




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

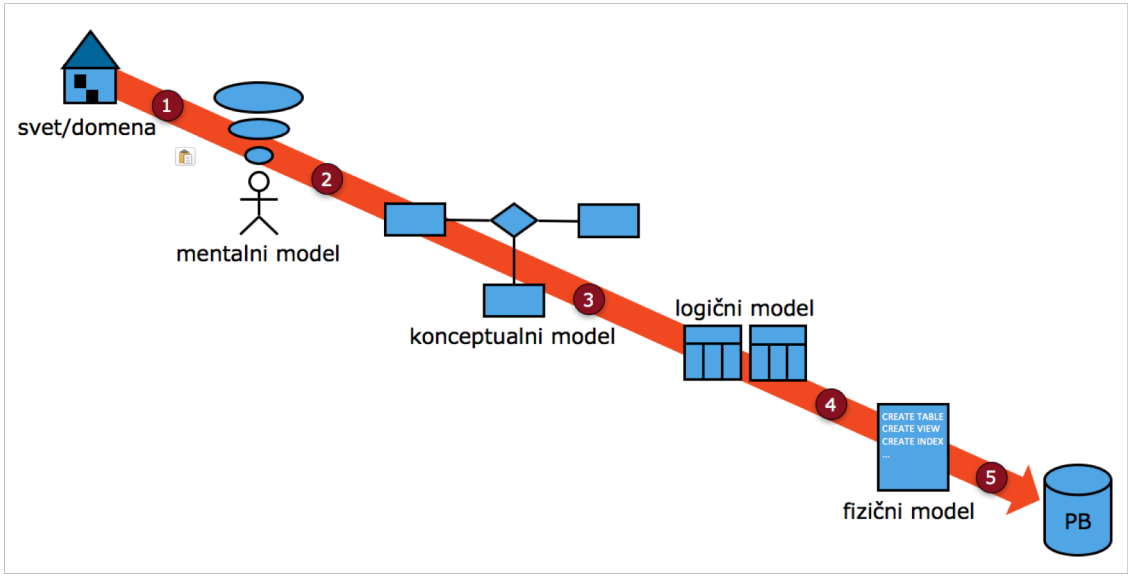
K1.9 – Preveri model z uporabnikom

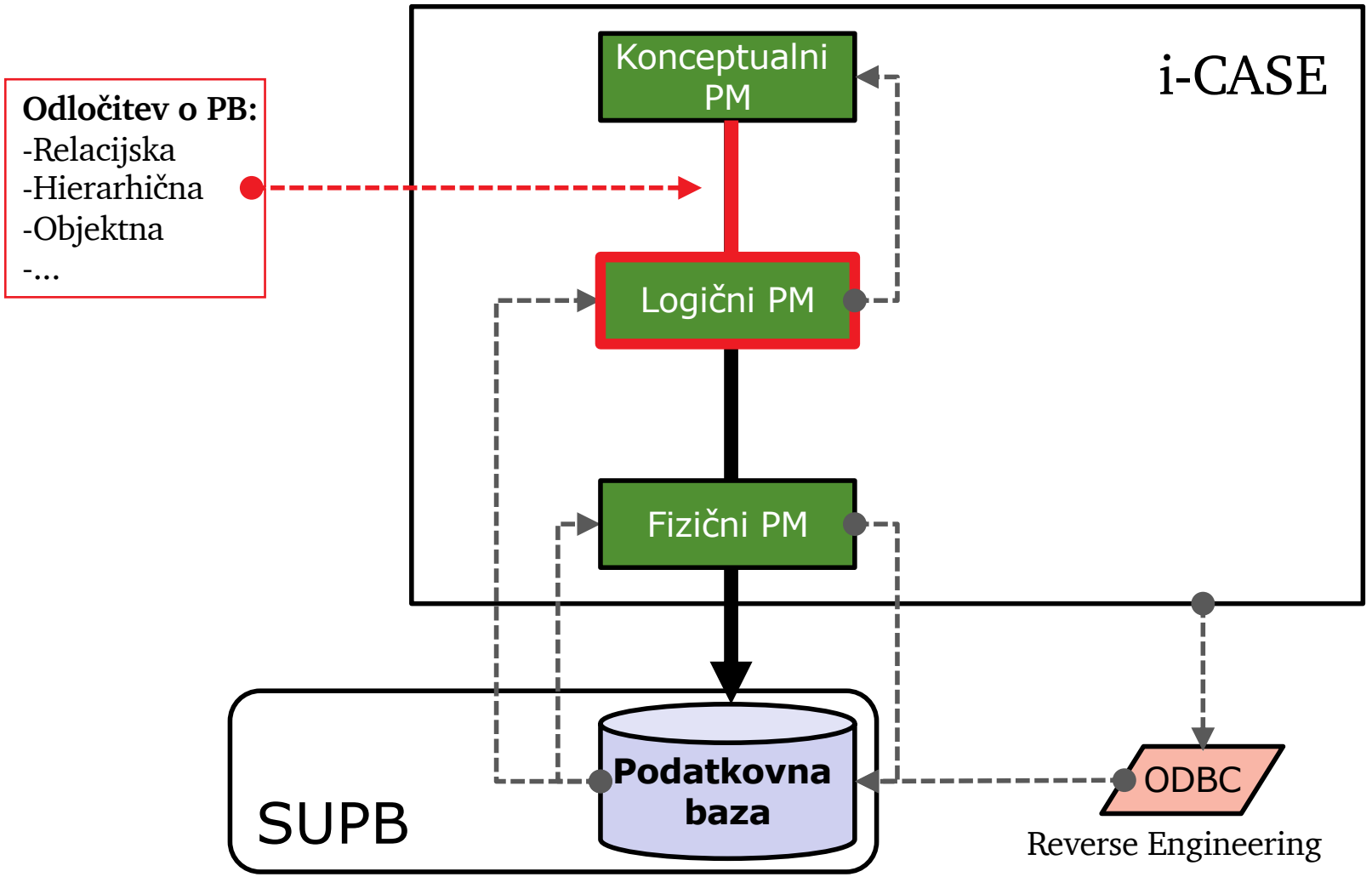
- Na koncu model preverimo z uporabnikom
- Anomalije, pomanjkljivosti, napake,... lahko vodijo v ponovitev korakov od K1 do K9.
- V mnogih podjetjih mora uporabnik podpisati podatkovni model


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
OBVESTILA
STAT_PR
STAT_POS
ROK
PRIJAVA_U
ROK_U

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.

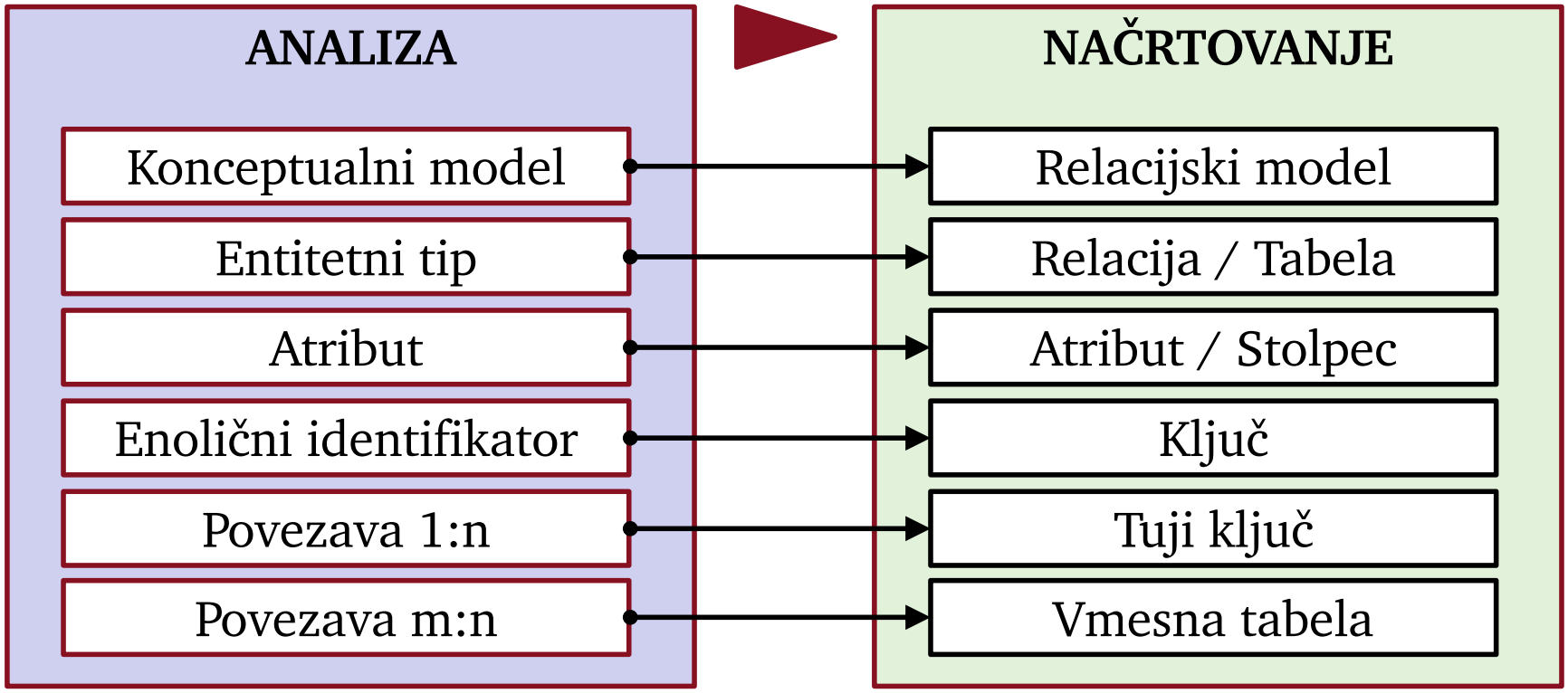



Odločitev o PB:
-Relacijska
-Hierarhična
-Objektna
-...
Konceptualni PM
Logični PM
Fizični PM
i-CASE
Podatkovna baza
SUPB
ODBC
Reverse Engineering

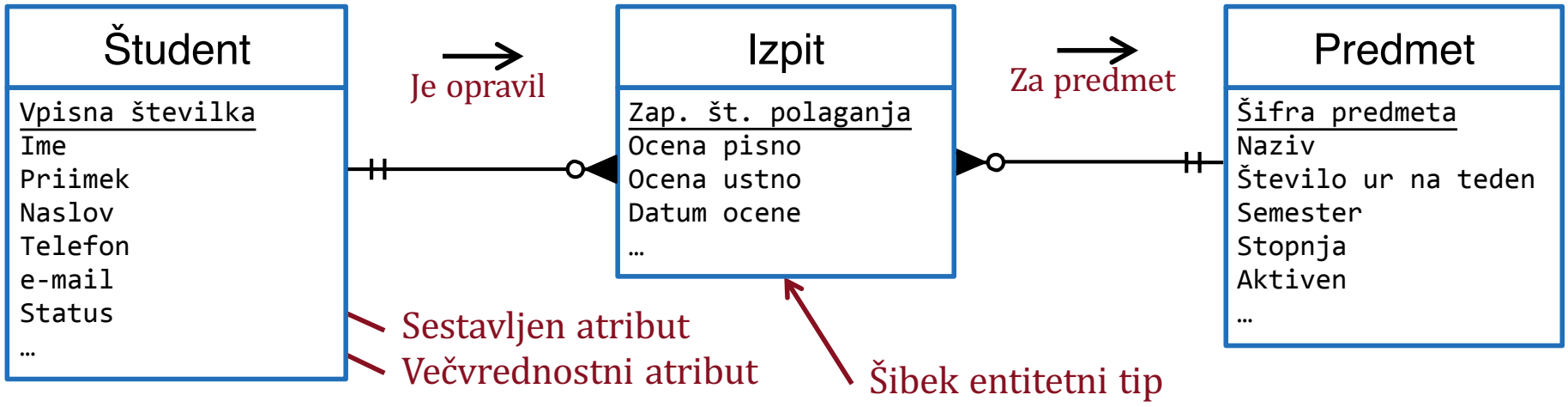
Prehod iz konceptualnega v logični model

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

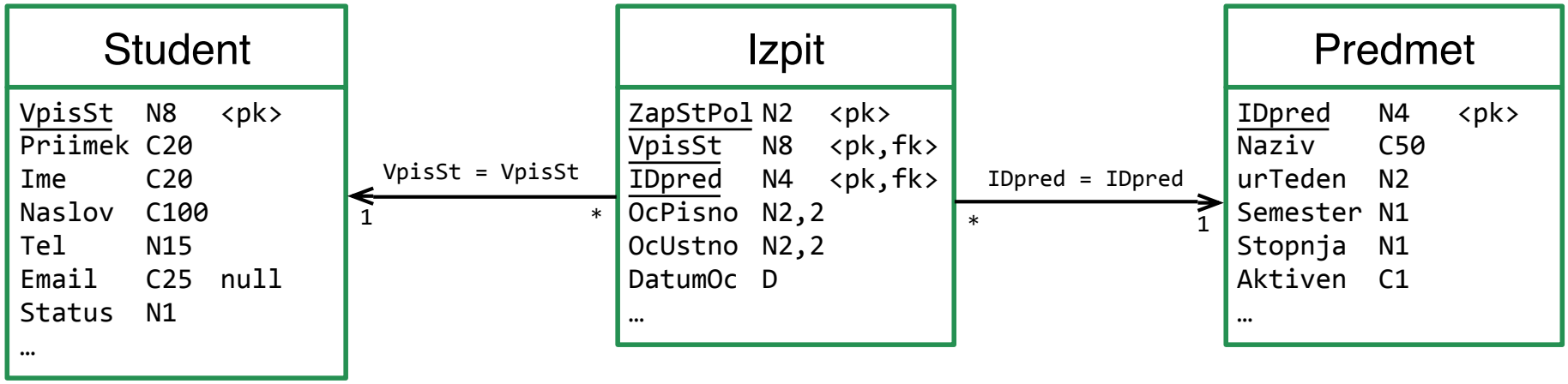
vrsta baze: relacijska
SUPB: Oracle 12c



Konceptualni model



Logični model



Logični načrtovanje

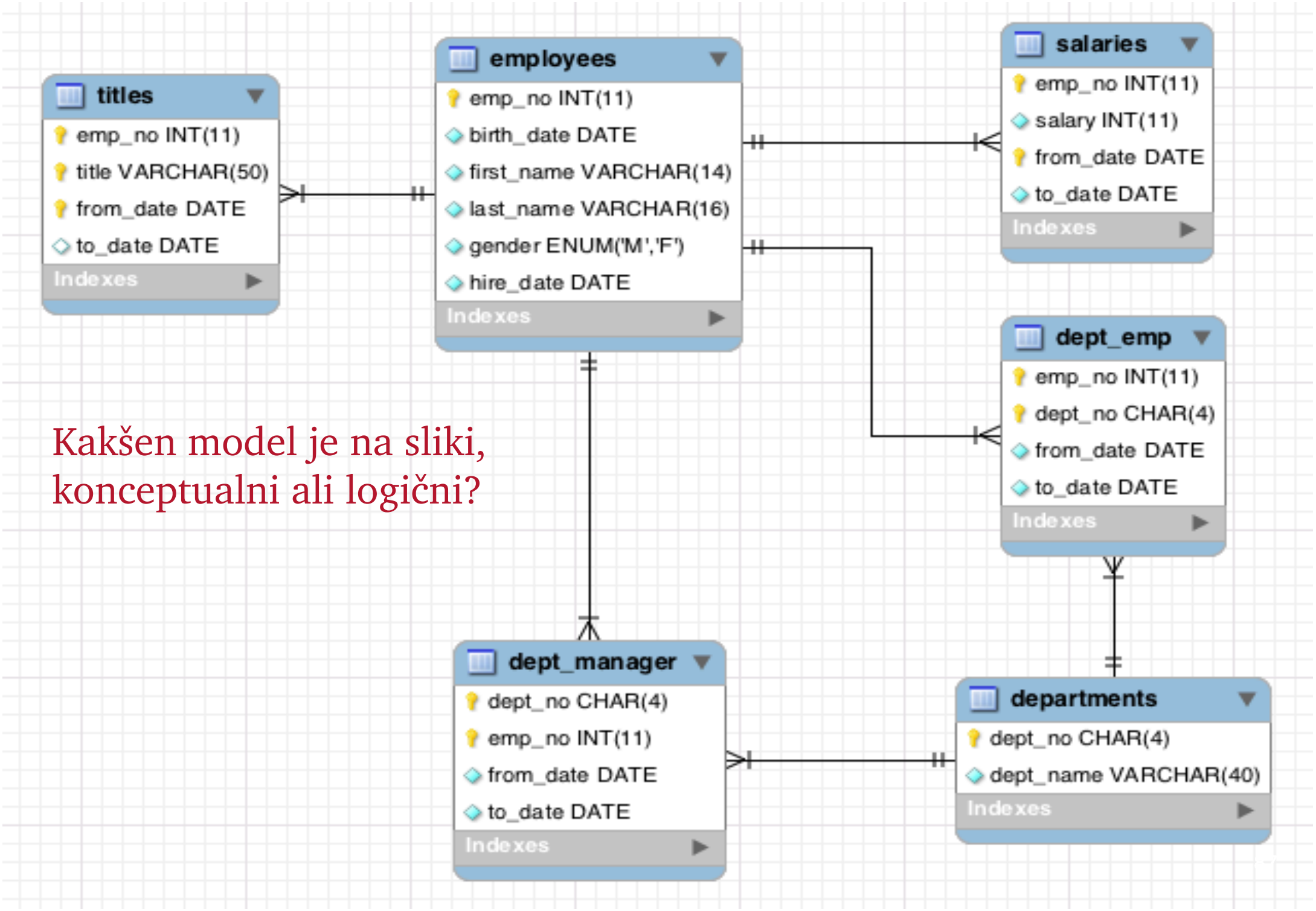
- Po pretvorbi poskrbimo še za:
  - Ustreznost imen
  - Pretvorbo sestavljenih, več-vrednostnih in izpeljanih atributov
  - Indekse po poljih, ki niso primarni ali tuji ključ
  - Določitev pogledov
  - Ustreznost ključev
  - Referencialno integriteto
  - Sprožilce in shranjene procedure
  - …

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



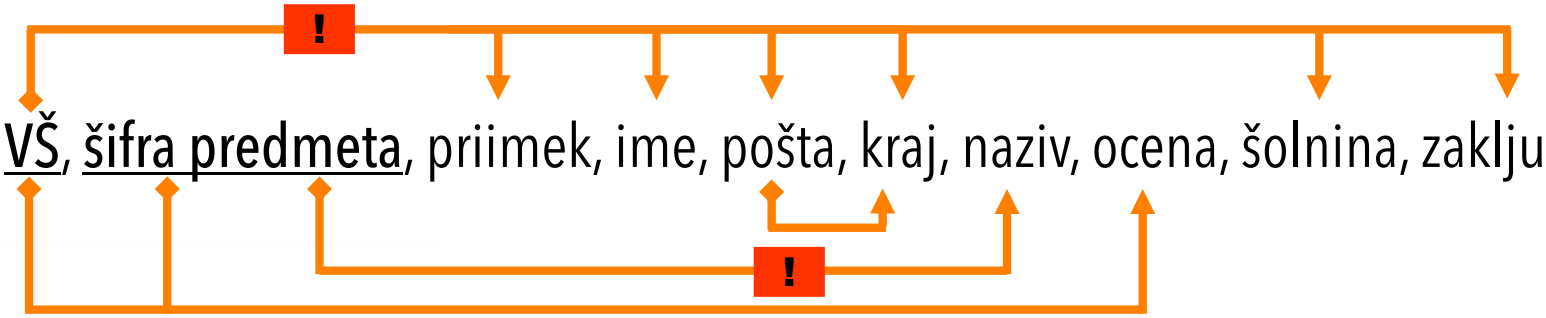
VŠ	priimek	ime	pošta	kraj	šifra predmeta	naziv	ocena	šolnina	zaključek
64010632	Bratina	Simon	4100	Kranj	20020	IS	10	10.1.2016	
64010632	Bratina	Simon	4100	Kranj	20021	TPO	8	10.1.2016	
64010632	Bratina	Simon	4100	Kranj	20033	IPI	8	10.1.2016	
64016209	Bizjak	Tadeja	2250	Ptuj	20060	E1	9		1.1.2022
64016209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

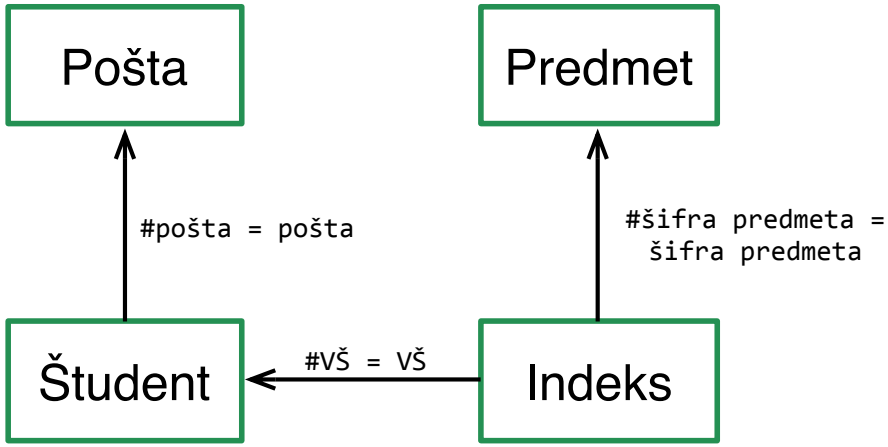
Indeks

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

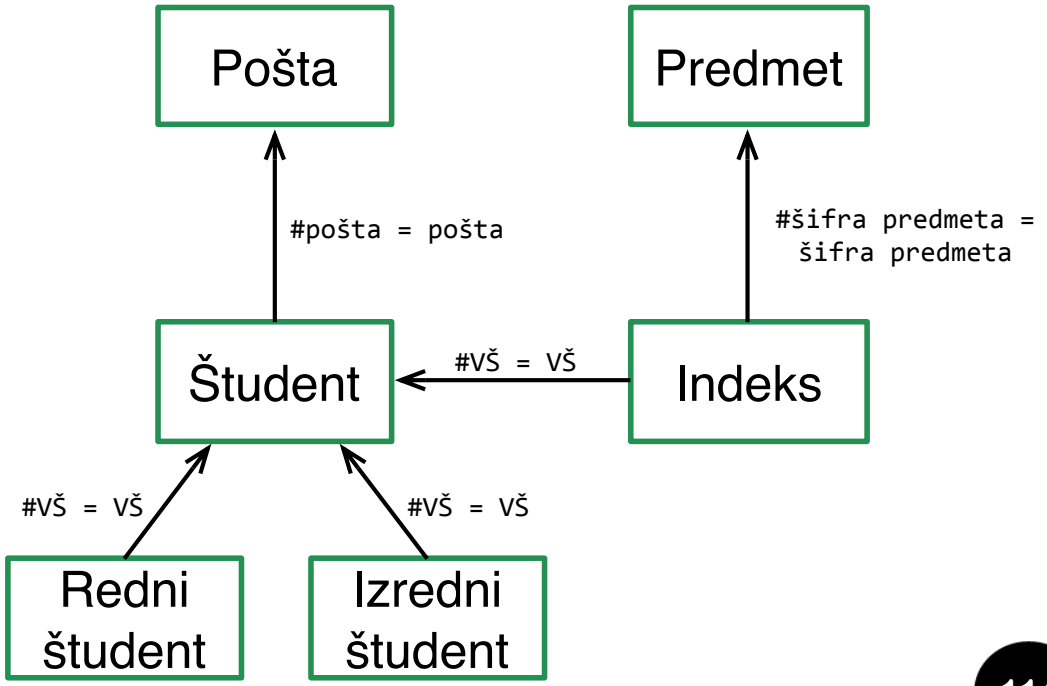
- Začetna shema

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

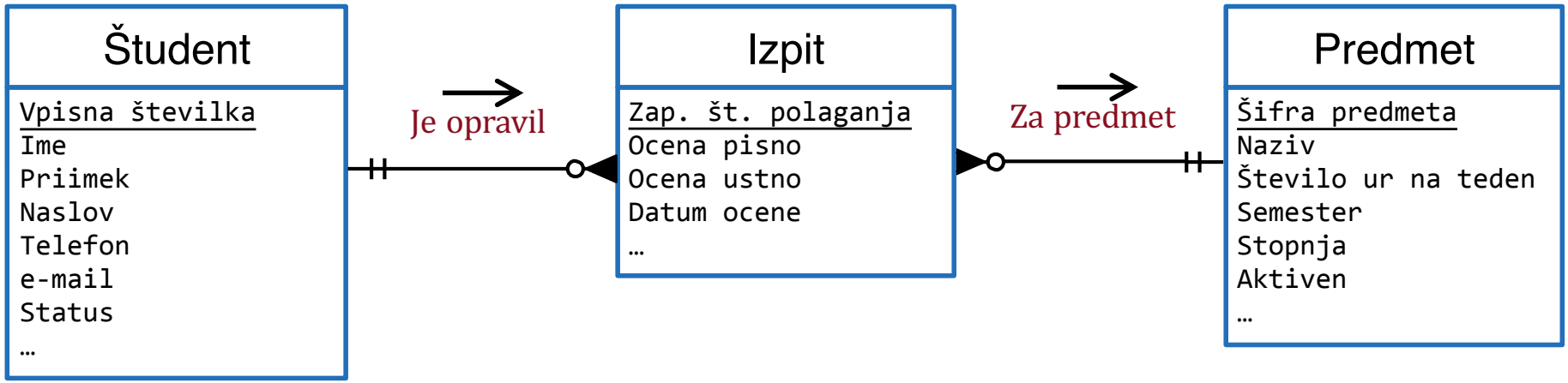
K2.1 – Za entitetne tipe kreiraj relacije...

- 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)

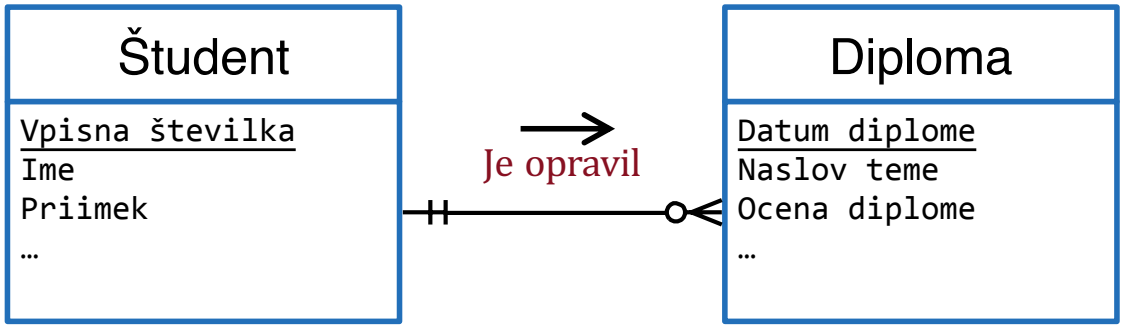
Izpit(ZapStPol, #VpisSt, #IDPred, OcenaPisno, OcenaUstno, DtmOc)

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

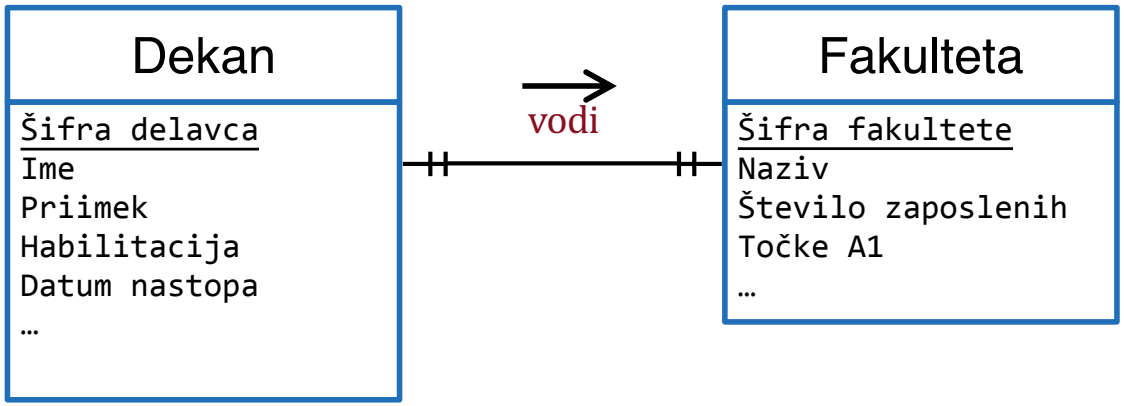


Študent(VpisSt, Ime, Priimek,...)

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

Primer: povezave 1:1

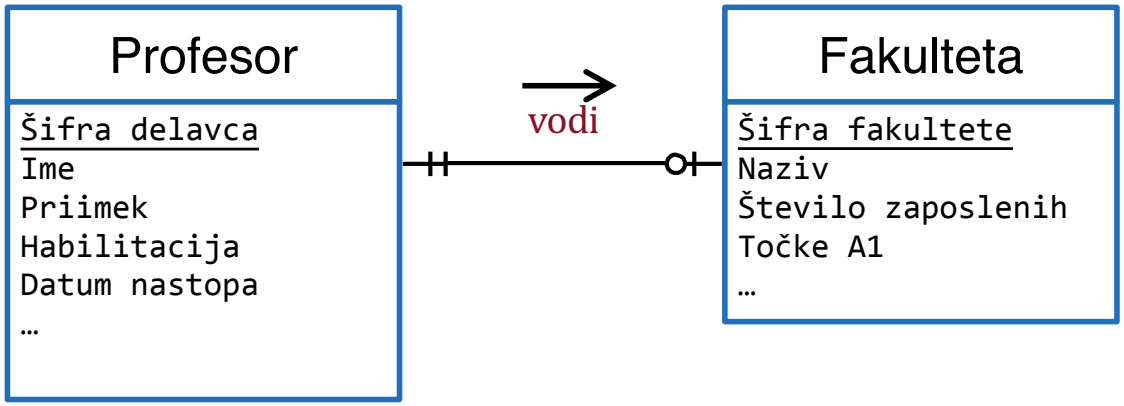
- Obveznost na obeh straneh: združimo v eno entiteto in izberemo ključ!



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...)

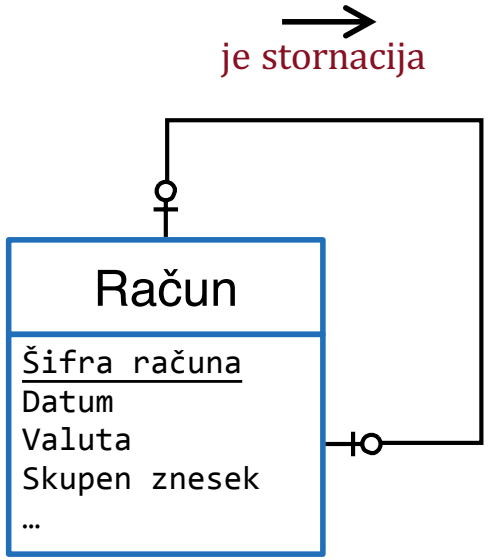
Poimenovani tuji ključ.

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 številnost modela pravilna?

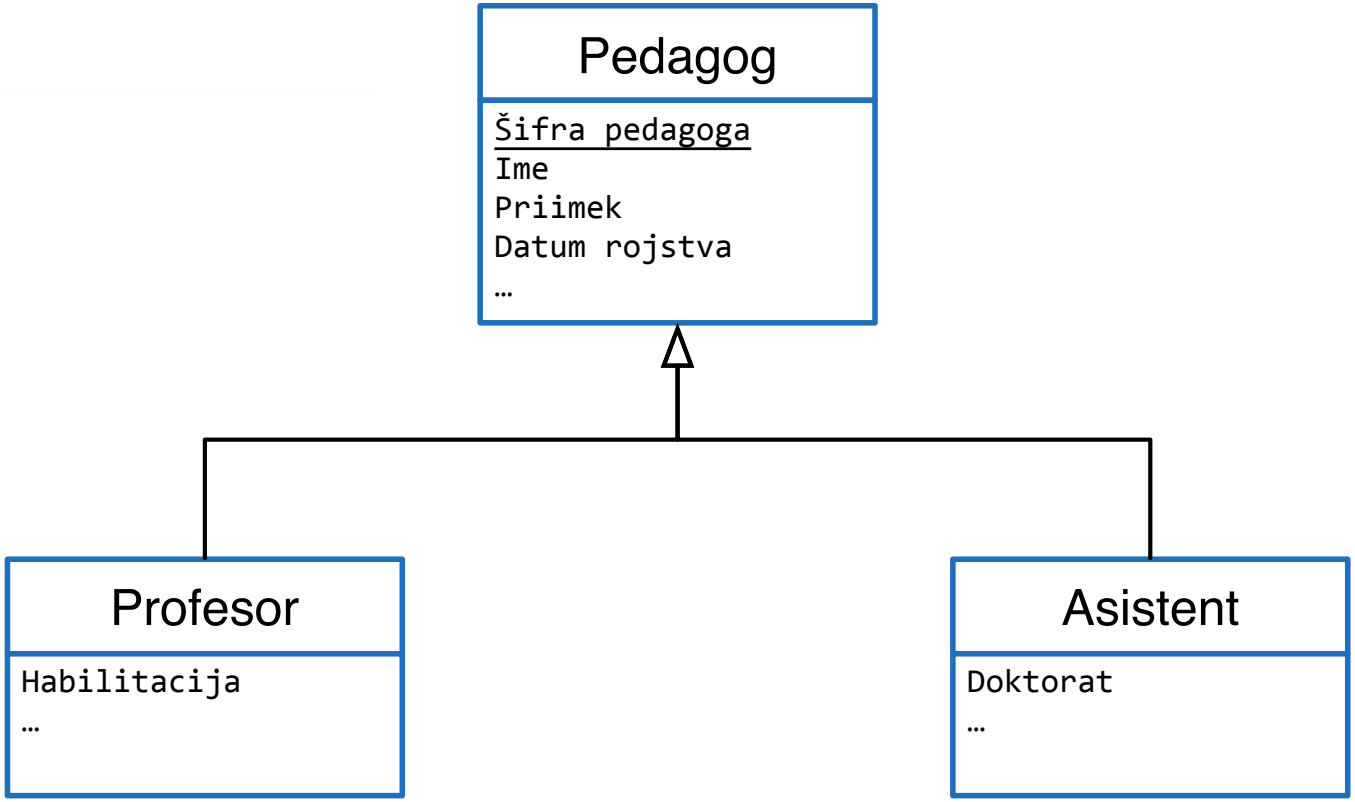
Racun(IDrac, #IDracStorno, Datum, Valuta, Znesek,…)

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?

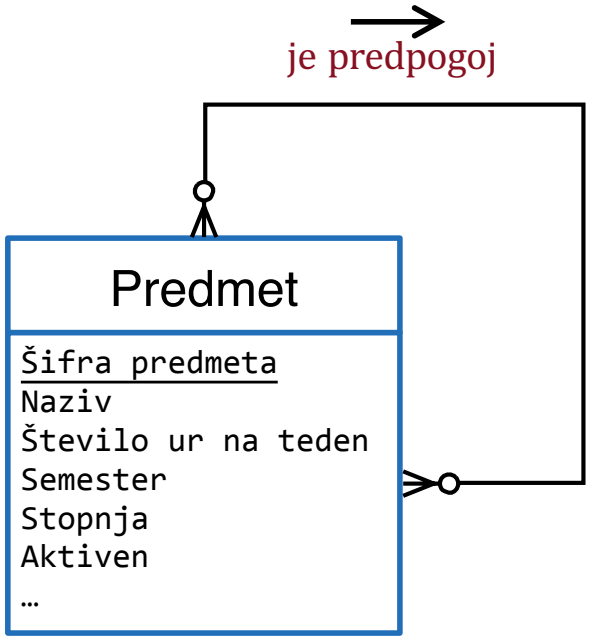


Povezave n:n

- 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...

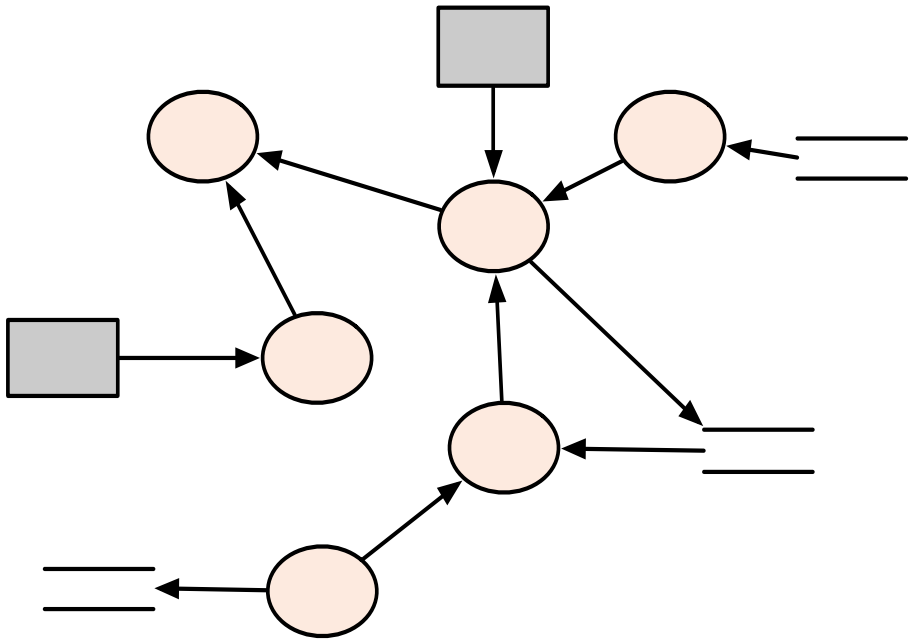
- V tem koraku preverimo pravila za zagotavljanje celovitosti podatkov:
  - Obveznost atributov
  - Omejitve domen atributov
  - Številnost
  - Omejitve entitet (celovitost entitet)
  - Omejitve povezav (celovitost povezav)
  - Splošne omejitve

K2.5 – Preveri model z uporabnikom...

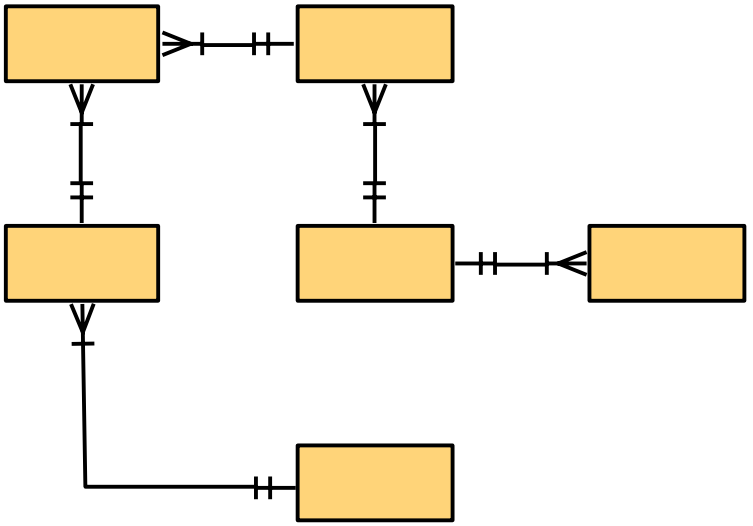
- Namen tega koraka je preveriti model z uporabnikom ter ugotoviti, če ustreza vsem uporabniškim zahtevam.
- Model lahko zajema več uporabniških pogledov. Pri pregledu lahko nastopa več uporabnikov.
- Odličen način za pregled celovitosti podatkovnega modela je specifikacija podatkovnih tokov s pomočjo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

Povezava podatkovni model in DFD

Diagram podatkovnih tokov



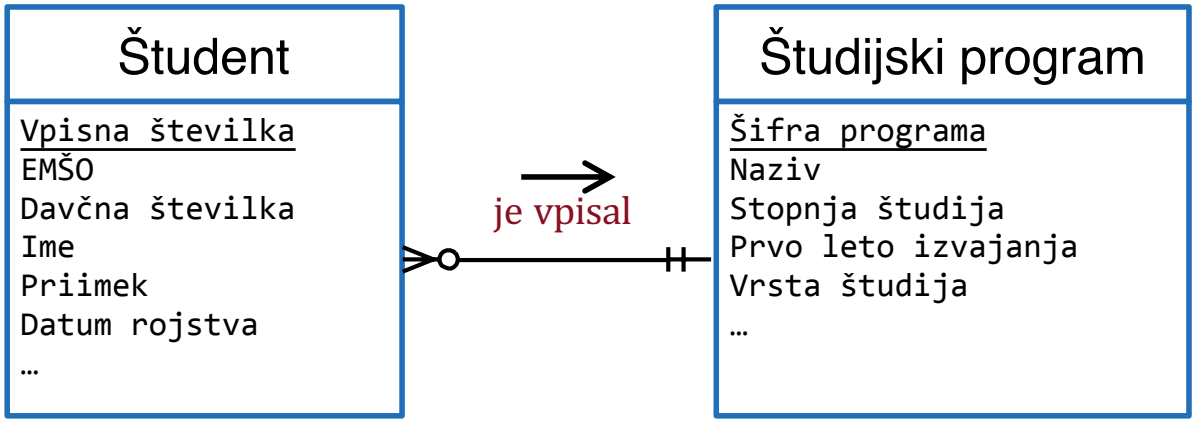
Podatkovni model

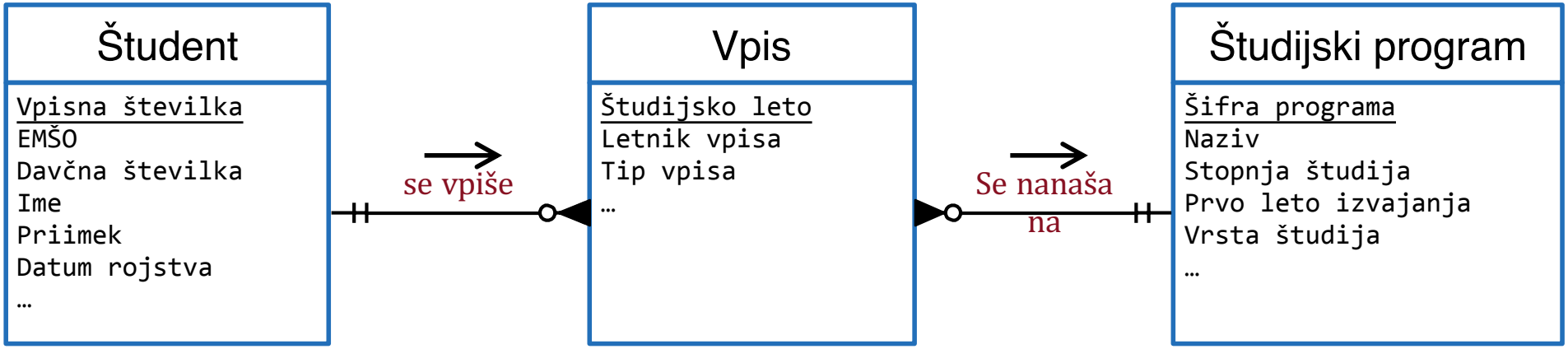


K2.6 – Preveri možnosti za razširitve...

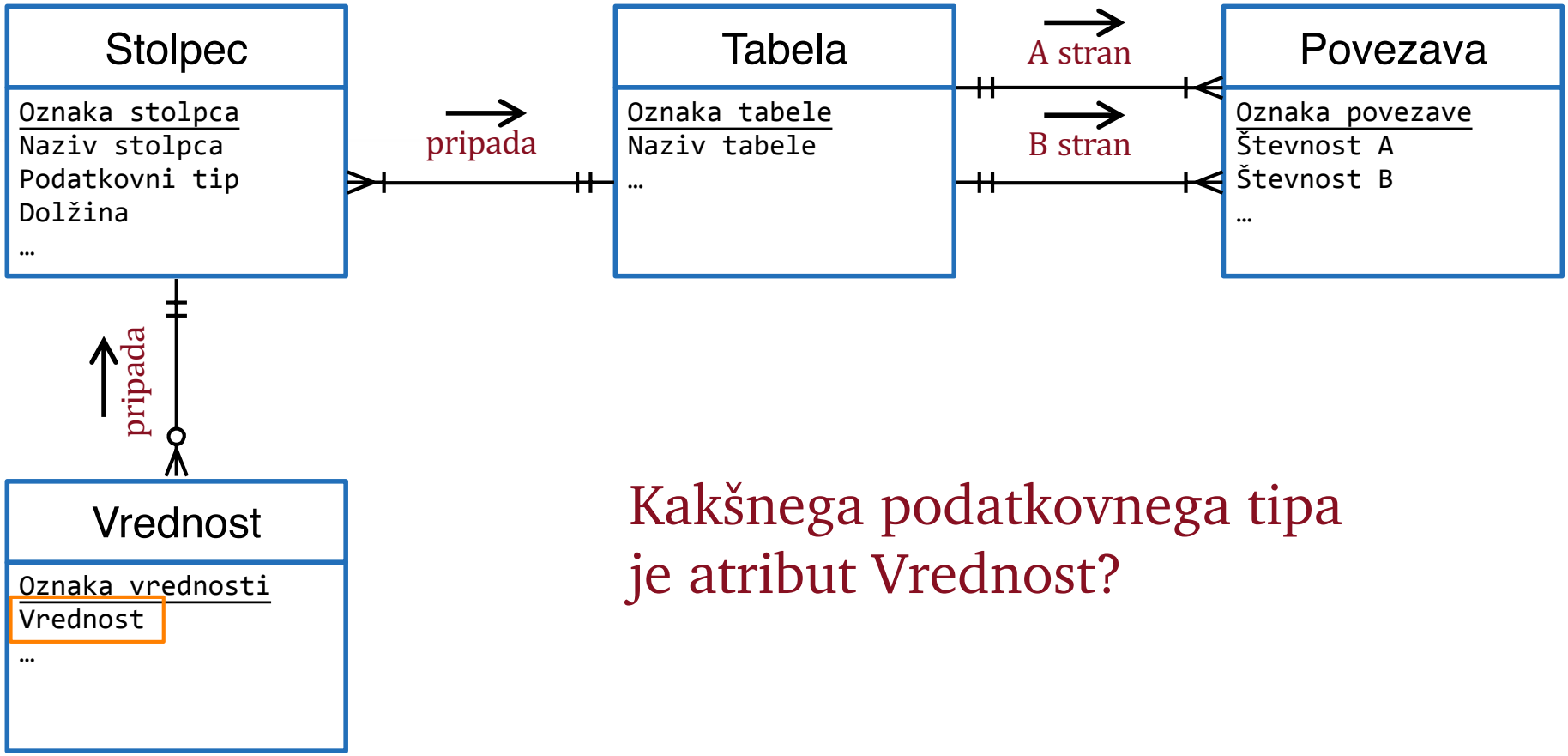
- V primeru, da so predvidene bodoče razširitve sistema, moramo preveriti, če logični model take razširitve podpira.
- Podatkovni model mora biti prilagodljiv; omogočati mora razširitve skladno z novimi zahtevami ter z minimalnim vplivom na obstoječe uporabnike.
- Popolnoma odprt sistem za razširitve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

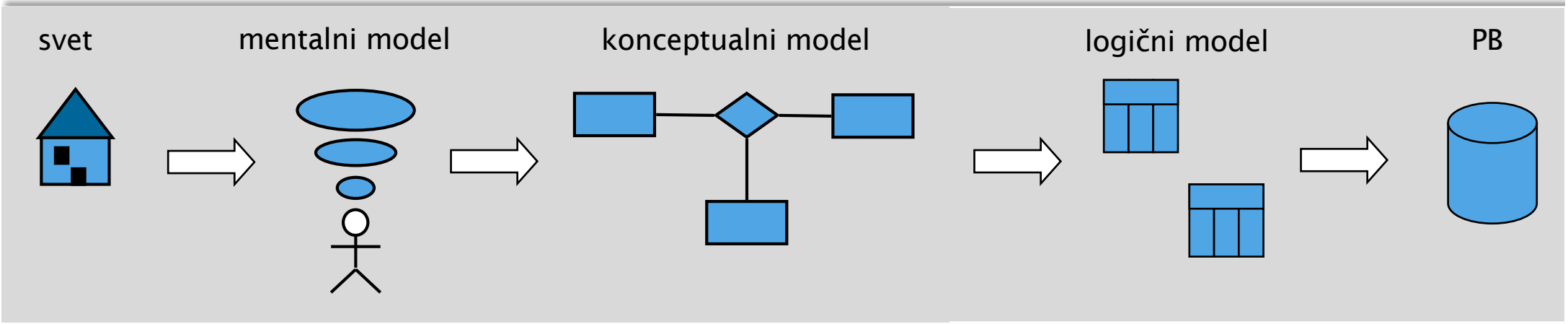
Primer generičnega podatkovnega modela

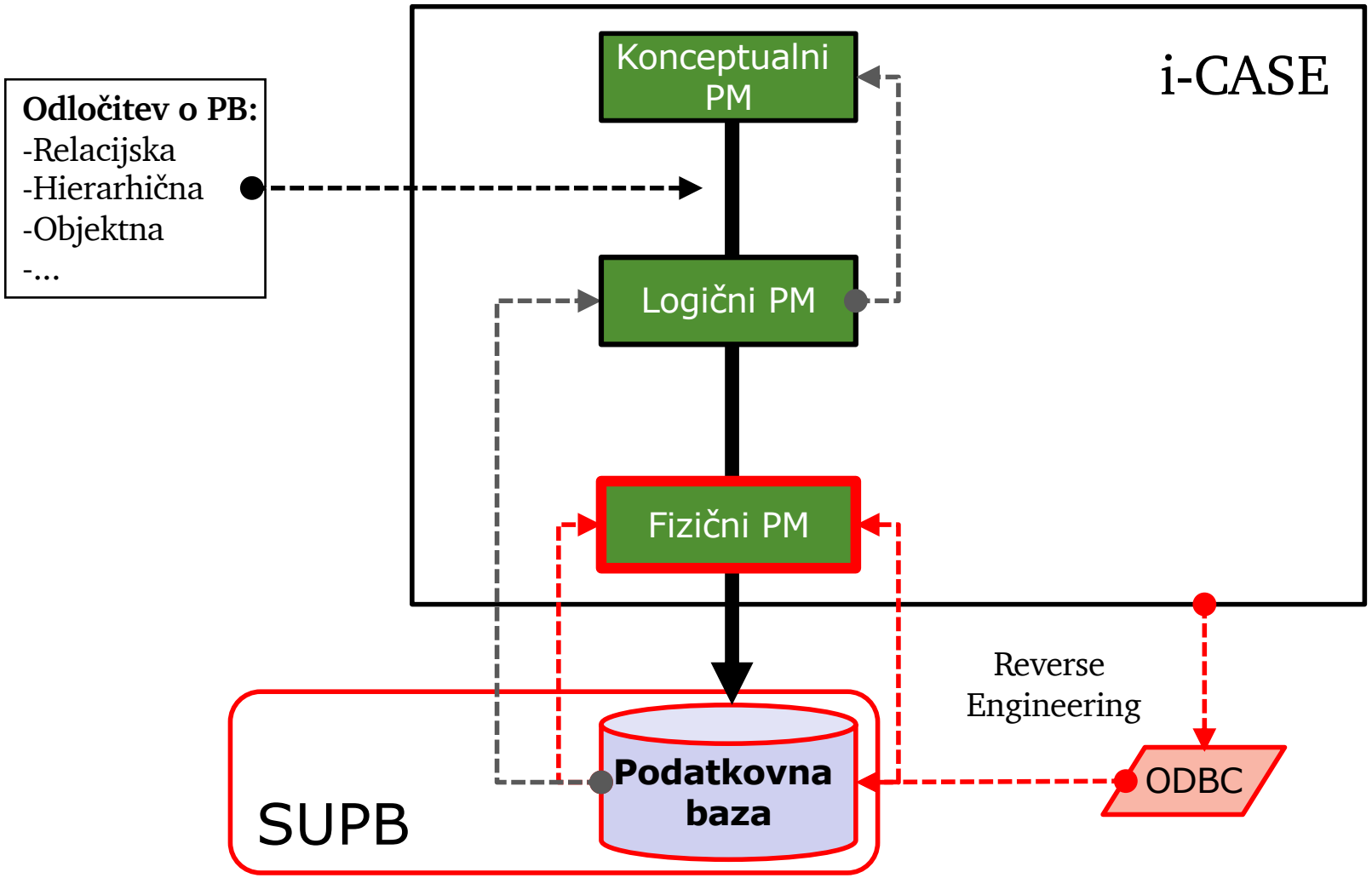


Kakšnega podatkovnega tipa je atribut Vrednost?

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.

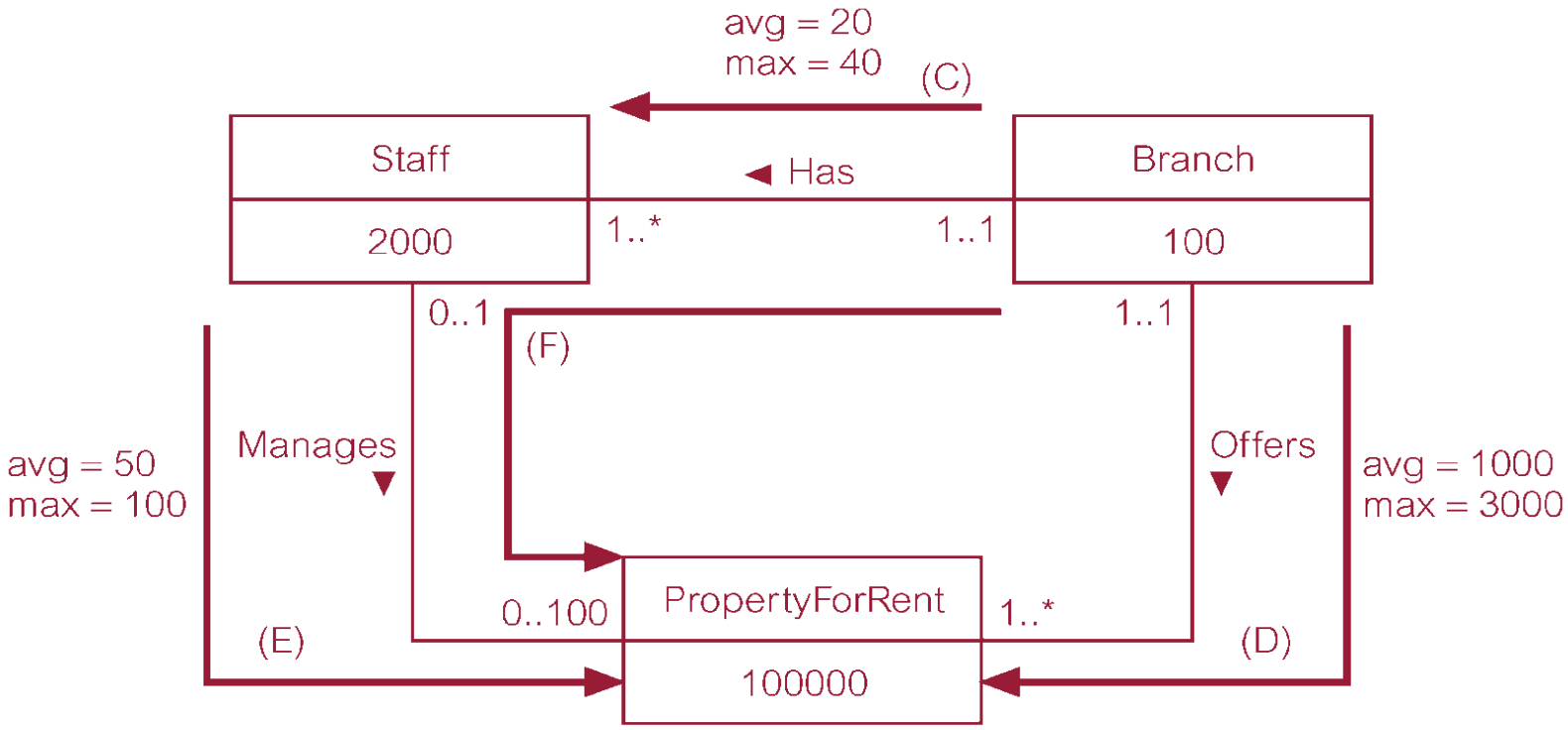
Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
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

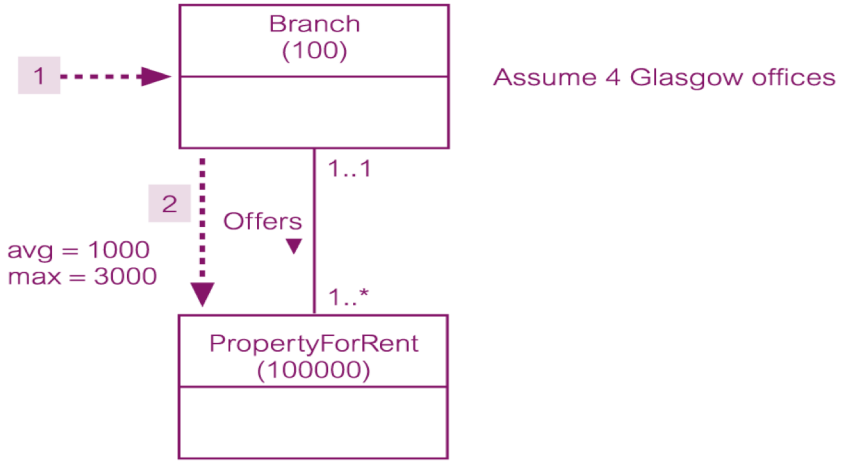
Transaction volume

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.

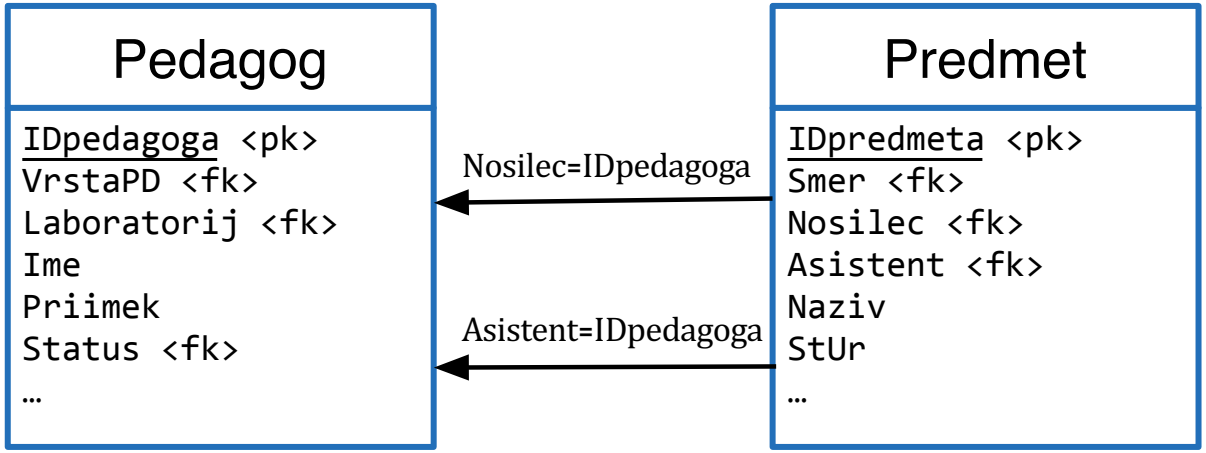
K5 – Načrt uporabniških pogledov

- 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

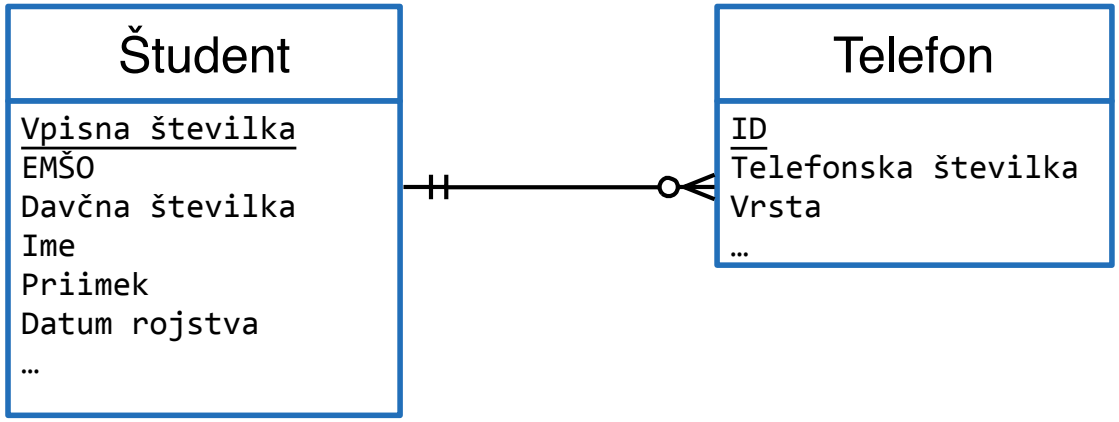
salaries

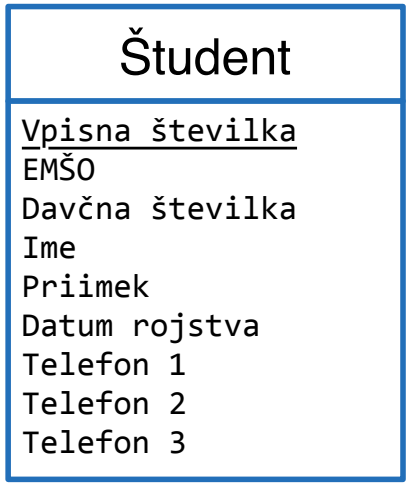
emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24
10001	74333	1992-06-24	1993-06-24
10001	75286	1993-06-24	1994-06-24
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

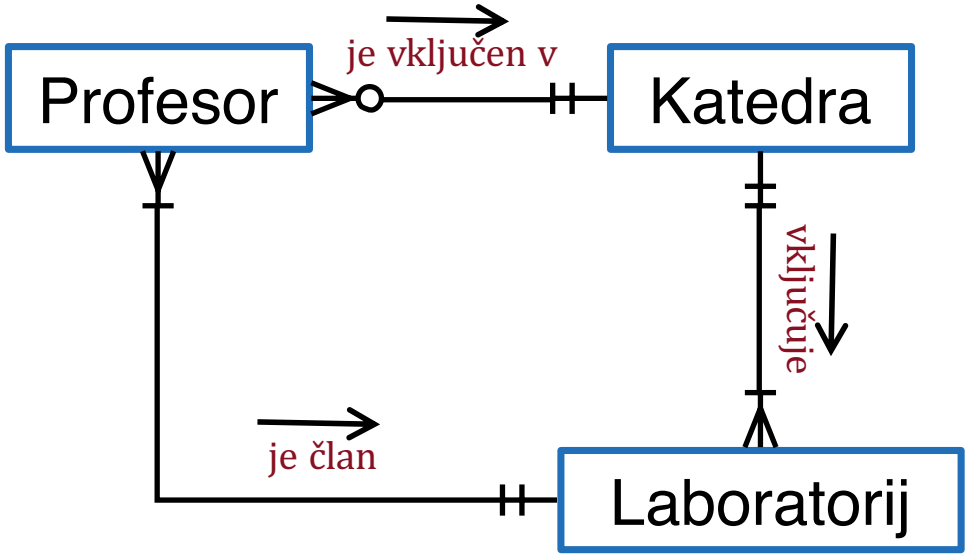
emp_no	birth_date	first_name	last_name	gender	hire_date	salary
10001	1953-09-02	Georgi	Facello	M	1986-06-26	88958
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	72527
10003	1959-12-03	Parto	Bamford	M	1986-08-28	43311
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	74057
10005	1955-01-21	Kyoichi	Maliniak	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



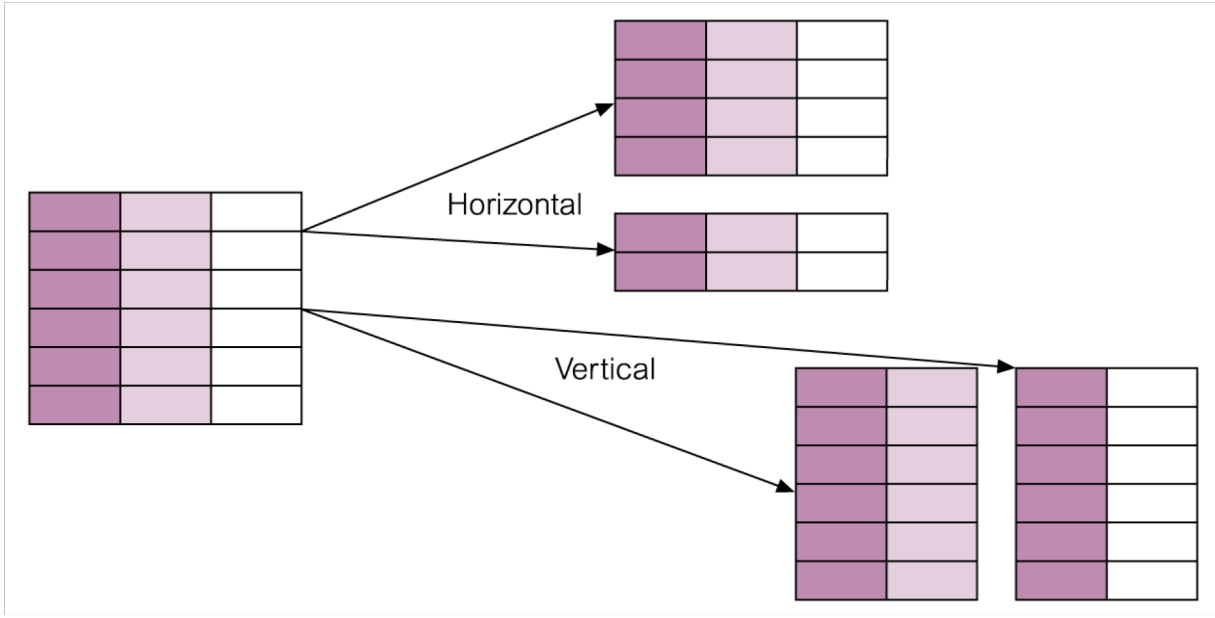


Primer: podvajanje tujih ključev



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)
```

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Predmet:
Osnove podatkovnih baz

Modul:
Podatkovna skladišča

Gradivo:
v.2015

Vsebina

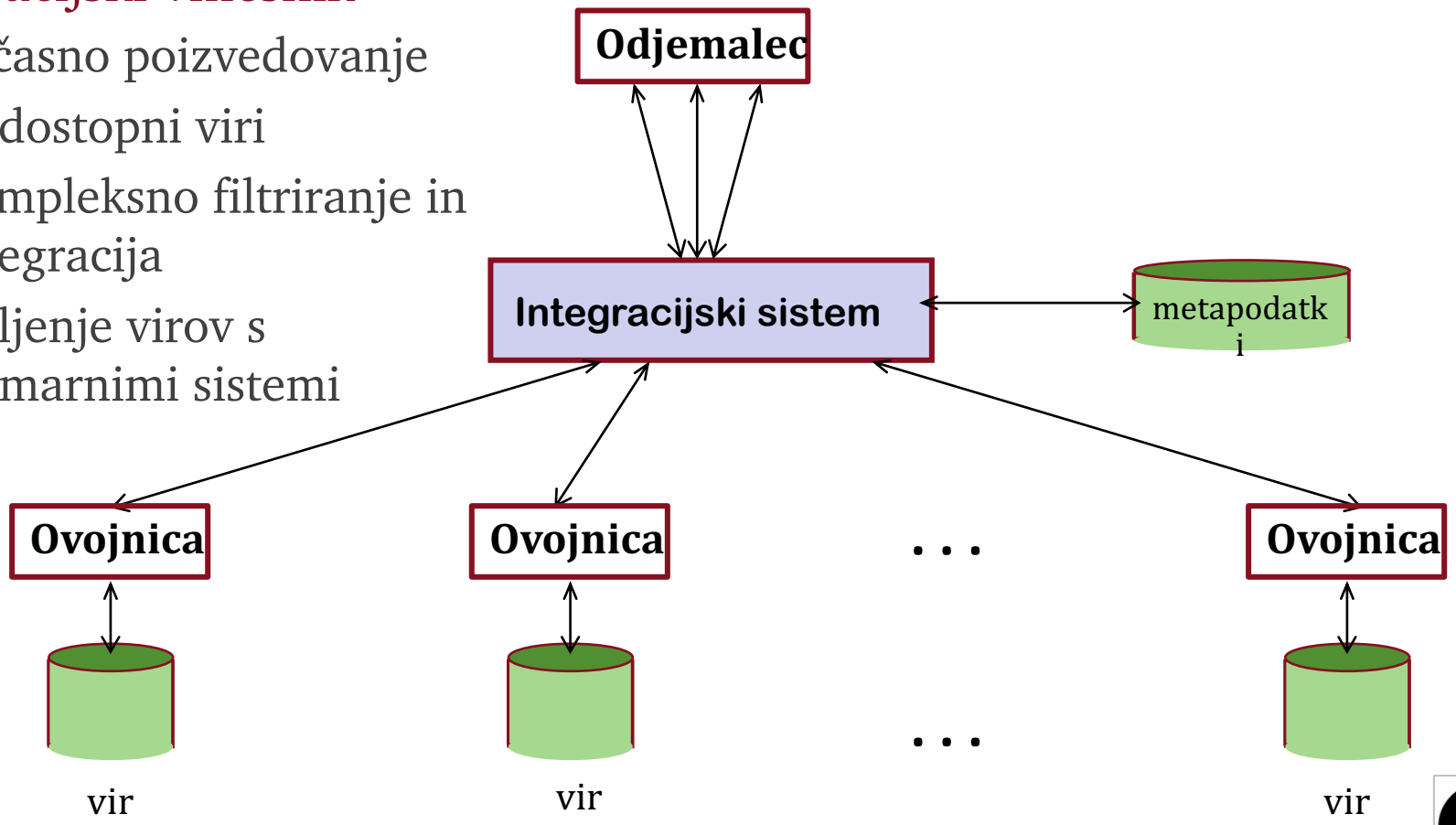
- Namen podatkovnih skladišč
- Lastnosti podatkovnih skladišč
- Arhitektura podatkovnih skladišč
- Orodja, ki uporabljajo podatkovna skladišča
- Načrtovanje podatkovnih skladišč

Zakaj podatkovna skladišča?

- Za poslovno odločanje so potrebni podatki:
  - iz različnih aplikacij informacijskega sistema, ne glede na uporabljeno platformo in fizično lokacijo aplikacije;
  - za daljše časovno obdobje – podatki, ki omogočajo pregled nad zgodovino vrednosti, ne le trenutne vrednosti podatkov in kazalnikov;
  - za identifikacijo trendov – podatki, ki omogočajo pregled nad trendi, kot so rast, padanje, stagnacija itn.

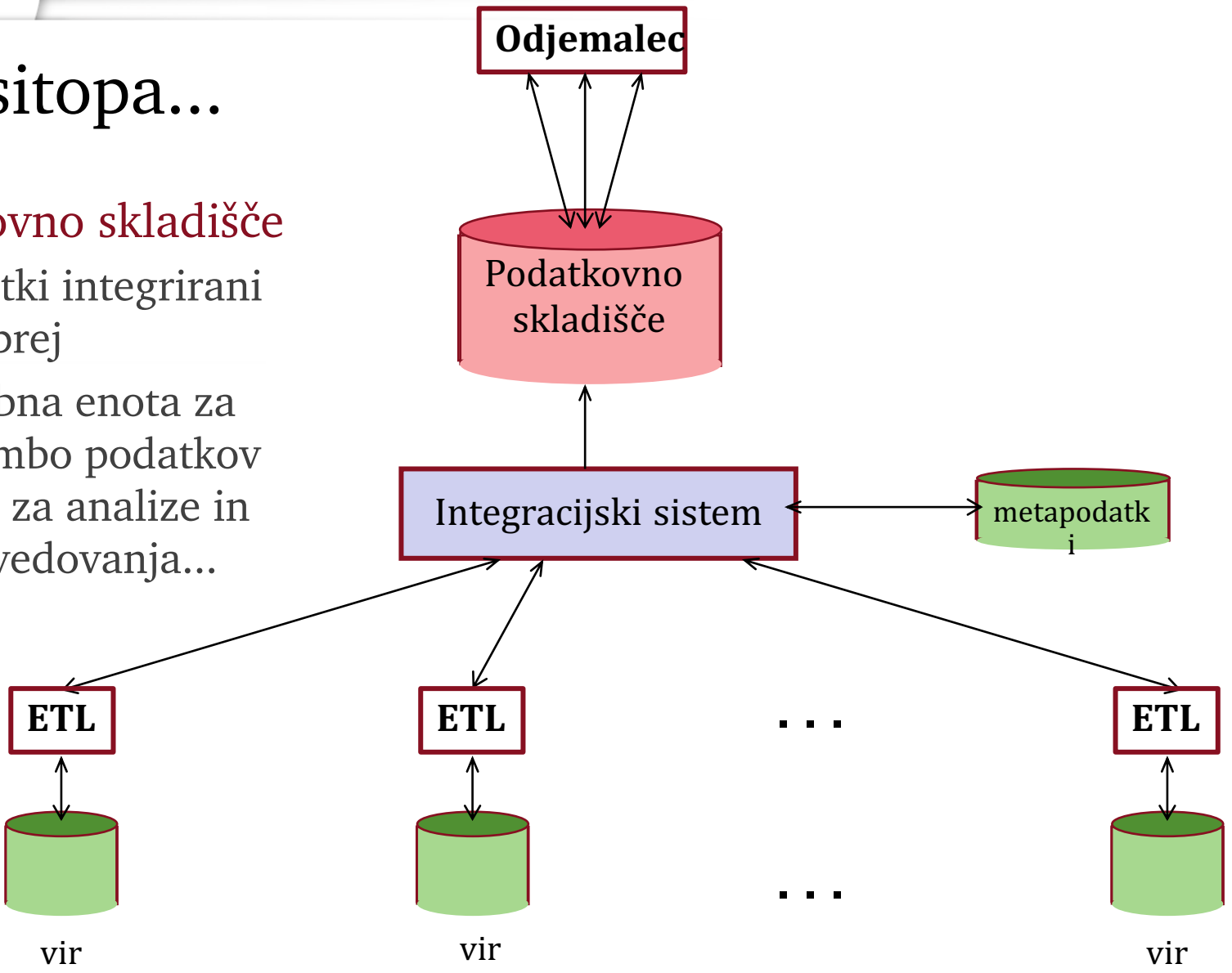
Dva prsitopa...

- Integracijski vmesnik
  - Počasno poizvedovanje
  - Nedostopni viri
  - Kompleksno filtriranje in integracija
  - Deljenje virov s primarnimi sistemi
  - ...

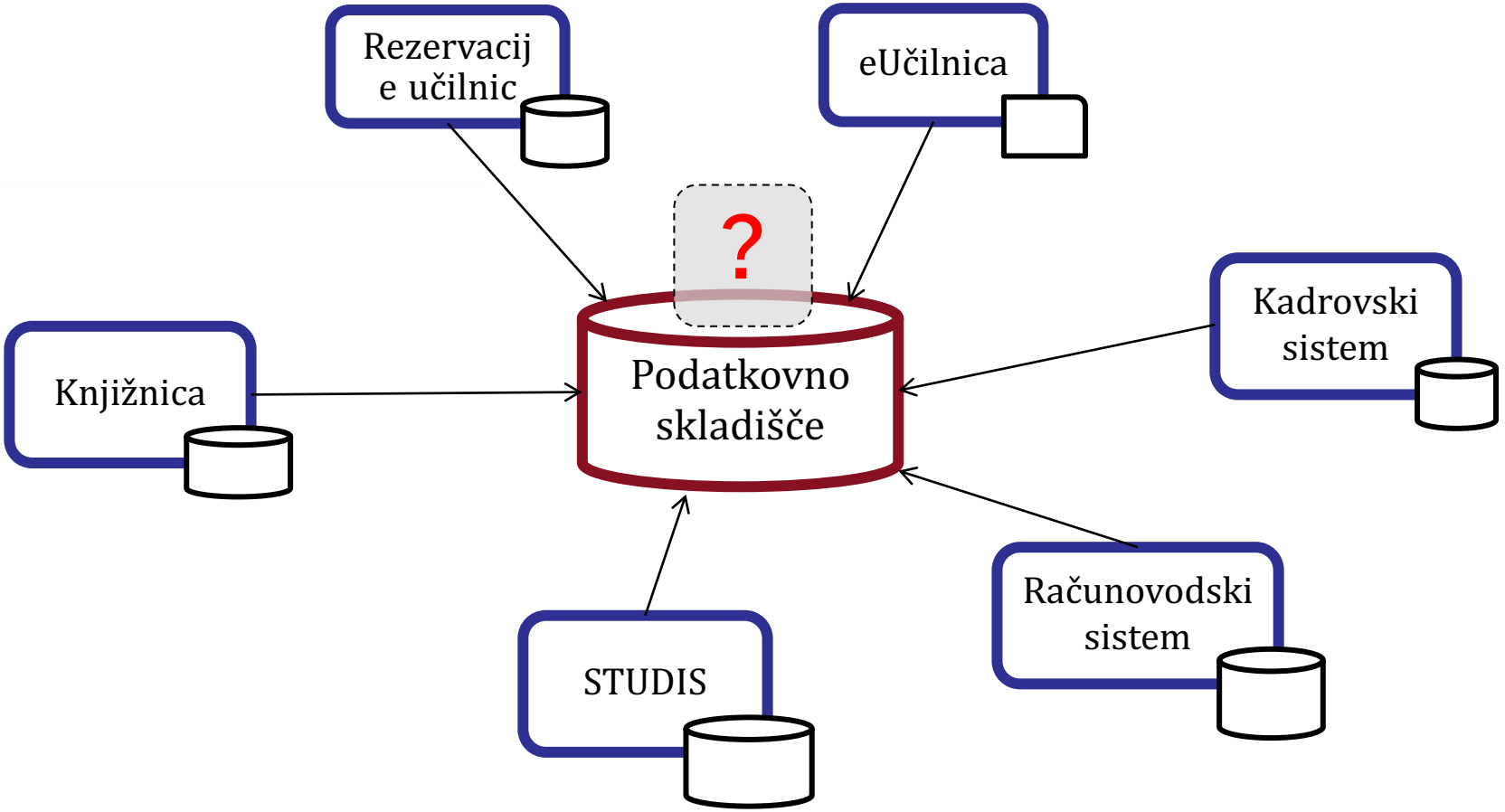


Dva prsitopa...

- Podatkovno skladišče
  - Podatki integrirani v naprej
  - Posebna enota za shrambo podatkov zgolj za analize in poizvedovanja...



Primer: IS FRI



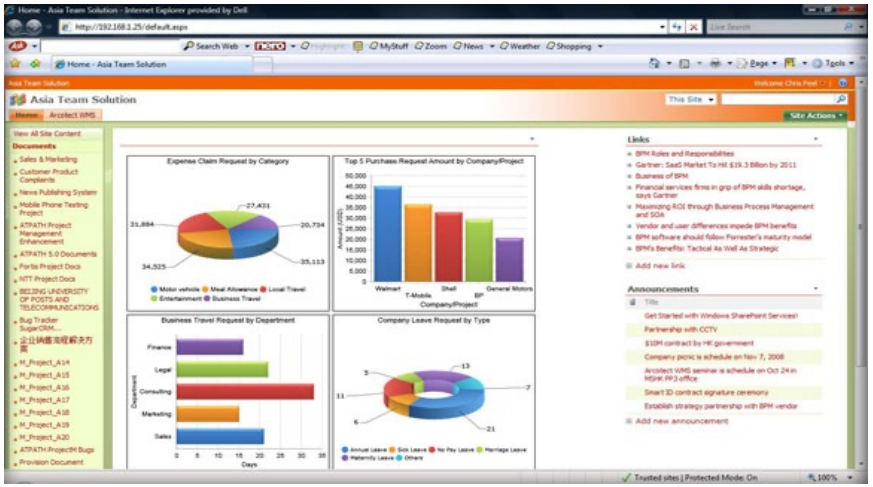
Evolucija podatkovnih skladišč

- V sedemdesetih aplikativni sistemi → avtomatizacija procesov na operativnem nivoju
- Posledica: akumulacija velikih količin podatkov v PB transakcijskih sistemov

- Danes poudarek različnim načinom uporabe teh podatkov za podporo odločitvenim procesom z namenom pridobiti strateško prednost pred konkurenco!

Podpora odločanju v transakcijskih IS

- Podpora odločanju na operativnem in taktičnem nivoju tudi v nekaterih transakcijskih IS → poročila, sumarni pregledi, grafi, …





- Problem več transakcijskih IS. Področja obravnavana ločeno, po svoje…

Lastnosti podatkovnega skladišča

- Definicija:
  - Podatkovno skladišče je vsebinsko organizirana, integrirana, časovno označena in nespremenljiva zbirka podatkov za namene podpore odločitvenim procesom.

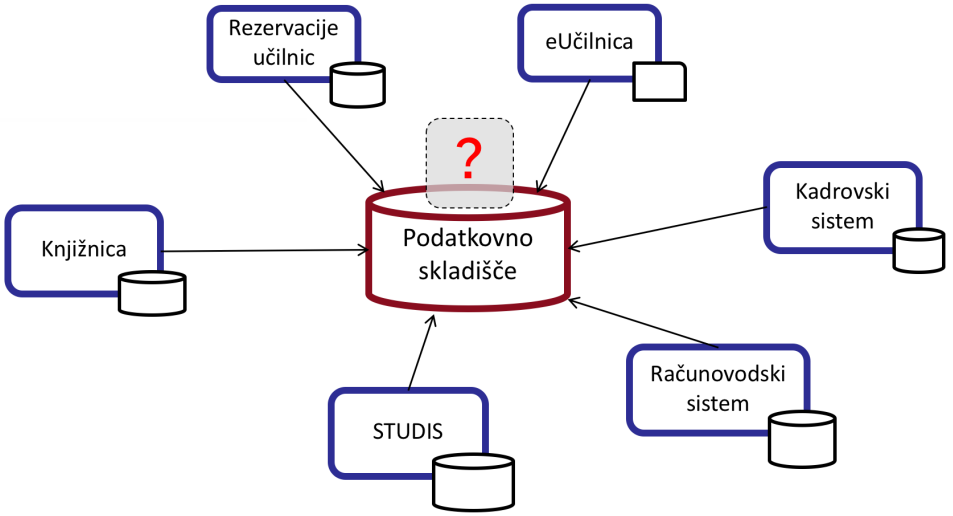


Vsebinska organizacija podatkov

- Organizacija podatkovnega skladišča temelji na entitetah, ki so ključne za neko poslovni sistem, npr.:
  - študent,
  - izpit,
  - profesor,
  - predmet,
  - račun…
- Ne temelji na funkcionalnih področjih oz. področjih, ki jih pokrivajo posamezni transakcijski sistemi!

Integriranost

- Podatkovno skladišče integrira podatke iz več aplikativnih sistemov v okviru IS podjetja.
- Podatki večkrat nekonsistentni. Naloga skladišča je, da omogoči konsistenten pogled na podatke!



Časovna označenost

- Podatki v skladišču so veljavni za določen čas oziroma interval.
- Za podatek v podatkovnem skladišču moramo poznati čas prenosa in čas veljavnosti.

- Časovna odvisnost večkrat prikazana v razširjenem časovnem formatu (poleg leta še kvartal, četrtletje).
- Časovna odvisnost omogoča opazovanje trendov.

Nespremenljivost

- Podatki v podatkovnem skladišču niso podvrženi spremembam v realnem času s strani aplikacij!
- Osveževanje (iz transakcijskih sistemov in ostalih virov) z neko (smiselno) frekvenco
  - Osveževanje: večinoma le dodajanje novih podatkov v podatkovno skladišče.

OLAP in OLTP...

- OLAP – On Line Analitical Processing (vir podatkovno skladišče)
- OLTP – On Line Transaction Processing (vir transakcijska baza)

- Za OLAP in OLTP praviloma različne instance podatkovne baze!

OLAP in OLTP...

OLTP	OLAP
Obdelujejo sveže (trenutne) podatke	Obdelujejo zgodovinske podatke
Baza vsebuje podrobne podatke	Baza vsebuje podrobne in sumarizirane podatke
Podatki so dinamični	Podatki so statični
Postopki uporabe se ponavljajo - transakcije, so predvidljivi	Postopki uporabe (analize) nepredvidljivi
Zelo veliko število transkacij na časovno enoto	Majhno število transakcij
Transakcijsko usmerjeni sistemi	Analitično usmerjeni
Aplikacijsko naravnani	Predmetno naravnani
Podpora vsakodnevnim odločitvam	Podpora taktičnim in strateškim odločitvam
Veliko število operativnih uporabnikov	Manjše število vodstvenih uporanikov

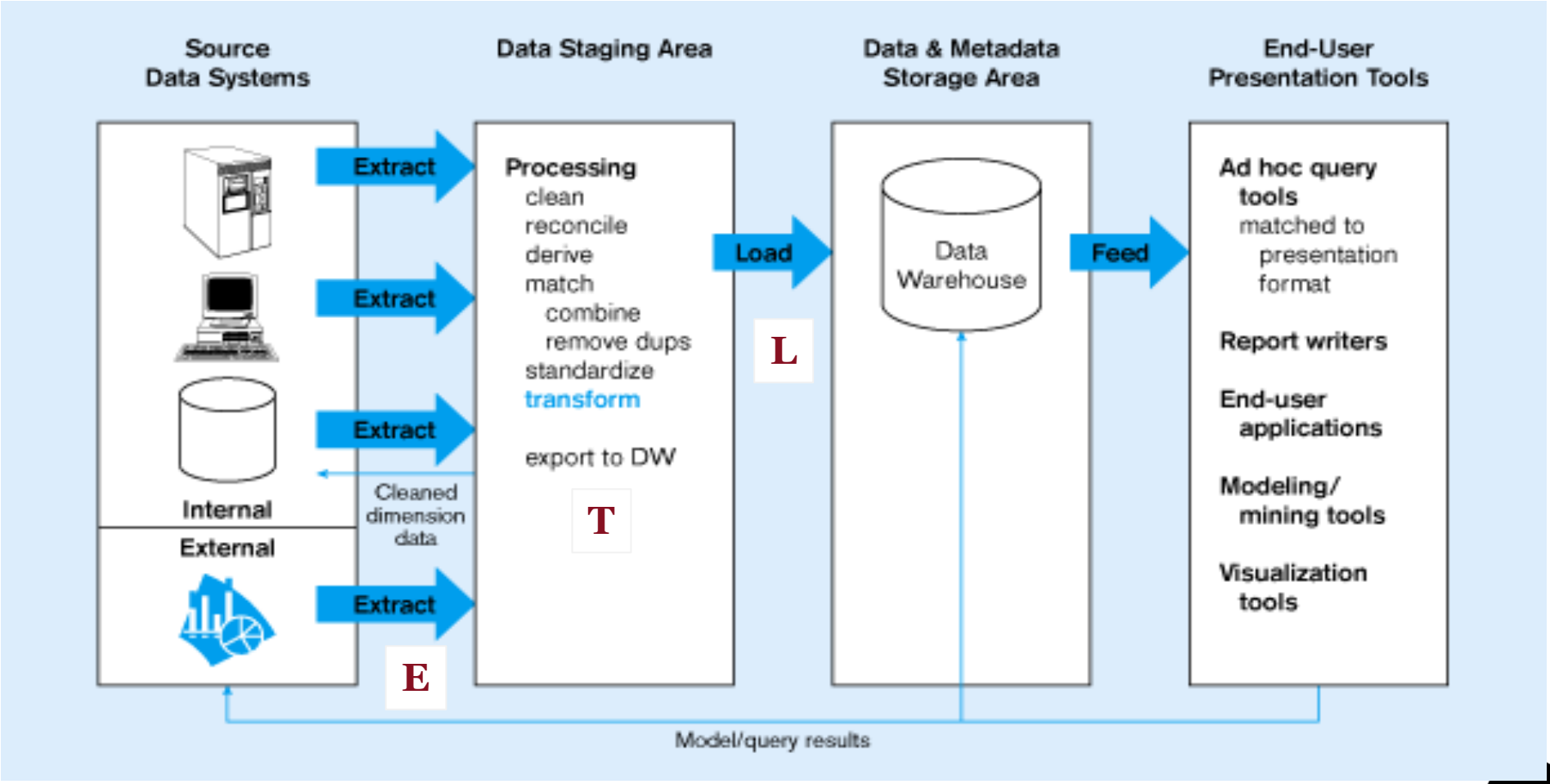
OLAP vs OLTP...

- Primeri vprašanj za OLTP…
  - Kakšen je bil celoten prihodek podjetja v prvem kvartalu 2016?
  - Kakšen je bil celoten prihodek iz prodaje nepremičnin za vsak tip nepremičnine v Veliki Britaniji v letu 2016?
  - Katera so tri najbolj popularna področja v velikih mestih za najem nepremičnin v 2016 in kako je to v primerjavi s preteklimi tremi leti.

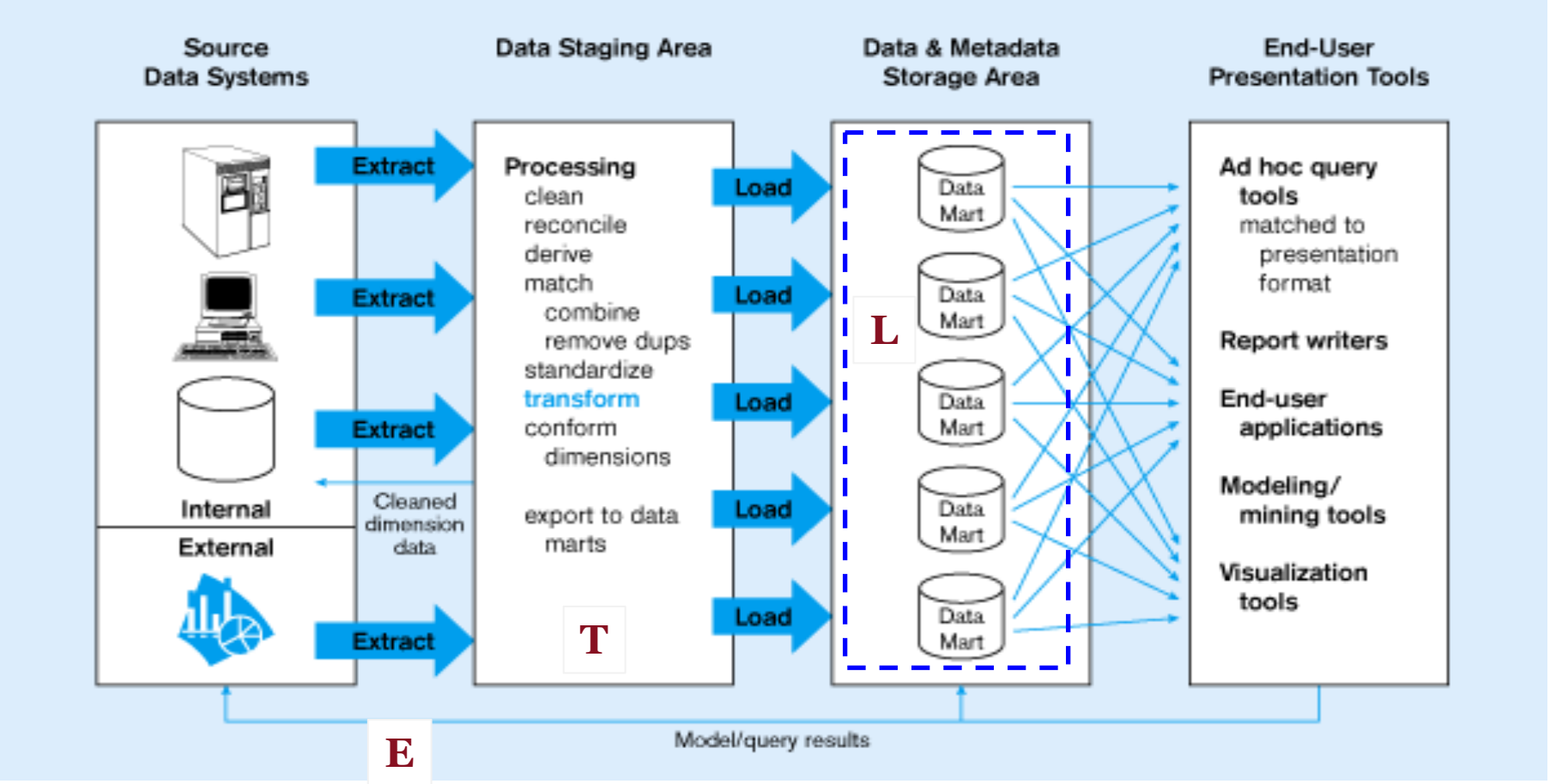
OLAP vs OLTP...

- Primeri vprašanj za OLAP
  - Kakšen bi bil učinek na prodajo nepremičnin v različnih regijah Velike Britanije, če bi cene nepremičnin, dražjih od 100.000 funtov povečali za 3,5%, obenem pa zmanjšali davke za 1,5%?
  - Katere vrste nepremičnin se prodajajo s ceno, ki je večja od povprečne cene nepremičnin v glavnih mestih Velike Britanije in v kakšni korelaciji je to z demografskimi podatki?
  - Ali obstaja signifikantna povezava med celotnim prihodkom, generiranim v posamezni nepremičninski agenciji, in številom agentov v tej agenciji?

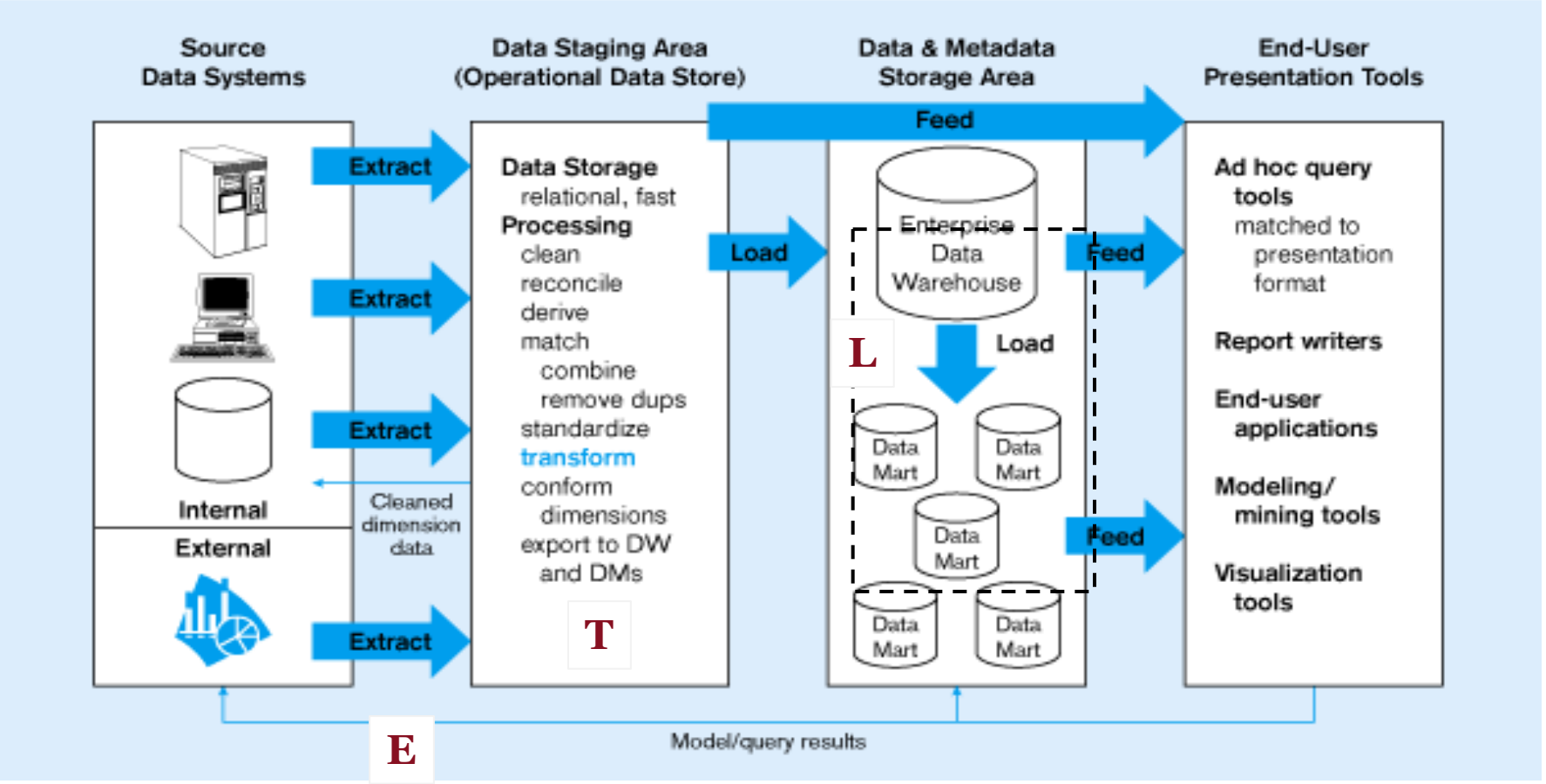
Generična dvo-nivojska arhitektura



Neodvisna področna skladišča



Odvisna področna skladišča



Orodja za dostop do DW

- DW mora podpirati učinkovite ad-hoc in rutinske analize.
  - Visoka učinkovitost dosežena s pred-planiranjem povezav, agregacij, periodičnih poročil itn.

- Tipična orodja končnih uporabnikov:
  - Orodja za poročila in poizvedovanje,
  - Razvojna orodja,
  - Direktorski IS,
  - OLAP orodja,
  - Orodja za interaktivno analitiko in vizualizacijo ter orodja za podatkovno rudarjenje.



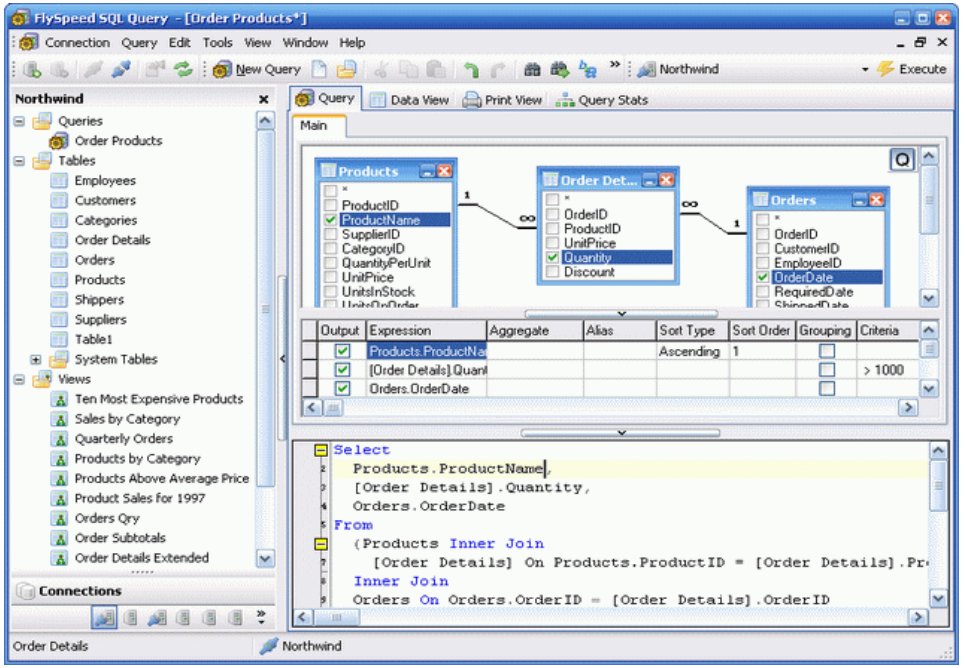
Generatorji poročil

- Priprava in generiranje poročil – različna orodja za pripravo standardnih poročil (navadno desktop aplikacije) ter generatorji poročil (strežnike aplikacije).



Orodja za poizvedovanje

- Orodja za poizvedovanje – orodja za poizvedovanje po relacijskih DW sprejmejo SQL na vhodu ali generirajo SQL.

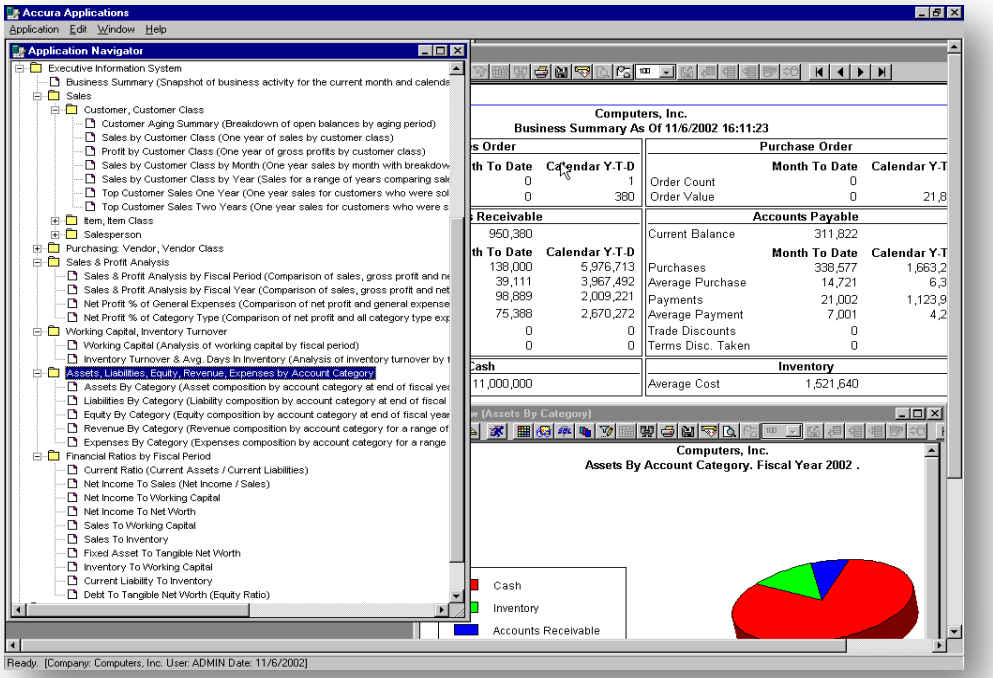


Razvojna okolja

- Razvojna okolja – če uporabniških zahtev ni moč realizirati z orodji za generiranje poročil ali poizvedovalnimi orodji, potrebno razviti uporabniško aplikacijo.
- Na voljo mnoga razvojna okolja. Večina omogoča povezavo s poljubnimi SUPB ter mnogimi OLAP orodji.

Direktorski informacijski sistemi

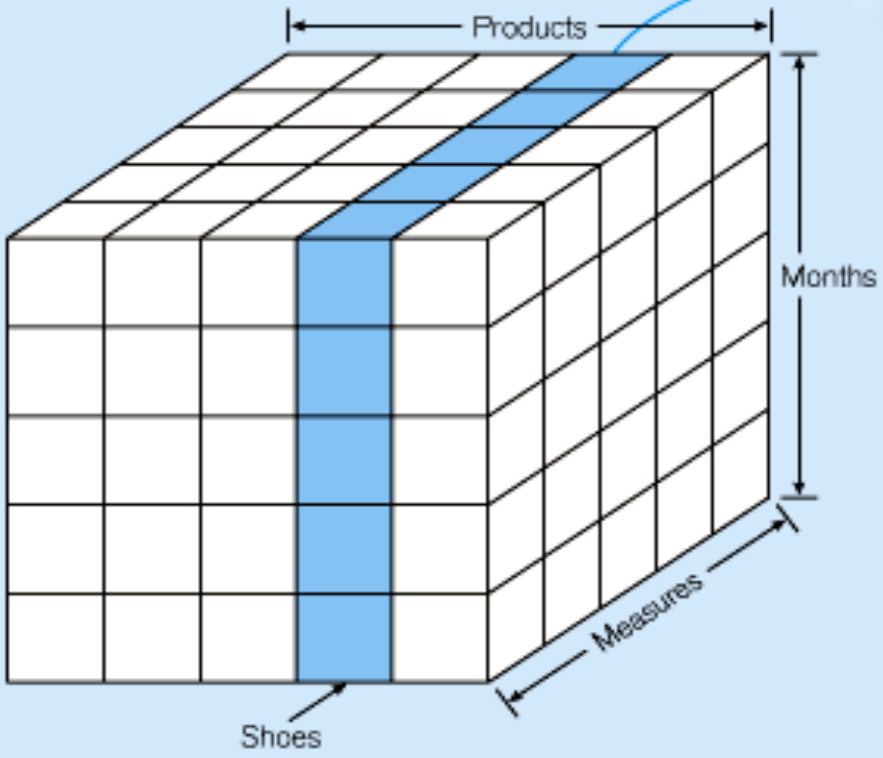
- Direktorski informacijski sistemi – inicialno razviti za potrebe najvišjega vodstva in podporo strateškemu odločanju. Danes podpirajo odločanje na vseh ravneh.





Roll up (drill-up)
Drill down (roll down)
Slice and dice

OLAP orodja



	Measure: Units	Measure: Revenue	Measure: Cost
January	250	1564	1020
February	200	1275	875
March	350	1800	1275
April	400	1935	1500
May	485	2000	1560

Product: Shoes

Enrollment

Required Parameters:

Academic Period(s):

- Spring 1990
- Fall 2004
- Spring 2005
- Session 2006-(
- Fall 2006
- Fall 1993
- Spring 1994

Event(s):

Description	Code
13 First Day of Classes	PRE13
14 Beginning of Classes	FA014
14 Beginning of Classes	REG014
90 Midterm of Classes	AR090
98 Final Academic Period	AR098
98 Final Academic Period	FA098

Begin Analysis

Available Dimensions

Acad. Year | Acad. Period | Event | Acad. Outcome | Award Categ. | Campus | College | Current Time St | Department

Enrolled Ind. | Enroll. Status | Ethnicity Categ. | Housing Ind. | Intended Time S | Nation of Citizen | Program | Prog. Classif. | Residency Ind.

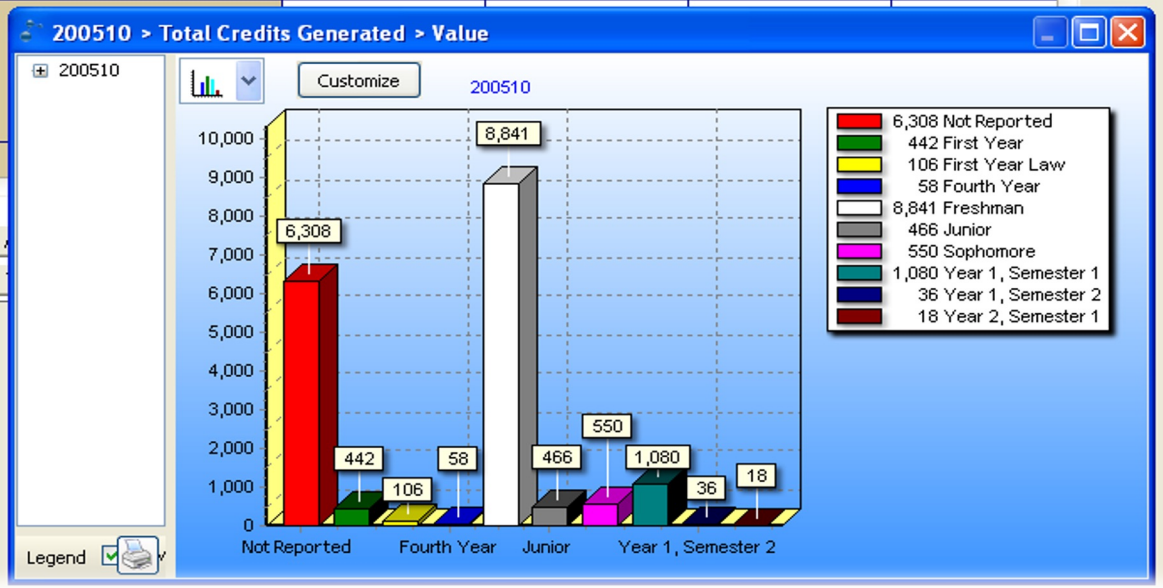
Columns: Academic Perio

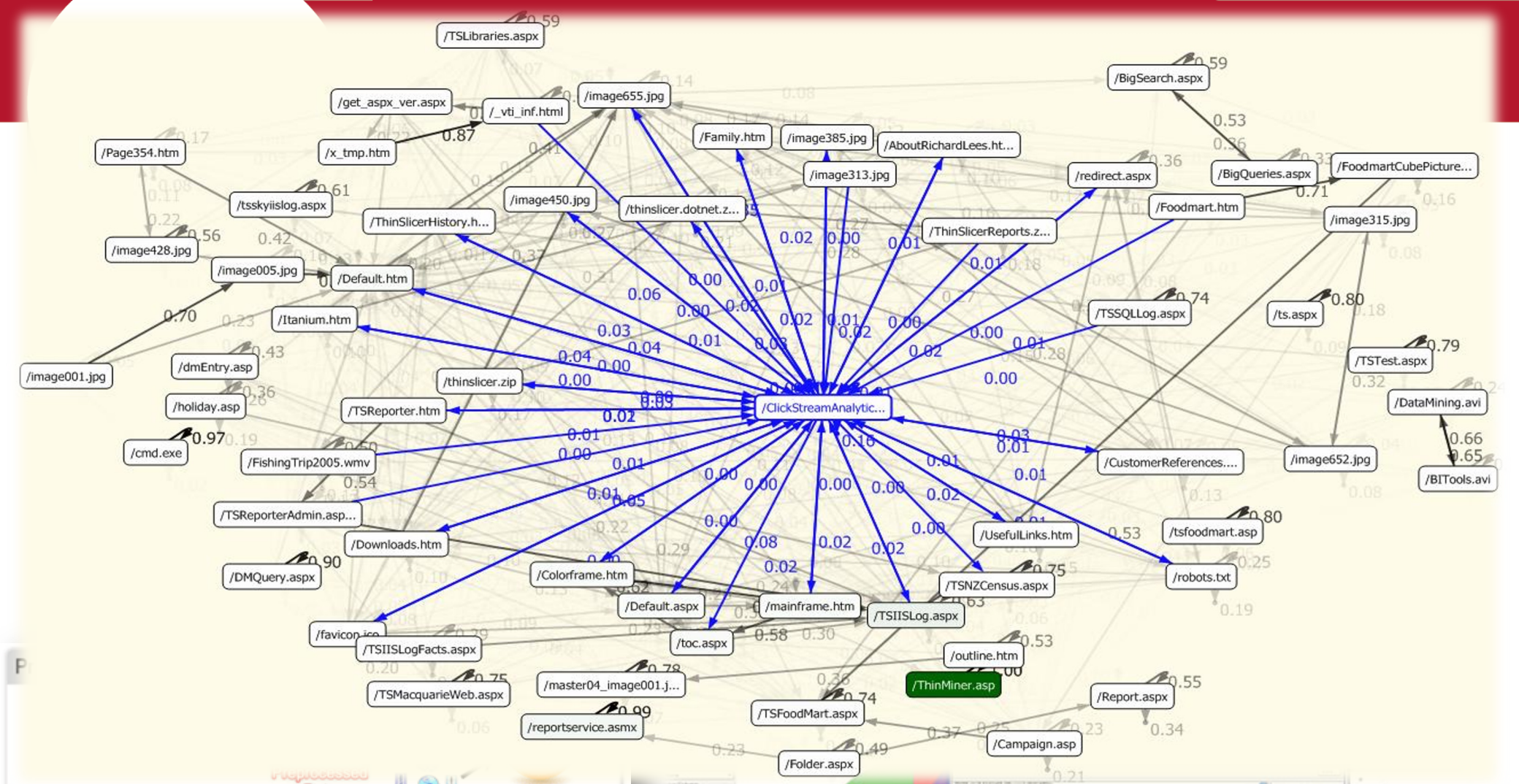
Student Classif. | Gender | Age Range

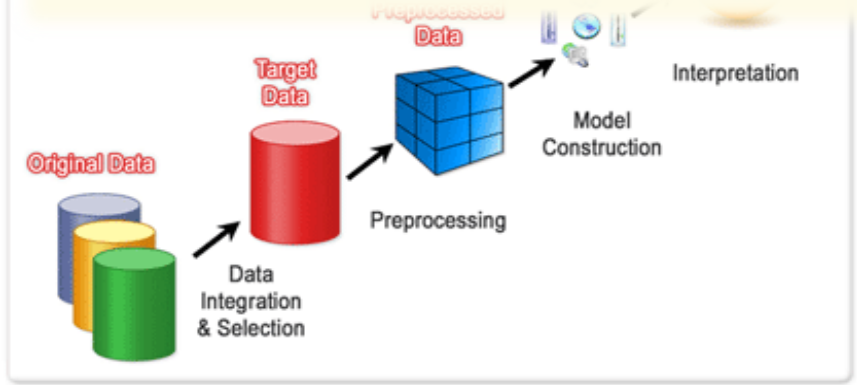
Student Classif.	Gender	Age Range	Academic Period 200510 Registered Count Value	Enrolled Count Value	Total Credits Generated Value	Acad. Period Cred Attempted Value
First Year			42	42	442.00	178.
First Year Law			8	8	106.00	48.
Fourth Year			6	6	58.00	52.
Freshman			938	938	8841.00	8489.
	Not Reported		212	212	1764.00	1628.
	Female					

Total by COLUMNS

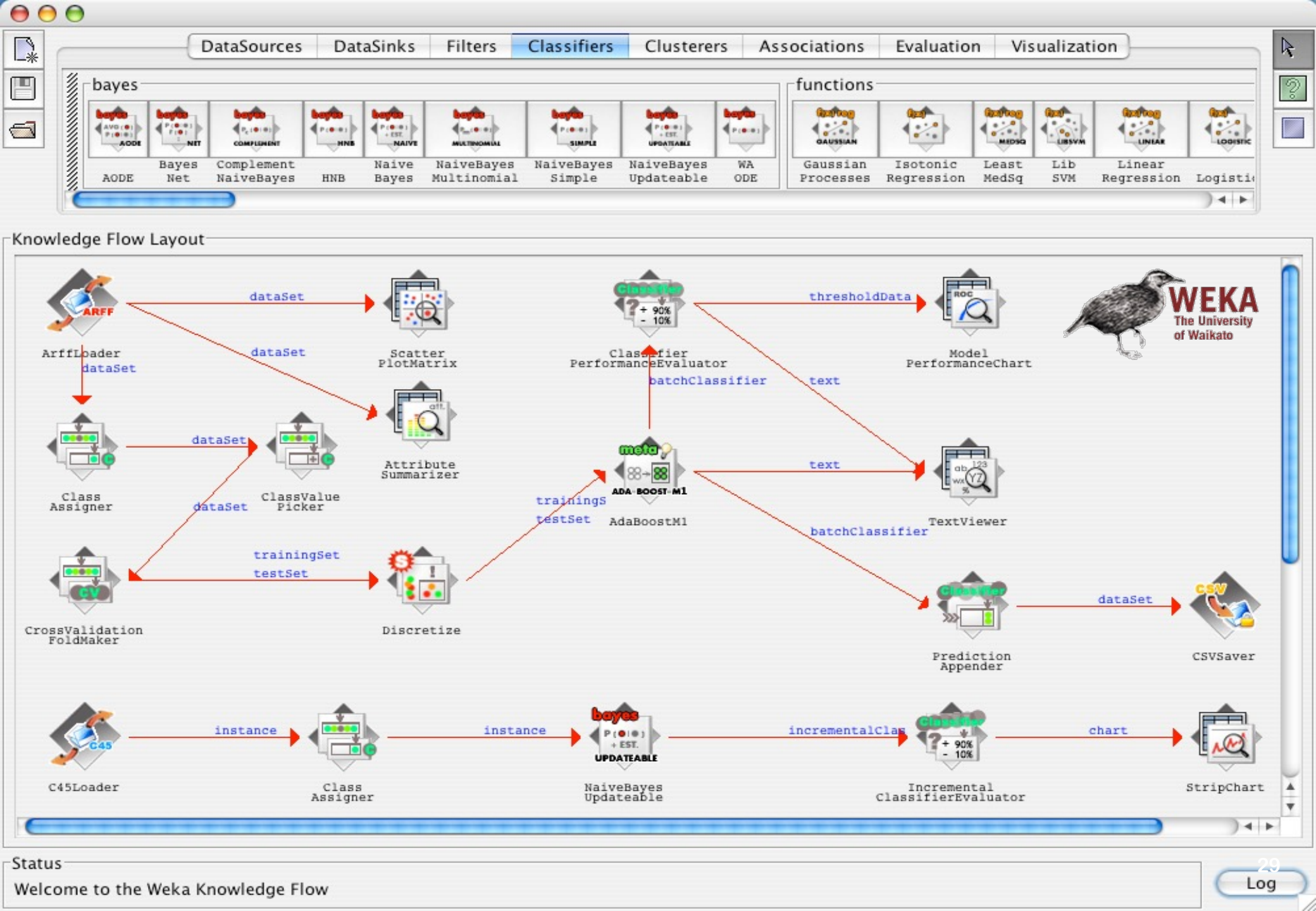
Measures: Registered Cou | Enrolled Count | Total Credits Ge | Avg. Tuition Ch | Fin. Aid Amt. | Avg. Fin. Aid Ar

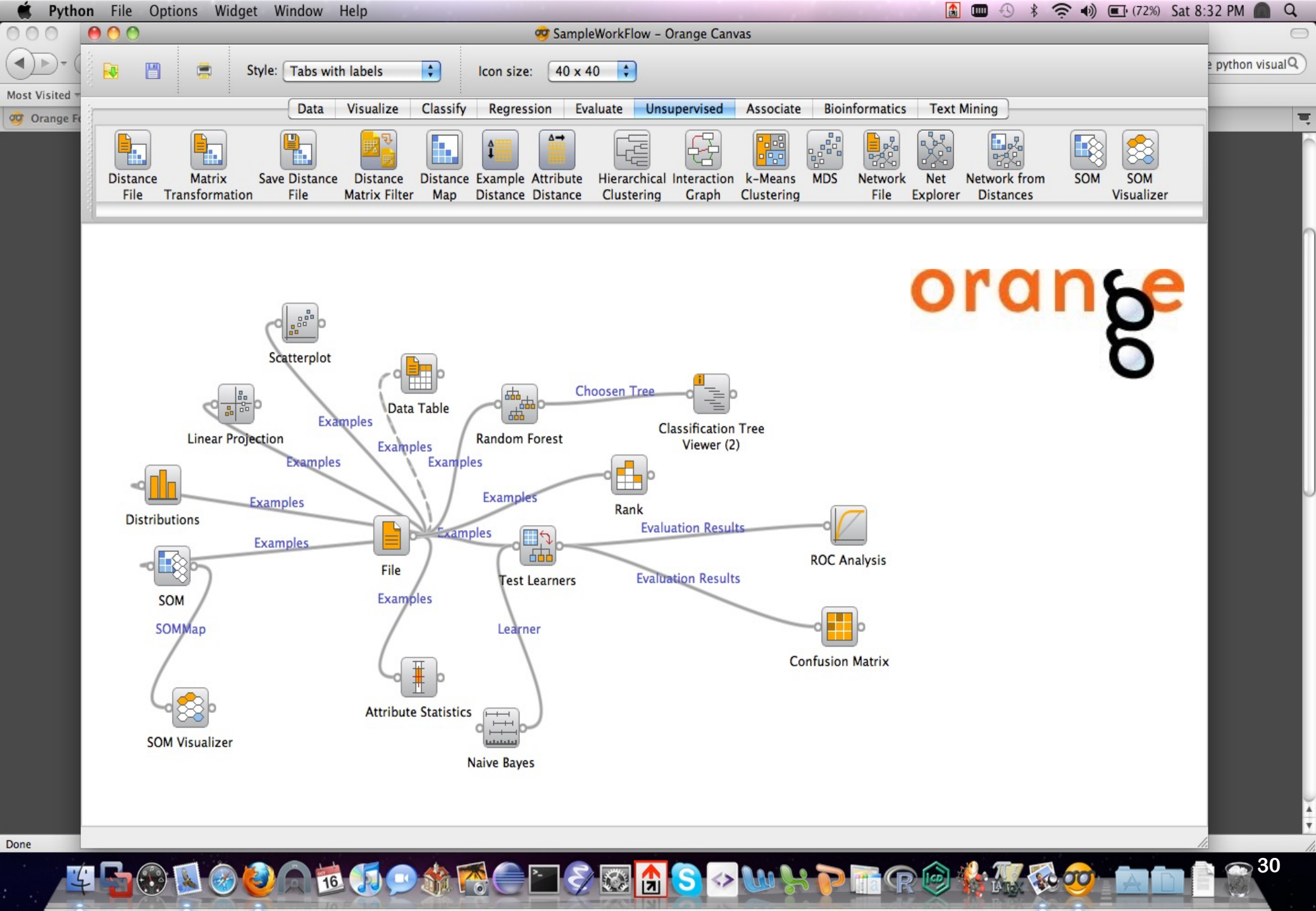



/TSLibraries.aspx
/BigSearch.aspx
/get_aspx_ver.aspx
/_vti_inf.html
/image655.jpg
/x_tmp.htm
/Family.htm
/image385.jpg
/AboutRichardLees.ht...
/Page354.htm
/image313.jpg
/redirect.aspx
/BigQueries.aspx
/FoodmartCubePicture...
/tsskyiislog.aspx
/image450.jpg
/thinslicer.dotnet.z...
/ThinSlicerHistory.h...
/Foodmart.htm
/image315.jpg
/ThinSlicerReports.z...
/image428.jpg
/image005.jpg
/Default.htm
/TSSQLLog.aspx
/ts.aspx
/Itanium.htm
/image001.jpg
/dmEntry.asp
/thinslicer.zip
/TSTest.aspx
/DataMining.avi
/holiday.asp
/TSReporter.htm
/ClickStreamAnalytic...
/cmd.exe
/FishingTrip2005.wmv
/CustomerReferences....
/image652.jpg
/BITools.avi
/TSReporterAdmin.asp...
/Downloads.htm
/UsefulLinks.htm
/tsfoodmart.asp
/DMQuery.aspx
/Colorframe.htm
/TSNZCensus.aspx
/robots.txt
/Default.aspx
/mainframe.htm
/TSIISLog.aspx
/favicon.ico
/TSIISLogFacts.aspx
/toc.aspx
/outline.htm
/TSMacquarieWeb.aspx
/master04_image001.j...
/ThinMiner.asp
/Report.aspx
/TSFoodMart.aspx
/reportservice.asmx
/Campaign.asp
/Folder.aspx


Data
Target Data
Interpretation
Original Data
Model Construction
Preprocessing
Data Integration & Selection




DataSources
DataSinks
Filters
Classifiers
Clusterers
Associations
Evaluation
Visualization
bayes
AODE
Bayes Net
Complement NaiveBayes
HNB
Naive Bayes
NaiveBayes Multinomial
NaiveBayes Simple
NaiveBayes Updateable
WA ODE
functions
Gaussian Processes
Isotonic Regression
Least MedSq
Lib SVM
Linear Regression
Logisti
Knowledge Flow Layout
ArffLoader
dataSet
Scatter PlotMatrix
Attribute Summarizer
Class Assigner
ClassValue Picker
CrossValidation FoldMaker
Discretize
trainingSet
testSet
AdaBoostM1
batchClassifier
Classifier PerformanceEvaluator
thresholdData
Model PerformanceChart
text
TextViewer
Prediction Appender
CSVSaver
C45Loader
instance
Class Assigner
NaiveBayes Updateable
Incremental ClassifierEvaluator
chart
StripChart
WEKA
The University of Waikato
Status
Welcome to the Weka Knowledge Flow
Log
29


Python File Options Widget Window Help
(72%) Sat 8:32 PM
SampleWorkFlow – Orange Canvas
Style: Tabs with labels
Icon size: 40 x 40
Data Visualize Classify Regression Evaluate Unsupervised Associate Bioinformatics Text Mining
Distance File
Matrix Transformation
Save Distance File
Distance Matrix Filter
Distance Map
Example Distance
Attribute Distance
Hierarchical Clustering
Interaction Graph
k-Means Clustering
MDS
Network File
Net Explorer
Network from Distances
SOM
SOM Visualizer
orange
Scatterplot
Data Table
Linear Projection
Random Forest
Choosen Tree
Classification Tree Viewer (2)
Examples
Distributions
Rank
File
Test Learners
Evaluation Results
ROC Analysis
SOM
SOMMap
Learner
Confusion Matrix
Attribute Statistics
SOM Visualizer
Naive Bayes
Done

Načrtovanje podatkovnega skladišča...

- Pred začetkom izdelave podatkovnega skladišča, moramo razčistiti:
  - Katere so najpomembnejše zahteve uporabnikov?
  - Katere podatke najprej uporabiti?
  - Ali s DW pokriti celoten poslovni sistem ali le določeno poslovno področje?
  - Če začnemo z manjšim obsegom, ali naj bo infrastruktura, na kateri se bo razvilo skladišče, pripravljena tudi za končno podatkovno skladišče?
  - …



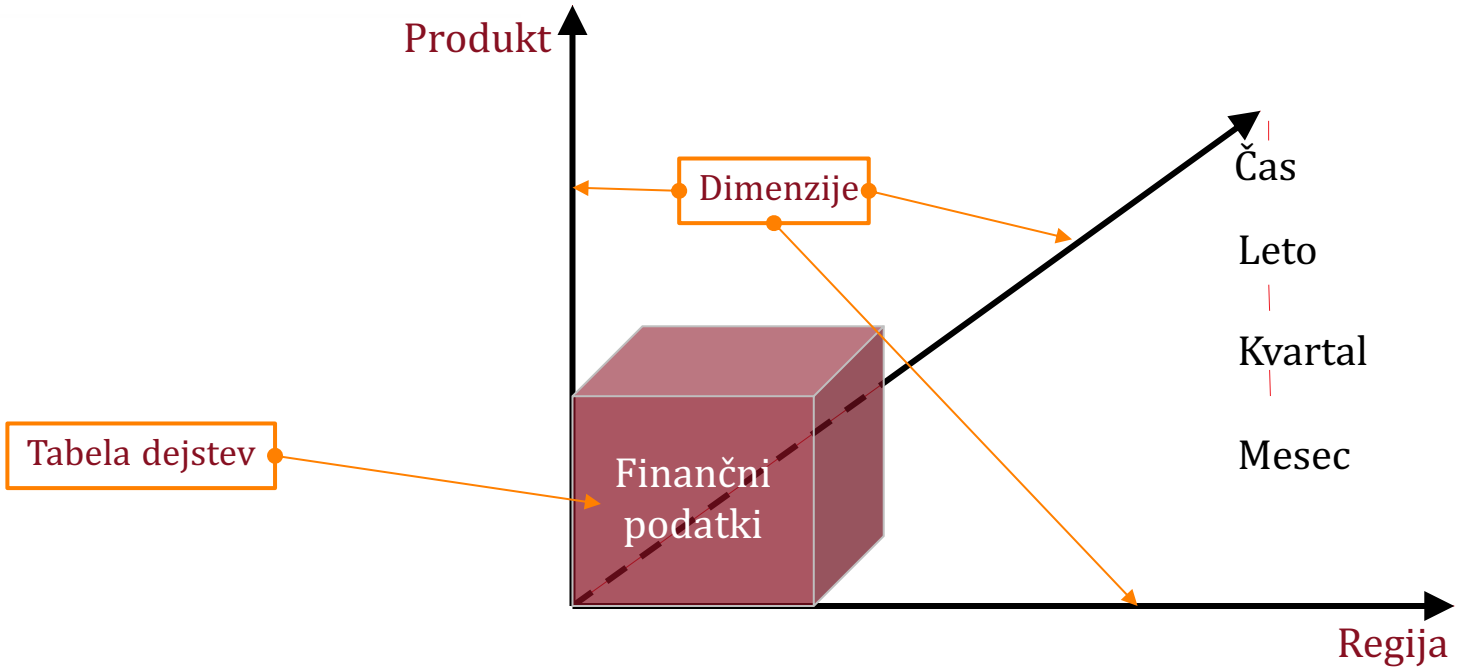
Načrtovanje podatkovnega skladišča

- Ključni koraki:
  - Izbira področja/vsebine podatkovnega skladišča
  - Določitev zrnatosti podatkov v tabeli dejstev
  - Izbira dimenzij
  - Identifikacija dejstev
  - Določitev atributov dimenzij

- Pomembno: pri zajemu zahtev se pogovorimo z:
  - odgovornimi za posamezna poslovna izbrana poslovna področja (npr. marketing, prodaja, finance, vodstvo…),
  - administratorji podatkovnih virov – kateri podatki obstajajo?

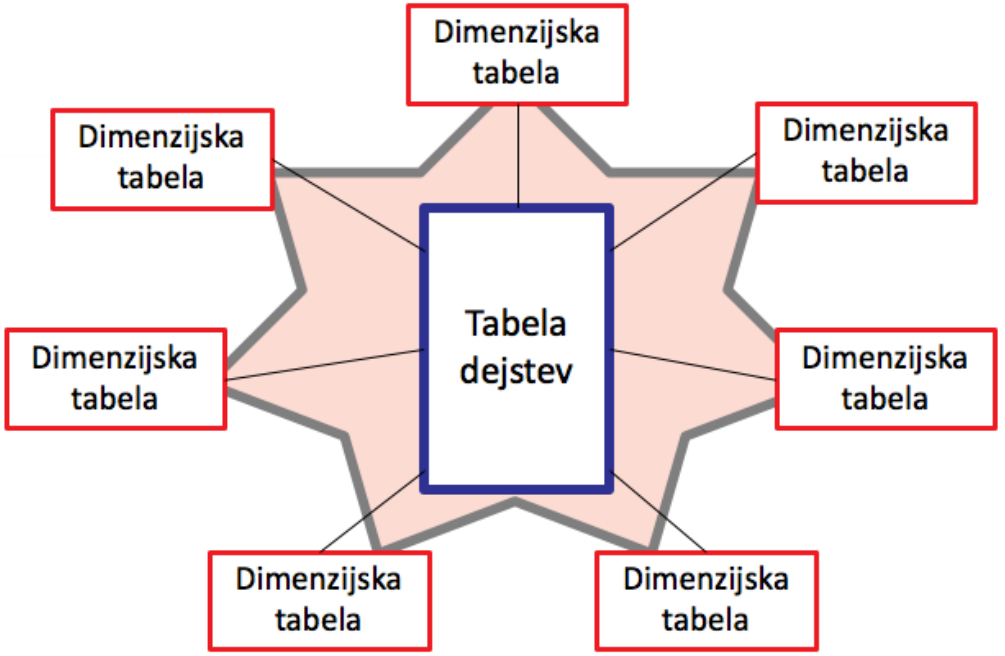
Modeliranje dimenzij...

- Logično načrtovanje podatkov na način, da bodo hitro dostopni…



Modeliranje dimenzij...

- Vsak model dimenzij vsebuje:
  - eno tabelo, ki vsebuje kompozitni primarni ključ – tabela dejstev in
  - množico manjših dimenzijskih tabel.
- Tabela dejstev skupaj z dimenzijskimi tabelamo tvori ti. zvezdno shemo.



Modeliranje dimenzij...

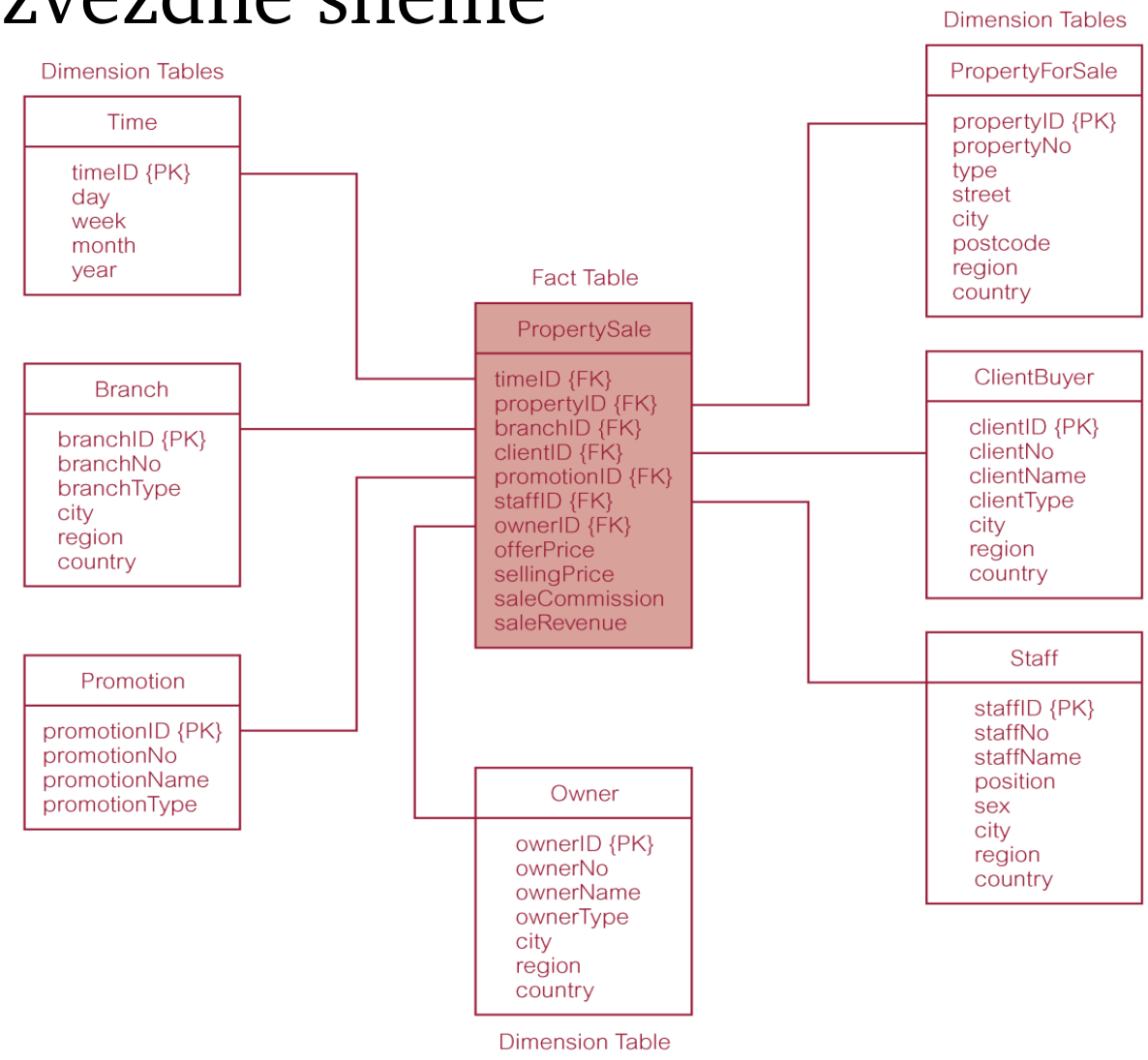
- Vsaka dimenzijska tabela ima enostaven ključ – komponenta kompozitnega ključa iz tabele dejstev
- Dejstva se generirajo na podlagi dogodkov, ki so se zgodila v preteklosti – spremembe malo verjetne.
- Primarni ključ tabele dejstev je sestavljen iz dveh ali več tujih ključev posameznih dimenzijskih tabel.

Uporaba umetnih ključev

- Vse naravne primarne ključe se nadomesti s surogati – umetni ključi.
- Surogati omogočajo neodvisnost med podatki podatkovnega skladišča in posameznimi viri.



Primer zvezdne sheme

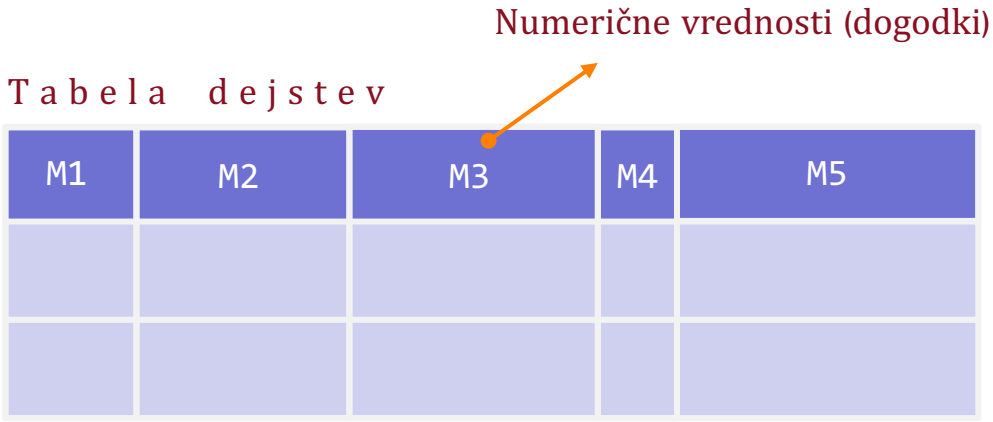


Primer skladišča za UL…

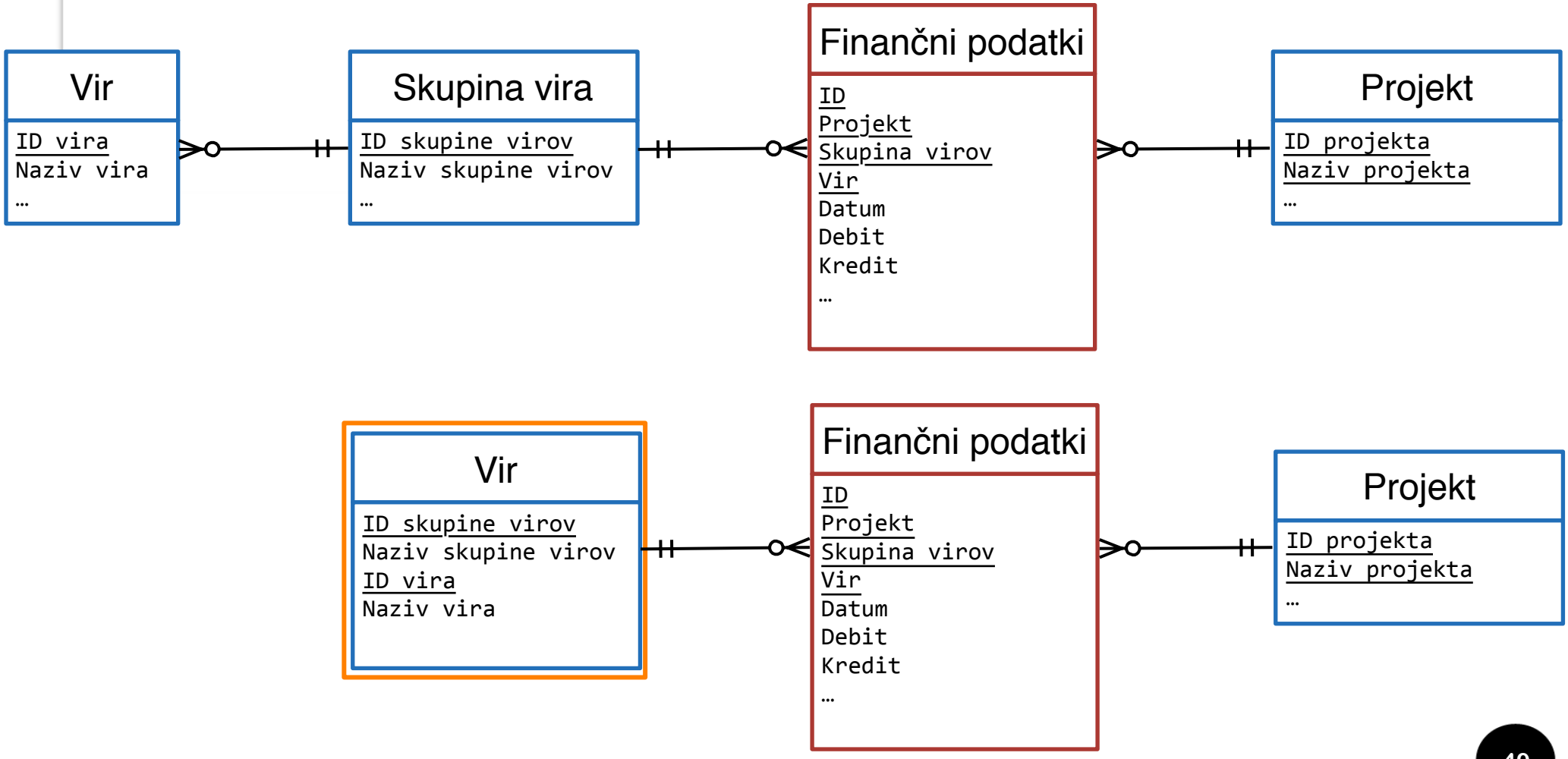
- Na UL želimo narediti skladišče za področje izpitne evidence.
  1. Kakšna je povp. ocena študentov 1. letnika po različnih fakultetah in kako je ta ocena v korelaciji z oceno na maturi/zaključnem izpitu?
  2. Pri katerih predmetih so v povprečju najnižje ocene?
  3. Kakšna je uspešnost študentov pri polaganju izpitov v posameznih izpitnih obdobjih?
  4. Pri katerih predmetih posameznih fakultet je napredovanje najpočasnejše?
  5. Kateri študijski program so najzahtevnejši (glede na vložene ure in končno oceno in čas, ki je potekel od prvega vpisa do opravljanja izpita)?
  6. Katera študijsko leto v zgodovini UL je bilo najmanj uspešno glede na število in povprečno oceno opravljenih izpitov?

Podatki v tabelah dimenzij in dejstev

- Podatki v tabelah dejstev
  - referenčni podatki – namenjeni branju,
  - skozi čas se ne spreminjajo (le dodajajo),
  - numerične vrednosti.
  - Večina podatkov v podatkovnem skladišču se hrani v tabeli dejstev → lahko ekstremno obsežne tabele.
- Podatki v tabelah dimenzij
  - navadno tekstovne vrednosti
  - filter



Denormalizirane dimenzije



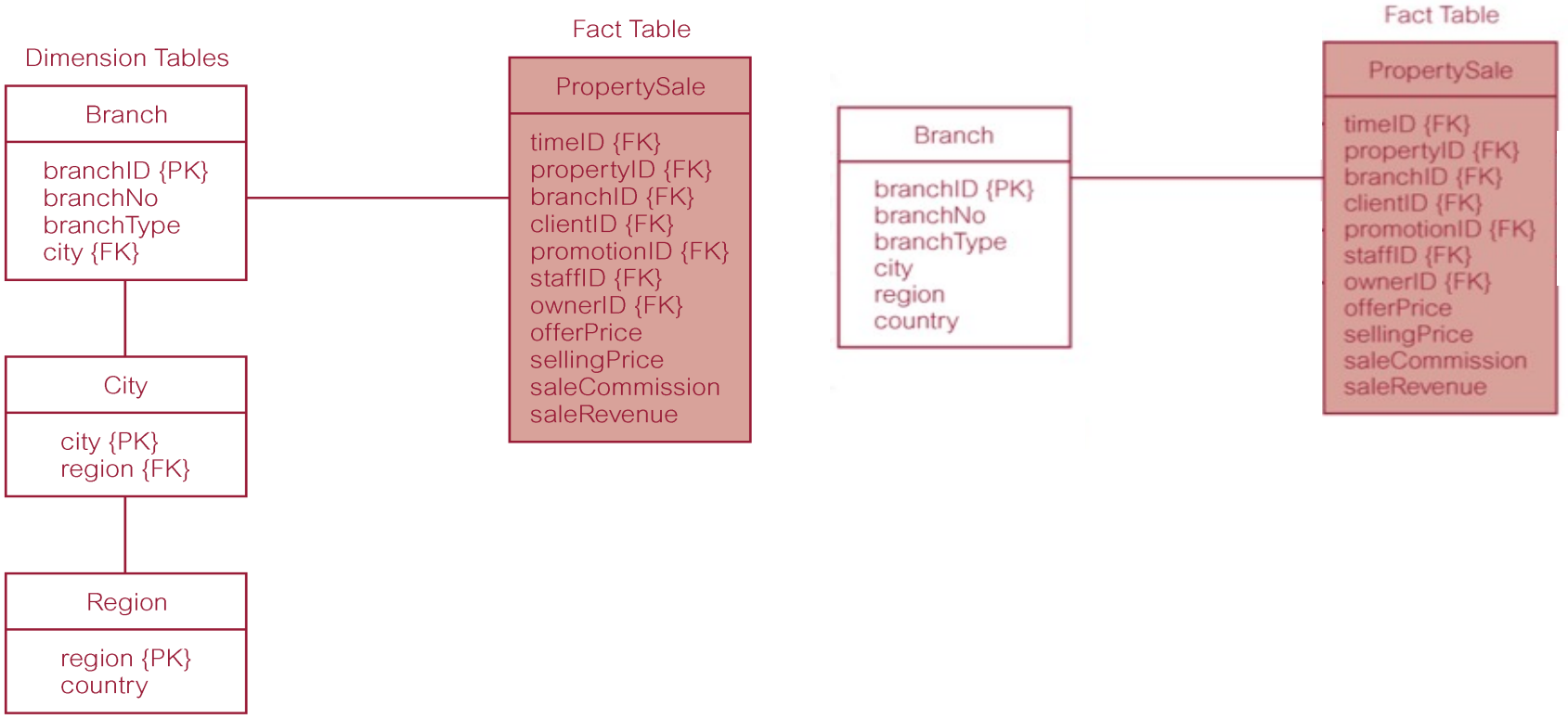
Snežna shema

- Snežena shema (snowflake) je shema brez denormaliziranih podatkov.

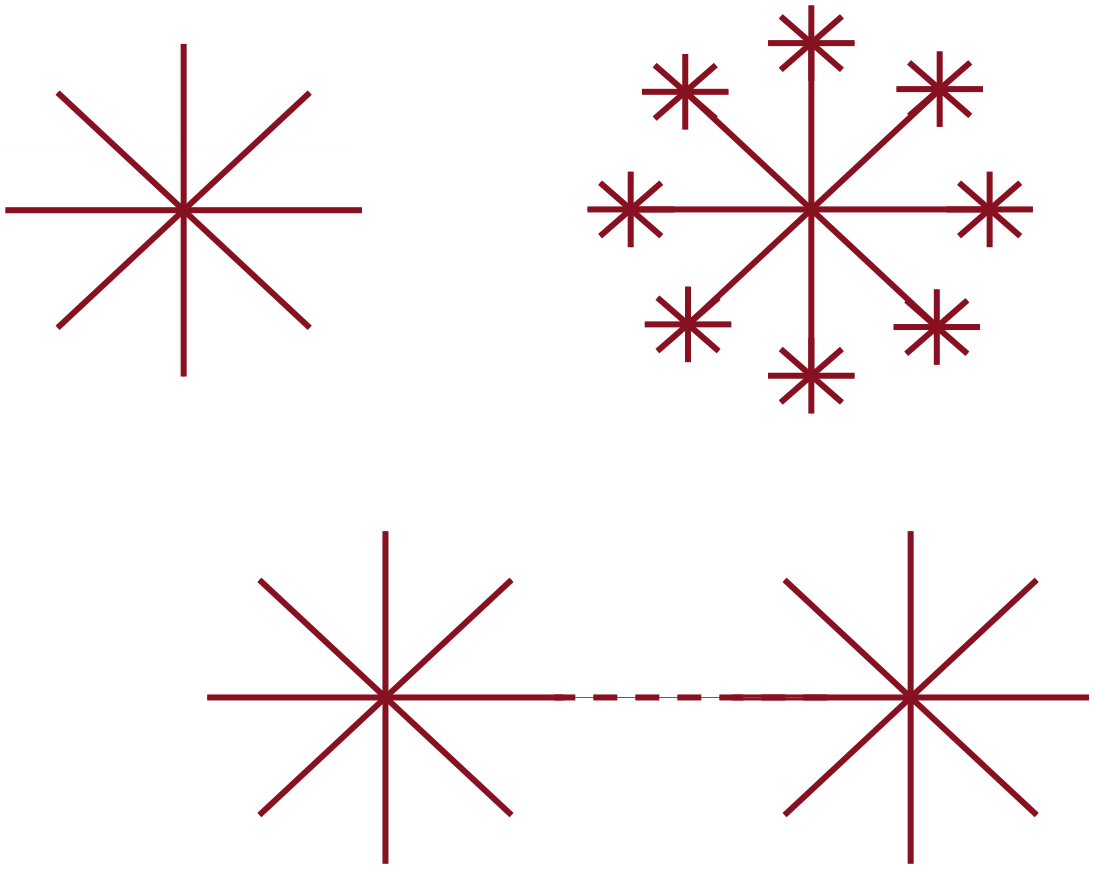
- Zvezdna snežna shema (starflake) –
  - hibridna struktura, ki vsebuje tako zvezdno (denormalizirano) kot sneženo (normalizirano) shemo.
  - shema omogoča predstavitev dimenzij na oba načina – ustreza različnim performančnim zahtevam poizvedb.

Modeliranje dimenzij...

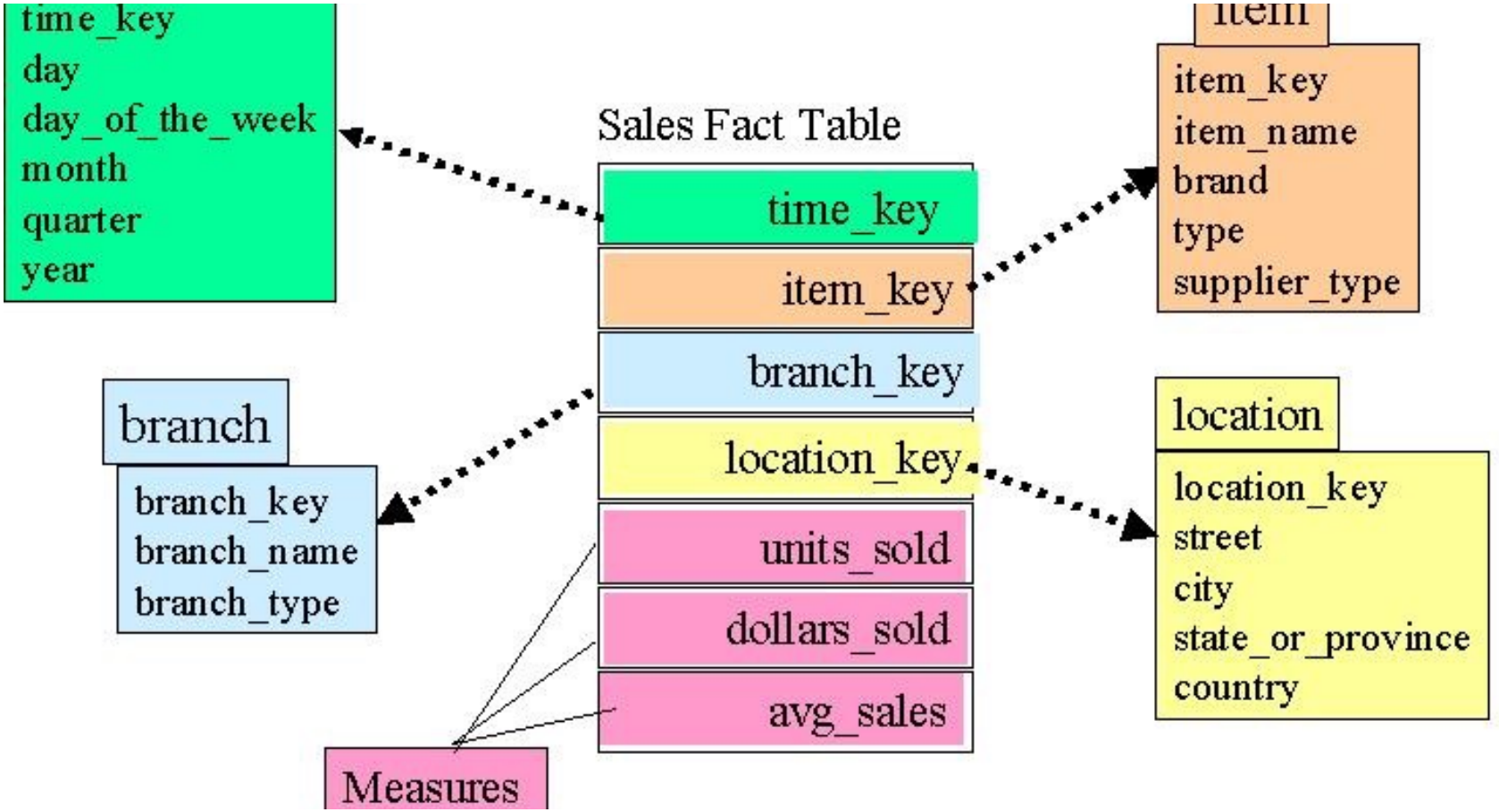
- Normalizirana vs denormalizirana varianta dimenzije Branch:



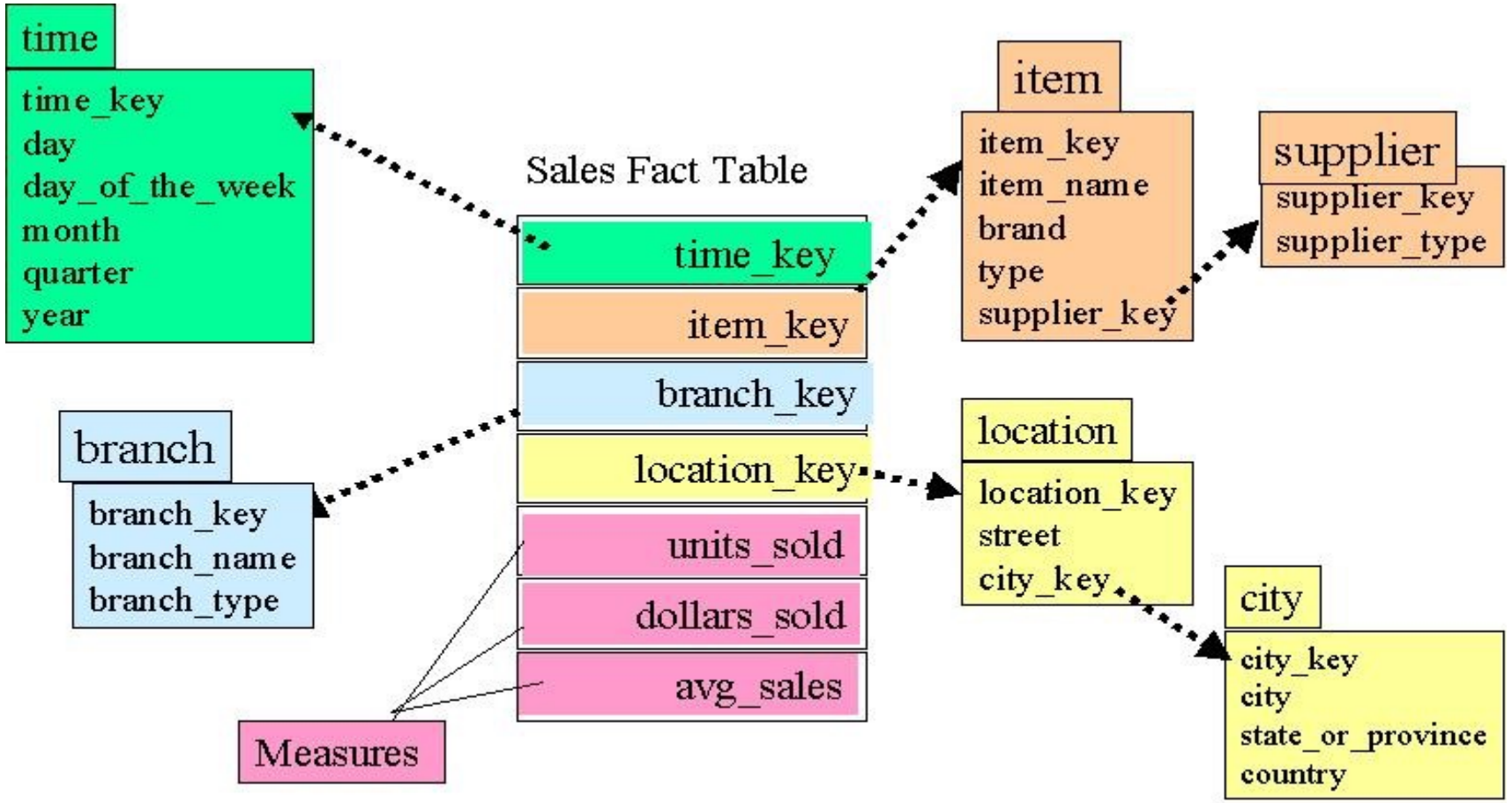
Zvezdna, snežna, sozvezdna shema...



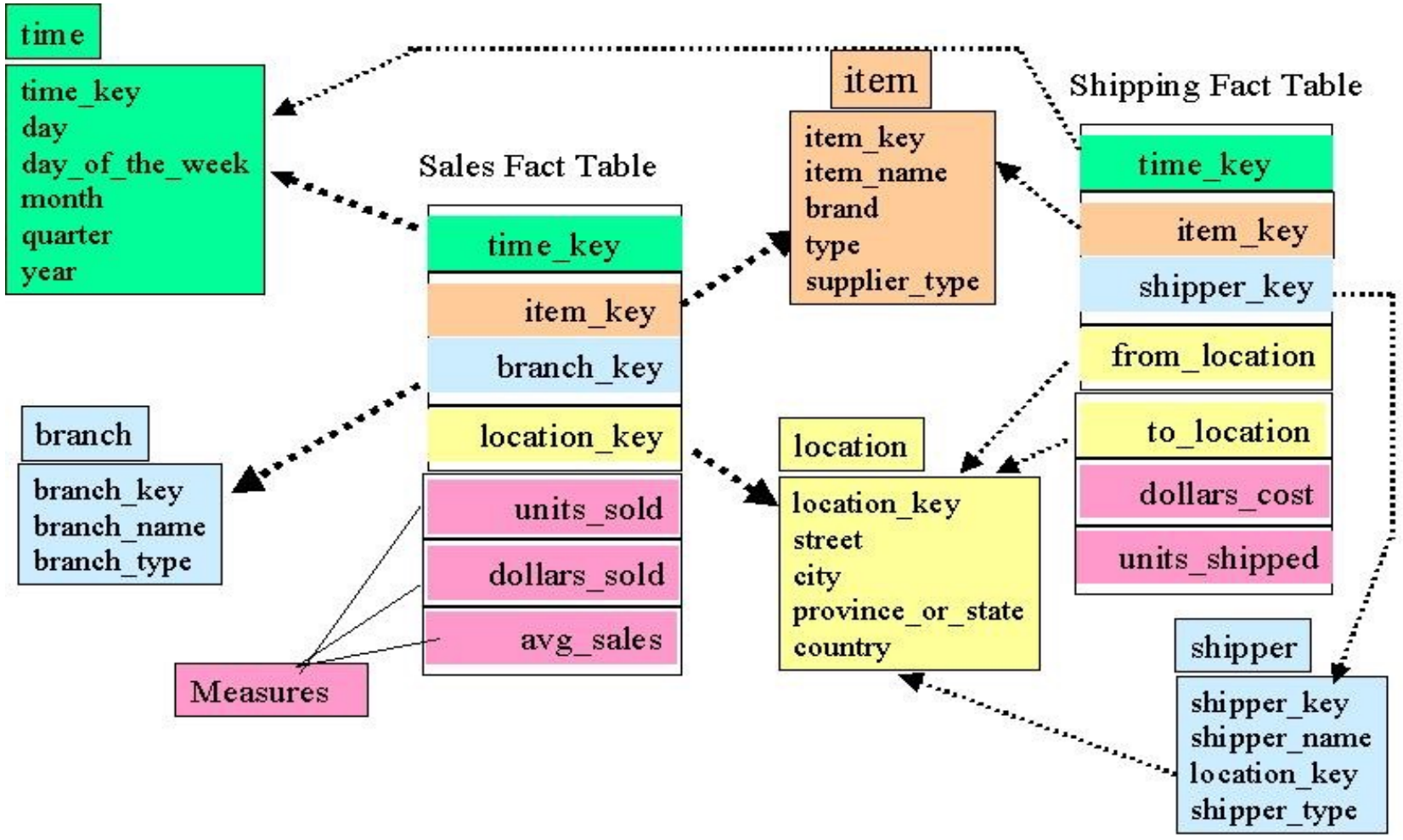
Primer zvezdne sheme

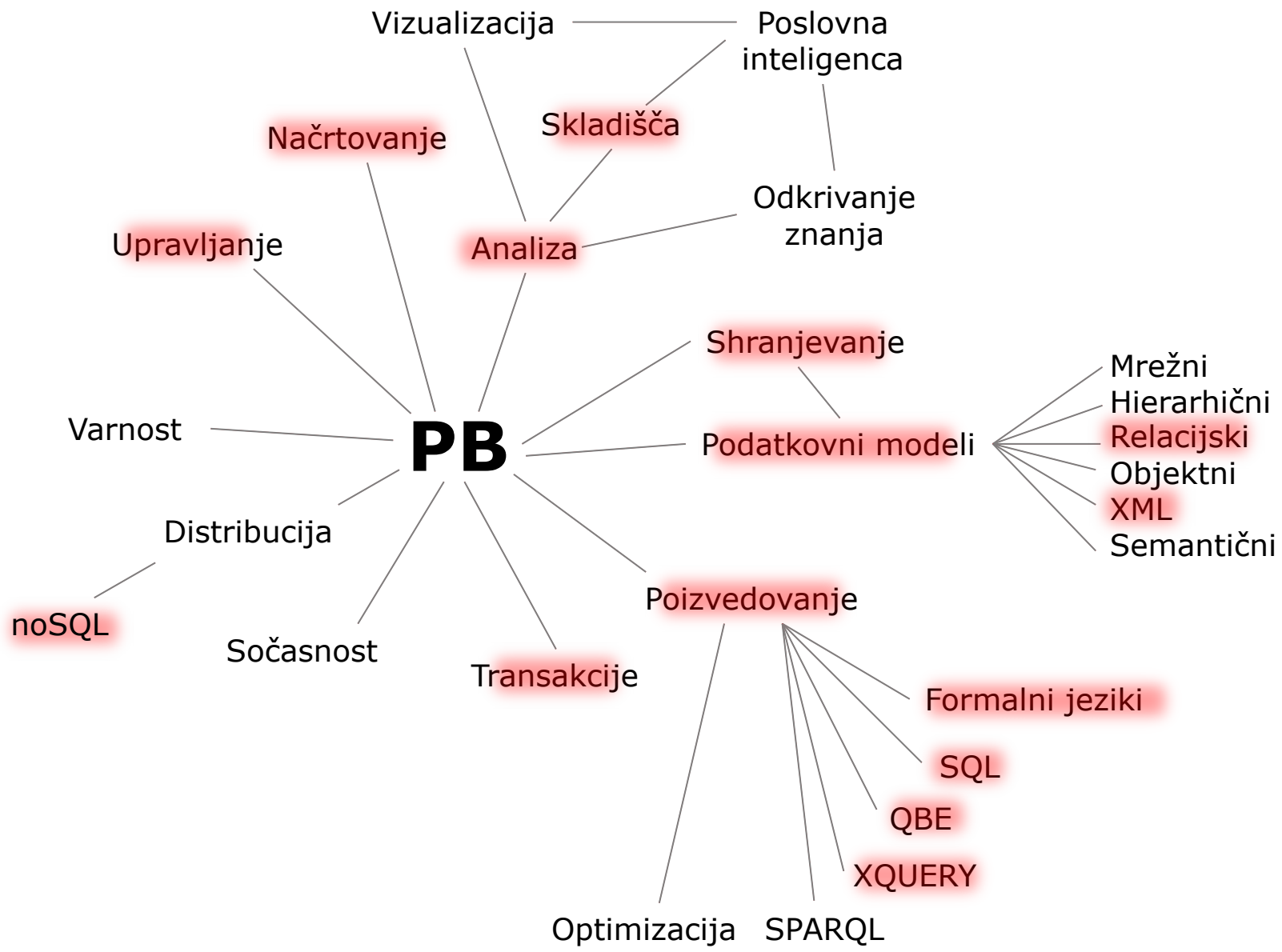


Primer snežne sheme



Primer sozvezdna shema




PB
Vizualizacija
Poslovna inteligenca
Skladišča
Načrtovanje
Odkrivanje znanja
Upravljanje
Analiza
Shranjevanje
Varnost
Podatkovni modeli
Mrežni
Hierarhični
Relacijski
Objektni
XML
Semantični
Distribucija
noSQL
Poizvedovanje
Sočasnost
Transakcije
Formalni jeziki
SQL
QBE
XQUERY
Optimizacija
SPARQL

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Predmet:
Osnove podatkovnih baz

Modul:
Uvod v XML

Gradivo:
v.2016

5.1.
2017

Vsebina

- Kaj je XML
- Primerjava s HTML in prednosti XML
- Sintaksa XML
- XML drevesna struktura
- Ravni skladnosti XML dokumentov
- XML shema
- XML orodja
- XML podatkovne baze
- Podpora XML v MySQL

Kaj je XML?...

- Razširljivi označevalni jezik (eXtensible Markup Language)
- Podoben jeziku HTML

Izsek primera HTML datoteke

```
<html>

<head>
    <title>My First Web Page</title>
</head>

<body>
    <h1>My First Web Page</h1>
    <p><b>Hello World Wide Web!</b></p>
    <p><i>Hello World Wide Web!</i></p>
    <p><u>Hello World Wide Web!</u></p>
    <p>This is my first web page.</p>
    <p>HTML tags can give <b><i>various</i></b>
    <u>looks and format</u> to the content of this web page.</p>
</body>

</html>
```

Kaj je XML?

- Namenjen prenosu, ne prikazu!
- Značke (tags) niso predpisane vnaprej.
- Je samopojasnjujoč.
- Priporočilo organizacije W3C (http://www.w3.org/).

Izsek primera XML datoteke

```
<person id="Big.Boss">
    <name>
        <family>Boss</family>
        <given>Big</given>
    </name>
    <email>chief@oxygenxml.com</email>
    <link subordinates="one.worker two.worker three.worker four.worker five.worker"/>
</person>
<person id="one.worker">
    <name>
        <family>Worker</family>
        <given>One</given>
    </name>
    <email>one@oxygenxml.com</email>
    <link manager="Big.Boss"/>
</person>
```

Razlike med HTML in XML

HTML	XML
vnaprej določen nabor značk	značke definiramo sami
značke namenjene določanju videza dokumenta	značke opisujejo pomen dokumenta
značke lahko izpuščamo	vse značke morajo biti prisotne
strani pogosto nepravilno zapisane – npr. napačne značke	dokumenti morajo biti „ustrezni“ – dobra definiranost

Prednosti XML pred HTML…

- HTML stran vsebuje podatke, ki so zapisani tako, da so razumljivi predvsem HTML prikazovalnikom (brskalniki).

<title> **Google ustavil digitalizacijo starih časnikov (Heise, 20. maj 2011)** </title>

<h1> Omrežja / internet </h1>

 Google je končal digitalizacijo starih časnikov, ki predstavljajo odsev dogajanja po svetu v zadnjih 250 letih. Projekt se je začel leta 2006 z indeksiranjem starih izvodov The New York Timesa, The Wall Street Journala in drugih ameriških časnikov, bistveno pa so ga razširili leta 2008 s skeniranjem mikrofilmov in digitalizacijo vsebine.



Googlov cilj je bil veličasten: želi so najti milijardo časopisnih strani, približno kolikor naj bi jih bilo v zgodovini napisanih, in jih zbrati na enem mestu v obliki, ki omogoča enostavno listanje. Do danes so zbrali več kot 2000 časnikov, ki so predstavljeni na spletu, med katerimi najdemo vse pomembne zgodbe zadnjih dveh stoletij, recimo pristanek na Mesecu.

Projekt je sedaj prekinjen, tako da Google ne bo več dodajal novih vsebin. Digitalizirana dela, vseh skupaj je 60 milijonov strani, bodo ostala prosto dostopna na internetu. Novih funkcionalnosti in vsebin ne bo, tako da bodo skenirana in neindeksirana dela žal ostala v vicah. Namesto tega se bo sedaj posvetil drugim nalogam, kot je recimo Google One Pass za prodajo vsebin založnikov neposredno s svojih strani prek Googla. Predčasni prekinitvi navkljub je Google ustvaril velik digitalni arhiv časnikov, kjer so vsi pomembnejši časniki z večjo naklado.

Matej Huš

HTML ne daje vsebinskih informacij:

- Kdo je avtor zgodbe?
- V katero kategorijo je umeščena?
- Kdaj je bila objavljena?
- …

- Programska obdelava HTML strani je zahtevna ker manjka struktura podatkov.

Prednosti XML pred HTML

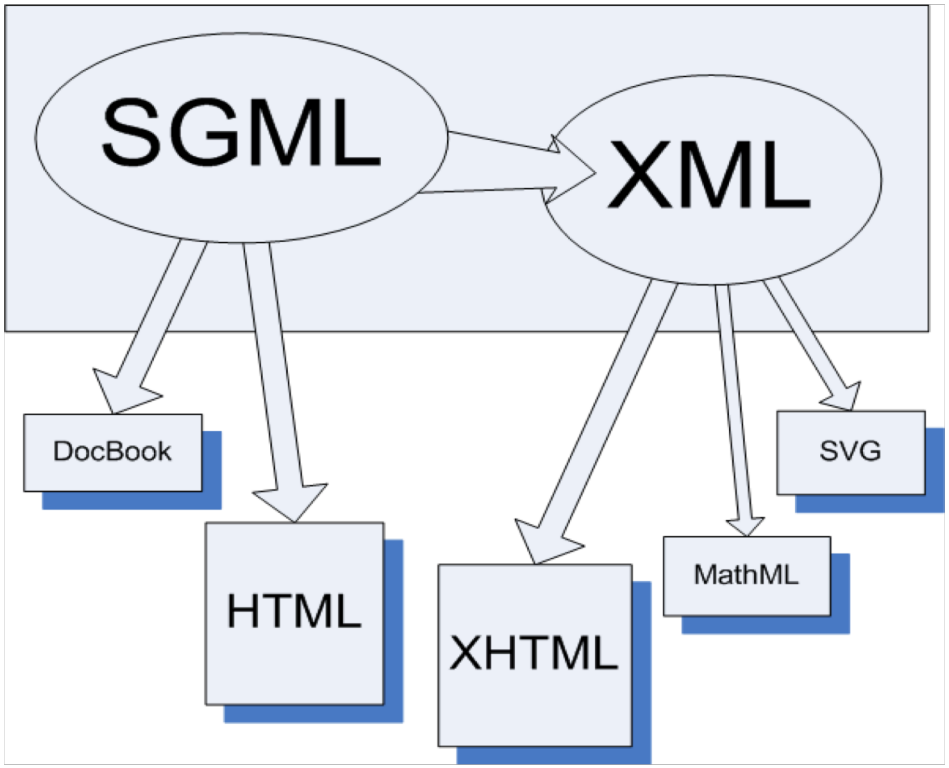
```
<B>John Q Public</B>
<P>
john.q.public.1@gsfc.nasa.gov<BR>
phone: 301-286-aaaa<BR>
fax: 301-286-bbbb<BR>
Bldg. 23, Rm. 999<BR>
NASA<BR>
Goddard Space Flight Center<BR>
588.0<BR>
Greenbelt, MD 20221<BR>
```



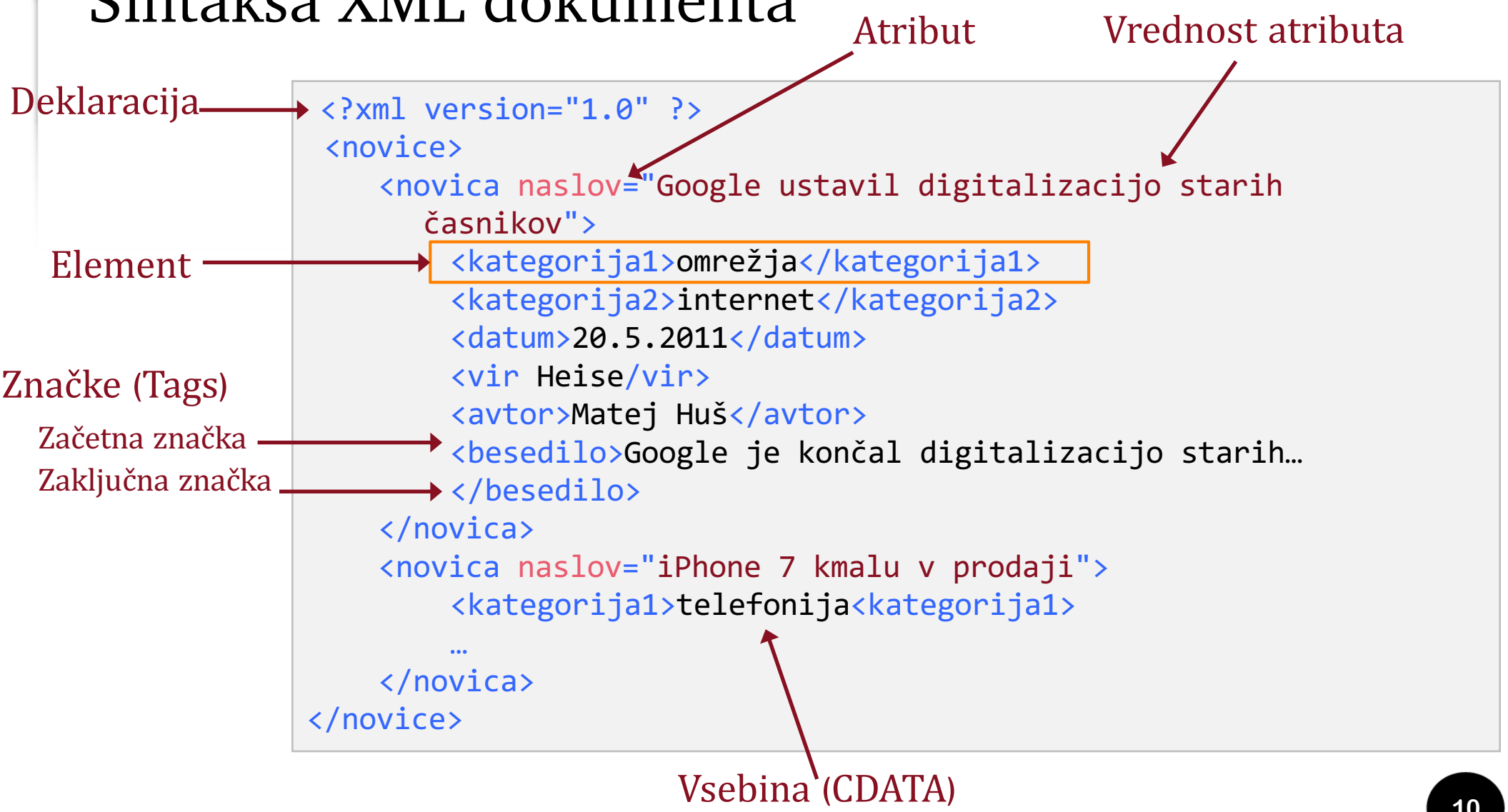

```
<EMPOLYEE>
  <NAME>
        <FIRST>John</FIRST>
        <MIDDLE>Q</MIDDLE>
        <LAST>Public</LAST>
  </NAME>
  <EMAIL>john.q.public.1@gsfc.nasa.gov</EMAIL>
  <PHONE>301-286-aaaa</PHONE>
  <FAX>301-286-bbbb</FAX>
  <LOCATION>
        <BUILDING>Bldg. 23</BUILDING>
        <ROOM>999</ROOM>
  </LOCATION>
  <ADDRESS>
        <ORG>NASA</ORG>
        <CENTER>Goddard Space Flight Center</CENTER>
        <MAILSTOP>588.0</MAILSTOP>
        <CITY>Greenbelt</CITY>
        <STATE>MD</STATE>
        <ZIP>20221</ZIP>
  </ADDRESS>
</EMPLOYEE>
```

Uveljavljena tehnologija

- XML je podmnožica standarda SGML (Standardized Generalized Markup Language), ki je bil definiran že leta 1969

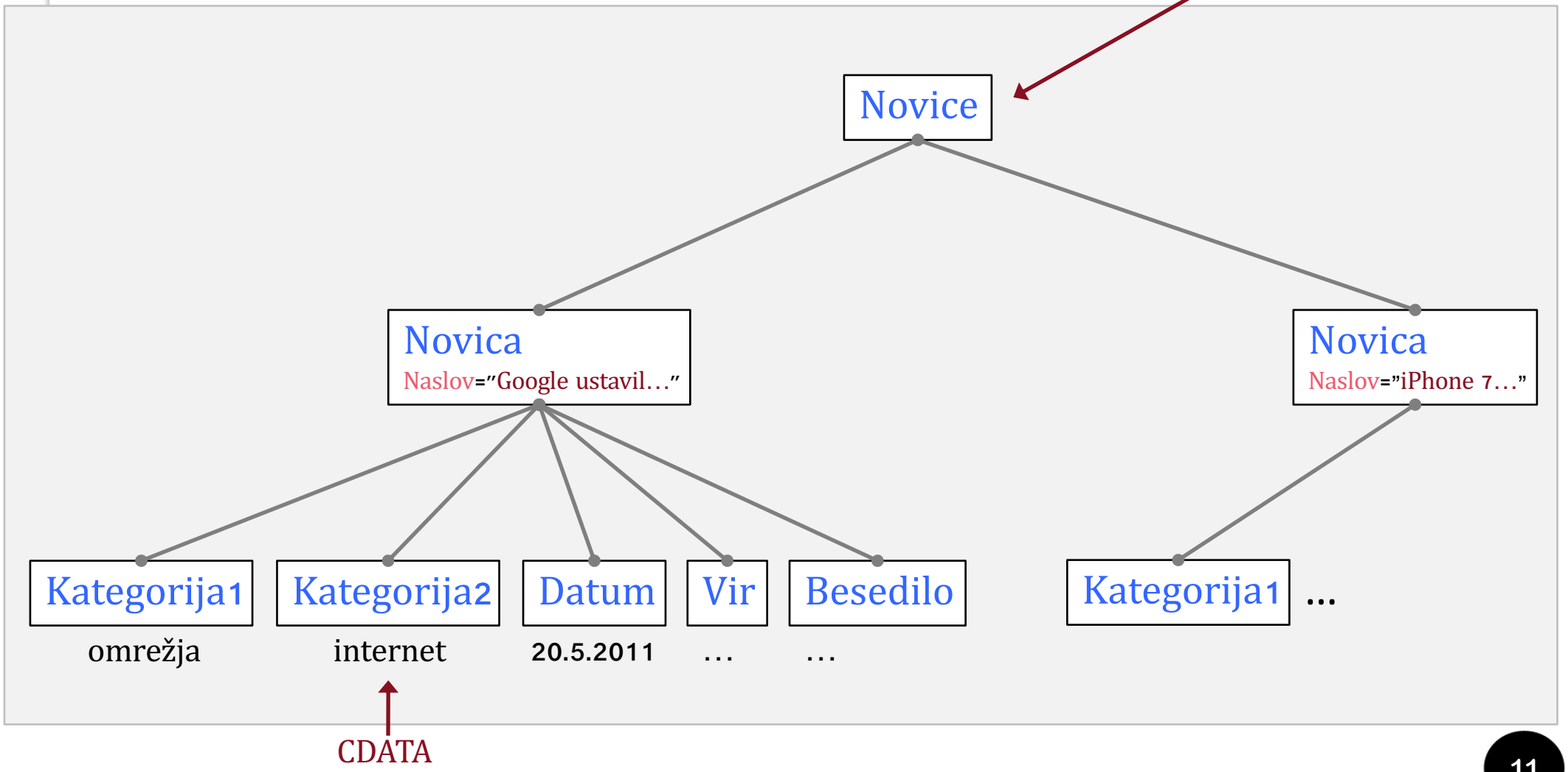


Sintaksa XML dokumenta




```
<?xml version="1.0" ?>
<novice>
   <novica naslov="Google ustavil digitalizacijo starih
      časnikov">
      <kategorija1>omrežja</kategorija1>
      <kategorija2>internet</kategorija2>
      <datum>20.5.2011</datum>
      <vir Heise/vir>
      <avtor>Matej Huš</avtor>
      <besedilo>Google je končal digitalizacijo starih…
      </besedilo>
   </novica>
   <novica naslov="iPhone 7 kmalu v prodaji">
      <kategorija1>telefonija<kategorija1>
      …
   </novica>
</novice>
```

XML dokument in drevesna struktura



Ravni skladnosti XML dokumentov

- Pravilno strukturiran (well formed): zadošča vsem WC3 sintaktičnim pravilom za XML
  - Pravila za poimenovanje, gnezdenje, označevanje atributov…
  - Dobra definiranost je za XML dokumente obvezna

- Veljaven (valid) dokument je dokument, ki je skladen s shemo
  - Formalnost je za XML dokument opcijska;
  - Lahko preverjamo z XML shemo;

Pravila dobro definiranih XML dokumentov...

- XML standard zahteva, da se v dokumentih upoštevajo naslednja pravila:
  - Obstajati mora en element, ki vsebuje vse ostale
  - Značke morajo biti uravnotežene

```
<BOOK>...</BOOK>
<BOOK />
```

  - Gnezdenje značk mora biti izvedeno pravilno.

```
<BOOK> <LINE> to je pravilno </LINE> </BOOK>
<LINE> <BOOK> to </LINE> gotovo ni </BOOK> pravilno
```

  - Tekst značk je občutljiv na velike in male črke

```
<P> različna začetna in končna značka – XML tega ne dopušča </p>
```

Pravila dobro definiranih XML dokumentov

- Atributi v značkah (tags) morajo biti znotraj narekovajev.

```
< ITEM Category="Home and Garden" Name="hoe-matic t500">
```

- Komentiranje je dovoljeno.

```
<!-- They are done just as in HTML… -->
```

- Dokument se mora začeti z ustreznim nizom (XML deklaracija).

```
<?xml version='1.0' ?>
```

- Posebni znaki se morajo pričeti z ubežnim znakom (&)

```
<formula> x &lt; y+2x </formula>
<cd title="&quot; music &quot;">
```

XML shema

- Shema je ločen dokument, ki definira elemente, atribute in strukturo XML dokumenta
- Shemo določimo tako, da
  1. definiramo slovar in
  2. določimo postavitev ter število elementov in atributov v označevalnem jeziku
- Shema definira točno določen razred dokumentov.

XML Shema - Primer

Author.xsd

Shema Author.XSD definira strukturo elementa Author

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Author">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FirstName" type="xs:string" />
        <xs:element name="LastName" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

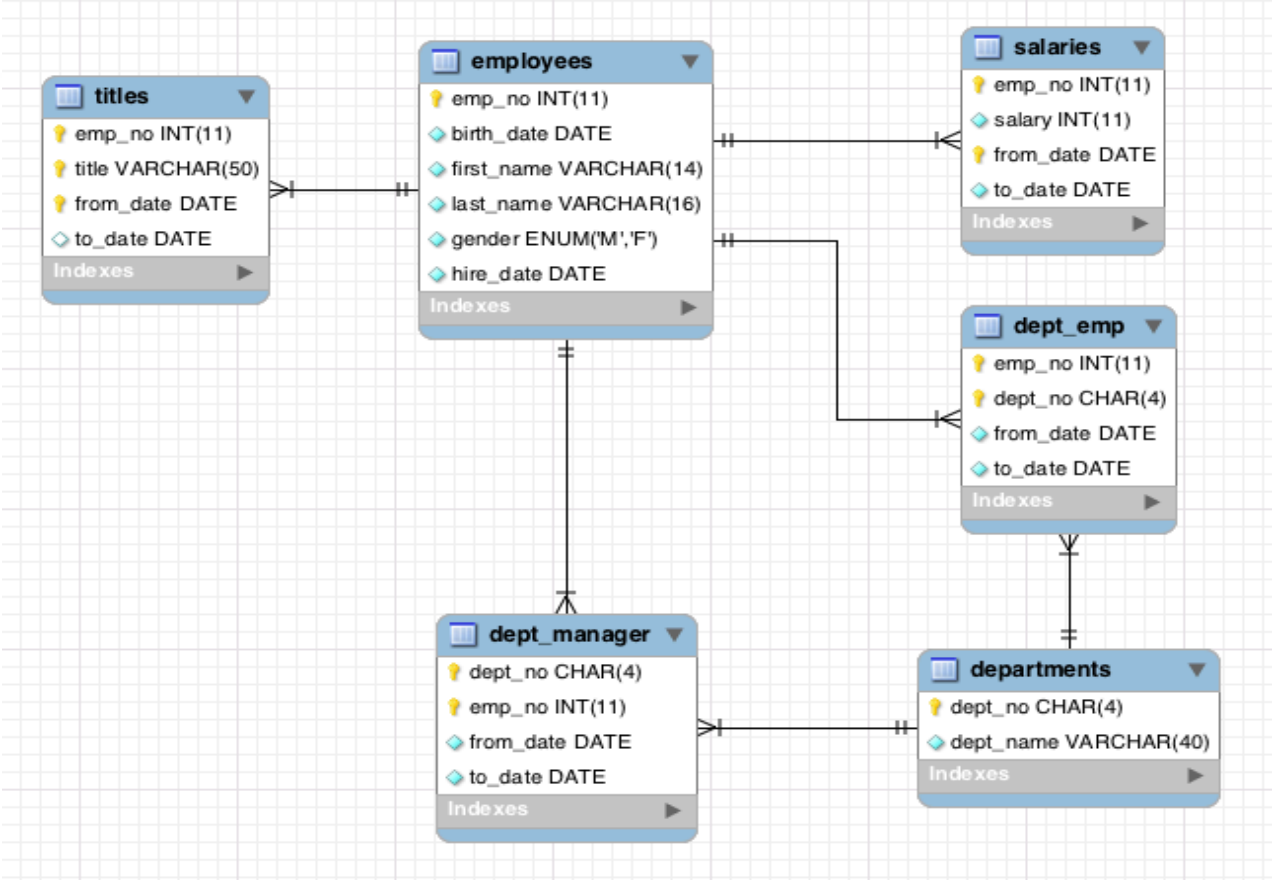
MarkTwain.xml

MarkTwain.XML je instanca oz. primerek zgoraj definirane sheme

```
<?xml version="1.0"?>
<Author xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="Author.xsd">
    <FirstName>Mark</FirstName>
    <LastName>Twain</LastName>
</Author>
```

Vaja

- Kako bi podatke, ki ustrezajo prikazanemu podatkovnemu modelu predstavili z XML datoteko?



Pretvorba relacijske sheme v XML…

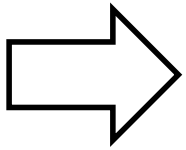
- Povezava 1 : 1

Relacija A

A1	A2
a11	a21
a12	a22

Relacija B

B1	B2	*A1
b11	b21	a11
b12	b22	a12



```
<A A1="a11" A2="a21">
    <B B1="b11" B2="b21"></B>
</A>

<A A1="a12" A2="a22">
    <B B1="b12" B2="b22"></B>
</A>
```

Pretvorba relacijske sheme v XML…

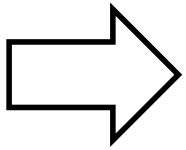
- Povezava 1 : n

Relacija A

A1	A2
a11	a21
a12	a22

Relacija B

B1	B2	*A1
b11	b21	a11
b12	b22	a12
b13	b23	a12



```
<A A1="a11" A2="a21">
    <B B1="b11" B2="b21"></B>
</A>

<A A1="a12" A2="a22">
    <B B1="b12" B2="b22"></B>
    <B B1="b13" B2="b23"></B>
</A>
```

Pretvorba relacijske sheme v XML…

- Povezava m : n

Relacija A

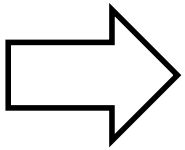
A1	A2
a11	a21
a12	a22

Relacija B

B1	B2
b11	b21
b12	b22

Relacija R

*A1	*B1
a11	b11
a12	b12



```
<A A1="a11" A2="a21" A_id="1"></A>
<B B1="b11" B2="b21" B_id="2"></B>
<R A_idref="1" B_idref="2"></R>

<A A1="a12" A2="a22" A_id="3"></A>
<B B1="b12" B2="b22" B_id="4"></B>
<R A_idref="3" B_idref="4"></R>
```

XML orodja…

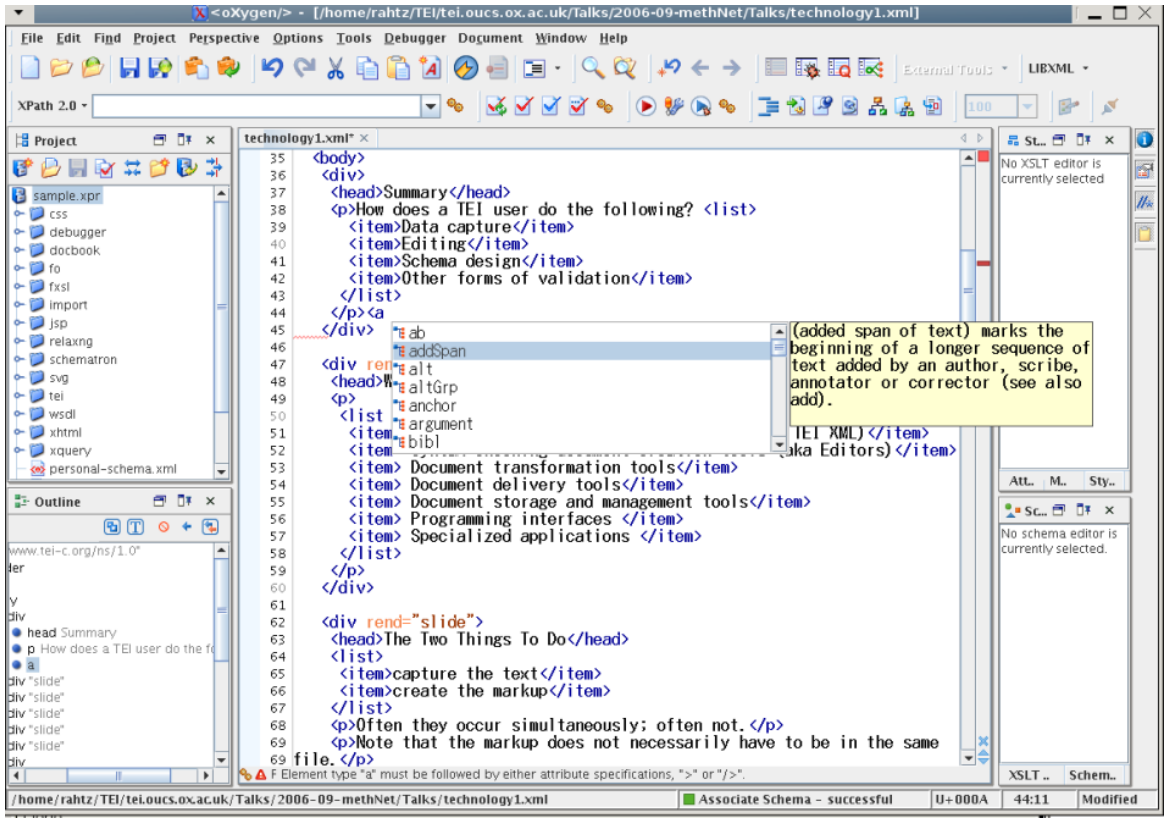
- XML urejevalniki:
  - Orodja, ki omogočajo izdelavo, urejanje in validacijo XML dokumentov.
- Orodja za obdelavo XML:
  - Orodja, ki omogočajo transformacijo XML podatkov (dokumentov) v druge standardizirane oblike… npr HTML.
- Orodja/standardi, ki omogočajo obdelavo XML iz programskih jezikov:
  - XML razčlenjevalci, ki omogočajo razčlenjevanje, validacijo, serializacijo in manipulacijo XML. Npr. Apache Xerces.
  - XML API-ji, ki omogočajo posodabljanje in branje XML podatkov potem, ko so ti že v podatkovnih strukturah nekega programskega jezika (DOM, SAX).

Izdelava XML dokumentov

- Primeri XML urejevalnikov:
  - EditiX – http://www.editix.com
  - XMLSpy – http://www.altova.com
  - Jedit - http://www.jedit.org/
  - Oxygen - http://www.oxygenxml.com/
  - XML Marker - http://symbolclick.com/
  - Sublime text - http://www.sublimetext.com/
  - Notepad

DEMO:

- Oxygen – XML urejevalnik
- http://www.oxygenxml.com/videos



XML podatkovne baze

- XML PB - nudijo upravljanje s podatki v XML formatu:
  - Shranjevanje
  - Poizvedovanje
  - Pretvorbe v različne formate
  - ...
- Tipično temeljijo na dokumentnih bazah.
- Razlog za pojav: XML standard za prenos podatkov. Nepotrebno pretvarjanje med formati...

Vrste XML PB

- PB, ki podpirajo XML dokumente:
  - Pretvarjajo XML v npr. relacije in na izhodu nazaj v XML...
  - Večina novejših SUPB podpira posebne XML podatkovne tipe.
- Naravne XML PB:
  - Interni podatkovni model temelji na XML - XML dokumenti osnovna enota shranjevanja.

PB, ki podpirajo XML

- Navadno ponujajo enega izmed naslednjih pristopov za shranjevanje XML podatkov v relacijsko bazo:
  - XML shranjen v CLOB (Character large object)
  - XML pretvorjen v množico tabel na osnovi sheme;
  - XML shranjen v naravnem XML formatu (XML tip, ISO standard)

- Relacijske PB, ki podpirajo ISO XML tip:
  - IBM DB2;
  - Microsoft SQL Server;
  - Oracle Database;
  - PostgreSQL.









Podpora XML v MySQL…

- Preusmerjanje izhoda poizvedb v XML

```
mysql –u root –p --xml;
mysql> select * from employees limit 1;
```

```
mysql> select * from employees limit 1;
<?xml version="1.0"?>

<resultset statement="select * from employees limit 1;" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
        <field name="emp_no">10001</field>
        <field name="birth_date">1953-09-02</field>
        <field name="first_name">Georgi</field>
        <field name="last_name">Facello</field>
        <field name="gender">M</field>
        <field name="hire_date">1986-06-26</field>
  </row>
</resultset>
1 row in set (0.00 sec)
```

Podpora XML v MySQL…

- Branje XML datotek

person.xml

```
<?xml version="1.0"?>
<list>
  <person person_id="1" fname="Pekka" lname="Nousiainen"/>
  <person person_id="2" fname="Jonas" lname="Oreland"/>
  <person person_id="3"><fname>Mikael</fname><lname>Ronström</lname></person>
  <person person_id="4"><fname>Lars</fname><lname>Thalmann</lname></person>
  <person><field name="person_id">5</field><field name="fname">Tomas</field>
  <field name="lname">Ulin</field></person>
  <person><field name="person_id">6</field><field name="fname">Martin</field>
  <field name="lname">Sköld</field></person>
</list>
```

```
mysql> LOAD XML LOCAL INFILE 'person.xml'
    ->   INTO TABLE person
    ->   ROWS IDENTIFIED BY '<person>';
```

```
mysql> SELECT * FROM person;
+-----------+--------+------------+---------------------+
| person_id | fname  | lname      | created             |
+-----------+--------+------------+---------------------+
|         1 | Pekka  | Nousiainen | 2007-07-13 16:18:47 |
|         2 | Jonas  | Oreland    | 2007-07-13 16:18:47 |
|         3 | Mikael | Ronström   | 2007-07-13 16:18:47 |
|         4 | Lars   | Thalmann   | 2007-07-13 16:18:47 |
|         5 | Tomas  | Ulin       | 2007-07-13 16:18:47 |
|         6 | Martin | Sköld      | 2007-07-13 16:18:47 |
+-----------+--------+------------+---------------------+
6 rows in set (0.00 sec)
```

Podpora XML v MySQL…

- Iskanje po XML datotekah

```
mysql> SET @xml = '<a><b>X</b><b>Y</b></a>';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @i =1, @j = 2;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @i, ExtractValue(@xml, '//b[$@i]');
+------+-------------------------------+
| @i   | ExtractValue(@xml, '//b[$@i]') |
+------+-------------------------------+
|    1 | X                             |
+------+-------------------------------+
1 row in set (0.00 sec)
```

Podpora XML v MySQL

- Spreminjanje XML datotek

```
mysql> SELECT
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '/a', '<e>fff</e>') AS val1,
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '/b', '<e>fff</e>') AS val2,
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '//b', '<e>fff</e>') AS val3,
    ->   UpdateXML('<a><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val4,
    ->   UpdateXML('<a><d></d><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val5
    -> \G

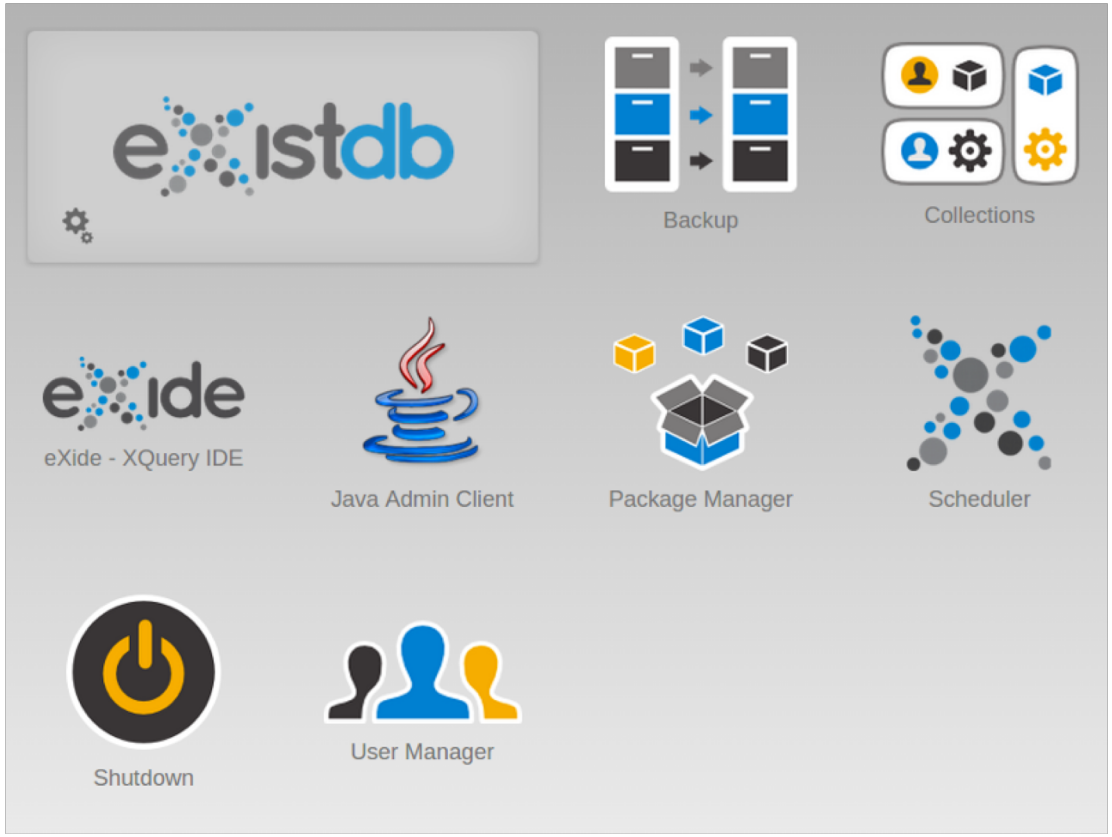
*************************** 1. row ***************************
val1: <e>fff</e>
val2: <a><b>ccc</b><d></d></a>
val3: <a><e>fff</e><d></d></a>
val4: <a><b>ccc</b><e>fff</e></a>
val5: <a><d></d><b>ccc</b><d></d></a>
```

Naravne XML podatkovne baze

- Naravne XML PB:
  - definirajo logični model za XML dokument – omogočajo shranjevanje dokumentov, ki ustrezajo logičnemu modelu in iskanje po njih;
  - Imajo XML dokumente kot osnovno logično enoto shranjavanja (fizično je lahko karkoli).
  - Mnoge ponujajo kolekcije – logične enote skupin dokumentov (collections). PB lahko upravljajo z več kolekcijami na enkrat...
  - Ponujajo vsaj en poizvedovalni jezik: večina XPath, mnoge tudi XQuery.
  - Mnoge ponujajo XSLT za pretvarjanje med formati dokumentov (npr XML – HTML).

DEMO

- Naravna XML podatkovna baza eXists db
- http://exist-db.org/exist/apps/homepage/index.html



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



Predmet:
Osnove podatkovnih baz

Modul:
Poizvedovanje po XML z XQuery

Gradivo:
v.2015

Vsebina

- XQuery, zgodovina, lastnosti
- XQuery in sorodni standardi
- Xpath sintaksa in primeri
- XQuery sintaksa in primeri

XQuery

- XQuery je poizvedovalni funkcijski jezik za poizvedovanje po kolekcijah XML dokumentov.
- XQuery uporablja XPath izrazno sintakso za naslavljanje posameznih delov XML dokumenta.
- Podoben SQL-u.
- FLWOR izrazi: poizvedbe sestavljene iz petih delov: `FOR`, `LET`, `WHERE`, `ORDER BY`, `RETURN`.

Zgodovina

- Nastal 1998 kot jezik za poizvedovanje po XML dokumentih – XQL.
- Skupaj z XML-QL prevzet s strani W3C. 1998 organizirana mednarodna konferenca, kjer narejena osnova za bodoči standard.
  - XQuery nastal kot kompozit dobrih delov sedmih poizvedovalnih jezikov.
- XQuery uporaben tudi za transformacije – izpodriva XSLT.

Dobre lastnosti jezika XQuery

- Osnovne dobre lastnosti:
  - Izraznost –lahko poizvedujemo po več različnih pod.strukturah. Omogoča rekurzijo – idealen za poizvedovanje po drevesih in grafih.
  - Jedrnatost - XQuery stavki so v primerjavi z SQL in XSLT stavki krajši.
  - Fleksibilnost – lahko poizvedujemo po hierarhičnih in tabelaričnih strukturah.
  - Konsistentnost – konsistentna sintaksa, uporaben z drugimi standardi.

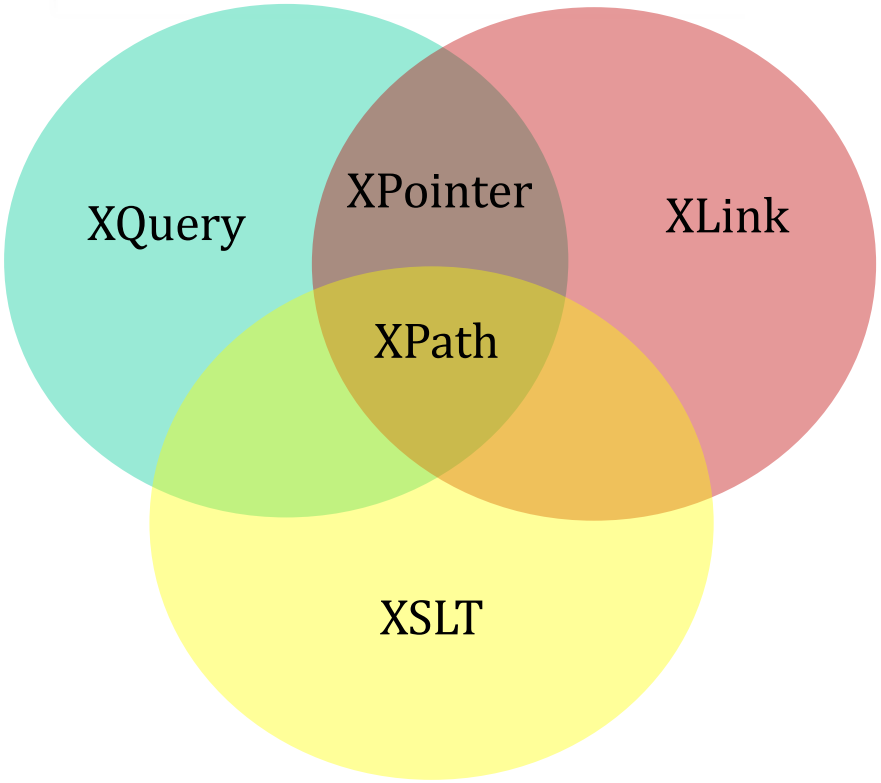
- XQuery veliko prednosti pred SQL in XSLT.

XQuery, XSLT, SQL

- SQL vrne relacijo (tabelo), XQuery vrača poljubne drevesne strukture (lahko npr. vrne XHTML strukturo).
- XQuery enostavnejši kot XSLT. XQuery nudi podobne konstrukte kot SQL:
  - Urejanje rezultatov: `ORDER BY`
  - Odprava duplikatov: `DISTINCT`
  - Selekcija: `WHERE` pogoji…
- XQuery naravni jezik WWW – uporaben za poizvedovanje po spletnih straneh.

XQuery in sorodni standardi

- XQuery 1.0 in XPath 2.0 sta osnovana na istem podatkovnem modelu, podpirata iste funkcije in operatorje...



- XLink standardizira način kreiranja hiperpovezav v XML dokumentih.
- XPointer omogoča, da s hiperpovezavami kažemo na več lokacij v XML dokumentu.
- XPath omogoča navigacijo po XML dokumentih.
- XSLT je standard za transformacijo XML podatkov.

Xpath

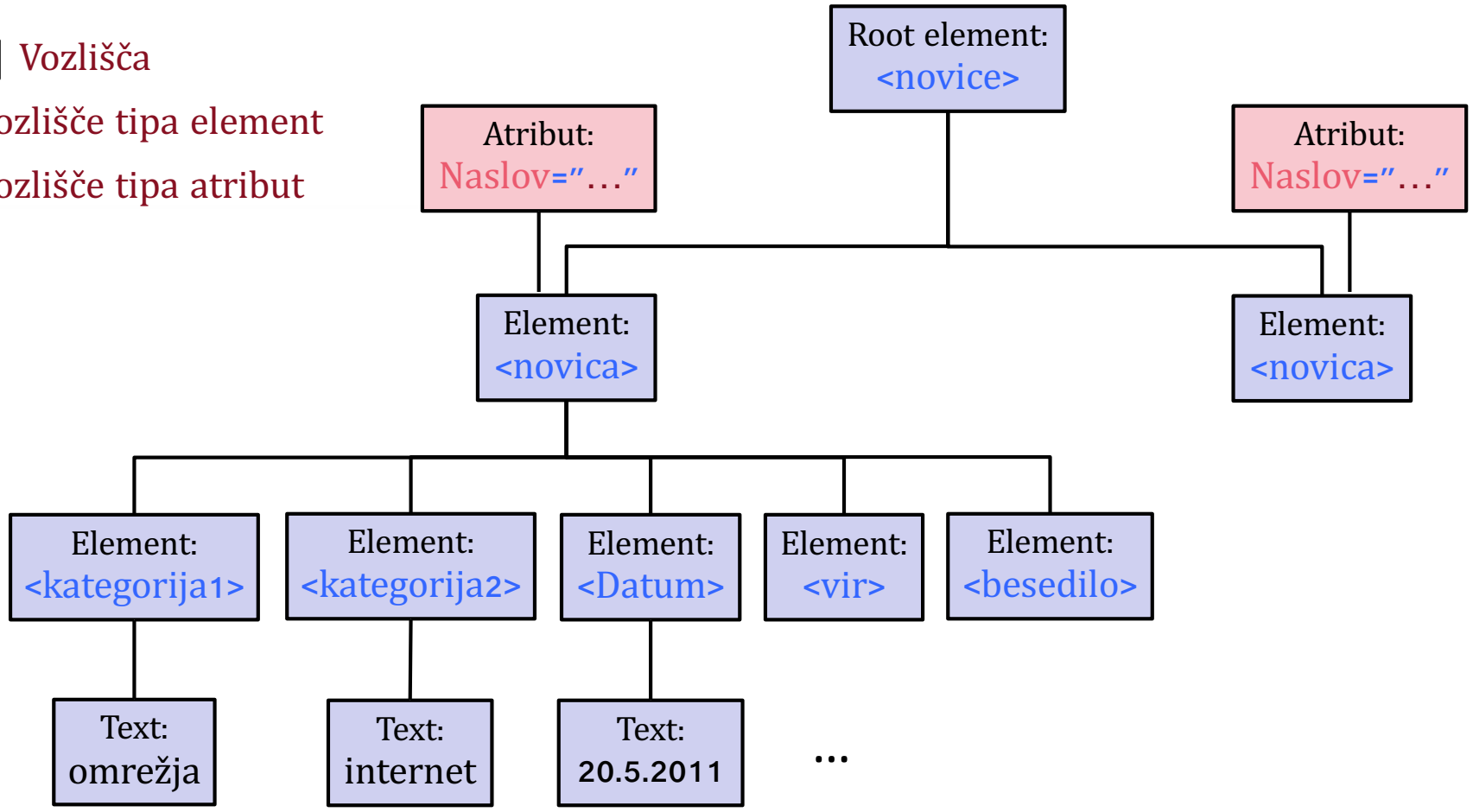
Sintaksa in primeri

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
        <price>49.99</price>
    </book>
    <book category="WEB">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

- bookstore
  - book "COOKING" Everyday Italian
    - title "en" Everyday Italian
    - author Giada De Laurentiis
    - year 2005
    - price 30.00
  - book "CHILDREN" Harry Potter
    - title "en" Harry Potter
    - author J K. Rowling
    - year 2005
    - price 29.99
  - book "WEB" XQuery Kick Start
    - title "en" XQuery Kick Start
    - author James McGovern
    - author Per Bothner
    - author Kurt Cagle
    - author James Linn
    - author Vaidyanathan Nagarajan
    - year 2003
    - price 49.99
  - book "WEB1" Learning XML
    - title "en" Learning XML
    - author Erik T. Ray
    - year 2003
    - price 39.95

Terminologija

- Vozlišča
- Vozlišče tipa element
- Vozlišče tipa atribut



Xpath, XQuery

Pravila, terminologija, sintaksa

Vrste vozlišč…

- XQuery in Xpath obravnavata XML dokument kot drevo vozlišč.
- Vozlišča so različnih tipov:
  - element,
  - atribut,
  - tekst,
  - imenski prostor (ang. namespace),
  - procesna instrukcija,
  - komentar in
  - dokument.

Vrste vozlišč

- Primer:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

dokument

atribut

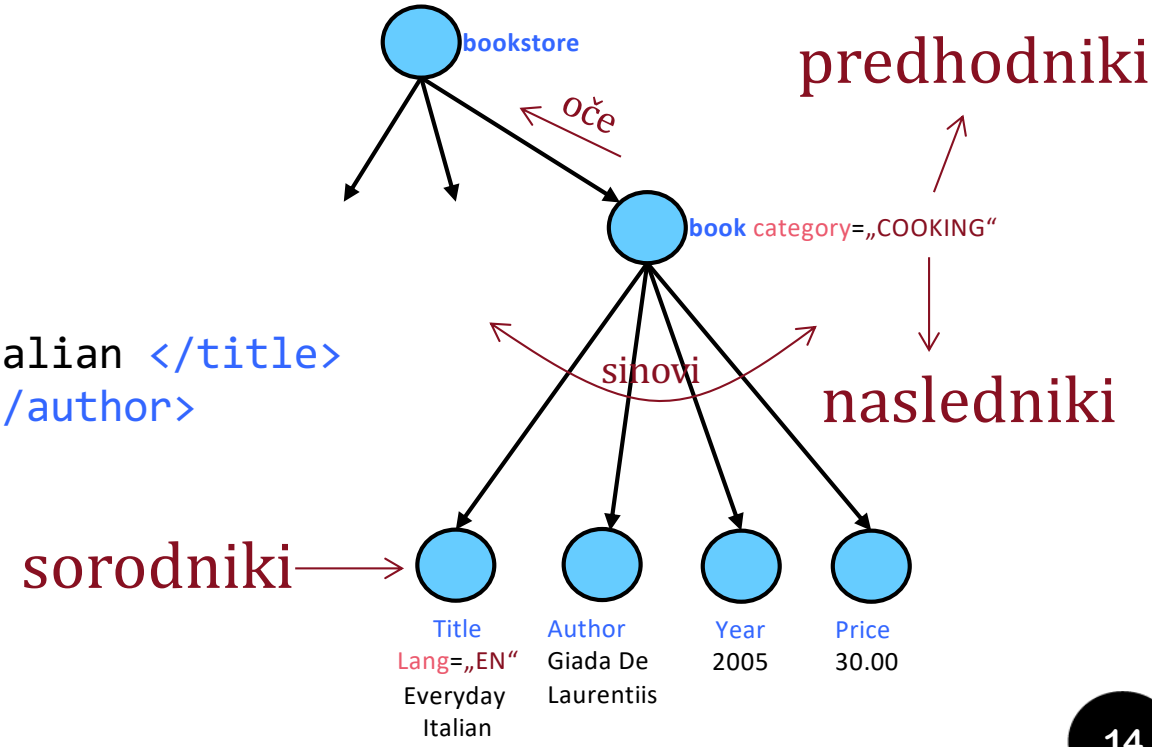
element

Tekst (CDATA)

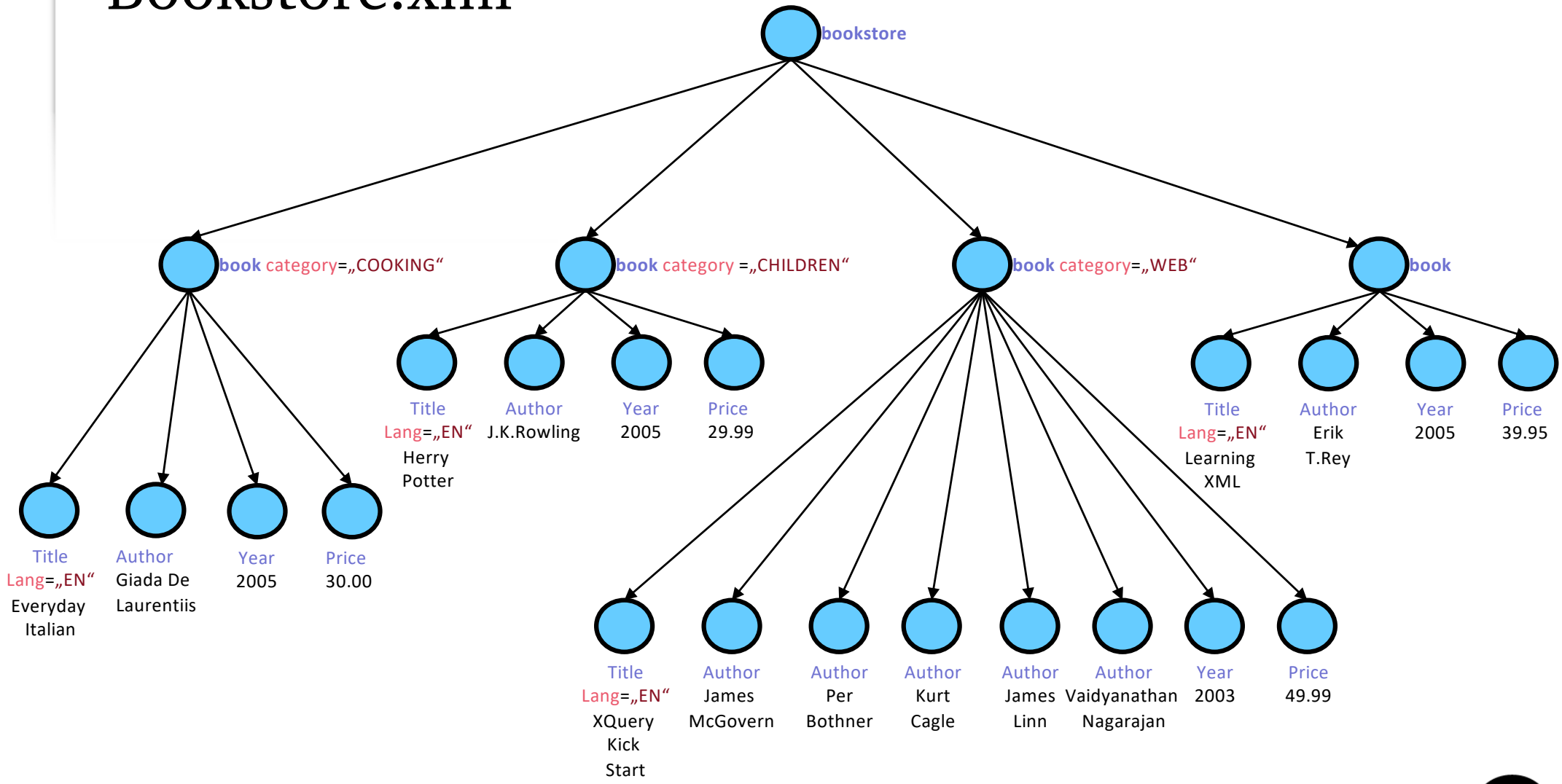
Relacije med vozlišči

- Vsak element in atribut ima enega očeta.
- Elementi imajo lahko poljubno sinov.
- Vozlišča z istim očetom so med seboj sorodniki.
- Poznamo še relacijo predhodnik in naslednik.

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian </title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```



Bookstore.xml



XPath sintaksa – izbira vozlišč...

Xpath>/

Izbere vsa vozlišča začenši s korenskim.
/ označuje absolutno pot.

Primer:
Xpath>/bookstore/book/author

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
        <price>49.99</price>
    </book>
    <book category="WEB1">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

XPath sintaksa – izbira vozlišč...

`Xpath>//`

Izbere vozlišča ne glede na njihovo pozicijo.

Primer:
`Xpath>bookstore//author`

Drugi ukazi za izbiro vozlišč:

`Xpath>.`	Izbere trenutno vozlišče
`Xpath>..`	Izbere očeta trenutnega vozlišča
`Xpath>@`	Izbere atribut

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <bookstore>
3      <book category="COOKING">
4          <title lang="en">Everyday Italian</title>
5          <author>Giada De Laurentiis</author>
6          <year>2005</year>
7          <price>30.00</price>
8      </book>
9      <book category="CHILDREN">
10         <title lang="en">Harry Potter</title>
11         <author>J K. Rowling</author>
12         <year>2005</year>
13         <price>29.99</price>
14     </book>
15     <book category="WEB">
16         <title lang="en">XQuery Kick Start</title>
17         <author>James McGovern</author>
18         <author>Per Bothner</author>
19         <author>Kurt Cagle</author>
20         <author>James Linn</author>
21         <author>Vaidyanathan Nagarajan</author>
22         <year>2003</year>
23         <price>49.99</price>
24     </book>
25     <book category="WEB1">
26         <title lang="en">Learning XML</title>
27         <author>Erik T. Ray</author>
28         <year>2003</year>
29         <price>39.95</price>
30     </book>
31 </bookstore>
```

XPath sintaksa – izbira vozlišč

Če ne vemo imen vozlišč, lahko uporabljamo posebne ukaze.

Xpath>//book/*
Vrne vsa vozlišča, ki so otroci elementa book.

Xpath>//@*
Izbere vse atribute

Xpath>//node()
Izbere vsa vozlišča ne glede na tip (element, atribut)

Primer:
Xpath>//book/*

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
        <price>49.99</price>
    </book>
    <book category="WEB1">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

XPath sintaksa – uporaba predikatov

Xpath>/bookstore/book[1]
Vrne pri element, tipa book

Xpath>/bookstore/book[last()-1]
Izbere predzadnji element tipa book

Xpath>//title[@lang]
Izbere vse atribute lang elementa title

Xpath>//title[@lang='en']
Izbere vse atribute lang elementa title z vrednostjo 'en'

Primer:
Xpath>/bookstore/book[last()-1]

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
        <price>49.99</price>
    </book>
    <book category="WEB1">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

Sintaksa jezika XQuery

- Osnovna pravila:
  - XQuery je občutljiv na velikost črk;
  - XQuery elementi, atributi in spremenljivke morajo biti poimenovani skladno z XML pravili;
  - XQuery črkovni nizi so lahko zapisani z enojnimi ali dvojnimi narekovaji;
  - XQuery spremenljivko definiramo z znakom `$`, ki mu sledi ime, npr. `$bookstore`
  - XQuery komentarje označujemo z `(: komentar :)`, npr. `(: XQuery Comment :)`

Pravila poimenovanja XML elementov in atributov

- ❑ Imena lahko vsebujejo črke, številke in druge znake
- ❑ Imena se ne smejo začeti s številko ali ločilom ali nizom XML, Xml ipd.
- ❑ Imena ne smejo vsebovati presledkov
- ❑ Imena so lahko kakršnakoli (skladno z zgornjimi pravili), rezerviranih besed ni.

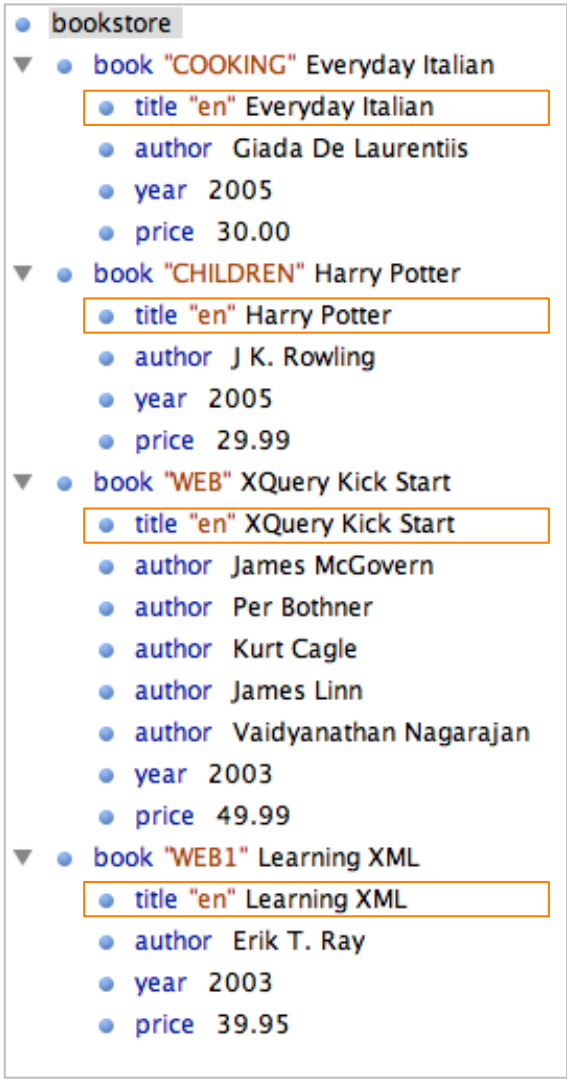
XQuery sintaksa – iskanje po vozliščih

- Za iskanje po vozliščih uporabljamo izraze `Path`
- Izrazi `Path` omogočajo navigacijo po elementih in atributih XML dokumenta.
- Izrazi `Path` temeljijo na Xpath standardu.

- Primer:

```
doc("books.xml")/bookstore/book/title
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <title lang="en">Everyday Italian</title>
3 <title lang="en">Harry Potter</title>
4 <title lang="en">XQuery Kick Start</title>
5 <title lang="en">Learning XML</title>
```



XQuery sintaksa – predikati…

- XQuery uporablja predikate za omejevanje podatkov, ki se vrnejo kot rezultat poizvedbe po XML dokumentu.
- Primer: izberi vse elemente `book` elementa `bookstore`, ki imajo element `price` manj kot `30`:

```
doc("books.xml")/bookstore/book[price<30]
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <book category="CHILDREN">
3       <title lang="en">Harry Potter</title>
4       <author>J K. Rowling</author>
5       <year>2005</year>
6       <price>29.99</price>
7   </book>
```

- bookstore
  - book "COOKING" Everyday Italian
    - title "en" Everyday Italian
    - author Giada De Laurentiis
    - year 2005
    - price 30.00
  - book "CHILDREN" Harry Potter
    - title "en" Harry Potter
    - author J K. Rowling
    - year 2005
    - price 29.99
  - book "WEB" XQuery Kick Start
    - title "en" XQuery Kick Start
    - author James McGovern
    - author Per Bothner
    - author Kurt Cagle
    - author James Linn
    - author Vaidyanathan Nagarajan
    - year 2003
    - price 49.99
  - book "WEB1" Learning XML
    - title "en" Learning XML
    - author Erik T. Ray
    - year 2003
    - price 39.95

Izrazi FLWOR…

- FLWOR akronim za `For`, `Let`, `Where`, `Ordrer By`, `Result`
- `For`:
  - Rezervirana beseda `for` dodeli spremenljivki vsa vozlišča, ki jih vrne nek izraz.
  - `For` povzroči iteriranje čez vse vrednosti spremenljivke.
  - V istem FLWOR izrazu lahko uporabimo več `For` delov.
  - Če želimo omejiti iteriranje, uporabimo rezervirano besedo `to`.

```
for $x in (1 to 5)
return <test>{$x}</test>
```

pove, da gre za spremenljivko

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <test>1</test>
3 <test>2</test>
4 <test>3</test>
5 <test>4</test>
6 <test>5</test>
```

Izrazi FLWOR…

- AT – štetje iteracije
  - Rezervirana beseda at omogoča štetje iteracij.

```
for $x at $i in
    doc("books.xml")/bookstore/book/title
    return <book>{$i}. {data($x)}</book>
```

Zanimajo nas samo podatki brez naziva elementa

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <book>1. Everyday Italian</book>
3 <book>2. Harry Potter</book>
4 <book>3. XQuery Kick Start</book>
5 <book>4. Learning XML</book>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <book>1. <title lang="en">Everyday Italian</title>
3 </book>
4 <book>2. <title lang="en">Harry Potter</title>
5 </book>
6 <book>3. <title lang="en">XQuery Kick Start</title>
7 </book>
8 <book>4. <title lang="en">Learning XML</title>
9 </book>
```

Izrazi FLWOR…

- `LET` – dodeljevanje vrednosti spremenljivkam.
  - Rezervirana beseda `let` omogoča dodeljevanje vrednosti spremenljivkam. Nima iteriranja.

```
let $x := (1 to 5)
   return <test>{$x}</test>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <test>1 2 3 4 5</test>
```

Izrazi FLWOR…

- `WHERE` – omejevanje rezultata.
  - Uporabimo za določanje enega ali več pogojev, ki jim mora zadoščati rezultat.

```
where $x/price>30 and $x/price<100
```

- `ORDER BY` – urejevanje rezultata.
  - Uporabimo za določanje načina sortiranja rezultata.

```
for $x in doc("books.xml")/bookstore/book
order by $x/@category, $x/title
return $x/title
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <title lang="en">Harry Potter</title>
3 <title lang="en">Everyday Italian</title>
4 <title lang="en">XQuery Kick Start</title>
5 <title lang="en">Learning XML</title>
```

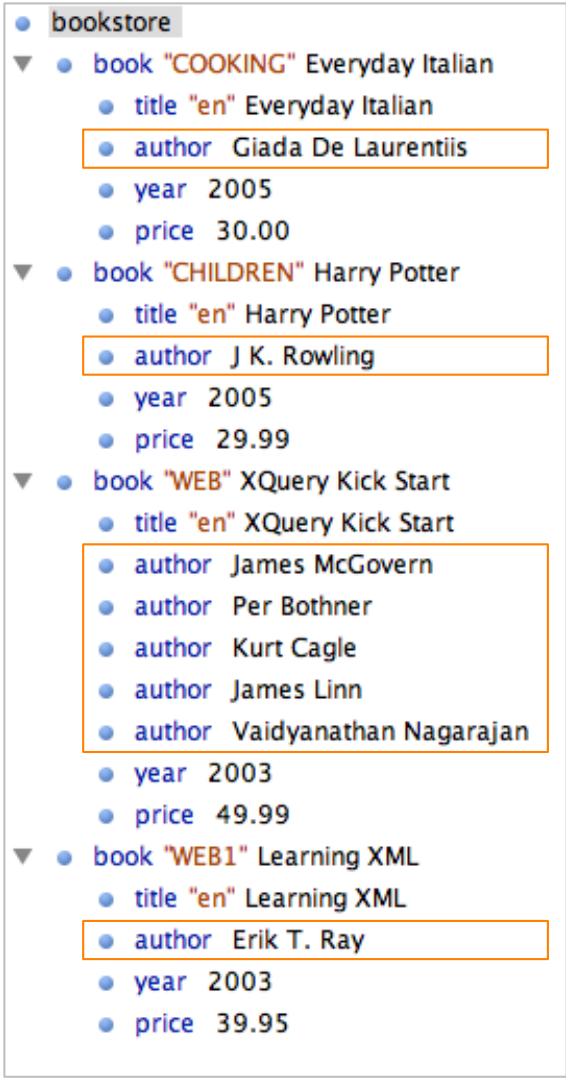
- bookstore
  - book "COOKING" Everyday Italian
    - title "en" Everyday Italian
    - author Giada De Laurentiis
    - year 2005
    - price 30.00
  - book "CHILDREN" Harry Potter
    - title "en" Harry Potter
    - author J K. Rowling
    - year 2005
    - price 29.99
  - book "WEB" XQuery Kick Start
    - title "en" XQuery Kick Start
    - author James McGovern
    - author Per Bothner
    - author Kurt Cagle
    - author James Linn
    - author Vaidyanathan Nagarajan
    - year 2003
    - price 49.99
  - book "WEB1" Learning XML
    - title "en" Learning XML
    - author Erik T. Ray
    - year 2003
    - price 39.95

Izrazi FLWOR…

- RETURN – določanje izhoda.
  - Uporabimo za določitev, kaj naj poizvedba vrne.

```
for $x in doc("books.xml")/bookstore/book
return $x/author
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <author>Giada De Laurentiis</author>
3 <author>J K. Rowling</author>
4 <author>James McGovern</author>
5 <author>Per Bothner</author>
6 <author>Kurt Cagle</author>
7 <author>James Linn</author>
8 <author>Vaidyanathan Nagarajan</author>
9 <author>Erik T. Ray</author>
```



Izrazi FLWOR…

- RETURN – vračanje sestavljenega izhoda.
  - Več izhodov ločimo z vejico. Vejica ima najnižjo prioriteto pri prevajanju, zato potrebno uporabiti oklepaje.

```
for $x in doc("books.xml")/bookstore/book
return ($x/title, $x/year)
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <title lang="en">Everyday Italian</title>
3 <year>2005</year>
4 <title lang="en">Harry Potter</title>
5 <year>2005</year>
6 <title lang="en">XQuery Kick Start</title>
7 <year>2003</year>
8 <title lang="en">Learning XML</title>
9 <year>2003</year>
```

FLWOR + HTML

- Želimo izpisati seznam naslovov vseh knjig v HTML seznam:
  - Dodamo `<ul>` in `<li>` značke v FLWOR izraz:

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ul>
3   <li>Everyday Italian</li>
4   <li>Harry Potter</li>
5   <li>Learning XML</li>
6   <li>XQuery Kick Start</li>
7 </ul>
```

- Everyday Italian
- Harry Potter
- Learning XML
- XQuery Kick Start

XQuery sintaksa – pogojni stavki

- XQuery omogoča izraze "If-Then-Else".

```
for $x in doc("books.xml")/bookstore/book
return if ($x/@category="CHILDREN")
then <child>{data($x/title)}</child>
else <adult>{data($x/title)}</adult>
```

Oklepaji so obvezni!

„else“ del obvezen, vendar lahko prazen, else()

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <adult>Everyday Italian</adult>
3 <child>Harry Potter</child>
4 <adult>XQuery Kick Start</adult>
5 <adult>Learning XML</adult>
```

XQuery sintaksa – primerjava...

- XQuery ponuja dva načina primerjave:
  - Primerjava vrednosti: `eq`, `ne`, `lt`, `le`, `gt`, `ge`
  - Splošna primerjava: `=`, `!=`, `<`, `<=`, `>`, `>=`

- Primer primerjave vrednosti:

```
for $b in //book
where $b/title eq "Harry Potter"
return $b/price
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <price>29.99</price>
```

- bookstore
  - book "COOKING" Everyday Italian
    - title "en" Everyday Italian
    - author Giada De Laurentiis
    - year 2005
    - price 30.00
  - book "CHILDREN" Harry Potter
    - title "en" Harry Potter
    - author J K. Rowling
    - year 2005
    - price 29.99
  - book "WEB" XQuery Kick Start
    - title "en" XQuery Kick Start
    - author James McGovern
    - author Per Bothner
    - author Kurt Cagle
    - author James Linn
    - author Vaidyanathan Nagarajan
    - year 2003
    - price 49.99
  - book "WEB1" Learning XML
    - title "en" Learning XML
    - author Erik T. Ray
    - year 2003
    - price 39.95

XQuery sintaksa – primerjava

- Primerjava vrednosti ni vedno mogoča: spodnji izraz vrača več vozlišč, zato vrne napako. Nadomestimo s splošno primerjavo!

error

```
for $b in //book
where $b/author eq "Kurt Cagle"
return $b/title
```

```
for $b in //book
where $b/author = "Kurt Cagle"
return $b/title
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <title lang="en">XQuery Kick Start</title>
```

- bookstore
  - book "COOKING" Everyday Italian
    - title "en" Everyday Italian
    - author Giada De Laurentiis
    - year 2005
    - price 30.00
  - book "CHILDREN" Harry Potter
    - title "en" Harry Potter
    - author J K. Rowling
    - year 2005
    - price 29.99
  - book "WEB" XQuery Kick Start
    - title "en" XQuery Kick Start
    - author James McGovern
    - author Per Bothner
    - author Kurt Cagle
    - author James Linn
    - author Vaidyanathan Nagarajan
    - year 2003
    - price 49.99
  - book "WEB1" Learning XML
    - title "en" Learning XML
    - author Erik T. Ray
    - year 2003
    - price 39.95

XQuery funkcije

- XQuery 3.0, XPath 2.0 in XSLT 2.0 imajo enak nabor vgrajenih funkcij.
- Na voljo preko 100 vgrajenih funkcij:
  - delo z alfa-numeričnimi in numeričnimi vrednostmi,
  - primerjava datumskih in časovnih vrednosti,
  - delo z vozlišči,
  - delo s sekvencami,
  - delo z logičnimi izrazi.

- Možna tudi definicija lastnih funkcij...

XQuery funkcije

- Nabor vgrajenih funkcij:
  - URI imenskega prostora XQuery funkcij je: http://www.w3.org/2005/02/xpath-functions
  - Privzet prefix funkcijskega imenskega prostora je `fn:` npr. `fn:string()`
  - Nabor vseh vgrajenih funkcij: http://www.w3schools.com/xpath/xpath_functions.asp

    Kategorije:

- Accessor
- Error and Trace
- Numeric
- String
- AnyURI
- Boolean
- Duration/Date/Time
- QName
- Node
- Sequence
- Context

XQuery funkcije

- Primeri funkcij iz skupine `String`:

```
fn:string(arg)
fn:codepoints-to-string(int,int,...)
fn:string-to-codepoints(string)
fn:codepoint-equal(comp1,comp2)
fn:compare(comp1,comp2)
fn:compare(comp1,comp2,collation)
fn:concat(string,string,...)
fn:substring(string,start,len)
```

…

XQuery funkcije

- Primer uporabe funkcije `substring`:
  - `fn:substring(string,start,len)`
  - Vrne niz, ki se začne na poziciji start in je dolžine len. Če drugi argument ni podan, vrne vse znake do konca vhodnega niza.

  - Primeri:

```
substring('Beatles',1,4)
```

```
1 <?xml version="1.0" encoding="UTF-8"?>Beat
```

```
substring('Beatles',2)
```

```
1 <?xml version="1.0" encoding="UTF-8"?>eatles
```

XQuery kvantificirani izrazi...

- XQuery omogoča uporabo eksistencialnega in univerzalnega kvantifikatorja.
- Za eksistencialni kvantifikator uporabljamo rezervirano besedo `some`, za univerzalni pa `every`.

- Primer uporabe univerzalnega kvantifikatorja:

```
every $book in /bookstore/book
satisfies $book/@category="CHILDREN”
```

```
1 <?xml version="1.0" encoding="UTF-8"?>false
```

```
every $book in /bookstore/book
satisfies $book/title/@lang=”en”
```

```
1 <?xml version="1.0" encoding="UTF-8"?>true
```

- bookstore
  - book "COOKING" Everyday Italian
    - title "en" Everyday Italian
    - author Giada De Laurentiis
    - year 2005
    - price 30.00
  - book "CHILDREN" Harry Potter
    - title "en" Harry Potter
    - author J K. Rowling
    - year 2005
    - price 29.99
  - book "WEB" XQuery Kick Start
    - title "en" XQuery Kick Start
    - author James McGovern
    - author Per Bothner
    - author Kurt Cagle
    - author James Linn
    - author Vaidyanathan Nagarajan
    - year 2003
    - price 49.99
  - book "WEB1" Learning XML
    - title "en" Learning XML
    - author Erik T. Ray
    - year 2003
    - price 39.95

XQuery kvantificirani izrazi

- Primer uporabe eksistencialnega kvantifikatorja:

```
some $p in /bookstore/book/price
Satisfies ($p > 30.00)
```

```
1 <?xml version="1.0" encoding="UTF-8"?>true
```

- Vaja: kakšna je najmanjša vrednost `P`, da naslednji izraz vrne „`false`“?

```
some $x in (3, 4, 5), $y in (1, 2, 3)
satisfies $x + $y > P
```

Domača naloga

- Imamo dva xml dokumenta: `depts.xml` s podatki o laboratorijih in `emps.xml` s podatki o zaposlenih v laboratorijih. Kaj vrne spodnja poizvedba?

```
for $d in document("depts.xml")//deptno
let $e := document("emps.xml")//employee[deptno = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
  <big-dept>
     { $d,
        <headcount>{count($e)}</headcount>,
        <avgsal>{avg($e/salary)}</avgsal>
     }
  </big-dept>
```

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



Predmet:
Osnove podatkovnih baz

Modul:
Kratek pregled NoSQL paradigme

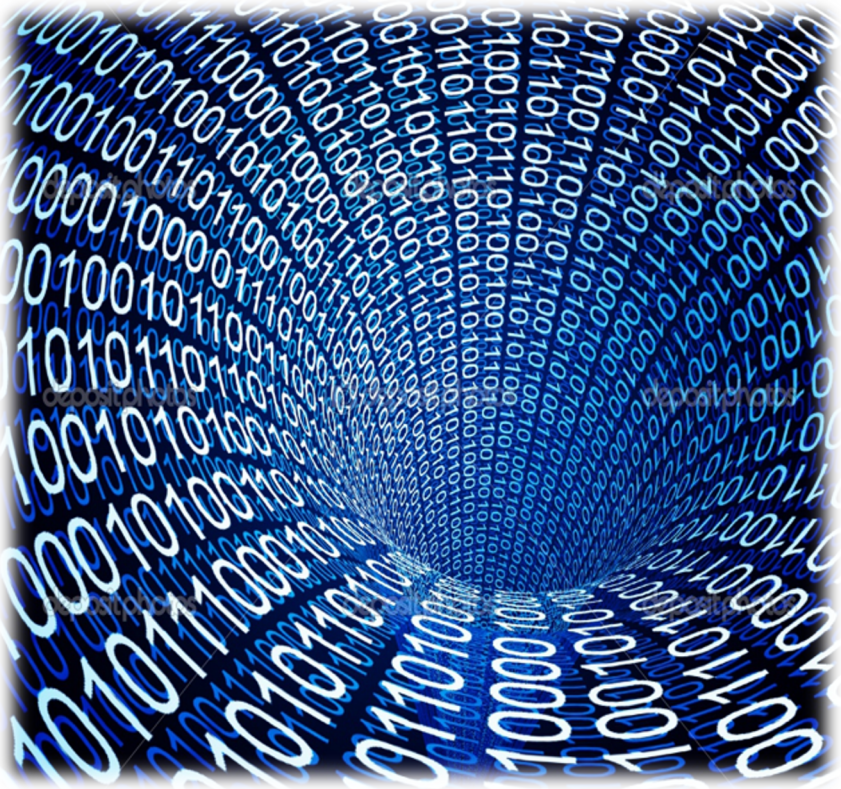
Gradivo:
v.2019

Vsebina

- Masovni podatki (angl. *BigData*)
- Relacijske vs noSQL podatkovne baze
- Lastnosti in tehnologije porazdeljenih baz
  - Fragmentacija in replikacija
  - ACID lastnosti
  - CAP izrek
  - Postopna konsistentnost
- Vrste noSQL podatkovnih baz

Masovni podatki...področja

- Številni primeri, kjer imamo opravka z ogromnimi količinami podatkov...
  - Senzorska omrežja,
  - Družbena omrežja,
  - Spletni dokumenti
  - CDR
  - Astronomija
  - Genomika
  - Vojaški nadzor
  - Video arhivi
  - ...



Narašča tudi število uporabnikov

- Posamezne spletne aplikacije uporabljajo milijoni uporabnikov... primer: družbeni mediji.

2,7 milijarde (2020)



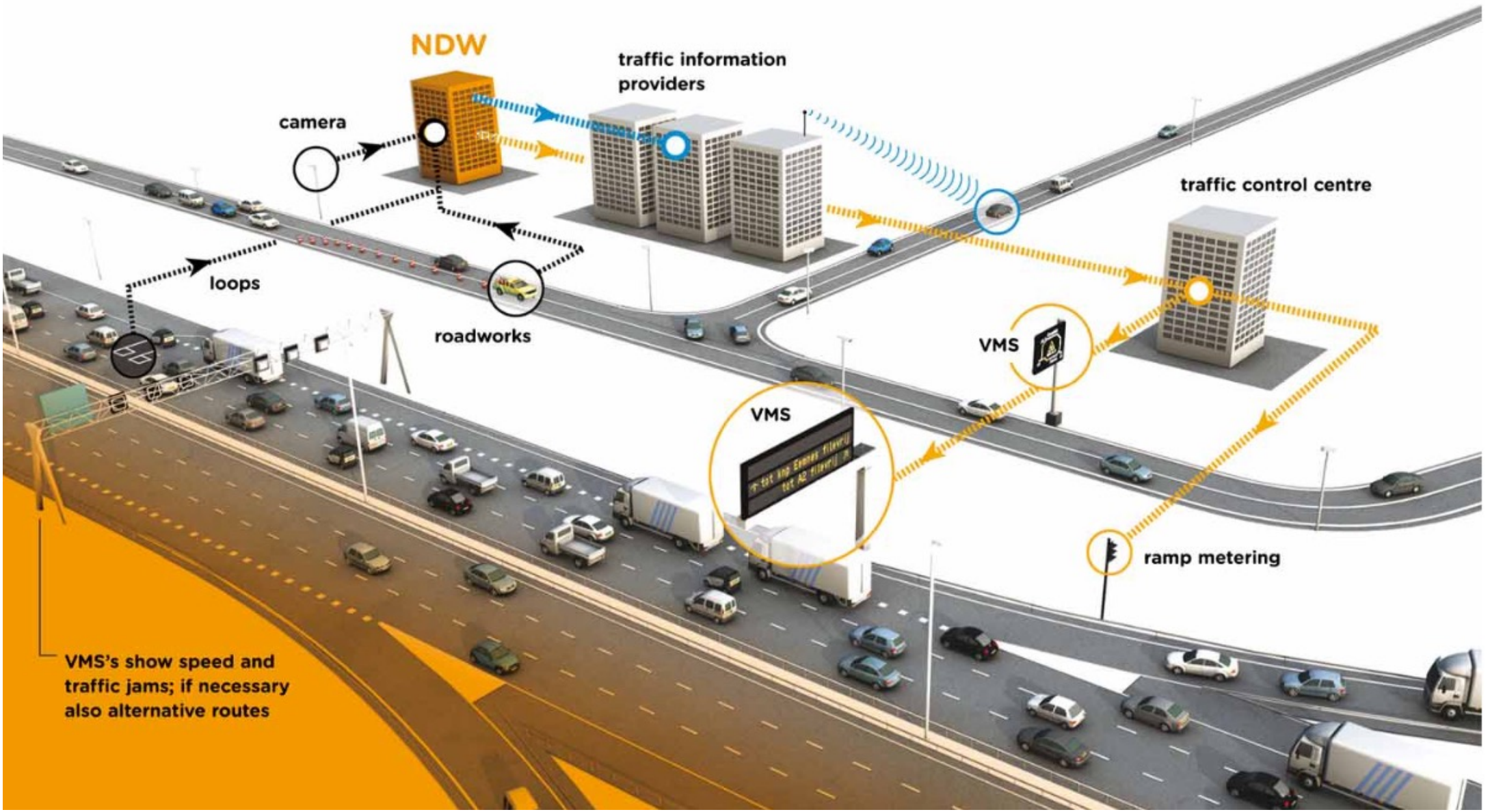
- Pričakujemo 100% razpoložljivost in zanesljivost...

Masovni podatki...trije V-ji

- **V**olume - **količina**: od terabajtov to petabajtov in več
- **V**ariety - **raznolikost**: podatki različnih formatov iz številnih virov (npr. web dnevniki, socialni mediji, e-poslovanje in on-line transakcije, finančne transakcije...)
- **V**elocity - **hitrost**: poslovna okolja zahtevajo čedalje krajše čase od trenutka, ko so podatki generirani, do trenutka, ko se na njihovi osnovi sprejema odločitve (od enega dneva do odločanja v realnem času)

Dodatno: **kontekstna odvisnost** (**V**ariability), **kakovost** (**V**eracity), **prikaz** (**V**isualization), **vrednost** (**V**alue).

Traffic information: collect, store and distribute



Masovni podatki...razširljivost

- Nujna lastnost: razširljivost (scalability).

- Razširljiva aplikacija:
  - če imamo dovolj strežnikov, lahko zagotovimo fiksne odzivne čase ne glede na količino podatkov...
  - Idealno: linearna odvisnost med obsegom dela, časom za procesiranje ter številom strežnikov (vozlišča).

- Razširljivost po podatkih
- Razširljivost po številu uporabnikov

Relacijske vs NoSQL baze...

- Relacijske baze
  - Zagotavljajo zanesljiv način izvajanja transakcij (ACID: Atomicity, Consistency, Isolation, Durability)
  - Standardizirani poizvedovalni jeziki
  - Podatkovna shema
  - Distribucija redko

Relacijske vs NoSQL baze

- noSQL
  - Distribucija prek mreže (poceni) računalnikov
  - Zagotavljena varnost pred odpovedjo (Fault Tolerant).
  - ACID le delno zagotovljen.
  - Navadno brez podatkovne sheme (scheme on read)
  - Optimizirane za spletne storitve
  - Lastni poizvedovalni jeziki

Lastnosti porazdeljeni baz

- Fragmentacija in replikacija
- ACID lastnosti
- CAP izrek
- Postopna konsistentnost

Replikacija in fragmentacija

- Tri dimenzije razširljivosti podatkovnih baz:
  - Po številu bralnih operacij
  - Po številu pisalnih operacij
  - Po velikosti podatkovne baze

- Razširljivost zagotavljamo s pomočjo replikacije in fragmentacije (sharding). Navadno s kombinacijo obojega.

Replikacija...

- Zapis podatkovnih enot na več kot eno vozlišče.
- Učinkovitejše branje (load balancer).
- Večja zanesljivost (odpornost proti odpovedi posameznega vozčišča)
- Replikacija na različne podatkovne centre v izogib katastrofalnim nesrečam...

- Replikacija lahko delna ali popolna (“povsod vse”)

Replikacija

- Negativen efekt na pisalne operacije... vse replike morajo biti zapisane.

- Možnosti:
  - Pisalna operacija izvedena na vseh vozliščih z repliko. Uspešna, ko potrjena na vseh vozliščih.
  - Pisalna operacija izvedena na enem ali nekaj vozliščih. Kasneje vzporedno asinhrono posredovana še drugim vozliščem.

Fragmentacija

- Fragmentacija (Sharding): razbitje podatkov na fragmente. Fragmenti zapisani na posameznih vozliščih.
- Vozlišča poljubno dodajamo… povečujemo kapaciteto ter učinkovitost branja in pisanja.

- Negativna plat – kompleksnost JOIN operacij. Navadno nepodprte v porazdeljenih bazah.

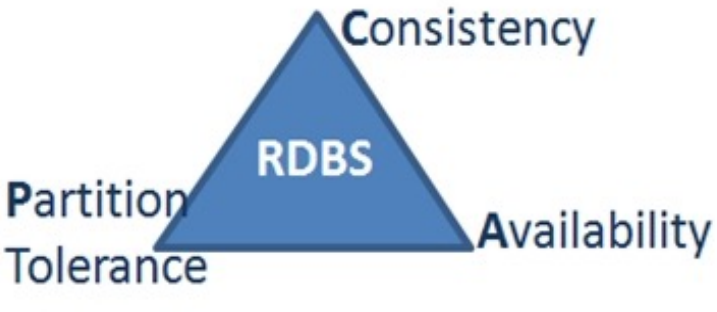
ACID lastnosti

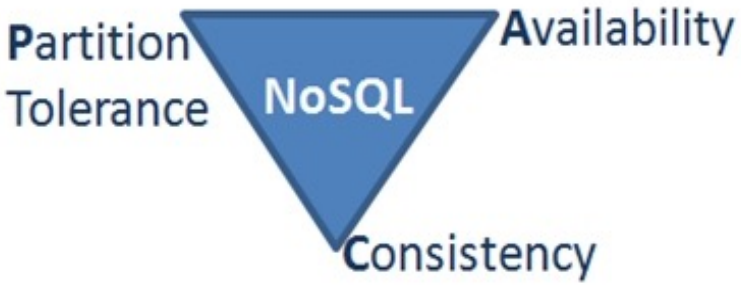
- ACID:
  - Atomarnost (**A**tomicity)
  - Konsistentnst (**C**onsistency)
  - Izolacija (**I**solation)
  - Trajnost (**D**urability)
- Relacijske baze koristijo vnaprejšen zapis v dnevnik (write-ahead log) za **D** ter centralno zaklepanje (central lock) za **ACI.**

- V porazdeljenih bazah centralno zaklepanje ozko grlo. noSQL uporabljajo poseben pristop.

Izrek CAP

- V porazdeljeni bazi lahko zagotovimo le dve od treh lastnosti*:
  - Razpoložljivost (availability)
  - Konsistentnost (consistency)
  - Odpornost proti odpovedim posameznih particij (partition tolerance).





* velja za primere, ko pride do odpovedi dela omrežja (network partitioning)

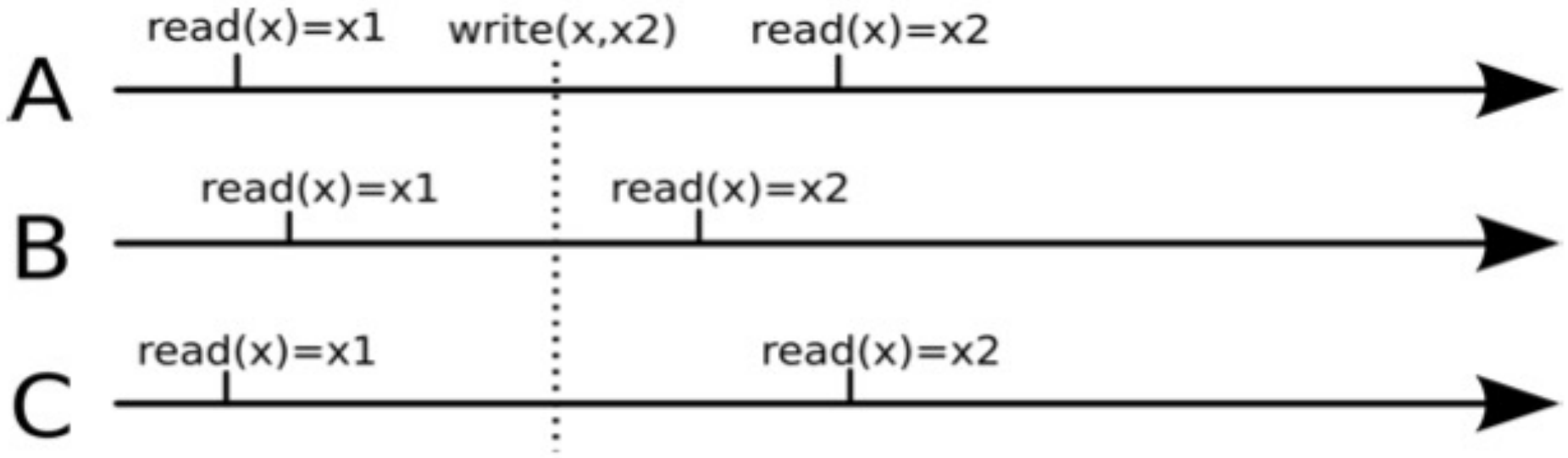
Konsistentnost...vrste

- Stroga konsistenost: vse verzije posamezne podatkovne enote so enake.
- Postopna konsistentnost: vse verzije posamezne podatkovne enote bodo sčasoma enake.

- Primer:
  - A, B, C: nepovezani procesi, ki berejo ali pišejo v bazo
  - x: opazovana podatkovna enota
  - x1, x2, x3: različne vrednosti podatkovne enote x

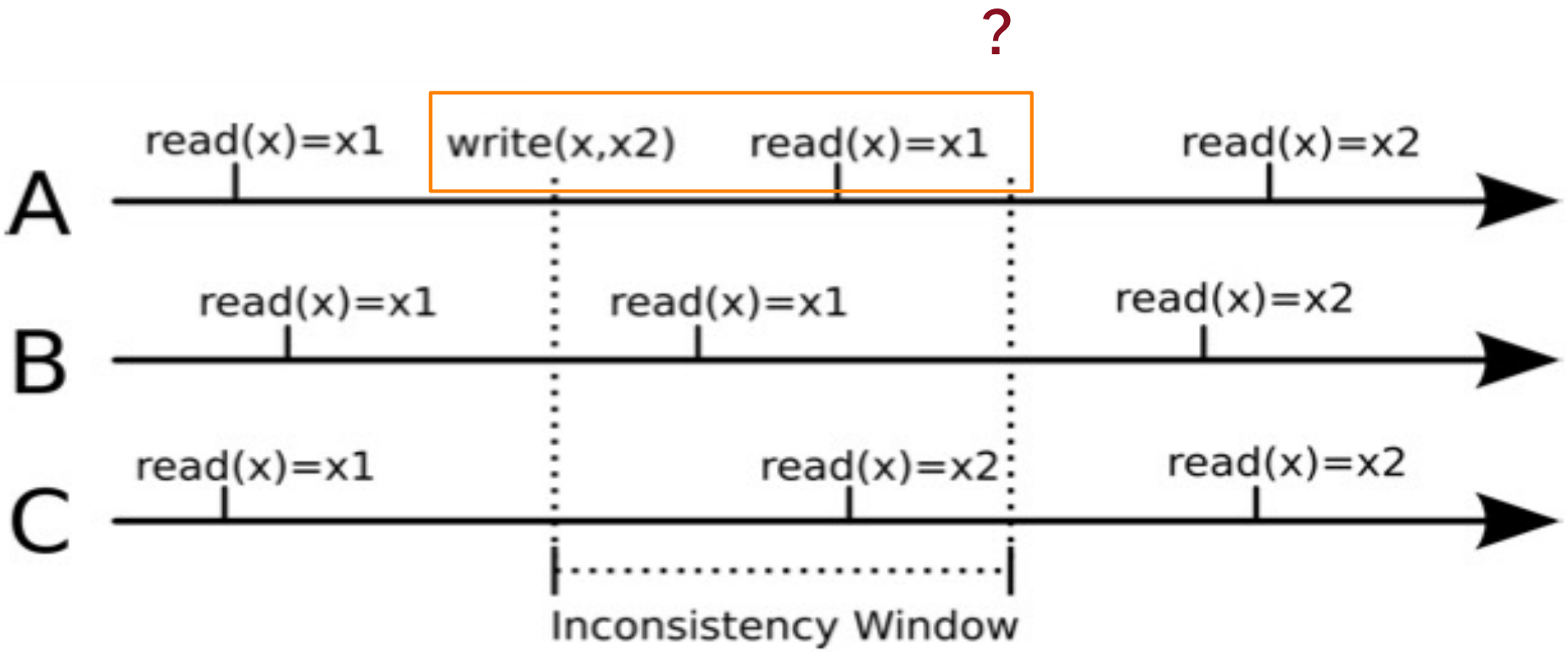
Konsistentnost...stroga

- Stroga konsistenost (Strict Consistency)



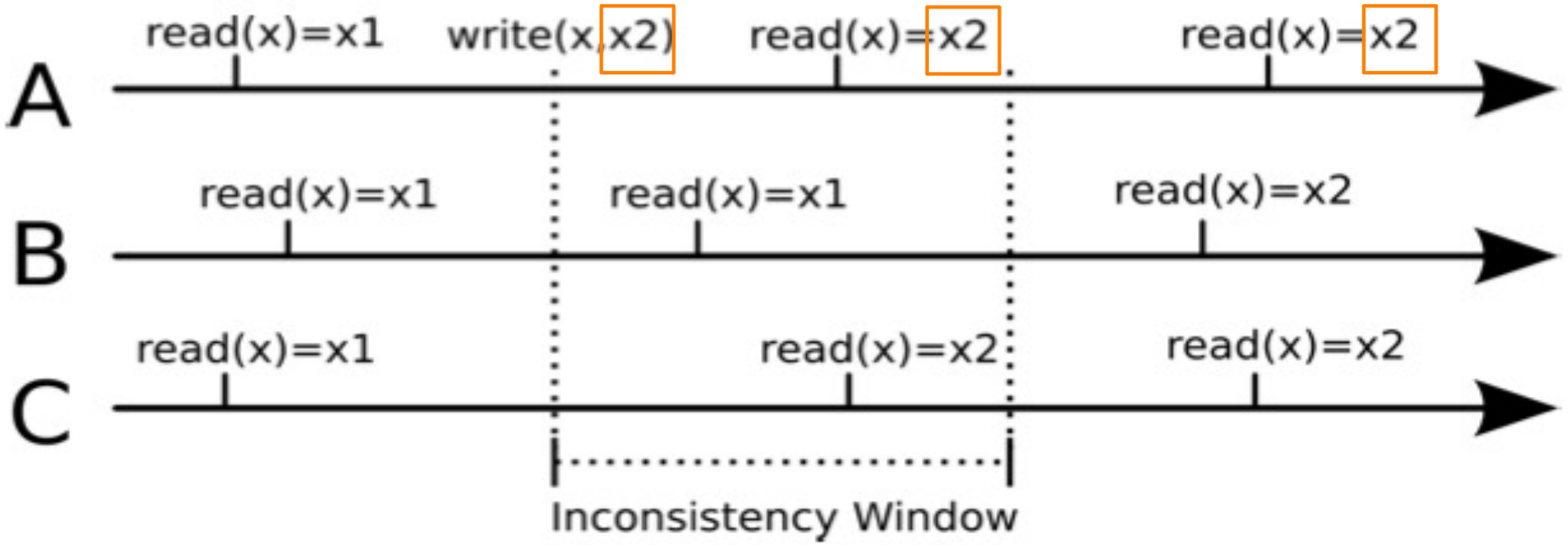
Konsistentnost...postopna...

- Postopna konsistenost (Eventual Consistency)



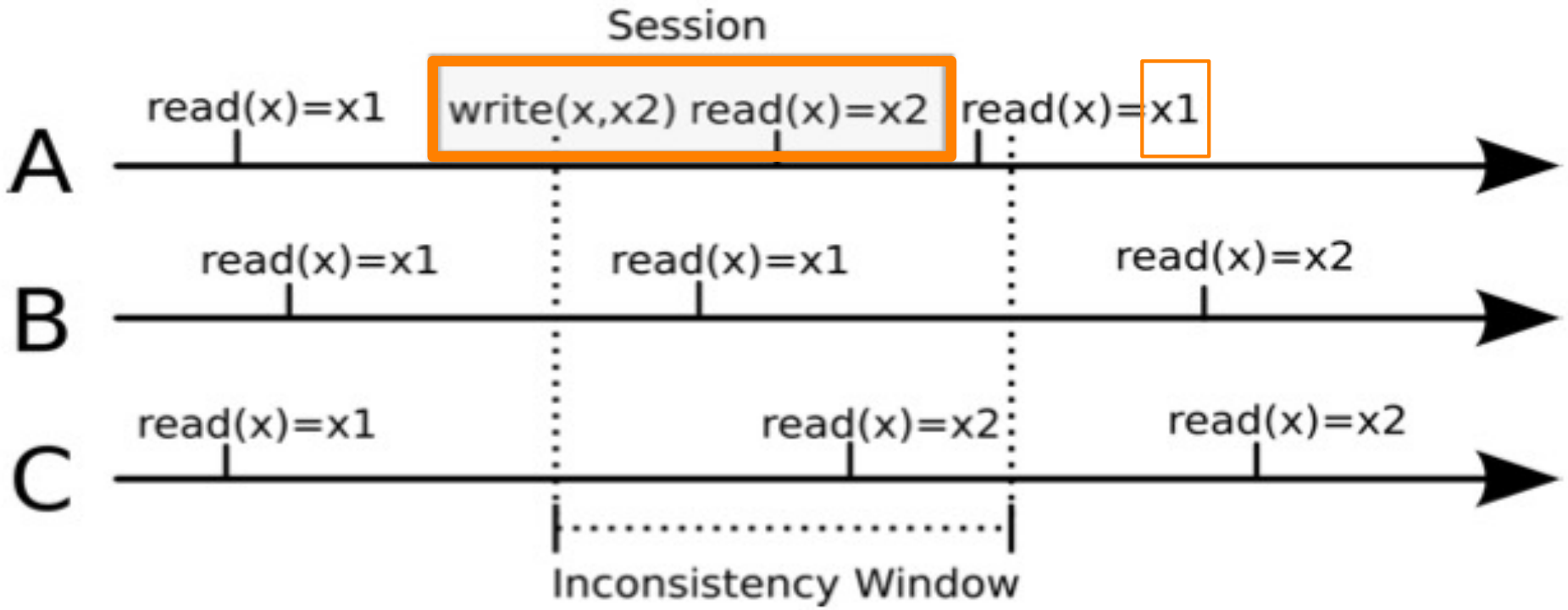
Konsistentnost...postopna

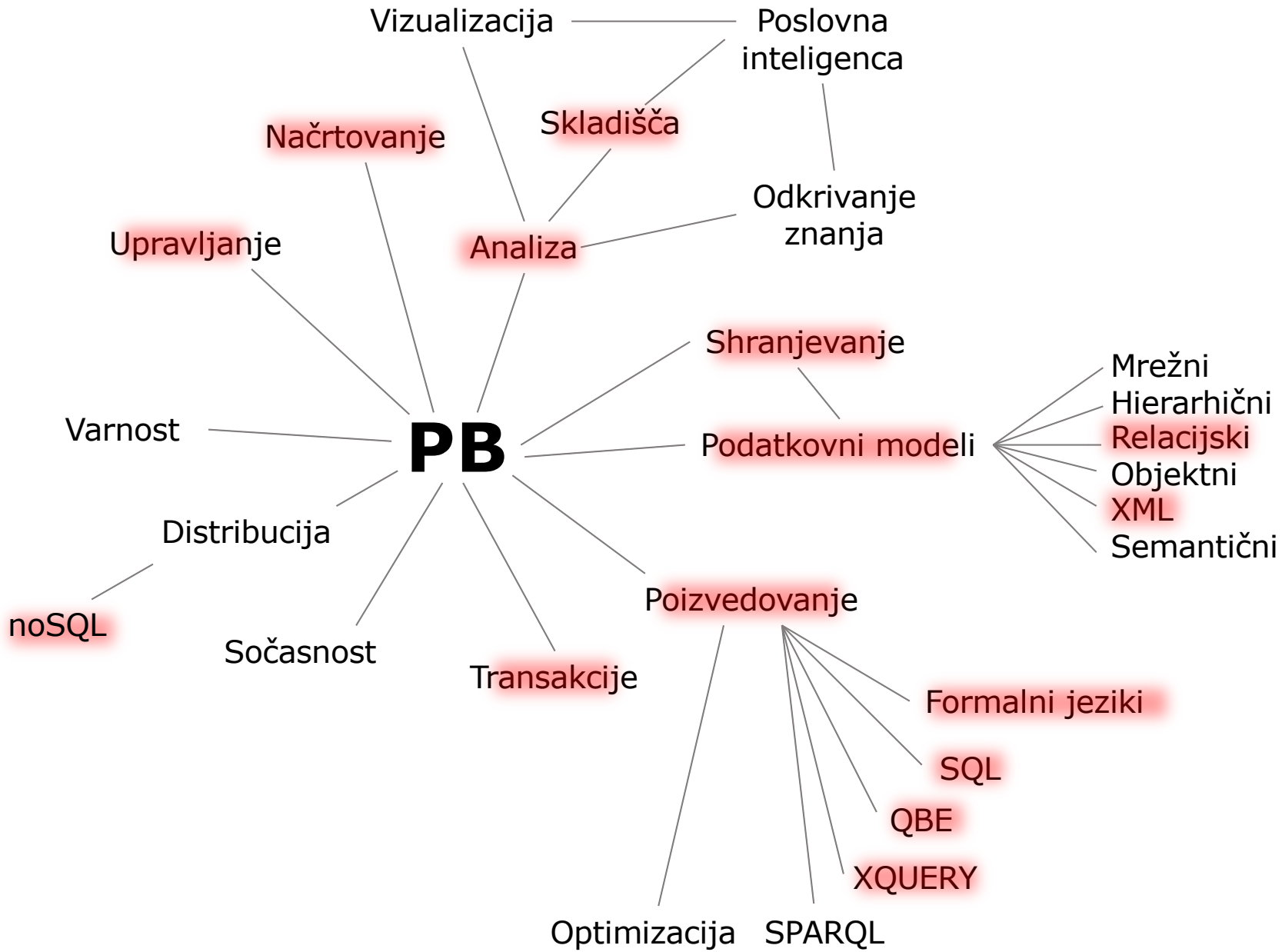
- Postopna konsistenost “beri-svoje-podatke” (Read-Your-Own-Writes Consistency)



Konsistentnost...seje

- Konsistenost seje (Session Consistency)




Vizualizacija
Poslovna inteligenca
Načrtovanje
Skladišča
Odkrivanje znanja
Upravljanje
Analiza
Shranjevanje
Varnost
PB
Podatkovni modeli
Mrežni
Hierarhični
Relacijski
Objektni
XML
Semantični
Distribucija
noSQL
Poizvedovanje
Sočasnost
Transakcije
Formalni jeziki
SQL
QBE
XQUERY
Optimizacija
SPARQL

Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



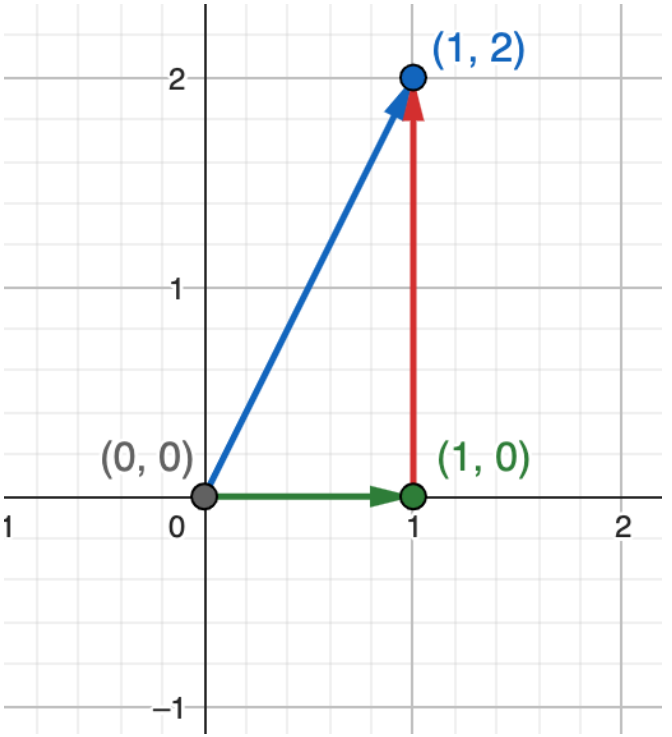
Predmet:
Osnove podatkovnih baz

Modul:
Vektorske podatkovne baze

Gradivo:
v.2025

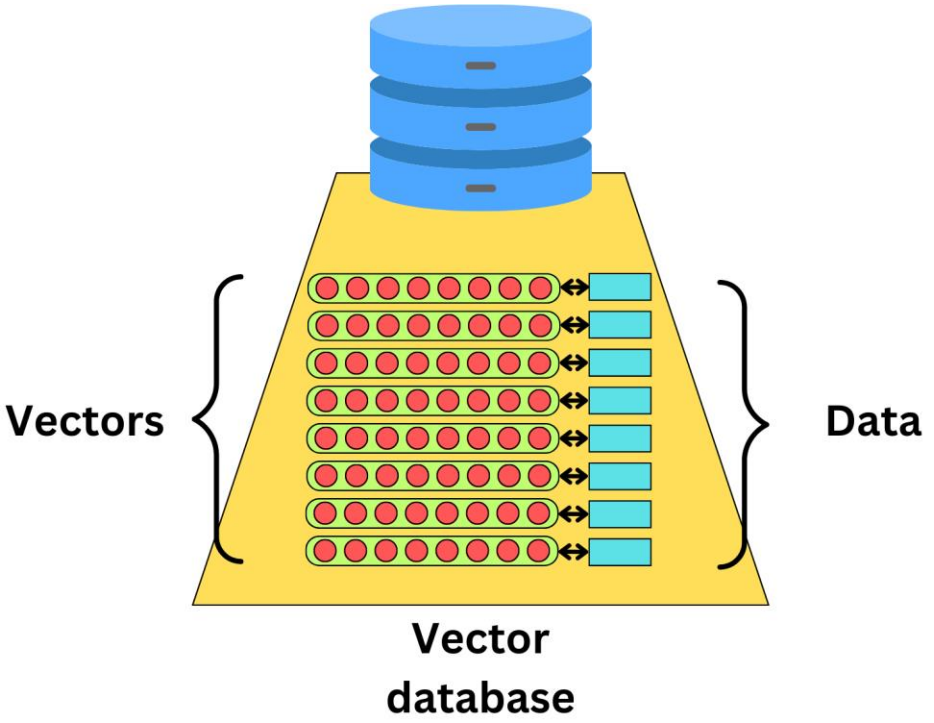
Vektorji

- Matematični objekti, ki imajo velikost in smer
- Predstavimo lahko s končnim zaporedjem števil
- Vizualiziramo lahko kot usmerjene črte



Vektorske podatkovne baze - VPB

- VPB je posebna vrsta PB, ki podatke hrani kot več-dimenzijske vektorje.
- Vektorji v VPB so matematične predstavitve lastnosti podatkov.
- Več lastnosti, kot jih lahko razberemo iz podatkov, več dimenzij ima vektor (tudi tisoč in mnogo več).



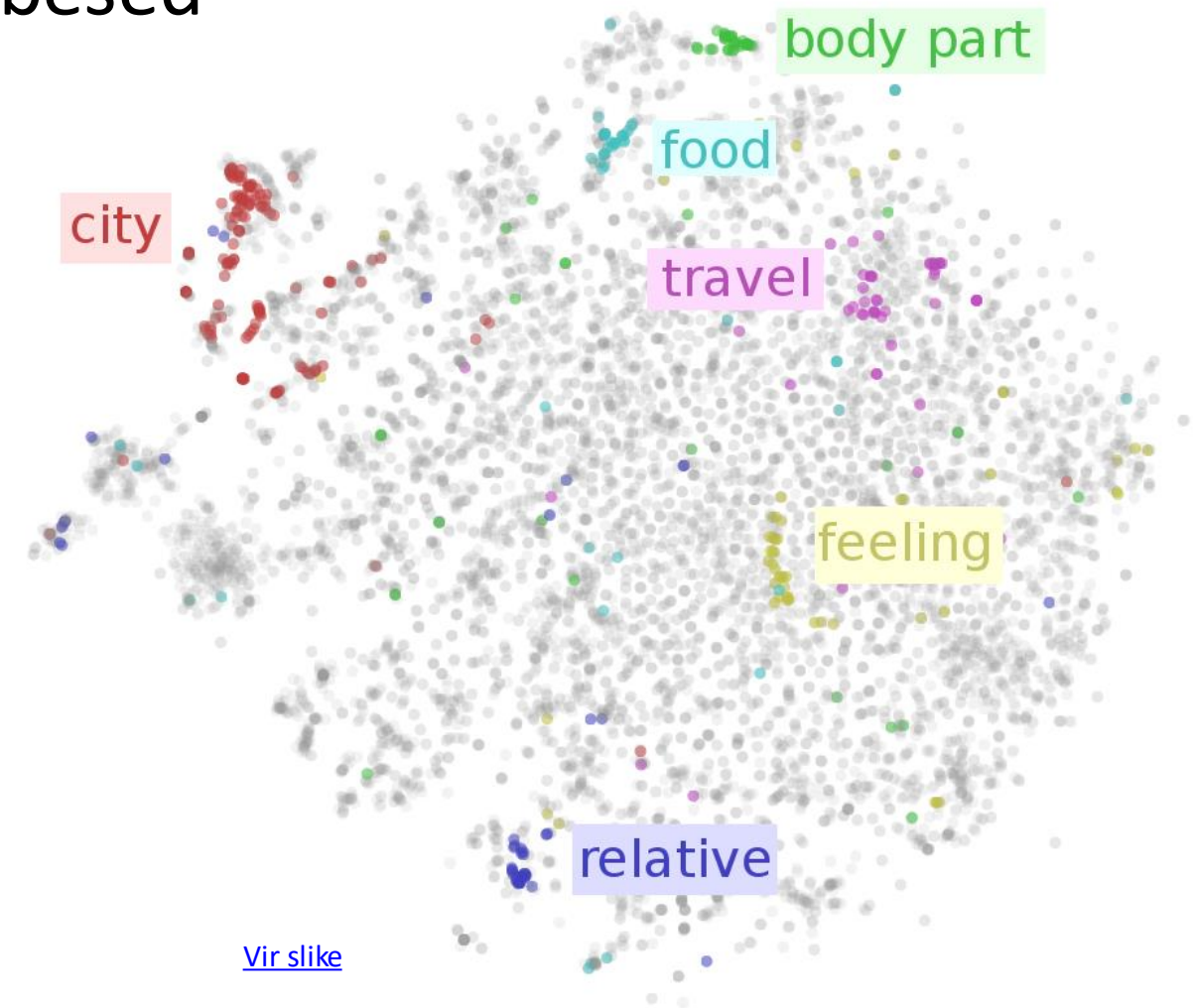
Vir slike

Vektorske podatkovne baze - VPB

- VPB se uporabljajo za hranjenje kompleksnih vektorjev, ki nastanejo s strojno obdelavo podatkov in iz podatkov izluščijo številne (tudi skrite) lastnosti.
- Različne metode za generiranje vektorskih predstavitev:
  - modeli strojnega učenja,
  - vložitve besed (*word embeddings*) in
  - algoritmi za ekstrakcijo lastnosti (feature extraction algorithms).

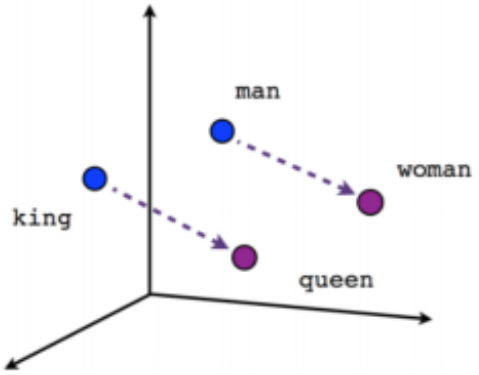
- Cilj: semantično podobni vhodni podatki se pretvorijo v vektorje, ki so si blizu v več-dimenzionalnem prostoru.

Vložitve besed

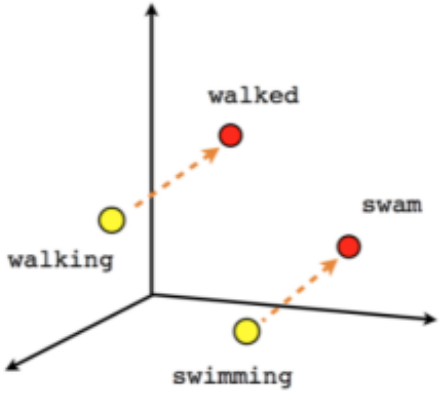


Vir slike

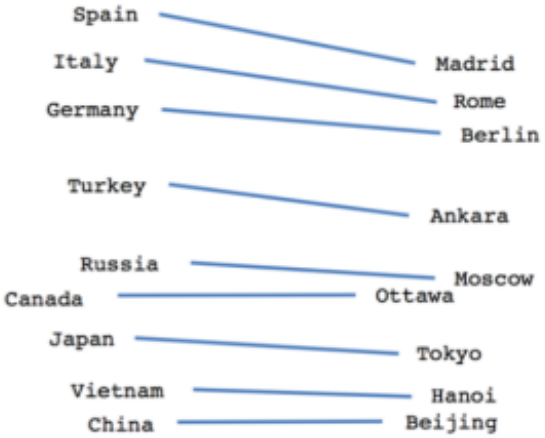
Lastnosti vložitev



Male-Female



Verb tense

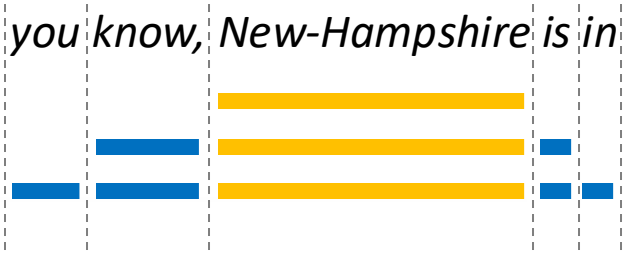


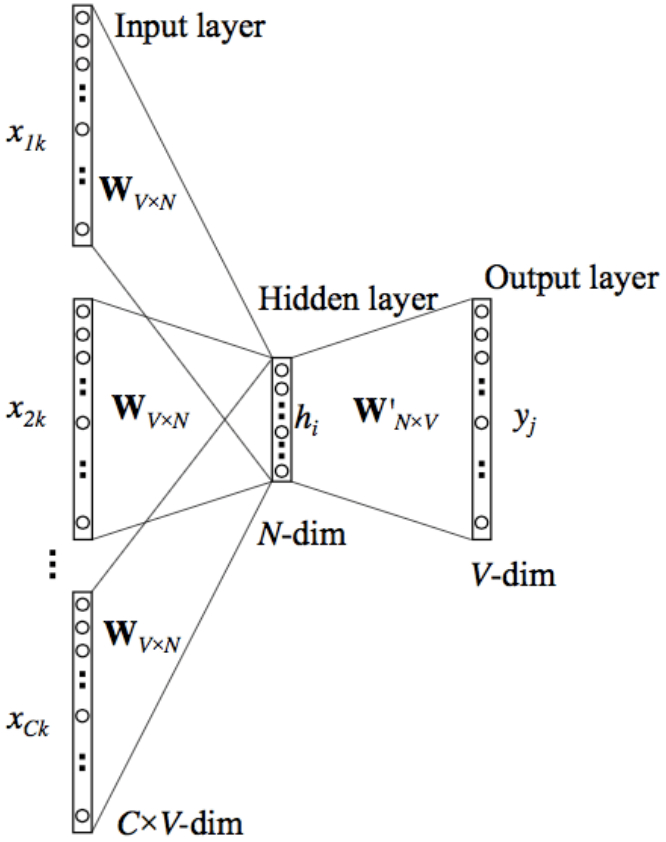
Country-Capital

Vir slike

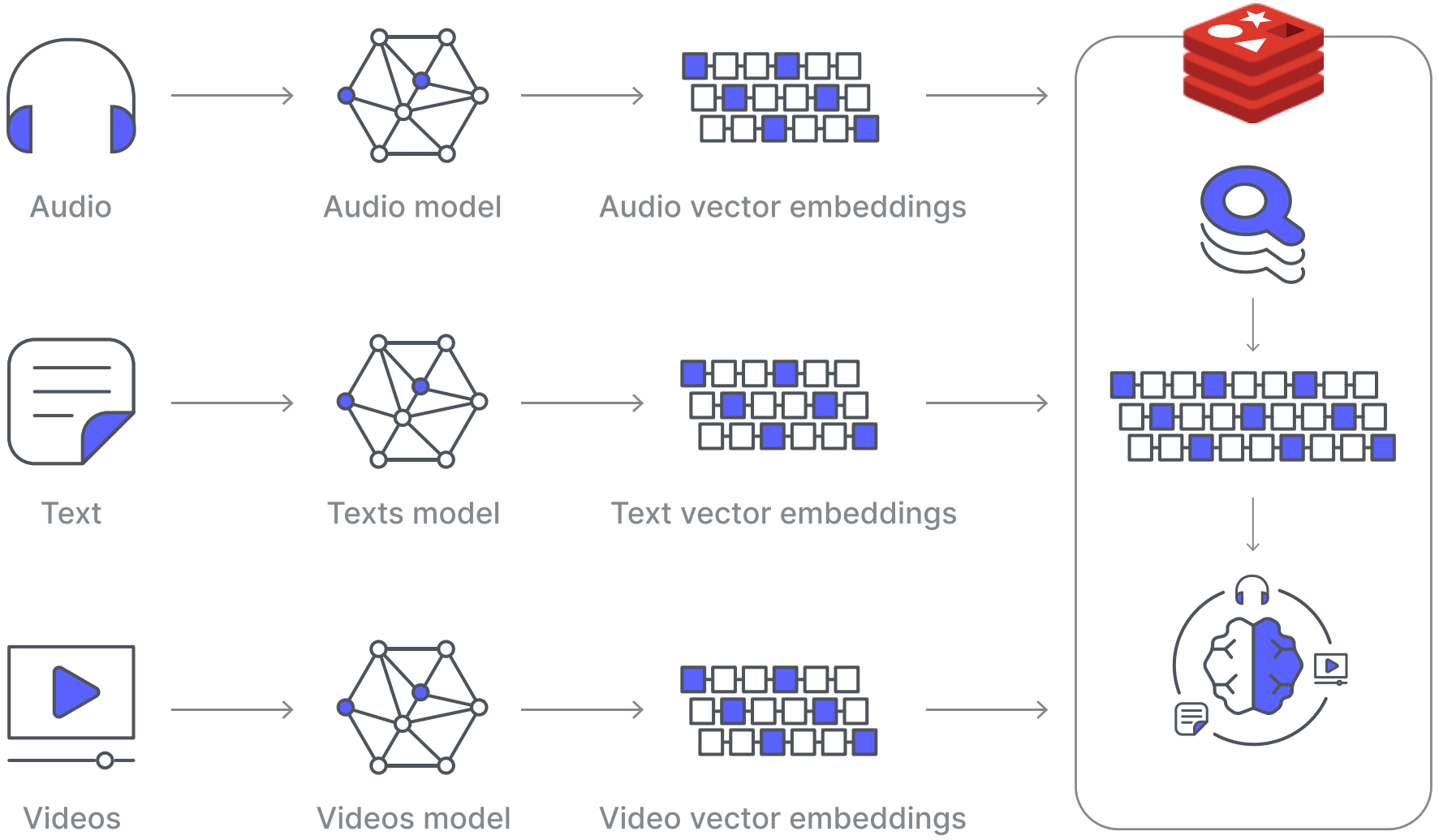
Pridobivanje vektorskih vložitev

Did you know, New-Hampshire is in New-England.





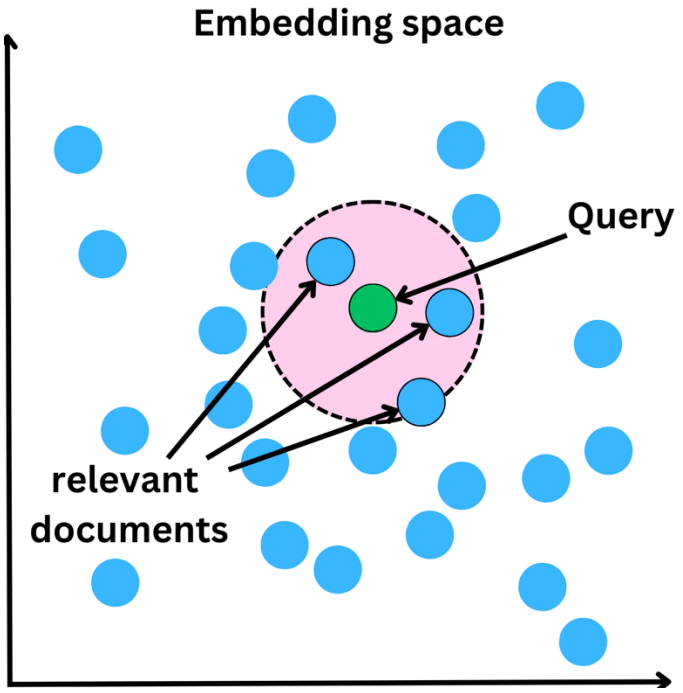
Vir slike


Audio
Audio model
Audio vector embeddings
Text
Texts model
Text vector embeddings
Videos
Videos model
Video vector embeddings

Vir slike

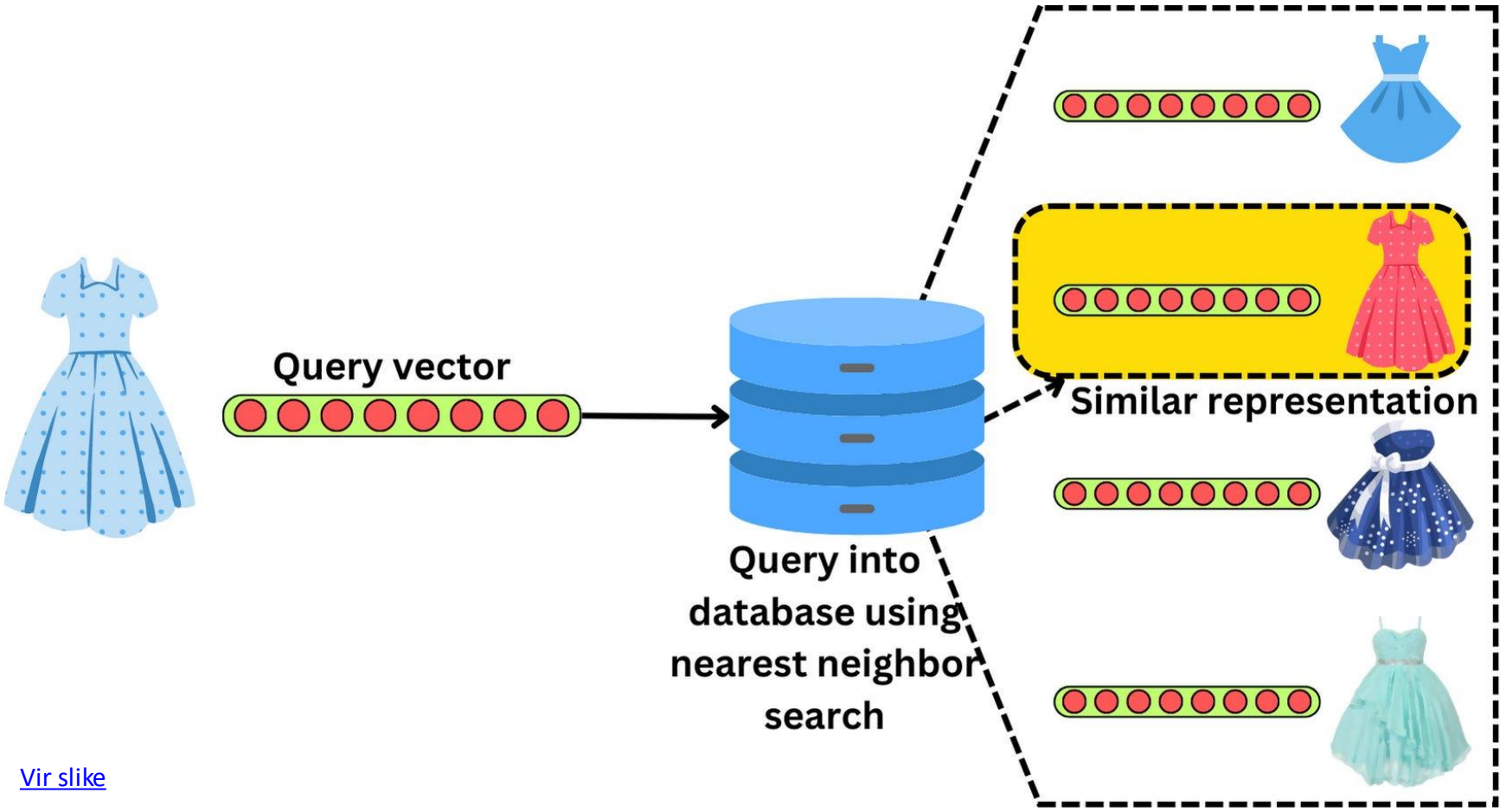
Prednosti VPB

- Učinkovito iskanje podobnih podatkov (bližina v vektorskem prostoru).
- Iskanje na podlagi semantične relevantnosti namesto zanašanja na natančno ujemanje.
- VPB so optimizirane za analitične poizvedbe; gručenje, kategorizacija podatkov, …



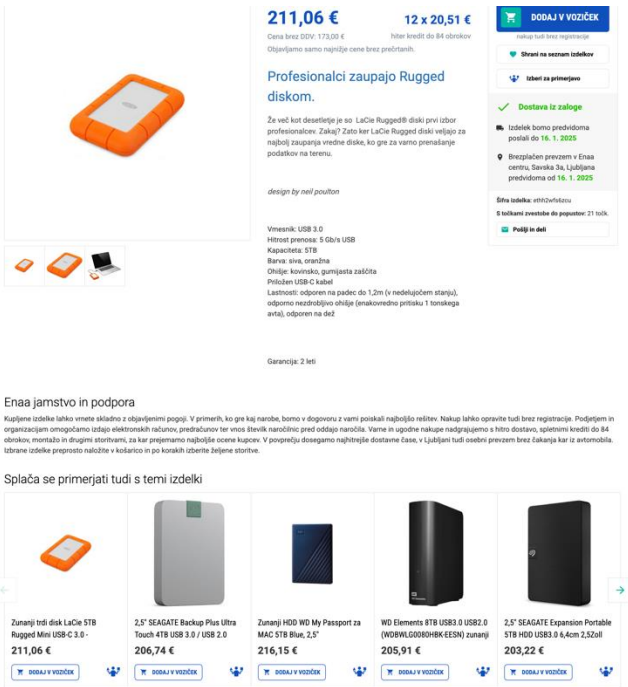
Vir slike

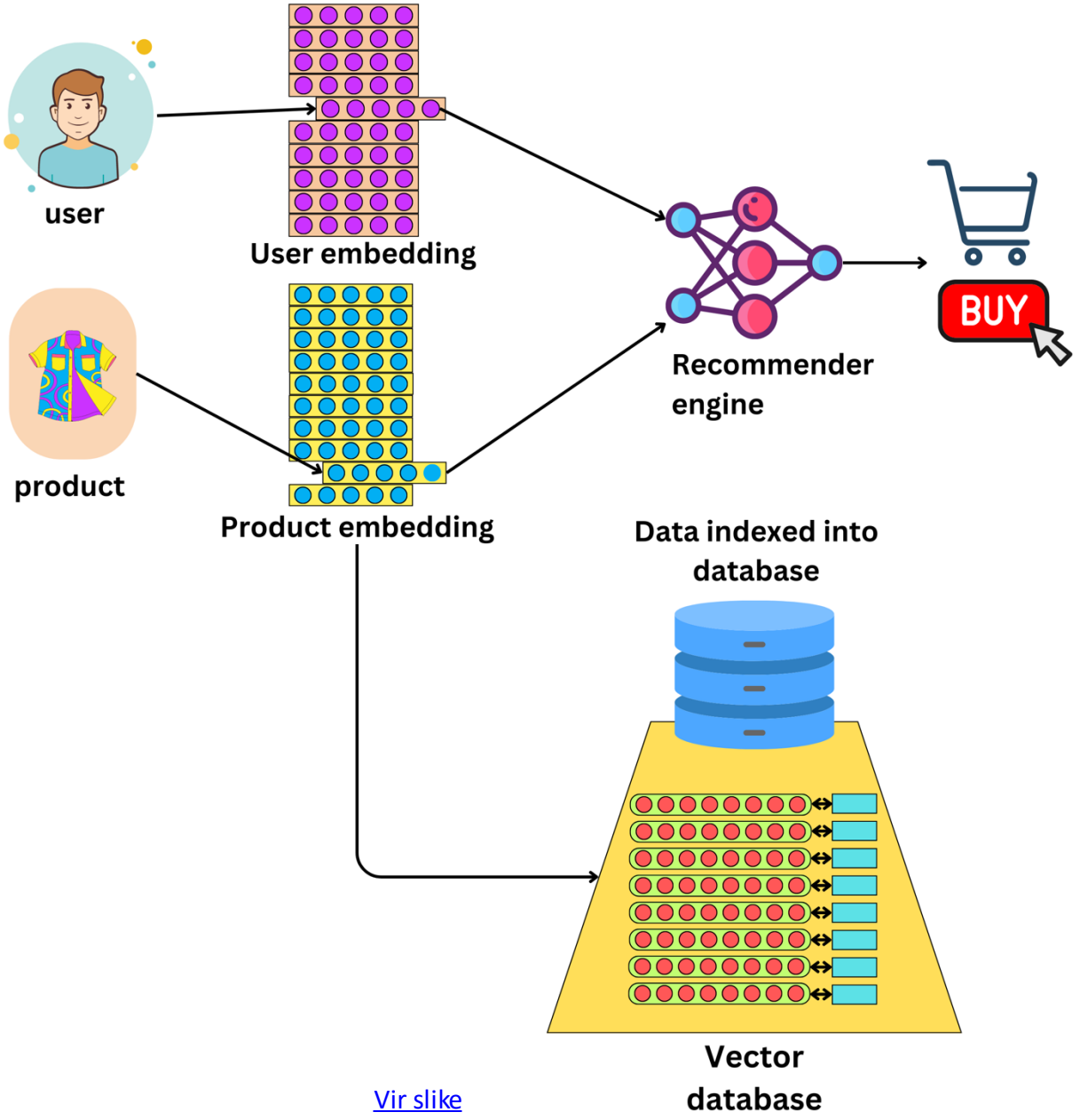
Iskanje po podobnosti



Vir slike

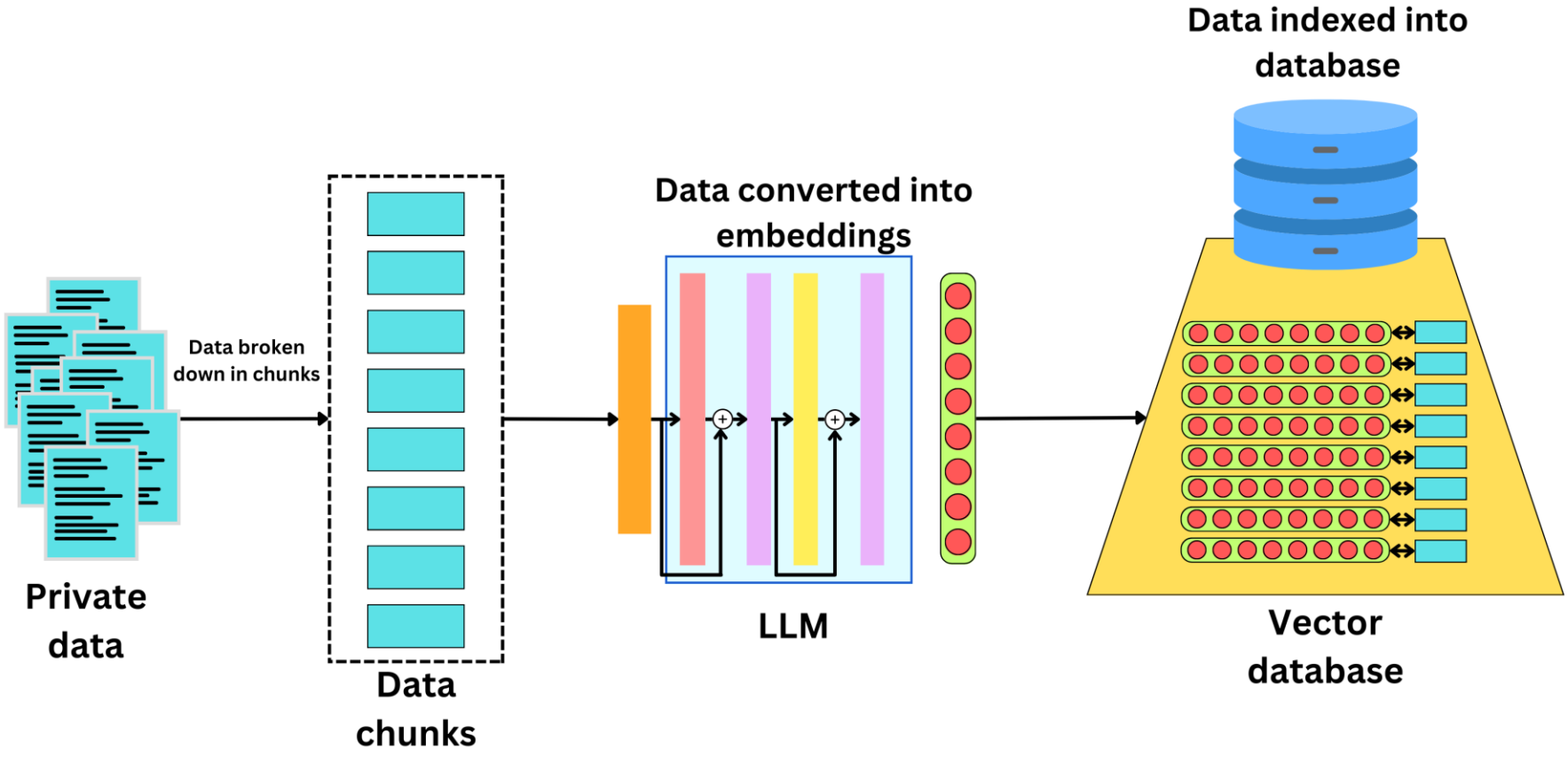
Priporočilni sistemi, e-poslovanje





Vir slike

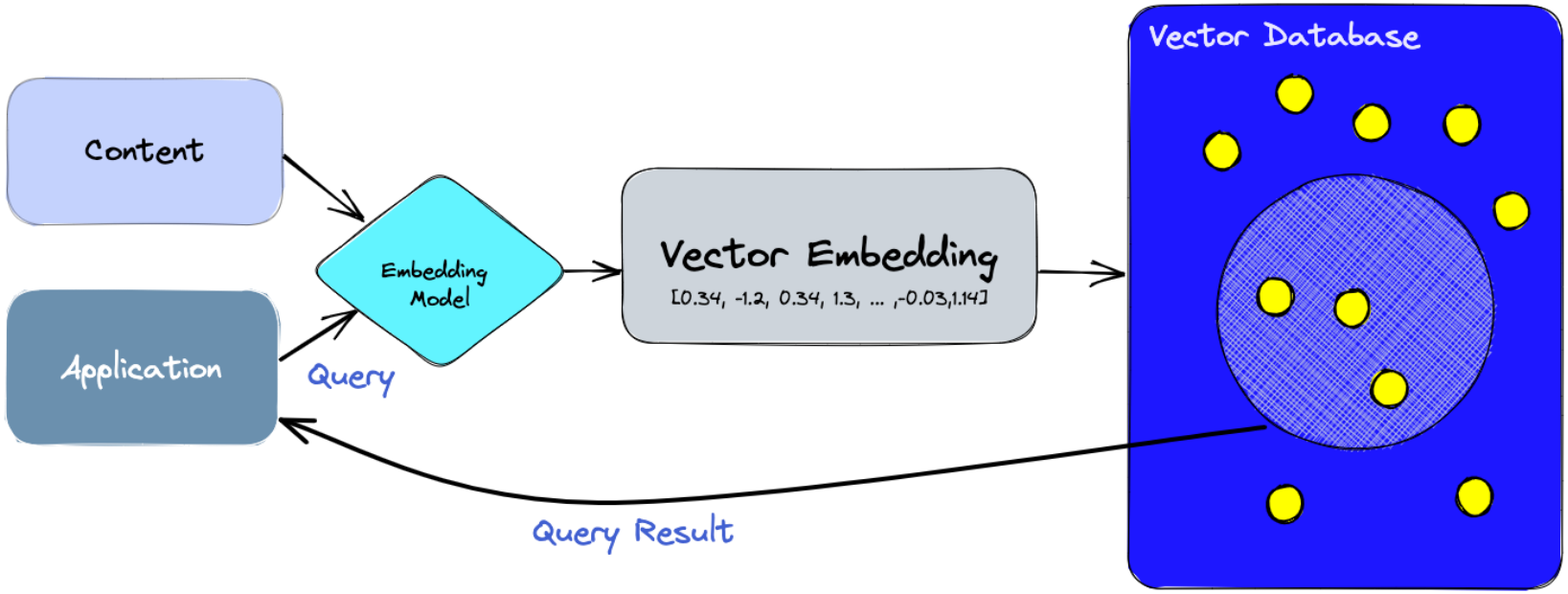
LLM in RAG



Vir slike

Delovanje VPB

- Polnjenje:
  - vektorje generiramo z uporabo ene izmed možnih funkcij za izračun vektorjev
  - izračunane vektorje zapišemo v VPB z referencami na izvirne podatke
- Poizvedovanje:
  - Poizvedbo pretvorimo v vektor (z isto funkcijo).
  - Z uporabo algoritmov v VPB poiščemo najbližje vektorje glede na izbrane metrike.


Content
Application
Embedding Model
Query
Vector Embedding
[0.34, -1.2, 0.34, 1.3, ... ,-0.03,1.14]
Vector Database
Query Result

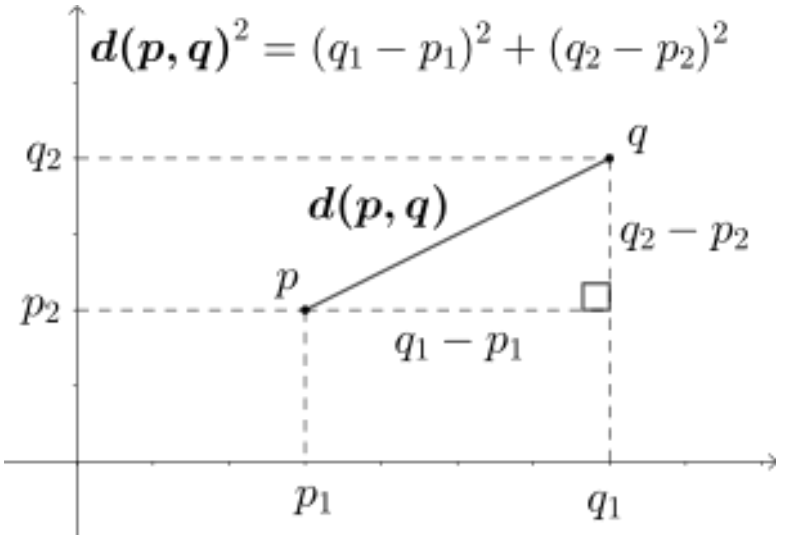
Vir slike

Metrike razdalje

- Matematične funkcije, ki določajo "razdaljo" med dvema vektorjema v vektorskem prostoru.
- Različne metrike razdalje zajamejo različne vidike podobnosti (izbira metrike ključna za specifične aplikacije).
- Najbolj priljubljene metrike razdalje so:
  - Evklidska razdalja
  - Manhattanska razdalja
  - Jaccardova podobnost
  - Skalarni produkt
  - Kosinusna razdalja

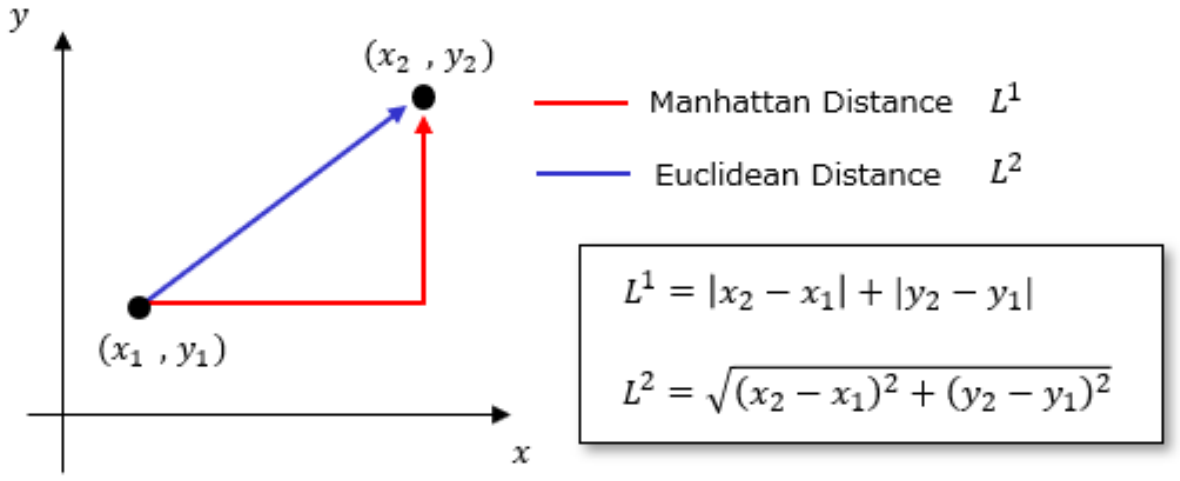
Evklidska razdalja

- Najpogostejša metrika razdalje, imenovana tudi L2 norma.
- Meri razdaljo med dvema točkama v vektorskem prostoru.
- Zelo občutljiva na velikost vektorjev.



Manhattanska razdalja

- L1 norma ali taksimetrična geometrija.
- Sešteje absolutne razlike med istoležnimi koordinatami vektorjev.
- Uporabimo, kadar želimo poudariti, kako se razlikuje vsaka značilnost, ne le kako različne so v celoti.
- Pogosto se uporablja za iskanje slik in finančno analizo.



$$L^1 = |x_2 - x_1| + |y_2 - y_1|$$

$$L^2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

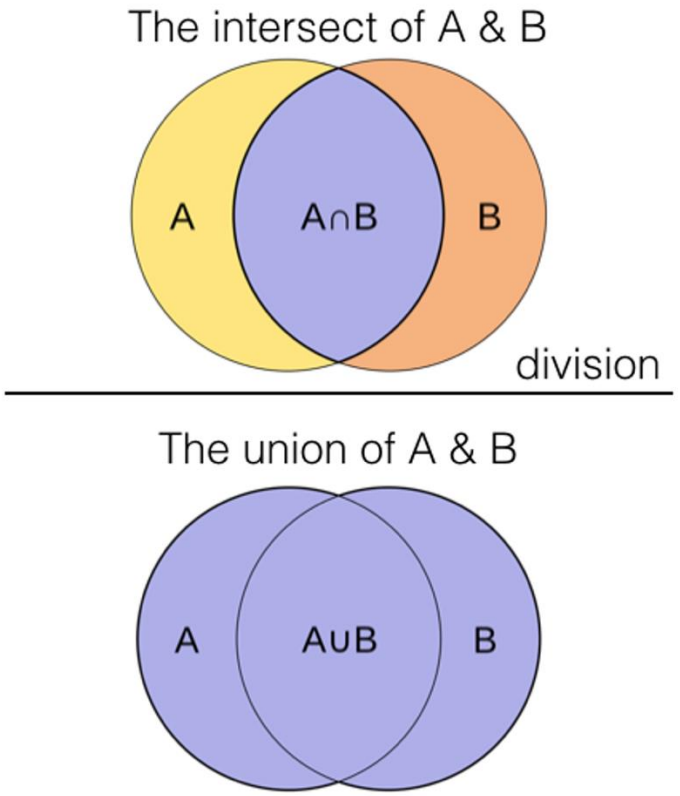
Vir

Jaccardova podobnost

- Jaccardova podobnost je podobnost med dvema množicama:

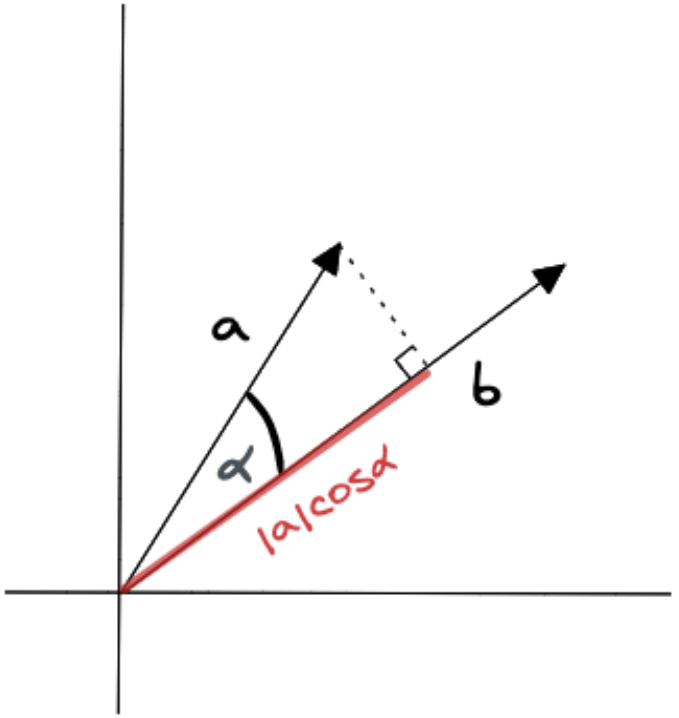
$$J(A,B) = \frac{A \cap B}{A \cup B}$$

- Primeri uporabe:
  - Gručenje in klasifikacija
  - Informacijsko poizvedovanje, npr. izračun podobnosti med dvema besediloma glede na besedišče, ki ga uporabljata.



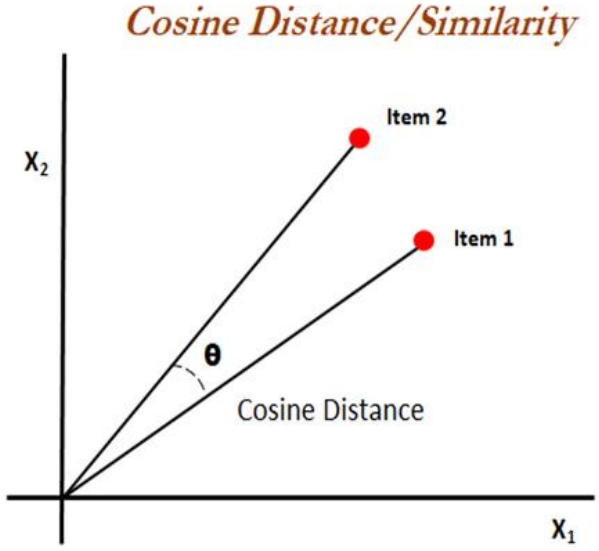
Skalarni produkt

- Meri produkt velikosti dveh vektorjev in kosinus kota med njima.
- Obseg je od -∞ do ∞, kjer pozitivna vrednost predstavlja vektorje, ki kažejo v isto smer, 0 predstavlja ortogonalne vektorje, negativna vrednost pa predstavlja vektorje, ki kažejo v nasprotnih smereh.
- Dolžina vektorjev ključna.



Kosinusna razdalja

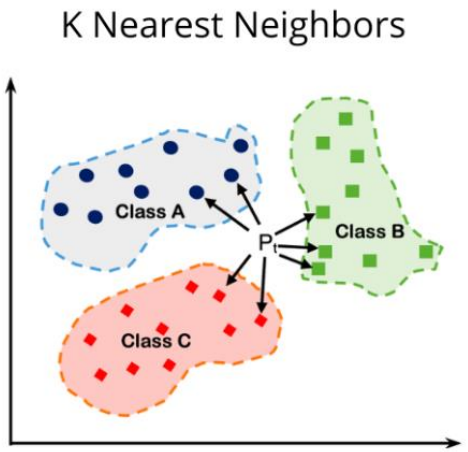
- Meri kot med dvema vektorjema – bolj kot sta vektorja narazen, manj podobne podatke predstavljata.
- Uporabna za iskanje podobnosti med besedili.
- Dolžina vektorjev nepomembna.



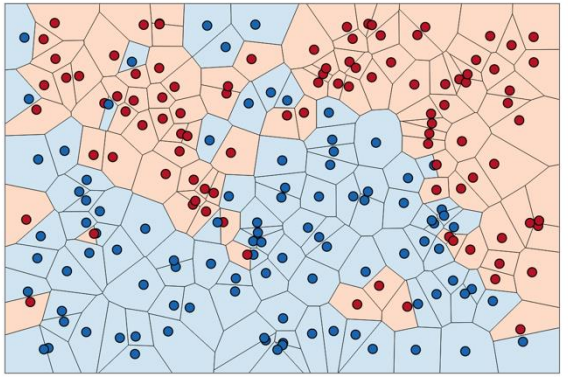
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Iskanje podobnosti

- VPB imajo implementirane optimizacijske algoritme za hitro iskanje, npr.:
  - Iskanje podobnih vektorjev
  - KNN – k najbližjih sosedov
  - ANN – približno najbližji sosed



Vir slike



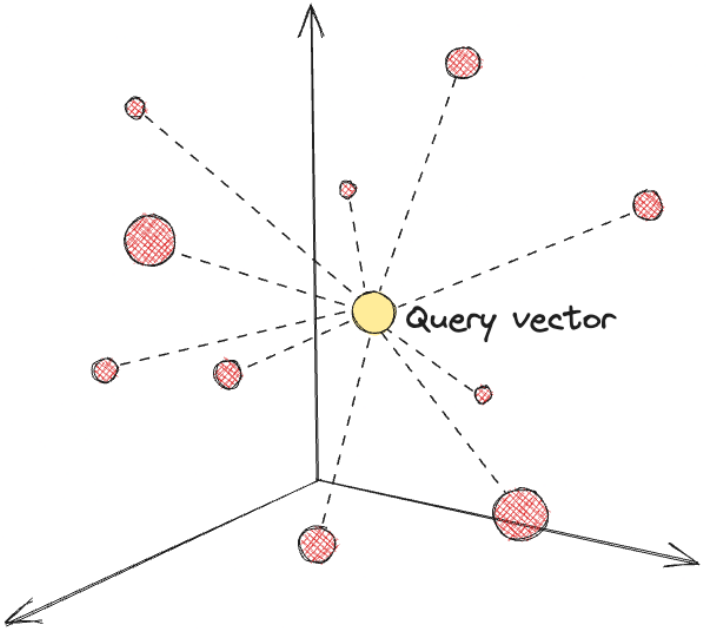
Vir slike

Indeksiranje

- Omogočajo hitro zoženje iskalnega prostora in pospešitev pridobivanja vektorjev.
  - Ravni indeks (*Flat index*)
  - Lokalno občutljivo zgoščevanje (*Locally Sensitive Hahsing - LSH*)
  - Približni najbližji sosed Oh Yeah (*ANNOY - Aproximate Nearest Neighbor Oh Yeah*)
  - …

Ravni indeks

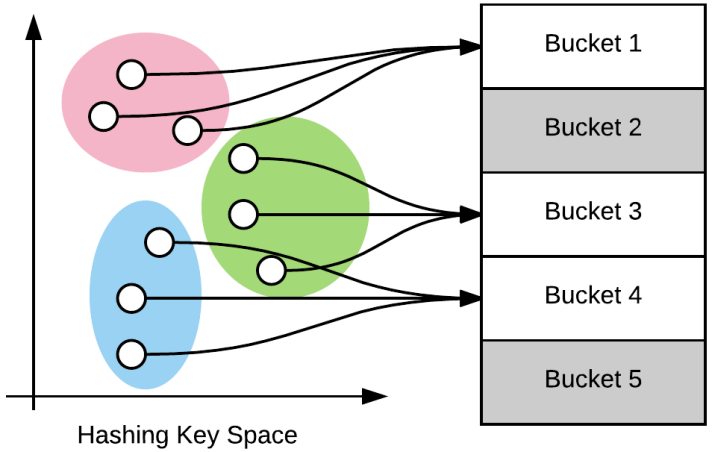
- Ravno indeksiranje (*Flat indexing*) – vektor hranimo v originalni obliki.
- Omogočajo pridobivanje natančnih rezultatov (natančno ujemanje)
- Hiter, natančen, majhna poraba prostora
- Primeren za majhne baze
  <10 000 vektorjev.
- Slabost: linearna časovna komp.



Vir slike

Lokalno občutljivo zgoščevanje (LSH)

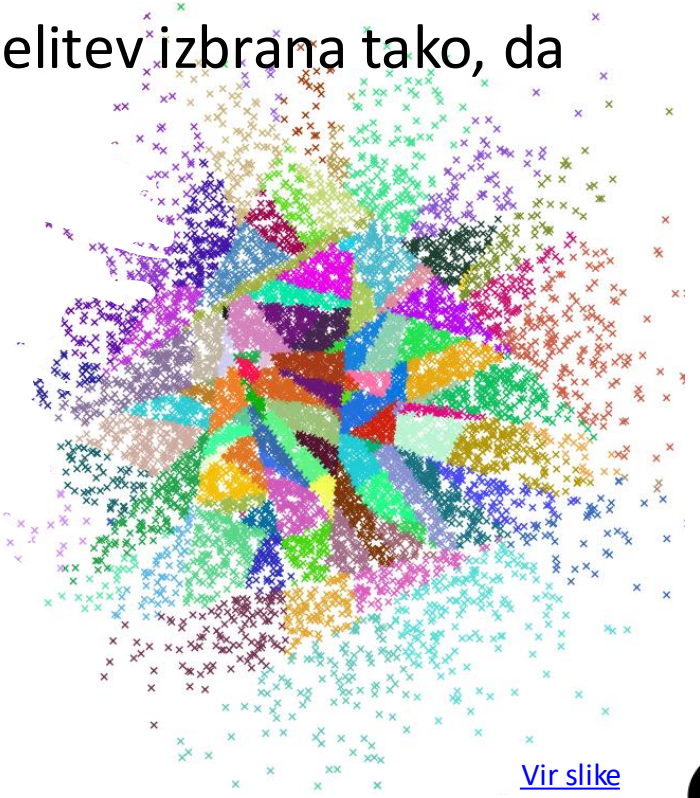
- Namenjen iskanju po visoko dimenzijskem prostoru.
- LSH podobne vektorje z visoko verjetnostjo vstavi v iste skupine (lokalno občutljive hash funkcije).
- Iskanje podobnosti zgolj znotraj istega ali sosednjih skupin



- Omogoča iskanje približno najbližjega soseda (ANN).

ANNOY – Appr. Nearest Neighbors Oh Yeah

- Knjižnica, zgrajena pri Sppotify-u
- Zgradi več dreves, tako da podatkovni nabor rekurzivno razdeli vzdolž naključno izbranih osi. Vsaka razdelitev izbrana tako, da čim bolj uravnoteži drevo.
- Ko so drevesa zgrajena, jih ANNOY indeksira za učinkovito poizvedovanje.
- Iskanje najbližjih sosedov temelji na pregledovanju dreves namesto vsakega posameznega vektorja.



Vir slike

Popularne vektorske baze

