

# OSNOVE PODATKOVNIH BAZ

## RELACIJSKI PODATKOVNI MODEL

### OSNOVNE INFORMACIJE:

- delamo s **podatkovnimi bazami**
- z njimi upravljamo s **sistemi za upravljanje PB: SUPB**
- poznamo relacijske, objektne, dokumentne,..
- relacijo si predstavljamo kot **tabelo**
- dobre lastnosti **relacijskega podatkovnega modela**: formalno definiran, neodvisen od fizičnega shranjevanja podatkov, močni poizvedovalni jeziki

### TERMINOLOGIJA:

- **relacija**: 2D tabela
- **atribut**: poimenovan **stolpec**
- **domena**: dovoljene vrednosti atributov
- **n-terica**: vrstica v tabeli
- **števnost relacije**: število vrstic
- **stopnja relacije**: število atributov
- **relacijska PB**: množica **normaliziranih** relacij z **enoličnimi** imeni

### LASTNOSTI RELACIJE:

- ime relacije **enolično**
- celica tabele ima **natančno** eno **atomarno** vrednost
- ime atributa **enolično** znotraj relacije
- vrednosti atributa so iz **iste domene**
- vrstica je **enolična** znotraj relacije
- vrstni red atributov **nepomemben**
- vrstni red vrstic **nepomemben**

### FUNKCIONALNE ODVISNOSTI:

- poleg funkcionalnih poznamo še **večvrednostne** in **stične** ki so pomembne na višji ravni **normalizacije**
- imamo relacijsko shemo **R**
- dve podmnožici atributov **X** in **Y**
- označimo **X → Y**
- **X** funkcionalno **določa Y** oziroma **Y** je funkcionalno **odvisen od X**
- to se zgodi ko **ne obstajata** vrstici ki se ujemata v **X** in se ne ujemata v **Y**
- označimo z črko **F**

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

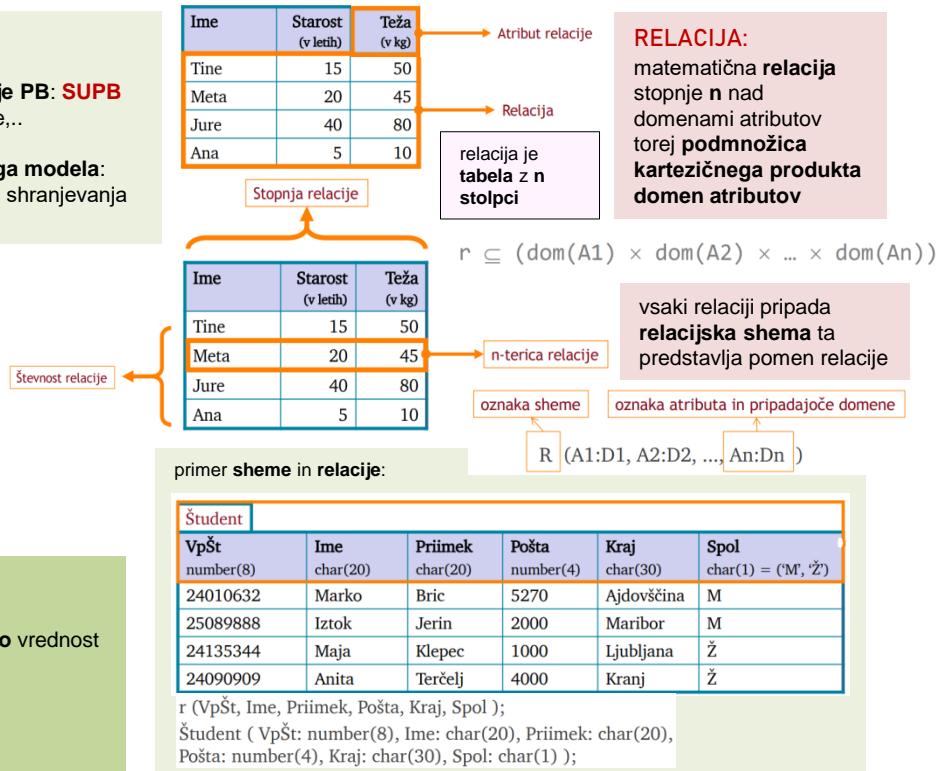
$t.X, u.X, t.Y, u.Y$  označujejo vrednosti atributov **X** oziroma **Y** v n-tericah **t** oziroma **u**.

### KLJUČI RELACIJE:

- katere vrednosti moramo poznati da se lahko **sklicujemo** na relacijo?
- **množica atributov** ki **enolično določajo** vsako **vrstico**

### POGOJA:

- **X → vsi ostali atributi**
- ne obstaja tako **manjša** množica **X**



### primer relacijske funkcionalne odvisnosti:

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:

$$F = \{ Vpšt \rightarrow (Priimek, Ime), (Vpšt, ŠifraPredmeta, DatumIzpita) \rightarrow (OcenaPisno, OcenaUstno) \}$$

Pri celovitosti podatkov **celovitost entitet in povezav**

omejitev **entitete** pomeni da nobena vrednost atributa ki je del **kluča ne sme** biti **NULL**.

### OMEJITVE:

- omejitev **domene**
- pravila za **celovitost** podatkov
- **števnost**
- **splošne omejitve**

**Vrednost NULL:** vrednost **nezdana** ali **nerelavantna** za to vrstico, je odsotnost podatka !

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

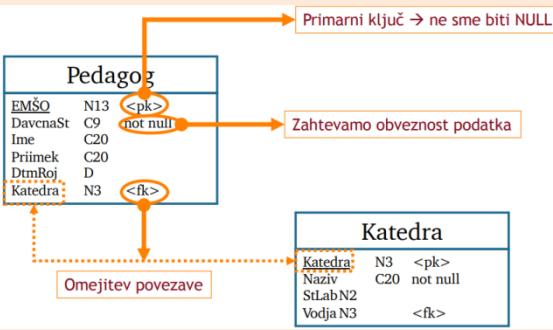
### VRSTE:

- **primarni kluč**: izberemo kandidata za kluč
- **tuji kluč**: referenca na primarni kluč druge tabele
- **kandidat** za kluč: vsi minimalni superkluči
- **superkluč** oz. **nadkluč**: množica atributov ki ni nujno najmanjša



omejitev **povezav** pomeni da se morajo tuji kluči **popolnoma skladati** ali pa bit **NULL** v celoti

### primer omejitve:



### POGLED OZ. VIEW:

- nad **osnovno relacijo**: poimenovana relacija katere n terice so **fizično** spravljene v bazi
- **pogled**: rezultat ene ali več **operacij** nad osnovnimi relacijami z namenom pridobitve **nove relacije**
- to je **navidezna** relacija spremembe vidne takoj
- **dinamično** kreiranje ob povpraševanju
- spremenjanje pogledov ni vedno mogoče
- zagotavljajo varnost: **enkapsulacija**
- prilagojen dostop za **uporabnike**
- **poenostavitev** kompleksnih operacij

### SELEKCIJA:

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

- deluje na **enojni** relaciji
- vrne samo **vrstice** ki zadostajo **predikatu** oz. pogoju

### PROJEKCIJA:

 $\Pi_{s_1, \dots, s_n}(R)$ 

- deluje na **enojni** relaciji
- vrne samo **stolpce** ki so določeni s **seznamom**
- eliminira **duplicate**

### primer selekcije in projekcije:

- 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

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

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

### R ∩ S

- deluje na **dveh** relacijah
- vrne **vrstice** ki v oben
- more se **ujemati** v **atributih** in podatkovnih tipih

### KARTEZIJSKI PRODUKT:

- deluje na **dveh** relacijah
- vrne vse možne **kombinacije** med vrsticami R in S

če združimo **kardezjski produkt** in **selekcijo** dobimo **stik** ki je časovno zelo zahteven s stališča implementacije

### primer pogleda:

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

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
...		

## RELACIJSKA ALGEBRA IN RAČUN

### 1. Relacijska algebra

#### OSNOVE

- visoko nivojski **postopkovni** jezik
- formalno ekvivalentna računu
- relacijsko **popolni** jeziki
- vhod in izhod so **relacije**
- vhodne relacije se ne **spremenijo!**
- **gnezdjenje** izrazov

#### OSNOVNE OPERACIJE:

- selekcija
- projekcija
- kartezijski produkt
- unija
- razlika

#### UNIJA:

- deluje na **dveh** relacijah
- vrne **vrstice oben** relacij
- eliminira duplike
- more se **ujemati** v **atributih** in podatkovnih tipih

 $R \cup S$ 

#### RAZLIKA:

- deluje na **dveh** relacijah
- vrne vrstice ki so v R ampak niso v S
- more se **ujemati** v **atributih** in podatkovnih tipih

 $R - S$ 

#### primer unije in kartezijskega produkta:

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

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

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

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

- stik theta
- ekvistik
- naravni stik
- odprtji stik
- delni stik

## KOLIČNIK:

- deluje na relacijah R in S
- vrne relacijo z atributi C
- C sestavljajo vrstice iz R ki so kombinacija vsake vrstice iz S

R / S

## ODPRTI STIK:

- deluje na relacijah R in S
- vrne tudi vrstice ki nimajo vrednosti v stičnem stolpcu
- imamo levi in desni odprtji stik

R  $\bowtie$  S

## DELNI STIK:

- deluje na relacijah R in S
- vrne vrstice iz R ki nastopajo v stiku z S po stolcu F

R  $\triangleright_F$  S

## THETA STIK:

- deluje na dveh relacijah R in S
- vrne samo vrstice ki zadostajo predikatu F kartezijskega produkta
- pri F gre za primerjave atributov

( $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ,  $=$ ,  $\neq$ ).

Če se omejimo na enakost, dobimo EKVISTIK!

R  $\bowtie_F$  S

## NARAVNI STIK:

- deluje na relacijah R in S
- v bistvu ekvistik ki deluje prek skupnih stolcev
- vzame se samo en primerek skupnega atributa

R  $\bowtie_S$

- AL je seznam agregatnih funkcij
- vsebuje pare agregat in atribut
- GA predstavlja združevalne funkcije
- vrne atribute GA

## AGREGATNE OPERACIJE

- count
- sum
- avg
- min
- max

$\Gamma_{AL}(R)$

GA  $\Gamma_{AL}(R)$

## 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

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

R=AČUN

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

## primer levega odprtega stika:

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

R=OSEBA

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

S=KRAJ

Primerik in ime	PTT	Naziv zač. preb.
Kante Janez	5270	Ajdovščina
Tratnik Jože	5000	Nova Gorica
Mali Mihael		

→

→

## primer količnika:

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

$\Pi_{\text{šifra, kupec}}(R)$

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

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

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

→

## 2. Relacijski račun

### OSNOVE

- nepostopkovni oz. deklarativni jezik
- formalno ekvivalenten algebi
- relacijsko **popolni** jezik

### DEKLARATIVNO POIZVEDOVANJE

- pri algebi določamo **postopek** po katerem prideamo do rezultata
- tukaj določimo le **kaj nas zanima**
- simbolična logika **predikatni račun prvega reda**
- trdite so **predikati**
- besede ob katerih predikati držijo so **konstante**
- imamo lahko **spremenljivke** katerih vrednosti so vezane z **kvantifikatorji**
- domene objektov določene z **funkcijami**
- če predikat vsebuje spremenljivko mora zanj obstajati **domena vrednosti**
- predikate povezujemo z **ne, in, ali**

### VRSTE

- n-terični
- domenski

### TUPLE RELATIONAL CALCULUS

- uporabljamo **n terične** spremenljivke
- spremenljivke kjer **domena** določena z relacijo
- dovoljene vrednosti torej **vrstice table**
- t je spremenljivka
- COND je nabor pogojev kjer iščemo vrstice
- zaloge vrednosti: **relacije**

$\{t \mid \text{COND}(t)\}$

uporaba kvantifikatorja:

- ekstencionalni
- univerzalni
- če kvantifikator potem **vezane** spremenljivke
- drugače so **proste** in so **definirane** torej lahko samo na **levi strani**

primer n teričnega računa:

- Poišči vse podatke o artiklih, ki imajo kritično zalogu manjšo od tri (zaloge < 3):
  - Označba domene! Za iskanje n-terice zahtevamo, da so iz relacije Artikel.
$$\{A \mid \text{Artikel}(A) \wedge A.\text{zaloga} < 3\}$$
- Če nas zanima samo določen atribut (npr. naziv artikla), zapišemo:
 
$$\{A.\text{naziv} \mid \text{Artikel}(A) \wedge A.\text{zaloga} < 3\}$$

S so spremenljivke, a so atributi in F je **formula**

Splošna oblika dobrega izraza je naslednja:

$\{S_1.a_1, S_2.a_2, \dots, S_n.a_n \mid F(S_1, S_2, \dots, S_m)\}$  mzn

dobro definirana formula je sestavljena iz **atomov**

atomi formule:

- R(S<sub>j</sub>): relacija in **spremenljivka**
- S<sub>i</sub>.a<sub>1</sub> \* S<sub>j</sub>.a<sub>2</sub>: atributi spremenljivk in **primerjalni operator**
- S<sub>i</sub>.a<sub>1</sub> \* C: c konstanta iz domene atributa

### DOMAIN RELATIONAL CALCULUS

→ imamo **domenske** spremenljivke

$\{d_1, d_2, \dots, d_n \mid F(d_1, d_2, \dots, d_m)\}$

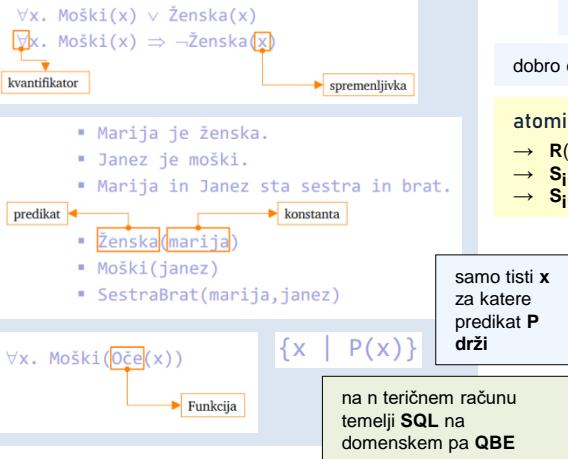
- atomi: R(d<sub>1</sub>...d<sub>n</sub>), d<sub>i</sub> \* d<sub>j</sub>, d<sub>i</sub> \* c
- relacija R, d<sub>i</sub> spremenljivka, C konstanta
- **rekurzivno** gradimo izraze
- zaloge vrednosti: **domena**

rekurzivna gradnja formule:

- atom je formula
- dva atoma povezana z **and or ali not**
- formula z spremenljivko če dodamo **kvantifikator**

varni izrazi:  $\text{dom}(E)$

- z relacijskim računom lahko **neskončne** množice
- **omejimo** tako da morajo biti vse vrednosti v rezultatu iz **domene** izraza E



## POIZVEDOVALNI JEZIK SQL

### OSNOVE :

- skupina ukazov **DDL**: za opredelitev **strukture** podatkovne baze
- skupina ukazov **DML**: za **poizvedovanje** in ažuriranje podatkov
- **DML**: select, insert, delete, update
- enostaven, **nepostopkov**, standardni jezik

### SELECT STAVEK

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

### OSNOVNA SINTAKSA:

- **SELECT**: določa **stolpce** v izhodni relaciji
- **FROM**: določa **tabele** za poizvedbo
- **WHERE**: filtrira vrstice z **pogoji**
- **GROUP BY**: zdržuje vrstice po vrednostih stolpcov
- **HAVING**: filtrira skupine glede na določene pogoje
- **ORDER BY**: določa **vrstni red** vrstic na izhodu

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

### SINTAKTIČNA PRAVILA:

- neobčutljivo st na **caps**
- barve niso predpisane
- imamo **rezervirane** besede

primerjava obeh:

N-terični relacijski račun

$\{H.\text{hotelName} \mid \text{Hotel}(H) \wedge (\exists R)(\text{Room}(R) \wedge H.\text{hotelNo}=R.\text{hotelNo} \wedge R.\text{type}=2 \wedge R.\text{free} = \text{true})\}$

Domenski relacijski račun

$\{hName \mid \exists hNo, hName, hAddress, rNo, rPrice: (\text{Hotel } (hNo, hName, hAddress) \wedge (\text{Room}(rNo, hNo, 2, rPrice, true)))\}$

Relacijska algebra

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

SQL

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

primer :

- Izpiši starost delavcev ob zaposlitvi

```
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
Christian	Koblick	1954-05-01	1986-12-01	32.6082
Miriam	Matsik	1964-01-21	1988-09-17	24.6658

## NAČINI ISKANJA Z SELECT:

- če hočemo izpisat vse: **SELECT \***
- če hočemo specifične: **SELECT atribut<sub>1</sub>, atribut<sub>2</sub>...**
- če hočemo različne: **SELECT DISTINCT atribut<sub>1</sub>, atribut<sub>2</sub>...**
- če hočemo preimenovati dodamo: **AS**
- če imamo pogoj imamo: **WHEN atribut pogoj**
- če želimo vrstni red: **ORDER BY desc** (če pada)

## WHEN:

- **between**: med dvema vrednostima atributa
- **in**: samo tiste vrednosti atributa
- **like**: podobnost v črkah
- **is null**: tisti ki nimajo podatka
- **primerjalni operatorji**

## AGREGACIJSKE OPERACIJE:

- **count**: število vrednosti v določenem stolpcu
- **sum**: seštevek vrednosti
- **avg**: povprečje vrednosti
- **min**: najmanjša vrednost
- **max**: največja vrednost

## AGREGIRANJE:

- agregiranja delujejo na enem stolpcu in vrnejo eno vrednost
- **COUNT MIN in MAX** lahko ne numerične vrednosti
- **AVG** in **SUM** samo numerične vrednosti
- vse agregacije najprej **odstranijo null** v stolpcu razen **COUNT**
- **COUNT** presteje čisto vse vrstice tudi null in duplike
- **DISTINCT** vpliva na **avg** in **sum** ne pa na **min** in **max**
- če želimo prešteti **različne COUNT ( DISTINCT atribut )**
- agregarne funkcije le v **SELECT** ali pa **HAVING** sklopu
- če v **SELECT** potem potrebujemo tudi **GROUP BY**

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

## GROUP BY:

- stolpci v **SELECT** tudi v **GROUP BY**
- izjema stolpc, ki nastopajo samo v **agregarnih** operacijah
- najprej se izvede **WHERE** združevanje na **preostalih** vrsticah
- pri združevanju **NULL** vrednosti obravnavane kot **enake**

## HAVING:

- stolcoli v **HAVING** sklopu morajo biti tudi v **SELECT** sklopu ali **agregatih**
- postavlja **pogoje** katerim morajo zadoščati skupine v **rezultatu**

## primer agregacij:

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

presteje število različnih delovnih mest

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

minimalna, maksimalna in povprečna plača

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

```
having count(*) > 9200;
```

presteje vse ženske rojene v vseh vrednostih let ki jih imamo v tabeli in ta count grupiramo po teh letih

Rojeni leta	Število	YEAR(birth_date)	count(*)
1952	8502	1954	9385
1953	9104	1955	9267
1954	9385	1956	9229
1955	9267	1958	9312
1956	9229	1959	9286
1957	9069	1960	9383
1958	9312	1961	9269
1959	9286	1962	9216
1960	9383		

če dodamo še to vrstico potem izpišemo samo tiste ki jih je več kot 9200

WHERE CustomerName LIKE 'a%"	Finds any values that start with "a"
WHERE CustomerName LIKE '%a"	Finds any values that end with "a"
WHERE CustomerName LIKE "%or%"	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%"	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%"'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%'	Finds any values that start with "a" and ends with "o"

## primer select stavkov:

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

delavci ki rojeni med določenimi datumimi

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

podatki o delovnih mestih delavcev na določenih vrednostih atributa title

```
select * from employees
where last_name like 'B%' and
birth_date like '_-6-----01';
```

delavci ki imajo priimek na B in so rojeni prvega v mesecu v 60' letih

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

zaposleni brez podatka o določenem atributu

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

zaposleni urejeni po priimku naraščajoče in letu rojstva padajoče

## GNEZDENJE SELECT STAVKA:

- lahko se pojavi v **WHERE** ali **HAVING drugega SELECT** stavka
- uporaba **order by** in ugnezdenem stavku **nemiselna**
- lahko zajema samo **en stolpec** razen če uporabimo **EXISTS**
- imena stolpcov v ugnezdenem stavku se nanašajo na tabele iz **ugnezdenega** ali **zunanjega** stavka
- če je ugnezden **SELECT** stavek operand v primerjavi se mora nahajati na **desni** strani enačbe
- lahko uporabljamo operatorja **ANY, SOME in ALL**
- **ANY** in **SOME** vrne **TRUE** če je to res za **vsaj** eno vrednost poizvedbe
- **ALL** vrne **TRUE** če res za **vse** vrednosti
- če rezultat poizvedbe **prazen** **ALL** vrne **TRUE** in **ANY** in **SOME FALSE**
- **EXISTS** samo v ugnezdenih in vrača **TRUE** če obstaja **vsaj ena** vrstica v tabeli ki je rezultat ugnezdenne poizvedbe in **FALSE** sicer
- tako lahko izvajamo poizvedbe v več tabelah
- v poizvedbah ki vračajo stolpc različnih tabel moramo uporabljati **stik**

primeri gnezdenja select stavek:

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

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

primer ko izbiramo iz tabele  
ki že je rezultat neke  
poizvedbe

vrne samo imena in priimek zaposlenih ki smo jih zbrali po ključu samo tistih ki so na delavnem mestu manager iz tabele titles

primer uporabe ANY  
izpišemo ime, primerek in datum rojstva tistih delavcev kjer so vsi ostali delavci mlajši od njih

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

pogojni stavek je torej  
rojstni datum mora biti  
manjši od vseh ostalih

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

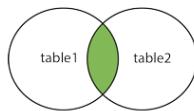
izpišemo ime in priimek tistega  
delavca ki ima največjo plačo

```
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  
)
```

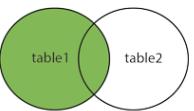
primer uporabe EXISTS  
za ločevanje med  
istoimenskimi stolpcii  
uporabljamo sinonime

izpišemo vse  
zaposlene kjer ne  
obstaja nihče ki  
zaposlen pred njim

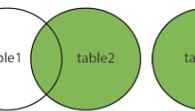
INNER JOIN



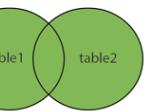
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



limit nam pri urejanju pomaga če  
želimo omejiti število vrstic na izhodu

izpišemo ime in priimek prvih 3 moških  
in prvih 3 žensk ki so bili prvi zaposleni

POIZVEDBA PO VEČ TABELAH:

→ uporabljamo stik

→ INNER JOIN: v bistvu inner join vrne samo tiste ki imajo enako vrednost v obeh tabelah

→ NATURAL JOIN: karteziski produkt tabel

→ LEFT / RIGHT JOIN: vrne vse vrstice iz leve / desne tabele in tiste ki se ujemajo iz druge tabele desne / leve

→ FULL OUTER JOIN: vrne vse vrstice kjer se ujemajo ali v desni ali v lev

na čem izvajamo JOIN:

- lahko napišemo ON in podamo enakost
- lahko napišemo USING in ne rabimo pogojnega stavka samo ime atributa ampak lahko uporabljamo samo kadar imena enaka shortcut !

OPERACIJE NAD MNOŽICAMI

- CORRESPONDING BY: operacija se izvede samo nad poimenovanimi stolpcii
- CORRESPONDING brez BY člena: operacija se izvede nad skupnimi stolpcii
- ALL: rezultat vključuje tudi duplike

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

primeri unije:

```
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

- seznam column ni obvezen
- vpisati moramo vrednost za vse razen tiste ki imajo default vrednost
- seznam vrednosti mora ustrezati column seznamu

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

primer vstavljanja v tabelo

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

primer  
vstavljanja iz  
drugih tabel

Shema relacije:  
departments (dept\_no, dept\_name)

## UPDATE STAVEK

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

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

posodobimo plačo vsem  
zaposlenim za 10%

## DELETE STAVEK

```
DELETE FROM TableName  
[WHERE searchCondition]  
delete from salaries  
where emp_no not in (
```

```
    select emp_no  
    from salaries  
    where to_date = '9999-01-01'
```

izbrišemo plače delavcev ki niso več zaposleni tam

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

imamo različne omejitve

- Obveznost podatkov
- Omejitve domene (Domain constraints)
- Pravila za celovitost podatkov (Integrity constraints)
  - Celovitost entitet (Entity Integrity)
  - Celovitost povezav (Referential Integrity)
- Števnost (Multiplicity)
- Splošne omejitve (General constraints)

## PODATKOVNI TIPI

Numeric  
Integer Types: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT  
Fixed-Point Types: DECIMAL, NUMERIC  
Floating-Point-Types: FLOAT, DOUBLE  
Bit-Value Type: BIT

String  
CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, SET.

Date & Time  
DATE, TIME, DATETIME, TIMESTAMP, YEAR.

## obveznost podatkov

`emp_no numeric(5) NOT NULL`

## omejivje domene

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

`gender CHAR NOT NULL`

`CHECK (gender in ('M', 'F'))`

`gender Tgender NOT NULL`

ISO standard podpira kreiranje 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 stolpcov.

`UNIQUE(last_name)`

`UNIQUE(last_name, first_name)`

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

`searchCondition` lahko vsebuje **iskalno tabelo**:

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

`DROP DOMAIN DomainName`

`[RESTRICT | CASCADE]`

**ukinemo** neko domeno

Pove, kako ravnati, če je domena trenutno v uporabi

**celovitost povezav**: če ima **tuji** ključ neko vrednost potem se more ta vrednost nahajati v **primarnem** ključu povezane tabele, če to ni **izpolnjeno** se zavrne z opcijami:

- CASCADE
- SET NULL
- SET DEFAULT
- NO ACTION

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

**primer**

## USTVARJANJE TABELE

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

- **relacije** in drugi **podatkovni objekti** so v nekem **okolju**
- vsako okolje ima **enega ali več katalogov**
- **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**

`CREATE SCHEMA [Name |  
AUTHORIZATION CreatorId ]`

`DROP SCHEMA Name [RESTRICT | CASCADE]`

`CREATE TABLE TableName`

`{colName dataType [NOT NULL] [UNIQUE]`

`[DEFAULT defaultOption]`

`[CHECK searchCondition] [...]`

`[PRIMARY KEY (listOfColumns),]`

`{[UNIQUE (listOfColumns),] [...]}`

`{[FOREIGN KEY (listOfFKColumns)`

`REFERENCES ParentTableName [(listOfCKColumns)],`

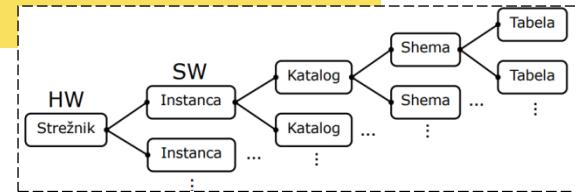
`[ON UPDATE referentialAction]`

`[ON DELETE referentialAction] ] [...]`

`[CHECK (searchCondition)`

**KREIRANJE  
TABELE** in  
zraven primer

**CASCADE** pomeni da se  
kaskadno brišejo objekti, če  
eno brisanje ne uspe se  
**zavrne celo** operacija



`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)`

## ALTER STAVEK

- dodajamo stolpce
- izbrisemo stolpce
- dodajamo in ukinemo **omejivje**
- določamo in ukinemo **privzete vrednosti**
- spremjamamo **podatkovne tipove** stolpcov

## BRISANJE

`DROP TABLE TableName [RESTRICT | CASCADE]`

## TRANSAKCIJE

- **atomarna**
- deluje z enim ali več ukazi **SQL**
- spremembe so drugim transakcijam **skrite**, dokler transakcija ni **končana**
- lahko se zaključi eksplicitno z ukazom **ROLLBACK** in **COMMIT**
- lahko se zaključi implicitno z programom, kjer **klicana**
- ne moremo jih gnezdit

## INDEKSI

- urejanje tabel po stolpcih
- ključni za **hitro poizvedbo**
- vedno narejen indeks po **primarnih ključih**

## POGLEDI

- **navidezne** tabele
- v ozadju vedno **select** stavek

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

**READ ONLY**: samo transakcije ki iz baze **berejo**, torej **SUBP** dovoli **INSERT**, **UPDATE** in **DELETE** samo nad začasnimi tabelami

**ISOLATION LEVEL**: stopnja interakcije ki je dovoljena by **SUBP**, varen je samo **SERIALIZABLE**

- **DIRTY READ**: pri izolaciji **UNCOMMITTED**
- **NON REPEATABLE READ**: pri **UNCOMMITTED** in **COMMITTED**
- **PHANTOM READ**: pri vseh razen pri tisti edini ki je varna
- prve dve **lock based** i have no idea kaj to pomeni not gonna lie

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

`ALTER TABLE salaries`  
`DROP CONSTRAINT PrevecSprememb;`

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

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

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	●	●	●
Read Committed		●	●
Repeatable Read			●
Serializable			

Samodejno zaključevanje transakcije:

SET autocommit = {0 | 1}

- INITIALLY IMMEDIATE – ob začetku transakcije;
- INITIALLY DEFERRED – ob zaključku transakcije.

SET CONSTRAINTS

{ALL | constraintName [, . . . ]}  
{DEFERRED | IMMEDIATE}

**SET CONSTRAINTS**  
nastavi omejitve

ALTER TABLE tab1 ADD CONSTRAINT fk\_tab1\_tab2  
FOREIGN KEY (tab2\_id)  
REFERENCES tab2(id)

DEFERRABLE

pomeni da lahko  
spreminjam  
**zakasnelost**

INITIALLY IMMEDIATE;

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

## IZVEDBENI OBJEKTI

- sprožilci: sprožijo se pred ali po insert, delete ali update
- podprogrami in funkcije
- hitro vendar poslovna logika na podatkovno raven

### TEMPLATE in PRIMERI

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

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

```
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;
```

```
create function hello (empno INT(5))
returns char(50) [deterministic]
return concat('Pozdravljen ',  

  (select first_name  

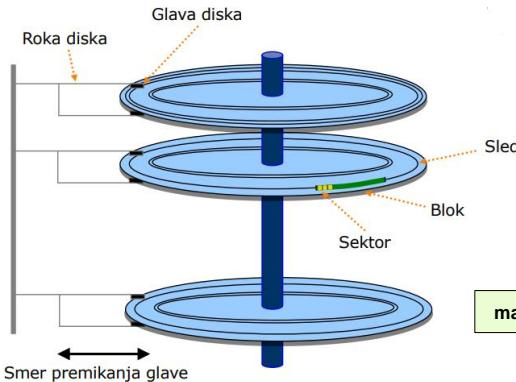
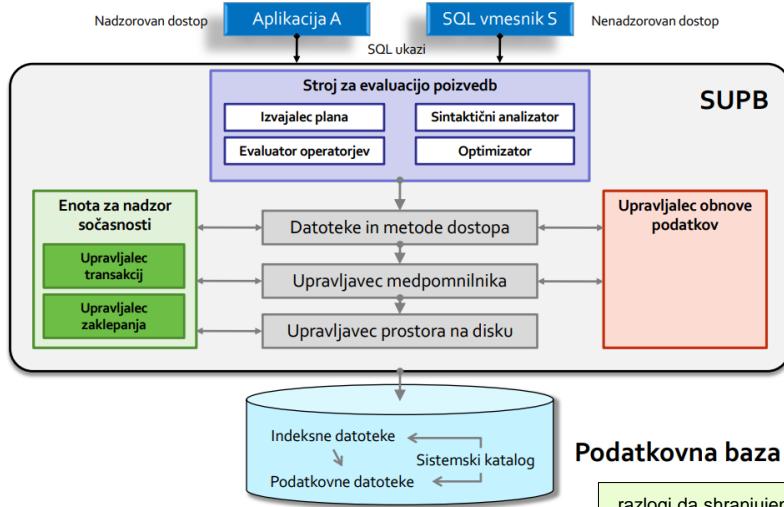
  from employees  

  where emp_no = empno));
```

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

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

## FIZIČNA ZGRADBA

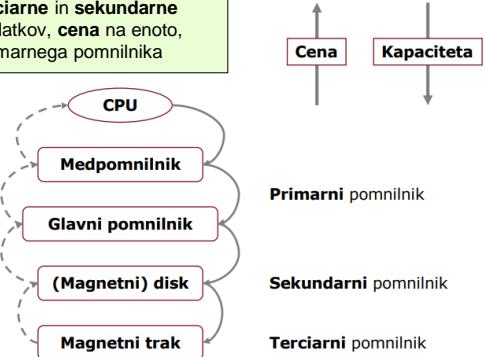


## Podatkovna baza

razlogi da shranjujemo na terciarne in sekundarne  
pomnilnike so **obstojnosc** podatkov, **cena** na enoto,  
**omejen naslovni prostor** primarnega pomnilnika

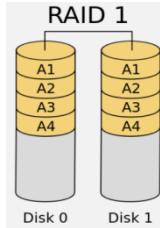
## BUFFER MANAGER

- prenos strani iz diska v **medpomnilnik**
- file manager** najde stran z zapisom

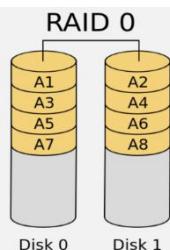


## MAGNETNI DISK

- povprečen **dostopni čas**: odvisen od iskalnega časa, **rotacijske zakasnitve**, časa prenosa
  - **organizacija** podatkov bistveno vpliva na dostopni čas
  - **HDD in SSD**
  - imamo **diskovna polja RAID**: za **porazdelitev** za boljšo učinkovitost in **podvajanje** podatkov za večjo zanesljivost
  - različni raid razmerje med **učinkovitostjo** in **zanesljivostjo**

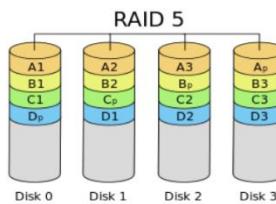
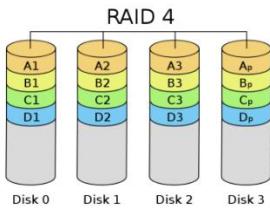


**RAID 1:** Dve kopije podatkov na dveh diskih, najdražja rešitev, to je **mirrored** disk. Vsako pisanje pomeni pisanje dvakrat. 50% izraba prostora, pisanje je **paralelno** možnost napake.



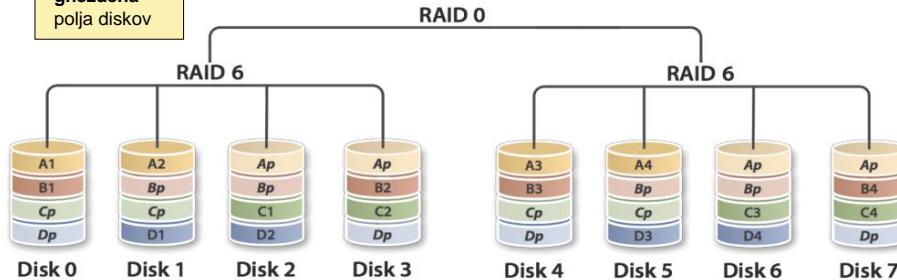
imamo  
**pomembn**  
parametre

- *Space Efficiency* - Izkoriščenost prostora
- *Fault Tolerance* - Št. diskov, ki lahko brezizgubno odgovedajo na napake
- *Array failure rate* - Verjetnost, da odpove polje
- *Read performance* - učinkovitost branja
- *Write performance* - učinkovitost pisanja



obstajajo tudi  
**gnezdna**  
polia diskov

**RAID 6:** porazdeljen z porazdeljeno redundanco. Enota porazdelitve je blok. Redundanca je **paritetni** bit, uporablja dva paritetna bita, oba sta **porazdeljena**. Neobčutljiv na hkratno odpoved do dveh diskov.



uporaba **datotečnega sistema**:  
disk space manager lahko uporablja datoteke iz OS  
saj je celotna podatkovna baza na eni ali več  
datotekah, veliko podatkovnih baz ima **svoj sistem**

**1. vzdrževanje seznama prostih blokov:** kazalec na prvi blok seznama se shrani na znano lokacijo na disku

## BUFFER MANAGER

- za vsak v medpomnilniku **dve spremenljivki**
  - **pin count**: število trenutnih uporabnikov na strani
  - **dirty**: ali je stran spremenjena ali ne **true / false**
  - začetno stanje **0** in **false**

POSTOPEK

Pojavi se zahteva po določeni strani upravljač pogleda če se stran **nahaja** v kakšnem **okvirju** vrne pomnilniški **naslov** in poveča pin count za 1. Sicer izbere okvir za **zamenjavo** in če je **dirty bit** okvirja **true** se stran prepriče na **disk**. Stran se prenese iz diska v okvir določen za zamenjavo in pin count okvirja gre na 1. Ko sistem določi da se je stran **sprostila** postavi pin count na **0** in če dirty bit true prepriče stran **nazaj na disk**.

če so vsi okvirji zasedeni izberemo stran ki pin count 0, če več **strategija** izbire. Če take ni čakamo na sprostitev

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

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

## KONFLIKTNE SPREMEMBE

- če neko stran zahteva več **neodvisnih transakcij**
- reši te z **zaklepanjem**
- za te protokole zaklepanja skrbi **upravljalec transakcij**
- vsaka transakcija lahko pridobi **deljeno** ali pa **ekskluzivno** zaklepanje preden se lahko stran bere ali spreminja
- ekskluzivno zaklepanje **ne more** biti odobreno **dvema** naenkrat

## ZAMENJAVA STRANI

- strategija zamenjave strani v medpomnilniku močno vpliva na učinkovitost
- različne strategije za različne situacije
- **LRU**: least recently used, vrsta kazalcev na okvirje z pin count 0
- **urne zamenjave**: clock replacement, vsaka stran ima reference bit R in pin count P, ko P postane nič postane R ena. Če je P več od 0 ali R 1 potem premaknemo kazalec naprej sicer stran uporabim za zamenjavo.

## ORGANIZACIJA DATOTEK

### BASIC FACTS

- podatkovna baza na **sekundarnem** pomnilniku v **eniji** ali **več datotek**
- vsaka datoteka zajema enega ali več **zapisov**
- zapis sestavlja **polja**
- zapis običajno označujejo **entitete**, polja pa njihove **attribute**
- **fizični zapis** lahko zajema **več logičnih**
- večji **logični zapis** lahko čez več **fizičnih**
- fizični zapis je v bistvu koncept **strani**

### ARTIKEL

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

Stran  
1  
2

### NEUREJENA

- imenujemo tudi **kopica** oz. **heap**
- najenostavnnejša organizacija
- zapisi shranjeni v **vrstnem redu** kot so **dodani**
- nov zapis dodan na **zadnjem** stran datoteke
- če ne dovolj prostora dodamo **novu** stran
- zelo učinkovito **dodajanje**
- uporabljamo za **masovni vnos**
- **neučinkovitost iskanja**
- brisanje pušča prazen prostor na straneh

### DINAMIČNE HASH DATOTEKE:

Uporabljamo ker postane datoteka premajhna, uporabljamo različne tehnike **razprševanja**

### Razširljivo razprševanje:

→ delamo po naslednjem ključu:

- Strani kreiramo po potrebi. V začetku gredo zapisi v prvo stran.
- Ko stran polna, razdelimo glede na prvih i bitov, kjer velja  $0 \leq i < b$  (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 21 kazalci.
- Vsaka stran ima tudi lokalno globino, ki pove, pri katerem i dobimo naslov te strani.

uporaba razpršenih datotek ni primerna za iskanje po **vzorcu**, iskanje po nizu **vrednosti**, iskanje po polju ki ni **hash polje**, spremenjanje zapisov je enostavno, razen v primerih ko spremenišmo hash polje

Logični pogled  
Tabela

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

Fizični pogled  
datoteka

SUPB naredi preslikavo logičnega zapisa **fizični** ko uporabnik zahteva neko poizvedbo, poišče **fizični zapis** in ga prepiše v **primarni pomnilnik** ali pa **medpomnilnik**

### DATOTEČNA ORGANIZACIJA

- **fizična urejenost** podatkov v **zapise** na strani na sekundarnem pomnilniku
- **kopica** oz. **neurejena** datoteka
- **zaporedno** urejena datoteka
- **razpršena** datoteka
- imamo različne **metode dostopa** ki so odvisne od datotečne organizacije

### UREJENA

- zapisi na datotekah urejeni na enem ali več poljih zato **zaporedna** datoteka
- možno je **binarno iskanje**
- **iskanje** učinkovito
- **neučinkovito** dodajanje in brisanje ker moramo vzdrževati **vrstni red**
- dodajanje na začetek **velike** datoteke posebaj problematično
- uporabljamo dodatno **neurejeno** datoteko oz. **overflow file**
- nov zapis dodan tja, ta se **periodično** prepiše v urejeno, najprej isčemo v urejeni in nato še v linearno v neurejeni

### R A Z P R Š E N A

- razpršene v skladu z **hash funkcijo** ki za vsak zapis izračuna **naslov** strani oz. **bloka** na disku kamor sodi glede na vrednost **hash polja**
- **iskanje** zapisa na strani se izvede v primarnem pomnilniku
- za večino zapisov rabimo prebrat samo **eno stran**
- hash funkcija more zagotavljati **enakomerno** porazdeljenost
- ponavadi deljenje po **modusu**

$f_{\text{hash}}(P) \rightarrow \text{naslov strani};$   
P: vrednost hash polja.

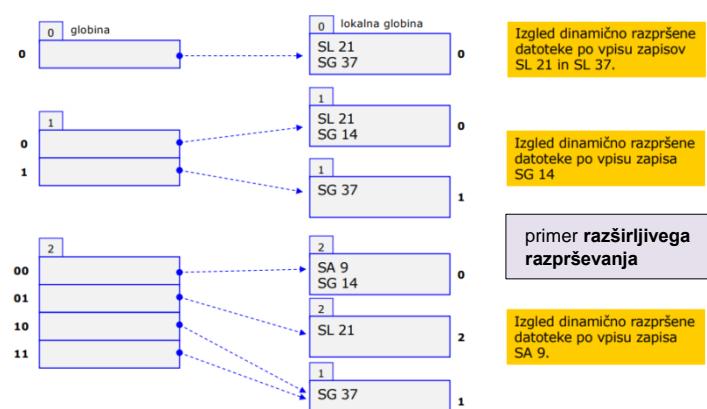
### SLABOSTI HASH DATOTEK:

Zaloga vrednosti hash polja po navadi **večja** od števila naslovov ki jih lahko vrne hash funkcija. Ko hash funkcija za nek zapis vrne naslov strani, ki je polna, pride do **kolizije**

### reševanje kolizije

vse te tehnike so **statične**  
poznamo tudi **dinamične**

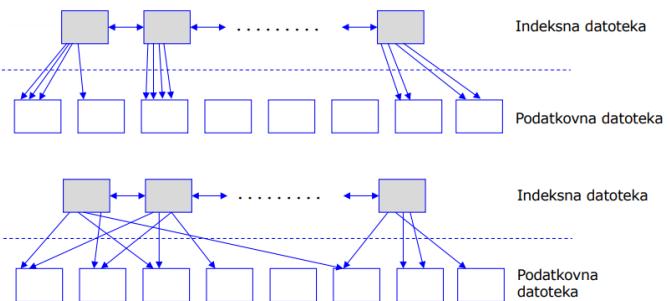
- Odprtvo naslavljanje (*open addressing*)
- Nepovezane dodatne strani (*unchained overflow*)
- Povezane dodatne strani (*chained overflow*)
- Večkratno razprševanje (*multiple hashing*)



# INDEKSI

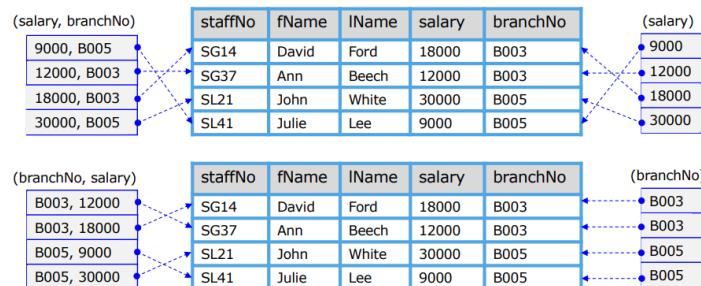
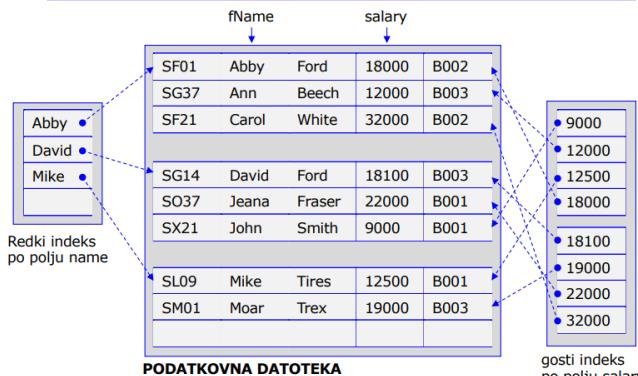
## TERMINOLOGIJA:

- **podatkovna datoteka:** osnovna datoteka z podatki
- **indeksna datoteka:** indeks sestavlja iskalni **ključ** ter **kazalec** na zapis v podatkovni datoteki
- **iskalni ključ:** indeksno polje sestavljeno iz vrednosti polj po katerih je datoteka **indeksirana**
- **primarni indeks:** indeks po poljih ki vsebujejo primarni ključ, vsaka datoteka ima to ali **indeks gruče**
- **sekundarni indeks:** vsak drug indeks ki ni po primarnem ključu in lahko jih imamo več
- **redki indeks:** indeksna datoteka vsebuje kazalce le na določene zapise v podatkovni datoteki po navadi po en zapis za vsako stran v podatkovni datoteki
- **gosti indeks:** indeksna datoteka na vse zapise v podatkovni datoteki



**INDEX GRUČE:**  
indeks po polju, po katerih je urejena podatkovna datoteka

**SESTAVLJEN INDEKS:**  
indeksi s sestavljenim iskalnim ključem, uporabljamo ga za kombinacije polj po katerih pogosto iščemo

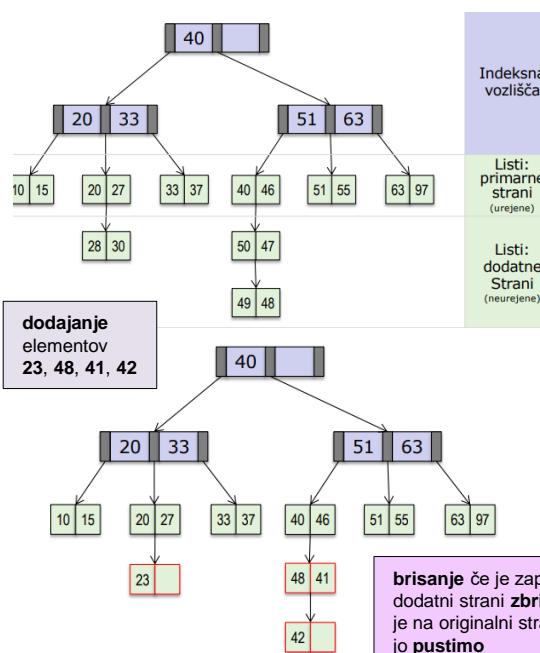


## DREVESNO INDEKSIRANJE:

- učinkovito pri **intervalnem iskanju**, dodajanju in brisanju
- iskanje po enakosti bolje pri **hash indeksiranju**

## I S A M

- **statična struktura**
- učinkovito ko se podatkovna struktura **ne spreminja** pogosto
- ni učinkovit če se datoteke pogosto širijo ali krčijo
- **gradnja:** vse strani so v listih urejene zaporedno in po iskalnem ključu, potem pri vstavljanju delamo nove strani če je stran kamor sodi podatek **polna**

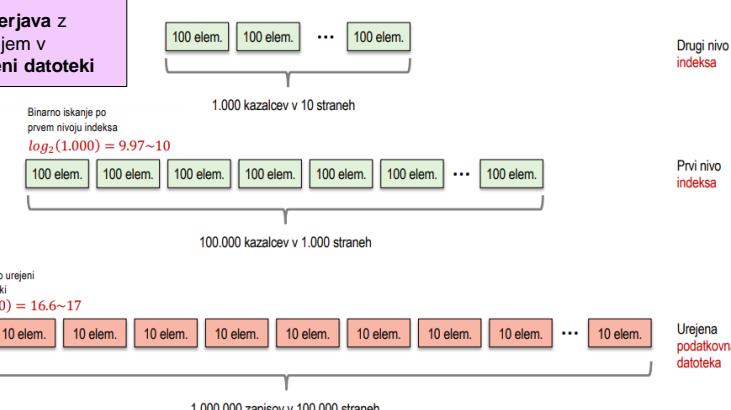


Število I/O operacij =  $\log_F N$ , kjer je N število primarnih strani listov, F število otrok vsake indeksne strani.

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)



problem če veliko dodajanja se lahko drevo **izrodi**, problem lahko rešimo tako da pustimo pri gradnji nekaj **prostih mest** v originalnih listih

## B + D R E V O

- dinamična struktura ki se sproti posodablja in je uravnoteženo
- liste drevesa uredimo z dvosmernim kazalcem
- operacije dodajanja in brisanja ohranajo drevo uravnoteženo
- vozlišča razen korena so vsaj polovično zasedena
- splača se ko drevo veliko spremijamo
- cena je več prostora

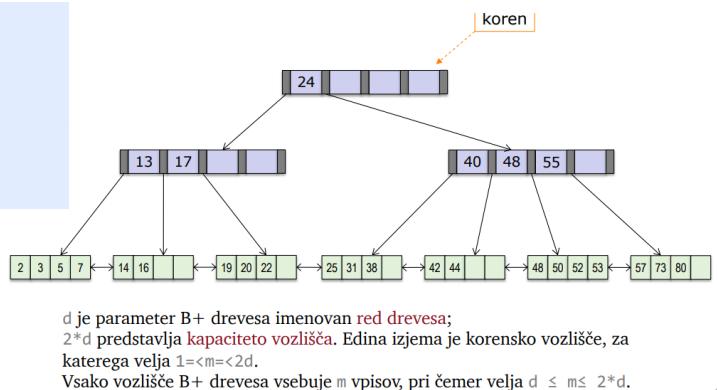
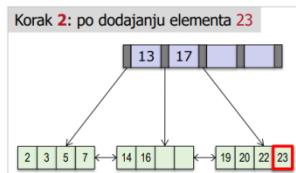
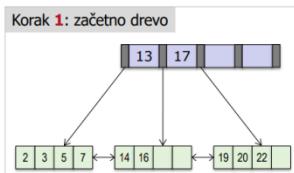


Tabela Staff

staffNo	fName	IName	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

Bitni indeks po polju position

Manager	Assistant	Supervisor
1	0	0
0	1	0
0	0	1
0	1	0
1	0	0

Bitni indeks po polju branchNo

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

STIČNI INDEKS

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
2004	B004	32 Manse Rd	Bristol	BS99 1NZ
2005	...	...		...

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

branch	rowID	city
20001	30002	London
20002	30001	Aberdeen
20003	30003	Glasgow
20004	30004	Glasgow
...	...	...

Tabela Property

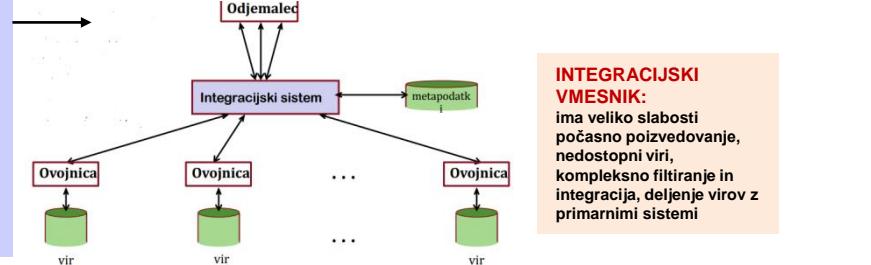
rowID	propertyNo	street	city	postcode	...
30001	PA14	16 Holhead	Aberdeen	AB2 3SU	...
30002	PL94	6 Argyll St	London	SW1 4EH	...
30003	PG4	6 Lawrence St	Glasgow	G11 9QX	...
30004	PG36	2 Manor Rd	Glasgow	G11 9QX	...
30005	...	...		...	...

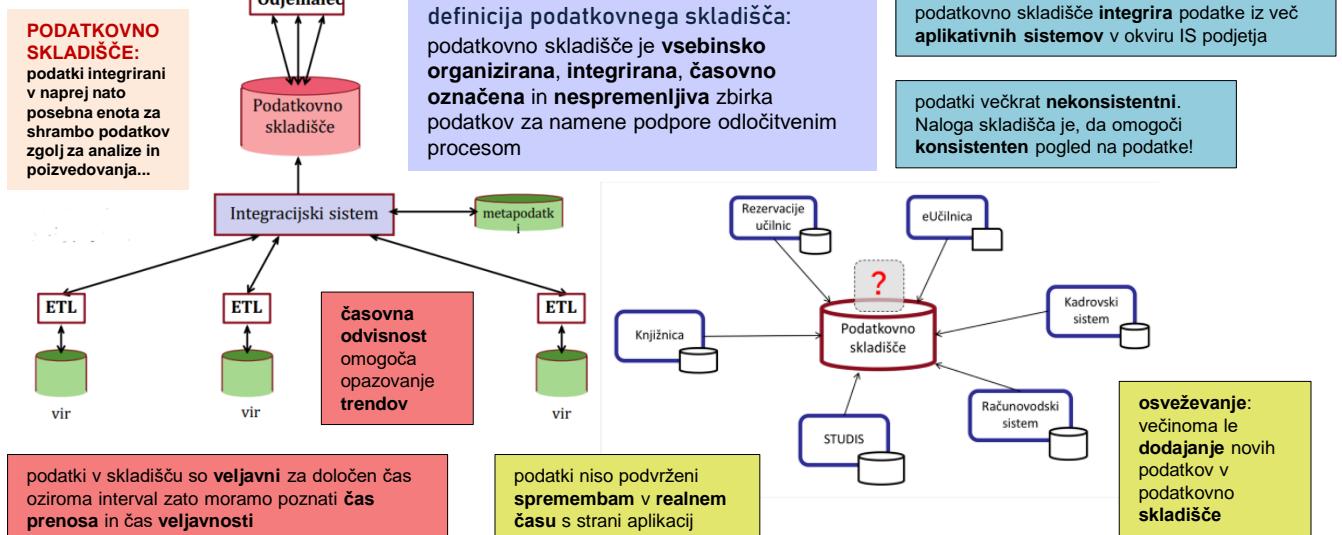
Koliko nepremičnin je v mestih, kjer so locirane organizacijske enote nepremičinske agencije?

## 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.





OLAP – On Line Analytical Processing (vir podatkovno skladišče)

OLTP – On Line Transaction Processing (vir transakcijska baza)

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 transakcij 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 uporabnikov

praviloma različne instance podatkovne baze

primeri vprašanj za OLTP:

- kakšen je bil celoten prihodek 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 primerjavi s preteklimi tremi leti.

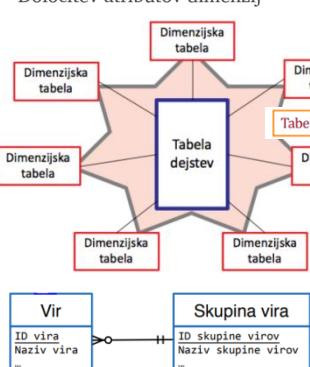
primeri vprašanj za OLAP:

- Iakšen 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 **korelacijski** je to z **demografskimi podatki**?
- ali obstaja signifikantna **povezava** med celotnim prihodom, generiranim v posamezni nepremičninski agenciji, in številom agentov v tej agenciji?

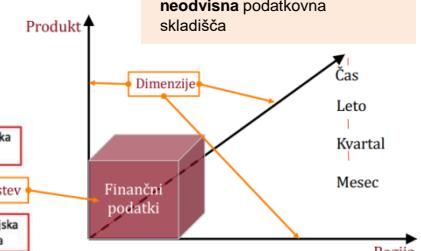
**NAČRTOVANJE PODATKOVNEGA SKLADIŠČA**

- katere so **najpomembnejše zahteve** uporabnikov
- katere podatke uporabiti najprej
- ali se da pokriti celoten poslovni **sistem** ali samo določeno poslovno **področje**
- če začnemo z manjšim obsegom, ali naj bo infrastruktura pripravljena tudi za **končno** podatkovno skladišče
- vsak **model dimenziij** vsebuje tabelo ki vsebuje **primarni kompozitni ključ** oz. **tabela dejstev**
- vsebuje tudi manjše **dimenzijske tabele**
- skupaj tvorijo **zvezdno shemo**
- vsaka dimenzijska tabela ima **enostaven ključ**: komponenta kompozitnega ključa
- **dejstva** se generirajo na podlagi dogodkov iz preteklosti spremembe so malo verjetne
- vse naravne **primarne ključe** se nadomesti z **surogati** oziroma **umetnimi ključi**
- ti ključi omogočajo **neodvisnost** med podatki podatkovnega skladišča in posameznimi viri

- Izbera področja/vsebine podatkovnega skladišča
- Določitev zrnatosti podatkov v tabeli dejstev
- Izbera dimenzij
- Identifikacija dejstev
- Določitev atributov dimenzij



imamo generično dvo nivojsko arhitekturo, odvisna in neodvisna podatkovna skladišča



### TABELA DEJSTEV:

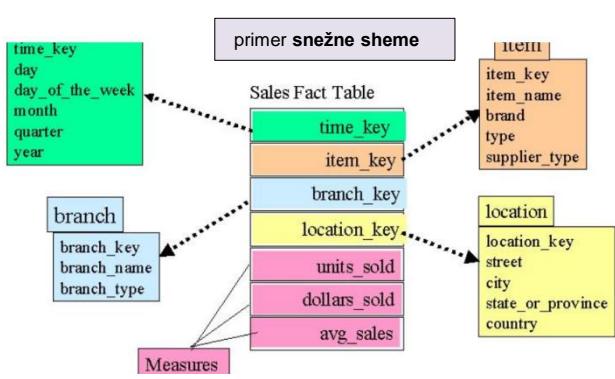
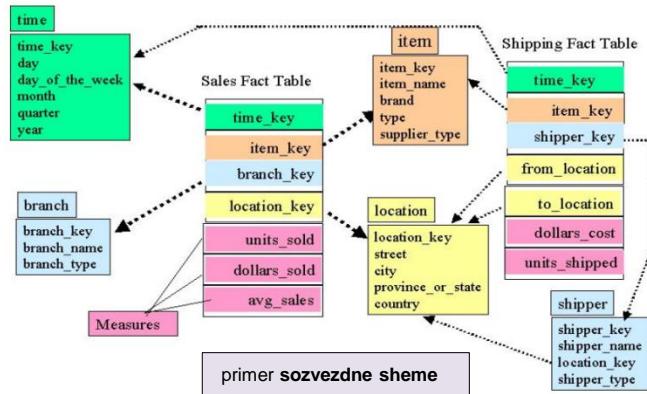
- referenčni podatki namenjeni branju
- skozi čas se ne spremenijo
- podatki se samo dodajajo
- numerične vrednosti
- večina podatkov tukaj

denormalizirane dimenzijske tabele

### DIMENZIJSKA TABELA:

- tekstovne vrednosti in filtri





## NAČRTOVANJE PODATKOVNIH BAZ

### PRISTOPI K NAČRTOVANJU:

- od **zgoraj** navzdol in od **spodaj** navzgor
- **od spodaj** je primeren za majhne PB
- začnemo z atributi in jih združujemo v skupine, temu rečemo **normalizacija**
- za večje PB od **zgoraj**
- tak pristop je ER: entiteta razmerje
- na začetku nekaj **osnovnih entitetnih tipov** in **razmerij** in nato zgradimo v podentitete, povezave in atribute

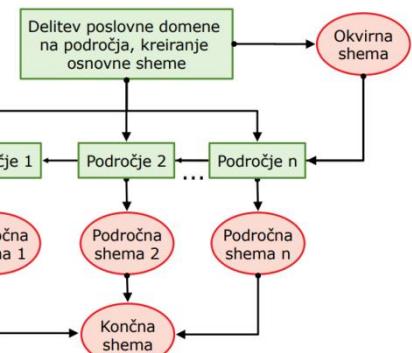
### NORMALIZACIJA

identifikacija potrebnih **atributov** in njihovih agregacij v normalizirane **relacije** na osnovi **funkcionalnih odvisnosti**

za kompleksne PB  
pristop **po delih**

### SNEŽNA SHEMA:

- shema brez **denormaliziranih** podatkov
- **zvezna snežna** shema: hibridna struktura ki ima ki ima denormalizirano in normalizirano shemo in omogoča predstavitev dimenzij na **oba** načina



### 1. KONCEPTUALNI PODATKOVNI MODEL:

- semantika obravnavne domene
- entitete, povezave med entitetami, atributi entitet, poslovna pravila...
- nepovezano od SUPB

### 2. LOGIČNI PODATKOVNI MODEL:

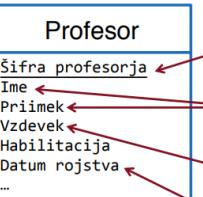
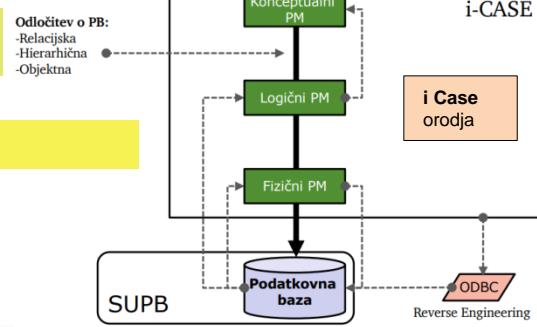
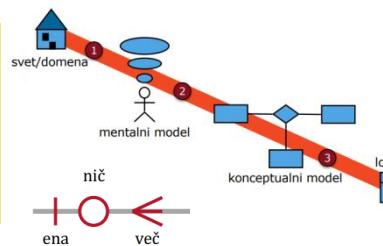
- model v jeziku izbranega SUPB
- v primeru izbere relacijskega SUPB je to **relacijski model**

### 3. FIZIČNI PODATKOVNI MODEL:

- optimizacija fizične podatkovne sheme

## KONCEPTUALNI PODATKOVNI MODEL

- **katero** podatke moramo hraniti?
- z načrtom na **formalen** način opredelimo koncepte domene **baze**
- cilj je razumeti **domeno**
- cilj je **identificirati** koncepte ki jih želimo hraniti v bazi
- po navadi uporabljamo koncept **ER**



TOTALNI atribut - vedno mora imeti vrednost!

VEČ-VREDNOSTNA atribut - oseba ima lahko tudi več imen in priimkov.

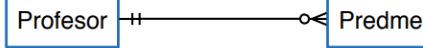
PARCIALNI atribut - ni nujno, da ima vrednost!

ENO-VREDNOSTNI atribut - ima lahko le eno vrednost - oseba se je rodila samo enkrat!



profesor ima nič ali več predmetov, vsak predmet pa točno enega profesorja

- **razmerje** označimo s **črto**
- **števnost**: koliko entitet iz ene strani je lahko povezano z **eno** entitetu na drugi strani
- na vsaki strani označimo **minimalno** in **maksimalno** števnost



### TERMINOLOGIJA:

- **entitetni tip**: zbirke istovrnnih objektov
- **entiteta**: posamezen objekt
- entitete ki pripadajo istemu **tipu** opisujemo prek istih **lastnosti** ki jim pravimo **atributi**

### IDENTIFIKATORJI:

- **enolični identifikator** je tista podmožica atributov ki enolično določijo entiteto
- podlaga za **ključ** v **relacijskem modelu**

## MOČNI ENTITETNI TIP:

- kadar je **vsako entiteto** entitetnega tipa moč enolično identificirati z **njenimi atributti**

$\{a_1, \dots, a_k\}$  je enolični identifikator entitete A, če ustreza naslednjim pogojem:

- $a_1, \dots, a_k$  so vsi **totalni enovrednostni atributi**, kar zagotavlja, da imajo vsi identifikacijski atributi definirano natanko eno vrednost (eno dimenzijo)
- T:  $V_1 \times \dots \times V_k \rightarrow E_A$  je **totalna ali parcialna enovrednostna funkcija**, kar zagotavlja, da se vsak element kartezjskega produkta vrednostnih množic  $V_i$ , ki so območja identifikacijskih atributov, preslikava v največ eno entitetu tipa A
- Je **minimalna podmnožica**: ne obstaja prava podmnožica, za katero bi tudi veljal pogoj b)

## ŠIBKI ENTITETNI TIP:

- enolični identifikator je sestavljen tudi iz razmerji in **drugih entitet v razmerju**

$\{a_1, \dots, a_k\} \cup I_{E1} \cup \dots \cup I_{En}$  je enolični identifikator entitete A, če ustreza naslednjim pogojem:

- $a_1, \dots, a_k$  so vsi totalni enovrednostni atributi,  $I_{Ti}$  pa identifikatorji entitetnih tipov
- T:  $V_{a1} \times \dots \times V_{ak} \times V_{IET1} \times \dots \times V_{IETn} \rightarrow E_A$  je **totalna ali parcialna enovrednostna funkcija**;
- Je **minimalna podmnožica**, ne obstaja prava podmnožica, za katero bi tudi veljal pogoj b).

## RAZŠERJENA ER NOTACIJA:

- generalizacija
- specializacija
- agregacija
- kompozicija

## AGREGACIJA

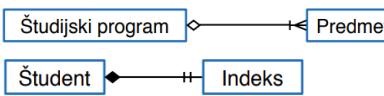
en entitetni tip predstavlja **celoto** drug pa njen **del**

## KOMPOZICIJA

ko tisti tip ki predstavlja le del celote **ne more obstajati sam**

**SPECIALIZACIJA**  
entitetni tip je **posebna vrsta** drugega entitetnega tipa

**GENERALIZACIJA**  
entitetni tip je **pospolitev** več podobnih entitetnih tipov



- totalno / delno** pokrivanje: profesorji in asistenti pokrivajo vse pedagoge / samo del pedagogov
- ekskluzivno / prekrivno** pokrivanje: če noben profesor ni asistent in noben asistent ni profesor / profesor je lahko asistent ali asistent profesor

**1. KORAK:** identificiramo entitetne tipove pregled **uporabniških zahtev**, možnih je več pravilnih rešitev, subjektivno. Ko to naredimo jih moramo **dokumentirati**.

## Pedagog

Šifra pedagoga  
Ime  
Priimek  
Datum rojstva  
...

## Profesor

Habilitacija  
...

## Asistent

Doktorat  
...

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
Izpiti rok	Predstavlja datum, na katerega je za nek predmet in določeno ciljno skupino (letnik, smer,...) razpisani izpitni rok.	Rok, pisni izpit, kolokvij	Na leto se razpiše okrog 300 pisnih izpitov. Vsak predmet mora imeti vsaj tri roke letno
...			

## 2. KORAK: identificiramo povezave

pozorni moramo biti na povezave ki povezujejo **več entitetnih tipov** in **rekurzivne zveze**. Potem moramo določiti **števnosti**, kjer smo pozorni na **dvojne in nepopolne** povezave. Tudi te potem dokumentiramo.

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

## 3. KORAK: identificiramo atribute

pažiti moramo pri **sestavljenih** atributih, **več vrednostnih** atributih in **izpeljanih** atributih. Na koncu jih **dokumentiramo**.

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

## 4. KORAK: atributom določimo domene

lahko določimo seznam **dovoljenih vrednosti**, **minimalno** in **maksimalno** vrednost, podatkovni **tip** in **dolžino** in dovoljene **operacije** nad atributom

## 5. KORAK: določimo kandidate za ključe in potem izmed teh kandidatov določimo ključ

## 8. KORAK: podpiranje transakcij

- preveriti moramo če model podpira vse **transakcije**
- če transakcije ne uspemo izvesti je model pomanjkljiv
- preverimo tako da vsako **transakcijo opišemo**.

## 6. KORAK: uporabimo EER elemente

Elementi razširjenega diagrama **povečajo semantiko** modela vendar negativno vplivajo na "berljivost" modela

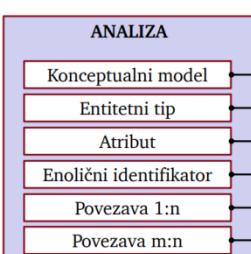
## 9. KORAK: preverjanje z uporabnikom

- anomalije ali pomanjkljivosti

## 7. KORAK: obstoj odvečnih elementov

- odstranimo **1 - 1** povezave: **zdržimo** v en entitetni tip in izberemo primarni ključ
- preverimo če imamo kakšne druge **odvečne povezave** in jih odstranimo

## LOGIČNI PODATKOVNI MODEL



## POSKRBIMO ZA:

- ustreznost imen
- pretvorbo **sestavljenih, več vrednostnih** in **izpeljanih** atributov
- indekse tam kjer jih rabimo
- **pogledi**
- ustreznost ključev
- referencialno integriteto
- sprožilce in shranjene procedure

## Predmet

Šifra predmeta  
Naziv  
Število ur na teden  
Semester  
...

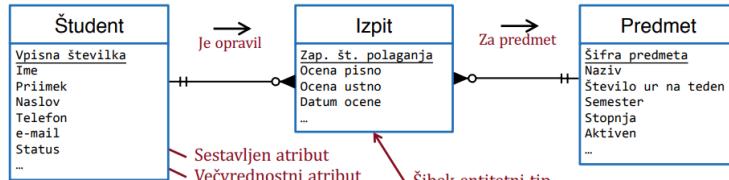
## Študent

Vpisna številka  
Ime  
Priimek  
...

## Ocena

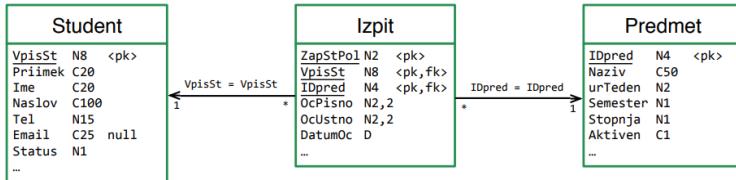
Datum ocene  
Ocena  
...

## Konceptualni in logični model



## NORMALIZACIJA

- preoblikovanje da ne pride do **ažurnih anomalij**
- **1NO**: prva normalna oblika
- **2NO**: druga normalna oblika
- **3NO**: tretja normalna oblika
- **4PNO**: četrta poslovna normalna oblika
- Boyce Coddova NO, 4NO, 5NO



primer  
normalizacije  
prvotne sheme

AŽURNE ANOMALIJE:

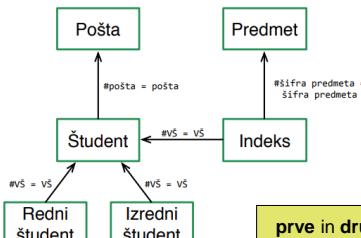
- pri dodajanju
- pri brisanju
- pri spremenjanju

Indeks

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

- Končna shema

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



### 1. KORAK: kreiramo relacije

potek po navadi **avtomatiziran**, pretvorba iz konceptualnega v logični model je podprtia s strani številnih CASE orodij. Če delamo na roke:

Močni entitetni tipi

- Za vsak močni 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 opazovan entitetni tip povezan. Da bi lahko določili ključ, moramo najprej pretvoriti vse povezave.

prve in druge normalne oblike nikoli ne kršimo ostale pa lahko da pridobimo na učinkovitosti

- 1 : 1 združimo v eno relacijo
- 1 : n dobimo tuji ključ
- n : n naredimo vmesno tabelo z tujima ključema

### 6. KORAK: preverimo možnost razširitve

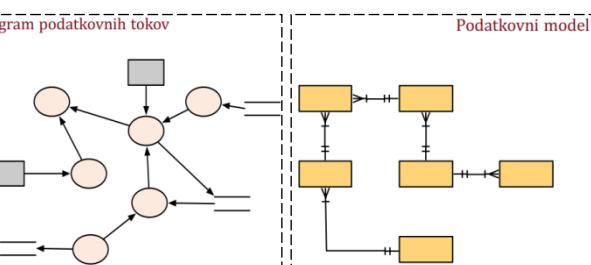
če imamo predvidene razširitve moramo pogledati ali jih model podpira. Model mora biti **prilagodljiv**.

## FIZIČNI PODATKOVNI MODEL

- osnovne relacije,
- datotečno organizacijo,
- indeks za dosego učinkovitega dostopa do podatkov,
- povezane omejitve in
- varnostne mehanizme.

opisuje več stvari

1. KORAK: pretvorba v jezik SUPB iz logičnega modela izdelati podatkovno shemo za ciljni SUPB. Vедeti moramo kaj SUPB podpira.



**2. KORAK:** izdelava osnovnih relacij  
vir podatkov je podatkovni slovar, jezik za opis pa **DDL**. Za vsako relacijo navedemo **naziv** relacije, listo osnovnih **atributov**, primarni ključ in **tuje** ključe ter **omejivte** povezav.

**3. KORAK:** predstavitev atributov  
za vsak atribut določimo ali je shranjen v **podatkovni bazi** ali se izračuna **vsakič znova**.

**4. KORAK:** načrt splošnih omejitvev

**5. KORAK:** datotečna organizacija  
izberemo optimalno **datotečno organizacijo** za shranjevanje **osnovnih relacij** ter potrebne **indeksse** za doseganje ustreznih učinkovitosti.  
Tu so velike **razlike** med SUPB

**6. KORAK:** analiza transakcij  
poskušamo razumeti namen transakcij, ki so **najpomembnejše**. preverimo tudi **atribute** ki se v transakcijah **spreminjajo**.

Table 17.1 Cross-referencing transactions and relations.

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

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

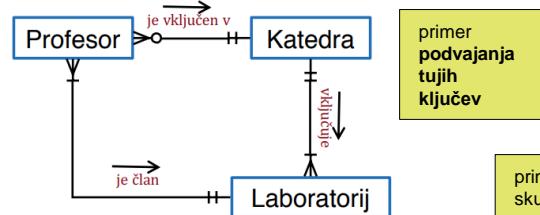
### 7. KORAK: izbiro indeksov

indeks gruče je **indeks po atributu**, ki je obenem tudi atribut, po katerem je **urejena relacija**, ni pa **primarni ključ**. Ključ ni unikaten! imamo **sekundarne indekse** ki niso po primarnih ključih.

**8. KORAK:** načrt varnostnih mehanizmov  
pomembna **sistemsko varnost** ki zagotavlja varnost dostopa in uporabe podatkovne baze in **podatkovno varnost**.

**9. KORAK:** uvedba nadzorovane redundance  
**denormalizacija** na primer. pospeši poizvedbe vendar upočasni spremenjanje podatkov. Laho **razbijemo relacije** horizontalno in vertikalno. Tako imamo boljšo **porazdelitev** vnosa, boljšo **razpoložljivost**, boljšo **obnovljivost** in več možnosti za zagotavljanje varnosti. Slaba je kompleksnost, slabša učinkovitost in podvajanje podatkov.

```
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;
```



primer podvajanja tujih ključev

primer podvajanja skupin namensko

primer podvajanja skupin namens

# PODATKOVNE BAZE XML

## OSNOVNE INFORMACIJE:

- podoben jeziku **HTML**
- namenjen **prenosu** ne prikazu
- **značke** oz. tags niso določeni vnaprej
- podmnožica standarda **SGLM**

## sintaksa:

- začnemo z neobveznim **prologom**
- ko odpremo en **tag** so vsi novi tagi noter njegovi **otroci**
- imamo **elemente** in **atribute**

## značke:

- **začetna**: < ... >
- **končna**: < / ... >
- so **case sensitive**
- začnemo z **korenom**

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

## ravni skladnosti:

- pravilno **strukturiran**
- **veljaven** glede na shemo
- obstajati mora **en** element ki vsebuje **vse** ostale
- značke **uravnovežene**
- pazimo na pravilno **gnezdjenje**
- **atributi** znotraj značk
- posebni znaki z &

```
<?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>
```

primerek definirane sheme

## pretvorbe relacijske sheme

### Relacija A

A1	A2
a11	a21
a12	a22

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

### Relacija B

B1	B2	*A1
b11	b21	a11
b12	b22	a12

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

pretvorba 1:1 povezave

### Relacija A

A1	A2
a11	a21
a12	a22

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

### Relacija B

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

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

pretvorba 1:n povezave

## Deklaracija

```
<?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>
        <b>besedilo</b>Google je končal digitalizacijo starih časnikov, kar pomeni, da so vse besedila v elektronskem obrazcu na voljo na internetu. To je del projekta Google News Archive, ki je bil uradno predstavljen leta 2004. Vse besedila so v slovenščini in so na voljo na spletni strani news.google.com. Projekat je potreboval več kot deset let in je bil finančno podprt z velikimi sredstvi. Vse besedila so v elektronskem obrazcu na voljo na spletni strani news.google.com. Projekat je potreboval več kot deset let in je bil finančno podprt z velikimi sredstvi.
```

## Značke (Tags)

### Začetna značka

```
<?xml version="1.0" ?>
<novice>
    <novica naslov="iPhone 7 kmalu v prodaji">
        <kategorija1>telefonija</kategorija1>
        ...
</novica>
</novice>
```

### Zaključna značka

```
</novica>
```

## primer sheme:

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

Author.xsd

Shema Author.XSD definira strukturo elementa Author

## Relacija A

A1	A2
a11	a21
a12	a22

## Relacija B

B1	B2
b11	b21
b12	b22

## Relacija R

*A1	*B1
a11	b11
a12	b12

pretvorba  
n:m  
povezave

## orodja:

- **urejevalniki**: izdelava, urejanje in validacija
- orodja za **obdelavo**: iz tega v druge oblike
- obdelava iz **programskih jezikov**

## podatkovne baze:

- **upravljanje** s podatki v tem formatu
- shranjevanje,
- **poizvedovanje**, pretvorbe...
- **dokumentne baze**
- standard za prenos podatkov

imamo **naravne** in take ki podpirajo ta format in pa pretvarjajo na **vhodu** iz XML in na izhodu **nazaj**

## poizvedovanje:

## X Q U E R Y

### OSNOVNE LASTNOSTI:

- **izraznost:** poizvedujemo po različnih podstrukturah
- omogoča **rekurzijo:** super za **drevesa** in **grafe**
- **fleksibilnost:** za hierarhične in tabelične strukture
- **konsistentnost:** kompatibilen z ostalimi standardi

■ Vozlišča

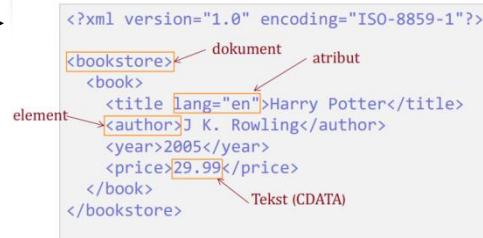
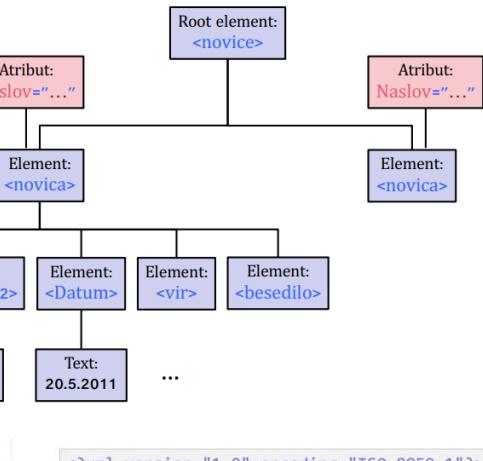
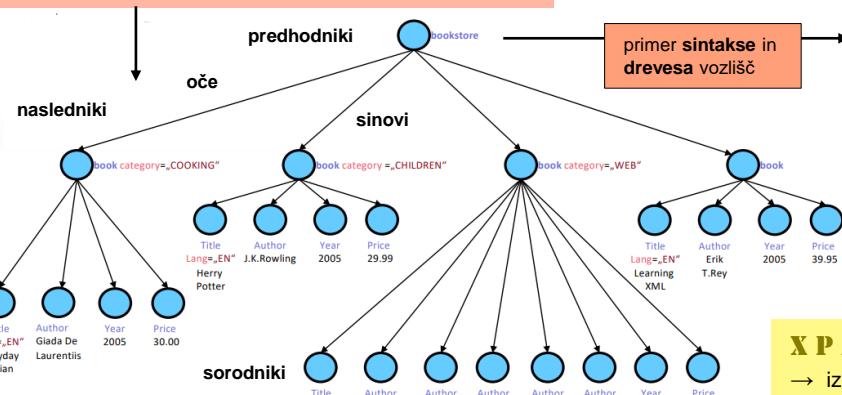
■ Vozlišče tipa element

■ Vozlišče tipa atribut

### SINTAKSA:

- obravnava kot **drevo vozlišč**
- vozlišča so **različnih tipov:**
  - vsak **atribut** vsaj enega **očeta**
  - vsak **element** vsaj enega **očeta**
  - **elementi** poljubno **sinov**
  - relaciji **predhodnik** in **naslednik**
  - vozlišča z istim očetom so **sorodniki**

- element,
- atribut,
- tekst,
- imenski prostor
- procesna instrukcija
- komentar in
- dokument.



### FLOWR :

- **for** : iteracija čez vse **vrednosti** spremenljivke, omejimo z **to**
- **let** : dodelimo **vrednost** spremenljivki
- **where** : dodelimo enega ali več **pogojev**
- **order by** : določamo način **sortiranja** rezultata
- **return**: določanje lahko sestavljenega **izhoda**, ločimo z vejico

z at štejemo iteracije

```
for $x at $i in
doc("books.xml")/bookstore/book/title
return <book>{$i}. {data($x)}</book>
```

Zanimajo nas samo podatki brez naziva elementa

```
let      let $x := (1 to 5)
        return <test>{$x}</test>
```

```
where $x/price>30 and $x/price<100
```

for

where

zapis v HTML :

- označe < ul >
- označe < li >

```
for $x in doc("books.xml")/bookstore/book
order by $x/@category, $x/title
return $x/title
```

order by

```
for $x in doc("books.xml")/bookstore/book
return ($x/title, $x/year)
```

return

```
for $b in //book
error
where $b/author eq "Kurt Cagle"
return $b/title
```

error

primerjave

pogojni izrazi:

- imamo if
- imamo then
- imamo else

```
for $x in doc("books.xml")/bookstore/book
return if ($x/@category = "CHILDREN")
```

kvantifikatorja

```
every $book in /bookstore/book
satisfies $book/@category = "CHILDREN"
```

```
1 <?xml version="1.0" encoding="UTF-8"?>false
```

```
some $p in /bookstore/book/price
Satisfies ($p > 30.00)
```

```
1 <?xml version="1.0" encoding="UTF-8"?>true
```

funkcije in kvantifikatorji:

- imamo **vgrajene** funkcije
- lahko definiramo **svoje** funkcije
- **ekstencionalni** kvantifikator: **some**
- **univerzalni** kvantifikator: **every**

# PODATKOVNE BAZE noSQL



**dodatni parametri:** kontekstna odvisnost, kakovost, prikaz, vrednost

**razširljivost:** še po podatkih in po številu uporabnikov

## BASICS:

- distribucija prek mreže
- zagotovljena varnost pred odpovedjo
- relacijske podatkovne baze imajo zanesljiv način izvajanja transakcij tukaj le delno zagotovljeno
- standardizirani poizvedovalni jeziki
- podatkovna shema
- distribucija redko

## C A P

- razpoložljivost: availability
- konsistentnost: consistency
- odpornost: partition tolerance

- A C I D**
- atomicity
  - consistency
  - isolation
  - durability

### porazdeljene baze:

- imajo ACID lastnosti
- fragmentacija
- replikacija
- postopa konsistentnost

vnaprejšnji zapis v dnevnik za D in centralno zaklepanje za ACI pri relacijskih bazah

pri porazdeljenih bazah centralno zaklepanje ozko grlo

**fragmentacija:** tri dimenzijske širjenje podatkovnih baz ki ga zagotavljamo z fragmentacijo in replikacijo.

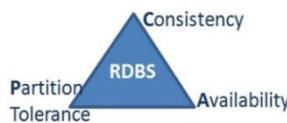
Podatke razbijemo na fragmente. Vsak pisan na svojem vozlišču, ki jih poljubno dodajamo.

Velika kompleksnost join operacij

- Po številu bralnih operacij
- Po številu pisalnih operacij
- Po velikosti podatkovne baze

**replikacija:** zapišemo podatkovne enote na več kot eno vozlišče. Bolj učinkovito beremo, večja zanesljivost če vozlišče odpove, lahko delna ali popolna.

Ima negativen efekt na pisanje, ker mora biti zapisano povsod



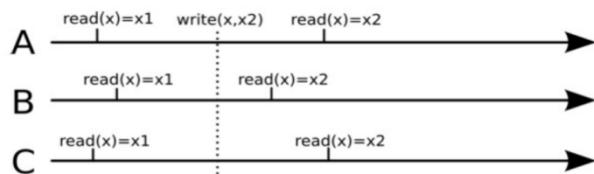
A, B, C: nepovezani procesi, ki berejo ali pišejo v bazo  
x: opazovana podatkovna enota  
x1, x2, x3: različne vrednosti podatkovne enote x

## konsistentnost:

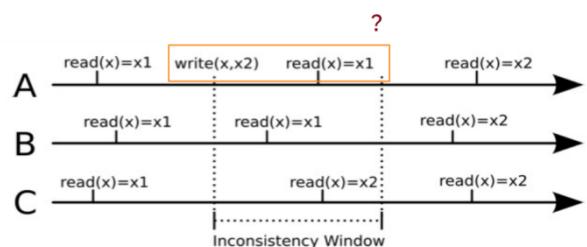
- **stroga**: vse verzije posamezne podatkovne enote so enake
- **postopna**: vse verzije posamezne podatkovne enote bodo sčasoma enake

primer konsistentnosti

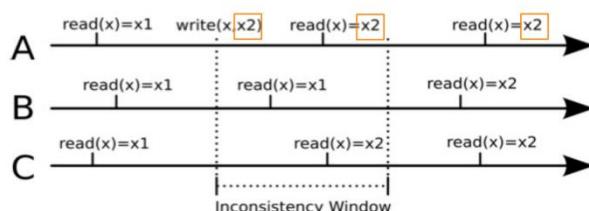
### Stroga konsistencija (Strict Consistency)



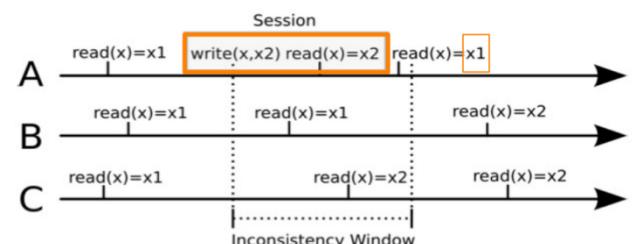
### Postopna konsistencija (Eventual Consistency)



### Postopna konsistencija "beri-svoje-podatke" (Read-Your-Own-Writes Consistency)



### Konsistencija seje (Session Consistency)



# VAJE: 1. KOLOKVIJ

## POIZVEDOVANJE:

```

5. SELECT [*|ALL|DISTINCT] A, B AS C -- projekcija in preimenovanje
1.   FROM t tabela ... -- preimenovanje (in stiki) tabel
2.   WHERE A = 1 -- selekcija po atributih
     /* GROUP BY ... */
     /* HAVING ... */
3.   ORDER BY A, B [ASC|DESC] -- urejanje izpisa
4.   LIMIT [0,] 10; -- omejevanje izpisa

```

$\rightarrow \text{SELECT [ALL|DISTINCT] A, B FROM t;}$  projekcija  $\pi_{A,B}(t)$

$\rightarrow \text{SELECT A AS Foo, B AS Bar FROM t;}$   $\rho(\text{Foo}, \text{Bar})(\pi_{A,B}(t))$  preimenovanje

$\rightarrow \text{SELECT * FROM t WHERE A > 100 AND B != 1;}$  selekcija

$\downarrow$   
 $t_1 \times t_2$   
 $\text{SELECT * FROM } t_1, t_2;$

$\downarrow$   
 $t_1 \bowtie t_2$   
 $\text{SELECT * FROM } t_1 \text{ NATURAL JOIN } t_2;$

kartezični produkt

naravni stik

$t_1 \bowtie_{t_1.A \geq t_2.B} t_2$

$\text{SELECT * FROM } t_1 \text{ LEFT [OUTER] JOIN } t_2 \text{ ON } t_1.A \geq t_2.B;$

odprtji stik

presek

$\text{SELECT * FROM } t_1 \text{ INTERSECT -- ni podprt? }$

$t_1 \cap t_2$

A je primarni ključ  
obeh tabel

$\text{SELECT * FROM } t_1 \text{ WHERE } t_1.A \text{ IN ( -- ugnezdena poizvedba }$   
 $\text{SELECT } t_2.A \text{ FROM } t_2);$

$\text{SELECT * FROM } t_1 \text{ MINUS -- ni podprt? }$

razlika

$t_1 - t_2$

$\text{SELECT * FROM } t_1 \text{ -- ali }$   
 $\text{SELECT * FROM } t_2;$   
 $\text{SELECT * FROM } t_1 \text{ WHERE } t_1.A \text{ NOT IN (SELECT } t_2.A \text{ FROM } t_2);$

## RELACIJSKI RAČUN:

oseba ali o	facebook ali f	twitter ali t						
ID	Ime	Rojen	ID	Ime	Rojen	ID	Ime	Rojen
1 Jill	9.3.1990		1	2		1	2	
2 Jack	2.6.1950		2	1		3	2	
3 Joe	1.8.1989		2	3		4	3	
4 Jenn	7.1.2001		3	2		2	4	

Relacijska algebra:

n-terični relacijski račun:

$\sigma$

$\{\sigma | \text{oseba}(O)\}$

$\pi_{ID, Ime}(O)$

$\{\sigma.ID, \sigma.Ime | \text{oseba}(\sigma)\}$

$\sigma_{Rojen>1.1.2000}(O)$

$\{\sigma | \text{oseba}(\sigma) \wedge \sigma.Rojen > 1.1.2000\}$

$\sigma_{ID=OID} f$

$\{\sigma | \text{oseba}(\sigma) \wedge \exists F(\text{facebook}(F) \wedge \sigma.ID = F.OID)\}$

Domenški relacijski račun:

$\{\sigma | \text{ID, Ime, Rojen} | \text{oseba}(\sigma | \text{ID, Ime, Rojen})\}$

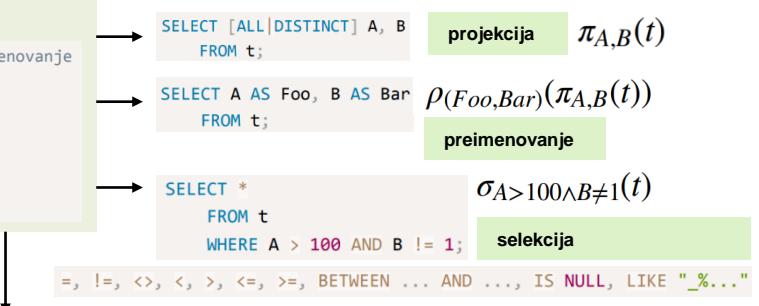
$\{\sigma | \text{ID, Ime} | \exists \text{Rojen}(\text{oseba}(\sigma | \text{ID, Ime}), \text{Rojen})\}$

$\{\sigma | \text{ID, Ime, Rojen} | \text{oseba}(\sigma | \text{ID, Ime, Rojen}) \wedge \text{Rojen} > 1.1.2000\}$

$\{\sigma | \text{ID, Ime, Rojen} | \text{oseba}(\sigma | \text{ID, Ime, Rojen})\}$

$\wedge \exists OID \exists PID (\text{facebook}(OID, PID) \wedge ID = OID)$

**primerjava sintakse**  
relacijske algebre in  
obeh relacijskih  
računov na neki  
podatkovni bazi



$\downarrow$   
 $t_1 \times t_2$

$\downarrow$   
 $t_1 \bowtie t_2$

kartezični produkt

naravni stik

$t_1 \bowtie_{t_1.A \geq t_2.B} t_2$

$\text{SELECT * FROM } t_1 \text{ LEFT [OUTER] JOIN } t_2 \text{ ON } t_1.A \geq t_2.B;$

odprtji stik

unija

$\text{SELECT * FROM } t_1 \text{ INTERSECT -- ni podprt? }$

$t_1 \cap t_2$

presek

$\text{SELECT * FROM } t_1 \text{ MINUS -- ni podprt? }$

razlika

$t_1 - t_2$

A je primarni ključ  
obeh tabel

$\text{SELECT * FROM } t_1 \text{ -- ali }$   
 $\text{SELECT * FROM } t_2;$   
 $\text{SELECT * FROM } t_1 \text{ WHERE } t_1.A \text{ NOT IN (SELECT } t_2.A \text{ FROM } t_2);$

$\text{SELECT * FROM } t_1 \text{ MINUS -- ni podprt? }$

razlika

$t_1 - t_2$

SELECT \* FROM t1

UNION [DISTINCT|ALL]

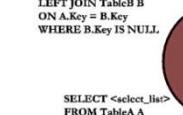
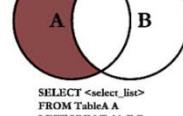
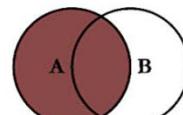
SELECT \* FROM t2;

$t_1 \cup t_2$

SELECT A, COUNT(\*) FROM t GROUP BY A;

grupiranje

$\sigma_{COUNT(B)}(t)$



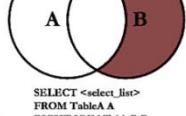
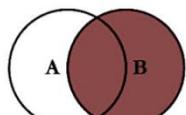
$\sigma_{<\text{select\_list}>} \text{FROM TableA A LEFT JOIN TableB B ON A.Key = B.Key WHERE B.Key IS NULL}$



$\sigma_{<\text{select\_list}>} \text{FROM TableA A RIGHT JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL OR B.Key IS NULL}$

$\sigma_{<\text{select\_list}>} \text{FROM TableA A FULL OUTER JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL OR B.Key IS NULL}$

© C.L. McCreath, 2008



$\sigma_{<\text{select\_list}>} \text{FROM TableA A INNER JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL OR B.Key IS NULL}$

$\sigma_{<\text{select\_list}>} \text{FROM TableA A FULL OUTER JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL OR B.Key IS NULL}$

AGREGACIJA:

- 7.  $\text{SELECT [A, B, ] SUM(C)} -- \text{agregacija atributov}$**
- 1.  $\text{FROM t tabela ...}$**
- 2.  $\text{WHERE A != 1 AND C > 0 -- selekcija po atributih}$**
- 3.  $\text{[GROUP BY A, B]} -- \text{grupiranje po atributih}$**
- 4.  $\text{HAVING SUM(C) > 100 -- selekcija po agregacijah}$**
- 5.  $\text{ORDER BY [B, ] SUM(C);}$**
- 6.  $\text{LIMIT 1;}$**

agregacija

$\tau_{\min A, \max A, \text{avg } A}(t)$

$\text{SELECT MIN(A), MAX(A), AVG(A) -- SUM(A) / COUNT(A)}$   
 $\text{FROM t;}$

$\text{SELECT A FROM t GROUP BY A HAVING COUNT(*) > 100;}$

imamo grupiranje  
in selekcijo

operatorji

$\dots \text{WHERE [NOT] EXISTS (SELECT ...); -- neprazna tabela}$   
 $\dots \text{WHERE A [NOT] IN (SELECT A FROM ...); -- vsebovanost v tabeli}$   
 $\dots \text{WHERE A >= ALL (SELECT A FROM ...); -- največja vrednost A}$   
 $\dots \text{WHERE A > ANY (SELECT A FROM ...); -- ne najmanjša vrednost A}$

# VAJE: 3. KOLOKVIJ

## X Q U E R Y

### POIZVEDBA:

- dobimo **vhodni** dokument
- napišemo **poizvedbo**
- dobimo nek **rezultat**

### Računanje ?

- operacije **levo asociativne?**

`true() and true() or false() and false()` → true  
`true() and (true() or false()) and false()` → false

### FUNKCIJE:

- **argument** poljuben izraz
- lahko zahtevamo določen **tip** argumenta

```
declare function local:vrniŠtezdelka($izdelek as element()) as element()
{ $izdelek/stevilka };
```

### LOGIČNI IZRAZI:

- **not** pred **and** pred **or**

`if (not($jePopust)) then 0 else $popust`

`if ($jePopust and ($popust > 10 or $popust < 0))
then 10 else $popust`

### Začnemo z prologom

- opredelitev **nastavitev**
- imenski prostor
- **funkcije**
- uvozimo zunanje **scheme**
- privzete **operacije**
- vsaka opredelitev ima **podpičje**

### self

- trenutni element (okrajšava je .)
- ancestor, ancestor-or-self**
- prednik, nad element nad elementa itd.
- following, preceding**
- nasledniki in predhodniki v dokumentu
- following-sibling, preceding-sibling**
- naslednji brat/sestra

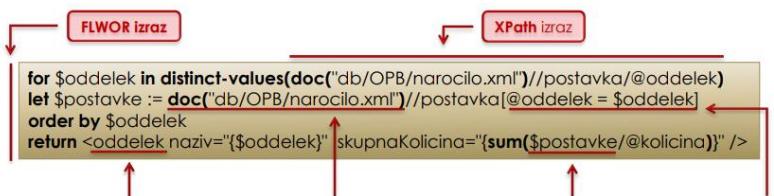
`doc("db/OPB/katalog.xml")/katalog/izdelek[4]`

izpišemo 4 izdelek kataloga

`doc("db/OPB/katalog.xml")/katalog/*[4]`

izpišemo 4 element kataloga

```
doc("db/OPB/narocilo.xml")//postavka[@barva = 'rdeča']
1 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava">
2 <postavka oddelek="ACC" stevilka="558" kolicina="11" barva="rdeča">
3 <postavka oddelek="ACC" stevilka="559" kolicina="21" barva="rdeča">
4 <postavka oddelek="MEN" stevilka="784" kolicina="11" barva="modra">
5 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča">
6 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična">
```



`for $izdelek in doc("db/OPB/katalog.xml")//izdelek
return $izdelek/stevilka`

primer FLWOR ukaza

`substring($imeŠtezdelka, 1, 5)`

ime funkcije  
seznam argumentov

vrne true če ima **poljuben** atribut količina vrednost **večjo** od 1

`doc("db/OPB/narocilo.xml")//postavka/@kolicina > 1`

zgodi se **napaka** če **obstaja** več kot en atribut količina

`doc("db/OPB/narocilo.xml")//postavka/@kolicina gt 1`

### PRIMERJAVE:

- primerjava po **vrednosti**: primerjamo **posamezne** vrednosti
- **splošna** primerjava: lahko uporabimo na **zaporedju** več argumentov

```
if ($izdelek/@oddelek = 'ACC')
then <dodatek>{data($izdelek/stevilka)}</dodatek>
else if ($izdelek/@oddelek = 'WMN')
then <zenski>{data($izdelek/stevilka)}</zenski>
else if ($izdelek/@oddelek = 'MEN')
then <moski>{data($izdelek/stevilka)}</moski>
else <ostalo>{data($izdelek/stevilka)}</ostalo>
```

### Pogojni stavki:

- **oklepaji** so nujni
- **else** del je **obvezen** vendar je lahko **prazen**
- if mora imeti **logično vrednost**
- če nima se uporabi **efektivna logična vrednost**

efektivna logična vrednost vrne **false**:

- xs:boolean vrednost false
- vrednost 0 ali NaN
- niz, dolžine 0
- prazen niz

drugače vrne **true** (npr. seznam elementov)

### Opredelitev poti !

- sprehajanje po **vhodnem** dokumentu
- izbiramo želene **elemente** in **attribute**
- **smer**: **child** je privzeta
- smer **parent** padajoča
- **testni pogoj**: ime elementa oz **tip**
- **predikat**: filtriramo izpis

`doc("db/OPB/narocilo.xml")//postavka`

`doc("db/OPB/narocilo.xml")/descendant-or-self::postavka`

`doc("db/OPB/narocilo.xml")/narocilo/postavka/..`

`doc("db/OPB/narocilo.xml")/narocilo/postavka/parent::narocilo`

izpišemo **nadelement** izbranega elementa

izpišemo vse otroke

vsi isti rezultat

smer testni pogoj predikat

`doc("db/OPB/katalog.xml")/katalog/izdelek[@oddelek='ACC']`

klic funkcije

testiranje elementov

predikat

`doc("db/OPB/narocilo.xml")/narocilo/postavka`

`doc("db/OPB/narocilo.xml")/child::narocilo/child::postavka`

`doc("db/OPB/narocilo.xml")/*/postavka`

`doc("db/OPB/narocilo.xml")/postavka/@oddelek`

`doc("db/OPB/katalog.xml")/katalog/element()`

`doc("db/OPB/katalog.xml")//stevilka/text()`

`doc("db/OPB/katalog.xml")//ime/node()`

### ime elementa

`doc("db/OPB/narocilo.xml")/narocilo/postavka/@oddelek`

### tip elementa

`doc("db/OPB/katalog.xml")/katalog/element()`

`doc("db/OPB/katalog.xml")//stevilka/text()`

`doc("db/OPB/katalog.xml")//ime/node()`

### poljubno ujemanje (\*)

`doc("db/OPB/narocilo.xml")/narocilo/*/@*`

vsi atributi vseh pod elementov naročila

1 557  
2 563  
3 443  
4 784  
lanena srajca  
kavbojski klobuk  
senčnik  
kratka majica

doc("db/OPB/katalog.xml")/katalog/izdelek[position() < 4]

doc("db/OPB/katalog.xml")/katalog/izdelek[stevilka < 500][@oddelek = 'ACC']

izpiše prve 3 izdelke kataloga

doc("db/OPB/katalog.xml")/katalog/izdelek[\* except (stevilka, ime)]

pogoji se preverjajo iz leve proti desni

doc("db/OPB/katalog.xml")/katalog/izdelek[\* except (stevilka, ime)]

vrne tistega ki vsebuje A

vse ki poleg številke in imena vsaj še en atribut

Kako dodamo atribut v rezultat?

- vključevanje iz vhodnega dokumenta
- z uporabo neposrednih konstruktorjev
- z uporabo dinamičnih konstruktorjev

1. Element je vključen v takšni obliku kakšen je v vhodni datoteki, skupaj z atributi!

Nimamo možnosti spremenjanja atributov.

2. Uporablja se XML zapis in njegova pravila. Laho vsebuje svašta svega idk and idc.

Vsi znaki zunaj {} gredo neposredno v rezultat

3. Omogoča dinamično opredelitev imen.

Uporaben za kopiranje elementov, pretvorbo imen in iskanje imen v ločenem slovarju

for \$i in (1 to 3)  
return <rezultat>{1}</rezultat>

primer let operacije

for \$izdelek in doc("db/OPB/katalog.xml")/katalog/izdelek  
return <li>{\$izdelek/@oddelek}

{concat(":", ".")}<br/>{\$izdelek/stevilka}</li>

<rezultat>1</rezultat>  
<rezultat>2</rezultat>  
<rezultat>3</rezultat>

let \$i := (1 to 3)  
return <rezultat>{1}</rezultat>

return vec elementov

Vozlišča atributov postanejo atributi  
Atomarne vrednosti postanejo znakovni podatki  
Vozlišča elementov postanejo pod elementi

primer uporabe kvantifikatorjev

some \$oddelek in doc("db/OPB/katalog.xml")/izdelek  
satisfies (\$oddelek = "ACC") → true

every \$oddelek in doc("db/OPB/katalog.xml")/izdelek  
satisfies (\$oddelek = "ACC") → false

for \$postavka in doc("db/OPB/narocilo.xml")/postavka  
order by \$postavka/@oddelek, \$postavka/@stevilka descending  
return \$postavka

primer sortiranja

reverse((6, 3, 2))

→ [2, 3, 6]

<katalog>  
<izdelek oddelek="WMN">  
  <stevilka>557</stevilka>  
  <ime jezik="sl">lanena srajca</ime>  
  <barve>rjava modra</barve>  
</izdelek>  
...  
</katalog>

Dokument 1  
Katalog izdelkov

zdrževanje rezultatov

Dokument 2  
Naročilo

<narocilo st="00299432" datum="2004-09-15" stranka="0221A">  
  <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava" />  
...  
</narocilo>

↓ integracija

<izdelek stevilka="557" ime="lanena srajca" kolicina="1" />  
...

Dokument 3  
Združeni podatki naročila  
in kataloga izdelkov

\_id: "joe",  
name: "Joe Bookreader",  
addresses: [  
  {  
    street: "123 Fake Street",  
    city: "Faketown",  
    state: "MA",  
    zip: "12345"  
  },  
  {  
    street: "1 Some Other Street",  
    city: "Boston",  
    state: "MA",  
    zip: "12345"  
  },  
  {  
    street: "2 Another Street",  
    city: "Canton",  
    state: "MA",  
    zip: "12345"  
  },  
  {  
    street: "3 Yet Another Street",  
    city: "Denton",  
    state: "MA",  
    zip: "12345"  
  },  
  {  
    street: "4 Final Street",  
    city: "Eenton",  
    state: "MA",  
    zip: "12345"  
  }  
]

razmerje 1 : n

uporaba ugnezenih elementov, imamo še eno opcijo z sklici na dokumente

id: "joe",  
name: "Joe Bookreader",  
address: {  
  street: "123 Fake Street",  
  city: "Faketown",  
  state: "MA",  
  zip: "12345"

(o1)-[:SLEDILEC]->(o2)

usmerjena povezava

CYPHER

CREATE ({ime: "Janez", starost: 7})

ustavljanje vozlišča brez oznake vs. oznaka oseba in vrnemo

CREATE (o:Oseba {ime: "Janez", starost: 7}) RETURN o

MATCH (a:Oseba), (b:Oseba) CREATE (a)-[r:PRIJATELJ]->(b) RETURN r

MATCH (POGOJ) RETURN SPREMENLJIVKA

posodabljanje grafa

iskanje po grafu

MATCH (o {ime: 'Janez'}) SET o.porocen = "DA" RETURN o

MATCH (n) DELETE n

izbrišemo vsa vozlišča vendor moramo prej povezave

MATCH ()-[r]-() DELETE r - Brisanje vseh povezav.

MATCH (n) DETACH DELETE n - Izbris celotne podatkovne baze.