Podatkovne baze 1
in
Osnove podatkovnih baz

Matjaž Kukar

2007-2008

Vsebina

1. Relacijski podatkovni model
   a. Relacijska algebra, osnovni SQL
   b. Množice v SQL
   c. Množice in gnezdene poizvedbe v SQL
   d. Skupinski operatorji v SQL
   e. Pogledi in indeksiranje v SQL
2. Indeksiranje na splošno
3. Nadzor nad sočasno uporabo PB
4. Obnavljanje podatkovne baze

Konfiguracija Oracle klienta

- Velja za Oracle klienta 10 (tako XP kot Vista)
- V direktorij C:\oracle\product\10.2.0\client_1\network\ADMIN skopirajte datoteki tnsnames.ora in sqlnet.ora

Režim izvajanja vaj

- Avditorni del
  - Osnove podatkovnih baz (prog. oprema)
    - Ponedeljek 14h-16h LRI-JD
    - Sreda 10h-12h LRI2
  - Podatkovne baze 1 (informatika)
    - Torek 14h-16h PR-JA (Pascalova predavalnica)
- Tutorski del
  - Ponedeljek 9h-14h v kabinetu ali laboratorijih (po dogovoru)
- Spletna stran za vaje:
  - **`http://pb.fri.uni-lj.si`**

Program dela na vajah

- Relacijska algebra
- SQL
- Teme s področja PB: sočasnost dostopa do PB, varovanje PB, indeksiranje

- Seminarska naloga iz rel. algebre in SQL!

Relacijski podatkovni model (RPM)

- Relacije in operacije nad njimi predstavljajo logično osnovo številnih povpraševalnih jezikov (npr. SQL), ki nudi številne možnosti optimizacije povpraševanj!
- Dve vrsti operacij:
  - Relacijska algebra: operativna; opišemo načrt izvajanja operacij
  - Relacijski račun: neoperativen, deklarativen; opišemo želen rezultat

Osnovni koncepti RPM

- Relacija in relacijska shema
- Atribut
- Vrednostna množica (območje) atributa
- Odvisnosti med atributi

Relacija

- Preslikava kartezičnega produkta vrednostnih množic

$$r : D_1 \times D_2 \times ... \times D_n \rightarrow \{ res, ni\ res \}$$

- Množica resničnih trditev:

  je_oseba={Janez, Peter}

Predstavitev relacije

- Predikatni zapis:
  - Shema: je_oseba(oseba)
  - je_oseba(Janez)
  - je_oseba(Peter)
- Predikatni zapis; opis objektov z atributi:
  - Shema: je_oseba(ime, priimek, kraj bivanja)
  - je_oseba(Janez,Novak,Ljubljana)
  - je_oseba(Peter,Klepec,Celje)

⇒ Naštejemo n-terice, za katere velja relacija

⇒ Kakšen je pomen gornjih relacij?

Predstavitev relacije s tabelo

Oseba	Ime	Priimek	Naslov
Janez (v celoti)	Janez	Novak	Ljubljana
Peter (v celoti)	Peter	Klepec	Celje

Objekti (elementi množice)

Atributni opis objektov (elementov);
ena vrstica = en objekt

Pomen relacije

- Relacija v dobesednem pomenu:
  - Elementi relacije (objekti, vrstice) izpolnjujejo določene pogoje
- Relacija v prenesenem pomenu:
  - Elementi v vrstici relacije (tabele) so med seboj v nekem razmerju
  - Uporaba za povezovanje drugih relacij (tabel) med seboj

Atribut

- Vsaka n-terica v relaciji predstavlja določen objekt
- Vsak objekt opišemo z lastnostmi – atributi
- Atribut kot preslikava objektov v pripadajočo domeno:

$$A_i : O \rightarrow D_i$$

Relacijska shema

- Vsaki relaciji pripada natanko ena relacijska shema, sestavljena iz oznake sheme R in oznakami imen in domen atributov

  $R(A_1 : D_1, A_2 : D_2, ..., A_n : D_n)$

  $\forall r \exists! R : Sh(r) = R$

- Eni shemi lahko pripada več relacij
- Shema relacije = glava tabele

Odvisnosti med atributi

- Omejevanje vrednosti relacij
  - Funkcionalne
  - Večvrednostne
  - Stične
- Veljajo v shemi R; torej v vseh relacijah, katerih shema je R

Funkcionalne odvisnosti

- Množica atributov {X} funkcionalno določa množico atributov {Y} če v nobeni relaciji s shemo R ne obstajata n-terici, ki bi se ujemali v vrednosti atributov {X} in ne ujemali v vrednosti atributov {Y}
- Zapišemo {X}→{Y} ali krajše X→Y
- Množico vseh funkcionalnih odvisnosti v shemi R označimo s F(R)

$$X \to Y \in F(R) \Leftrightarrow \forall r \left(Sh(r) = R \Rightarrow \forall t \forall u (t \in r \wedge u \in r \wedge t.X = u.X \Rightarrow t.Y = u.Y) \right)$$

Ključ relacijske sheme

- Relacija je množica, toraj morajo biti vsi elementi (n-terice) unikatni
- Minimalna podmnožica atributov, ki enolično identificira vsako n-terico je ključ
- Ključ:
  1. $X \rightarrow R$
  2. $\neg \exists A : A \subseteq X \wedge (X - A) \rightarrow R$
- Nadključ: vsebuje vsaj en ključ

- V relacijski shemi ključ podčrtamo

Operacije nad relacijami – relacijska algebra

- Tradicionalni operatorji za delo z množicami: unija $\cup$, presek $\cap$, razlika -, kartezični produkt $\times$
- Posebni relacijski operatorji: selekcija σ, projekcija π, stik $| \times |$, deljenje /

Množiški operatorji

Relacija r:

A	B	C
a	b	c
d	a	f
c	b	d

Relacija s:

D	E	F
b	g	a
d	a	f

Pomembna kompatibilnost atributov!

Unija, presek, razlika

Relacija r ∪ s:

G	H	I
a	b	c
d	a	f
c	b	d
b	g	a
d	a	f

Relacija r ∩ s:

G	H	I
d	a	f

Relacija r - s:

G	H	I
a	b	c
c	b	d

Kartezični produkt

Velja asociativnost: $(r \times s) \times t = r \times (s \times t)$.
Relacija $r \times s$:

A	B	C	D	E	F
a	b	c	b	g	a
d	a	f	b	g	a
c	b	d	b	g	a
b	g	a	b	g	a
d	a	f	b	g	a
a	b	c	d	a	f
d	a	f	d	a	f
c	b	d	d	a	f
b	g	a	d	a	f
d	a	f	d	a	f

Relacijski operatorji

- Projekcija π: zmanjševanje števila stolpcev
- Selekcija σ: zmanjševanje števila vrstic
- Stik $| \times |$: zmanjševanje števila stolpcev in vrstic kartezičnega produkta; zelo pogosta operacije, lahko realiziramo z drugimi operatorji
- Deljenje $/$: lahko realiziramo z drugimi operatorji

Projekcija π

$\pi_{A,B}(r)$

A	B
a	b
d	a
c	b

$\pi_{B}(r)$

B
b
a
~~b~~

Sintaksa: $\pi_{A1,A2,...Ak}$ - naštejemo atribute

Včasih se lahko zmanjša tudi število vrstic!

Selekcija σ

$\sigma_{B<b}(r)$

A	B	C
d	a	f

$\sigma_{B=b \wedge C=d}(r)$

A	B	C
c	b	d

Sintaksa: $\sigma_P(r)$

Logični pogoj P je lahko poljubno kompleksen!

Pogojni (theta) stik

$$r \left|\underset{\theta}{\times}\right| s = r \left|\underset{P}{\times}\right| s \equiv \sigma_P (r \times s)$$

- Alternativna sintaksa: $|\times|$ je isto kot $\bowtie$

Pogojni stik (1. korak)

$$r \underset{(B=D)\vee(C=c)}{\bowtie} s =$$

A	B	C	D	E	F
a	b	c	b	g	a
d	a	f	b	g	a
c	b	d	b	g	a
b	g	a	b	g	a
d	a	f	b	g	a
a	b	c	d	a	f
d	a	f	d	a	f
c	b	d	d	a	f
b	g	a	d	a	f
d	a	f	d	a	f

Pogojni stik (2. korak)

$$r \underset{(B=D)\vee(C=c)}{\bowtie} s =$$

A	B	C	D	E	F
a	b	c	b	g	a
~~d~~	~~a~~	~~f~~	~~b~~	~~g~~	~~a~~
c	b	d	b	g	a
~~b~~	~~g~~	~~a~~	~~b~~	~~g~~	~~a~~
~~d~~	~~a~~	~~f~~	~~b~~	~~g~~	~~a~~
a	b	c	d	a	f
~~d~~	~~a~~	~~f~~	~~d~~	~~a~~	~~f~~
~~c~~	~~b~~	~~d~~	~~d~~	~~a~~	~~f~~
~~b~~	~~g~~	~~a~~	~~d~~	~~a~~	~~f~~
~~d~~	~~a~~	~~f~~	~~d~~	~~a~~	~~f~~

Ekvistik in naravni stik

- Ekvistik: v pogoju lahko od operatorjev nastopajo samo enačaji
- Naravni stik: ekvistik po vseh istoimenskih atributih
  - Oznaka brez pogoja P: $| \times |$
  - Ker je nekaj atributov po naravnem stiku odveč, jih izločimo

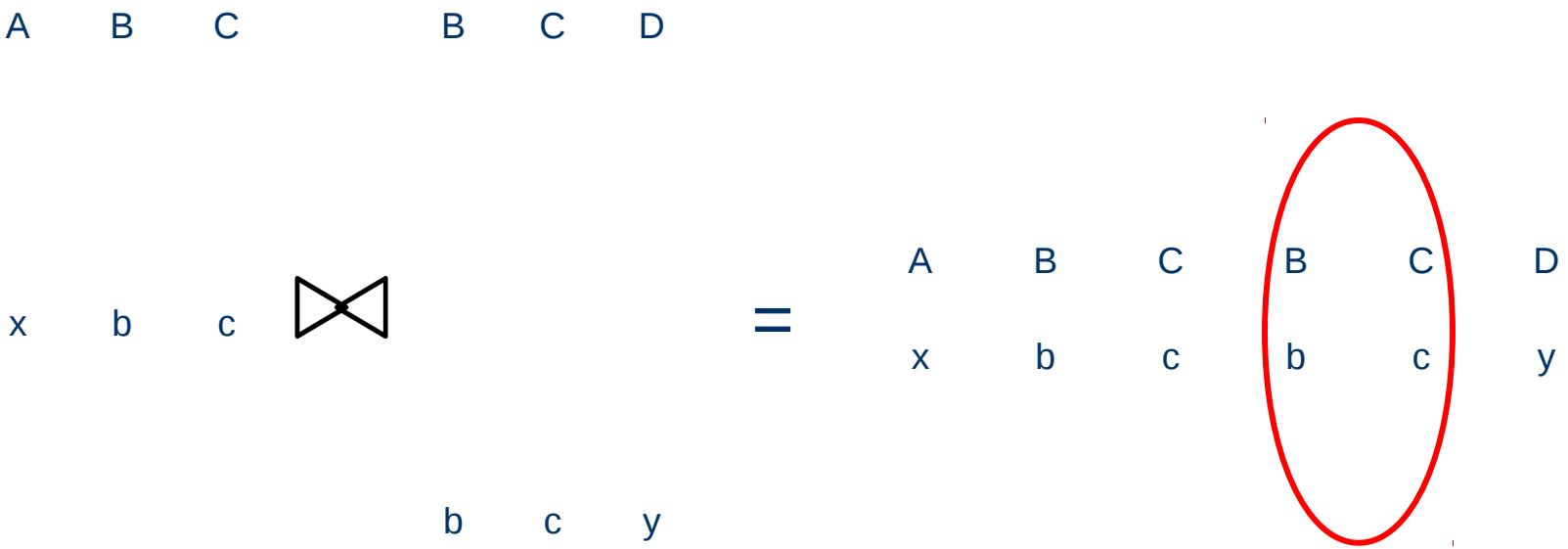
Naravni stik (1. korak)

A B C B C D

x b c  b c y = A B C B C D

x b c b c y

Naravni stik (2. korak)



Naravni stik (3. korak)

A	B	C
x	b	c

⋈

B	C	D
b	c	y

=

A	B	C	D
x	b	c	y

Relacijska algebra - ponovitev

- Tradicionalni operatorji za delo z množicami: unija $\cup$, presek $\cap$, razlika -, kartezični produkt $\times$
- Posebni relacijski operatorji: selekcija σ, projekcija π, stik $| \times |$, deljenje /

Deljenje /

- Imejmo relaciji A z atributi XY in B z atributi Y (X in Y sta množici atributov)
- A / B definiramo kot množico tistih vrednosti X, za katere velja, da za vsako vrednost Y v B obstaja XY v A

$$A / B \equiv \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$$

- Deljenje ni prav pogosta operacija in nima ekvivalenta v SQL

Primeri relacijske algebre

- Sheme za primere rel. algebre:

  `Jadralec(jid, ime, rating, starost)`

  `Coln(cid, ime, dolzina, barva)`

  `Rezervacija(jid, cid, dan)`

- Pomen in povezava relacij:

Primeri tabel

Jadralec:

jid	ime	rating	starost
22	Darko	7	45
29	Borut	1	33
31	Lojze	8	55.5
32	Andrej	8	25.5
58	Rajko	10	35
64	Henrik	7	35
71	Zdravko	10	16
74	Henrik	9	35
85	Anze	3	25.5
95	Bine	3	63.5

Rezervacija:

jid	cid	dan
22	101	2006-10-10
22	102	2006-10-10
22	103	2006-10-08
22	104	2006-10-07
31	102	2006-11-10
31	103	2006-11-06
31	104	2006-11-12
64	101	2006-09-05
64	102	2006-09-08
74	103	2006-09-08

Coln:

cid	ime	dolzina	barva
101	Elan	34	modra
102	Elan	34	rdeca
103	Sun Odyssey	37	zelena
104	Bavaria	50	rdeca

```
Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)
```

Projekcija

- Poišči (izpiši) šifre in imena vseh jadralcev:

$$\pi_{jid,ime}(\text{jadralec})$$

- Poišči barve vseh čolnov

$$\pi_{barva}(\text{coln})$$

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Selekcija

- Poišči (izpiši) šifre in imena vseh jadralcev, starejsih od 50 let:

$$\pi_{jid,ime}(\sigma_{starost>50}(\text{jadralec}))$$

- Poišči barve vseh čolnov krajših od 40 čevljev

$$\pi_{barva}(\sigma_{dolzina<40}(\text{coln}))$$

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Stik

- Poišči vse pare ime jadralcev in čolnov, kjer je jadralec rezerviral ustrezen čoln

$$\pi_{ime,ime}(\text{jadralec} \mid\times\mid \text{rezervacija} \mid\times\mid \text{coln})$$

$$\pi_{\substack{jadralec.ime,\\ coln.ime}}(\text{jadralec} \mid\times\mid_{\substack{jadralec.jid=\\ rezervacija.jid}} \text{rezervacija} \mid\times\mid_{\substack{rezervacija.cid=\\ coln.cid}} \text{coln})$$

$$\pi_{\substack{jadralec.ime,\\ coln.ime}}(\text{jadralec} \mid\times\mid_{jid} \text{rezervacija} \mid\times\mid_{cid} \text{coln})$$

- Poišči vse pare ime jadralcev in čolnov, kjer je jadralec starejši od 50 let rezerviral ustrezen čoln

$$\pi_{\substack{jadralec.ime,\\ coln.ime}}(\sigma_{starost>50}(\text{jadralec}) \mid\times\mid_{jid} \text{rezervacija} \mid\times\mid_{cid} \text{coln})$$

Structured Query Language - SQL

- Rezultat projektov v IBM (1974-77)
- Vsak proizvajalec ga po svoje razširja
- Standardi:
  - ANSI SQL (1987)
  - ANSI/ISO SQL (1992) = SQL'92
  - SQL'99 (1999): objekti, rekurzija, dogodki
- V praksi: le delna podpora standardom

SQL 92

- Data definition language (DDL)
- Data manipulation language (DML)
- Varnost
- Transakcije
- Client / server podpora
- Embedded/dynamic SQL

DML

- Delo nad obstoječimi tabelami!
- Povpraševanja
- Dodajanje vrstic
- Brisanje vrstic
- Spreminjanje vrstic

Osnovni SELECT stavek

```
SELECT A1, A2, ..., Ak

FROM T1, T2, ..., Tn

WHERE P;
```

$$\pi_{A1,A2,...,Ak}$$

$$T1 \times T2 \times ... \times Tn$$

$$\sigma_P$$

$$\pi_{A1,A2,...,Ak}(\sigma_P(T1 \times T2 \times ... \times Tn))$$

- Rezultat `SELECT` stavka kot začasna tabela!
- `SELECT DISTINCT` ali `ALL: DISTINCT` izloči duplikate iz rezultata; privzeta vrednost `ALL` jih ohrani!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Projekcija

- Poišči (izpiši) šifre in imena vseh jadralcev:

```
SELECT jid, ime
FROM jadralec;
```

$$\pi_{jid,ime}(\text{jadralec})$$

- Poišči barve vseh čolnov

```
SELECT barva
FROM coln;
```

$$\pi_{barva}(\text{coln})$$

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Selekcija

- Poišči (izpiši) šifre in imena vseh jadralcev, starejsih od 50 let:

$$\pi_{jid,ime}(\sigma_{starost>50}(\text{jadralec}))$$

```
SELECT jid, ime
FROM jadralec
WHERE starost > 50;
```

- Poišči barve vseh čolnov krajših od 40 čevljev

$$\pi_{barva}(\sigma_{dolzina<40}(\text{coln}))$$

```
SELECT barva
FROM coln
WHERE dolzina < 40;
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Stik (1)

- Poišči vse pare imen jadralcev in čolnov, kjer je jadralec rezerviral ustrezen čoln

$$\pi_{\substack{jadralec.ime,\\ coln.ime}}(\text{jadralec} \mid\times\mid_{jid} \text{rezervacija} \mid\times\mid_{cid} \text{coln})$$

```
SELECT jadralec.ime, coln.ime
FROM jadralec, rezervacija, coln
WHERE jadralec.jid=rezervacija.jid
      AND rezervacija.cid=coln.cid;
```

ime	ime
Darko	Elan
Darko	Elan
Darko	Sun Odyssey
Darko	Bavaria
Lojze	Elan
Lojze	Sun Odyssey
Lojze	Bavaria
Henrik	Elan
Henrik	Elan
Henrik	Sun Odyssey

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Stik (2)

- Poišči vse pare ime jadralcev in čolnov, kjer je jadralec starejši od 50 let rezerviral ustrezen čoln

$$\pi_{\substack{jadralec.ime,\\ coln.ime}}(\sigma_{starost>50}(\text{jadralec}) \mid\times\mid_{jid} \text{rezervacija} \mid\times\mid_{cid} \text{coln})$$

```
SELECT jadralec.ime, coln.ime
FROM jadralec, rezervacija, coln
WHERE jadralec.jid=rezervacija.jid AND
      rezervacija.cid=coln.cid AND
      starost > 50;
```

```
+-------+-------------+
| ime   | ime         |
+-------+-------------+
| Lojze | Elan        |
| Lojze | Sun Odyssey |
| Lojze | Bavaria     |
+-------+-------------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL

- Poišči vse pare ime jadralcev in čolnov, kjer je jadralec starejši od 50 let rezerviral ustrezen čoln

Nekatere nerodnosti prejšnje rešitve:
• Imena stolpcev nejasna - potrebno preimenovanje
• Dolgovezno pisanje imen tabel -uporaba okrajšanih imen - aliasov

```
SELECT j.ime AS "Jadralec", c.ime AS "Coln"
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid
      AND j.starost > 50;
```

Jadralec	Coln
Lojze	Elan
Lojze	Sun Odyssey
Lojze	Bavaria

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL

- Poišči imena jadralcev, ki so rezervirali čoln s številko 103.

$$\pi_{ime}(\sigma_{cid=103}(\text{Rezervacija}) \ |\times|\ \text{Jadralec})$$

```
SELECT jadralec.ime
FROM jadralec, rezervacija
WHERE jadralec.jid=rezervacija.jid AND
      jadralec.cid = 103;
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Alternativna sintaksa za stik: operator JOIN

```
SELECT jadralec.ime
FROM  jadralec, rezervacija
WHERE jadralec.jid=rezervacija.jid AND
      jadralec.cid = 103;

SELECT jadralec. ime
FROM  jadralec NATURAL JOIN rezervacija   -- Naravni stik
WHERE jadralec.cid = 103;

SELECT jadralec. ime
FROM  jadralec JOIN rezervacija ON (rezervacija.jid = jadralec.jid)
WHERE jadralec.cid = 103;

SELECT jadralec.ime
FROM  jadralec JOIN rezervacija ON (rezervacija.jid = jadralec.jid AND
                                    rezervacija.cid = 103);
```

SQL: osnovni SELECT stavek (ponovitev)

```
SELECT A1, A2, ..., Ak

FROM T1, T2, ..., Tn

WHERE P;
```

$$\pi_{A1,A2,...,Ak}$$

$$T1 \times T2 \times ... \times Tn$$

$$\sigma_P$$

$$\pi_{A1,A2,...,Ak}(\sigma_P(T1 \times T2 \times ... \times Tn))$$

- Rezultat `SELECT` stavka si lahko predstavljamo kot začasno tabelo, ki jo izpišemo, ali z njo počnemo kaj drugega
- `SELECT DISTINCT:` izloči duplikate iz rezultata
- Operator JOIN

Komentarji v SQL

- Dve vrsti komentarjev:
  - večvrstični: /* */
  - enovrstični:
    -- (dva minusa in presledek)
    # (lojtra - mySQL)

```
SELECT *          -- Izberi vse
FROM jadralec    /* iz tabele jadralcev */
WHERE starost <18; # Mladoletni jadralci
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL

- Poišči imena jadralcev, ki so rezervirali rdeč čoln.

$$\pi_{jadralec.ime}(\sigma_{barva=rdeca}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec})$$

```
SELECT DISINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid = c.cid AND
      c.barva='rdeca';
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

- Pisanje znakovnih nizov v narekovajih.
- Zakaj je potreben DISTINCT?

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (eksistenčna kvantifikacija)

- Poišči imena jadralcev, ki so rezervirali VSAJ EN čoln.

$$\pi_{ime}(\text{Rezervacija} \mid \times_{jid} \mid \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r
WHERE j.jid=r.jid;
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

- Univezalna kvantifikacija (rezervirali VSE čolne) je bistveno bolj zapletena za implementacijo!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (kartezični produkt)

- Poišči imena in ratinge vseh parov jadralcev, ki imajo enak rating.
  ρ: operator preimenovanja

$$\pi_{\substack{j1.ime,\\ j2.ime,\\ j1.rating}} (\rho(\text{j1,Jadralec}) \mid \times_{rating} \mid \rho(\text{j2,Jadralec}))$$

```
SELECT j1.ime, j2.ime, j1.rating
FROM jadralec j1, jadralec j2
WHERE j1.rating = j2.rating
ORDER BY j1.rating;
```

ime	ime	rating
Borut	Borut	1
Anze	Anze	3
Bine	Anze	3
Anze	Bine	3
Bine	Bine	3
Darko	Henrik	7
Darko	Darko	7
Henrik	Henrik	7
Henrik	Darko	7
Lojze	Lojze	8
Andrej	Lojze	8
Lojze	Andrej	8
Andrej	Andrej	8
Henrik	Henrik	9
Rajko	Zdravko	10
Zdravko	Zdravko	10
Rajko	Rajko	10
Zdravko	Rajko	10

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (kartezični produkt)

- Poišči imena in ratinge vseh parov jadralcev, ki imajo enak rating.
  ρ: operator preimenovanja

$$\pi_{\substack{j1.ime,\\ j2.ime,\\ j1.rating}} (\rho(\text{j1,Jadralec}) \mid \times_{rating} \mid \rho(\text{j2,Jadralec}))$$

```
SELECT DISTINCT j1.ime, j2.ime,
                                j1.rating
FROM jadralec j1, jadralec j2
WHERE j1.rating = j2.rating
ORDER BY j1.rating;
```

ime	ime	rating
Borut	Borut	1
Anze	Anze	3
Bine	Anze	3
Anze	Bine	3
Bine	Bine	3
Darko	Henrik	7
Darko	Darko	7
Henrik	Henrik	7
Henrik	Darko	7
Lojze	Lojze	8
Andrej	Lojze	8
Lojze	Andrej	8
Andrej	Andrej	8
Henrik	Henrik	9
Rajko	Zdravko	10
Zdravko	Zdravko	10
Rajko	Rajko	10
Zdravko	Rajko	10

Zakaj ni razlike od prej?

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (kartezični produkt)

- Poišči imena in ratinge vseh parov jadralcev, ki imajo enak rating.
  ρ: operator preimenovanja

$$\pi_{\substack{j1.ime,\\ j2.ime,\\ j1.rating}} (\rho(\text{j1,Jadralec}) \mid \times_{rating} \mid \rho(\text{j2,Jadralec}))$$

```
SELECT DISTINCT j1.ime, j2.ime,
                        j1.rating
FROM jadralec j1, jadralec j2
WHERE j1.rating = j2.rating AND
      j1.ime <> j2.ime
ORDER BY j1.rating;
```

ime	ime	rating
Bine	Anze	3
Anze	Bine	3
Darko	Henrik	7
Henrik	Darko	7
Lojze	Andrej	8
Andrej	Lojze	8
Zdravko	Rajko	10
Rajko	Zdravko	10

Kaj še ni v redu?

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (kartezični produkt)

- Poišči imena in ratinge vseh parov jadralcev, ki imajo enak rating.
  ρ: operator preimenovanja

```
+--------+---------+--------+
| ime    | ime     | rating |
+--------+---------+--------+
| Anze   | Bine    |      3 |
| Darko  | Henrik  |      7 |
| Andrej | Lojze   |      8 |
| Rajko  | Zdravko |     10 |
+--------+---------+--------+
```

$$\pi_{\substack{j1.ime,\\ j2.ime,\\ j1.rating}} (\rho(\text{j1,Jadralec}) \mid \times_{rating} \mid \rho(\text{j2,Jadralec}))$$

```
SELECT DISTINCT j1.ime, j2.ime,
                        j1.rating
FROM jadralec j1, jadralec j2
WHERE j1.rating = j2.rating AND
      j1.ime < j2.ime
ORDER BY j1.rating;
```

Operatorji v SQL (WHERE vrstica)

- =
- LIKE: približna primerjava nizov znakov
- != ali <>
- <=, >=, <, >
- BETWEEN x AND y: $x \leq$ vrednost $\leq y$
- AND, OR, NOT
- NULL (vrednost atributa nedefinirana)

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL: operator LIKE

- Poišči starost jadralcev, katerih imena se začnejo na B in imajo najmanj 5 črk.

```
+--------+
| starost|
+--------+
|     33 |
+--------+
```

```
SELECT j.starost
FROM jadralec j
WHERE j.ime LIKE 'B____%'; /* 4 podcrtaji */
```

- _ ustreza natanko eni poljubni črki
- % ustreza nič ali več poljubnim črkam
- SQL'99: operator SIMILAR z regularnimi izrazi

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL

- Poišči imena jadralcev, ki so rezervirali rdeč ALI zelen čoln.

$$\pi_{jadralec.ime}(\sigma_{\substack{\text{barva=rdeca}\vee\\ \text{barva=zelena}}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      (c.barva='rdeca' OR c.barva='zelena');
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL

- Poišči imena jadralcev, ki so rezervirali rdeč IN zelen čoln.

$$\pi_{jadralec.ime}(\sigma_{\substack{\text{barva=rdeca} \wedge \\ \text{barva=zelena}}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND
      (c.barva='rdeca' AND c.barva='zelena');
```

```
+--------+
| ime    |
+--------+
+--------+
```

- Rezultat – prazna množica???
- Kje je napaka?
- Reševanje s pomočjo množic

Operatorji za delo z množicami

- UNION, UNION ALL: unija $\cup$ in unija s ponavljanjem elementov
- INTERSECT: presek $\cap$
- MINUS ali EXCEPT: razlika –
- IN, NOT IN (tabela): pripadnost $\in$ in $\notin$
- ALL, ANY: kvantifikatorja $\forall$ in $\exists$
- EXISTS, NOT EXISTS (tabela): praznost
- UNIQUE (tabela): enoličnost elementov v tabeli
- Operatorja IN in EXISTS sta osnova za gnezdenje poizvedb

SQL: operatorji za delo z množicami

- Unija: UNION, UNION ALL (ohrani duplikate)
- Presek: INTERSECT (mySQL ne podpira)
- Razlika: MINUS ali EXCEPT (mySQL ne podpira)
- Sintaksa:
  ```
  SELECT ...
  <OPERATOR>
  SELECT ...;
  ```
- Kompatibilnost tabel (ali rezultatov SELECT stavka): isto število stolpcev, istoležni stolpci istega tipa

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (množice)

- Poišči imena jadralcev, ki so rezervirali rdeč ALI zelen čoln.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec}) \cup$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
UNION
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='zelena';
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (množice)

- Poišči imena jadralcev, ki so rezervirali rdeč IN zelen čoln.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec}) \cap$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
INTERSECT
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='zelena';
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (množice)

- Poišči imena jadralcev, ki so rezervirali rdeč čoln, vendar nikoli zelenega.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec}) -$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
MINUS           -- ali EXCEPT
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='zelena';
```

```
+--------+
| ime    |
+--------+
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (množice)

- Poišči imena jadralcev, z ratingi 1, 3 ali 7.

$$\pi_{ime}(\sigma_{\substack{\text{rating}=1 \vee \\ \text{rating}=3 \vee \\ \text{rating}=7}}(\text{Jadralec}))$$

```
SELECT ime
FROM jadralec
WHERE (rating = 1) OR
      (rating = 3) OR
      (rating = 7);
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Borut  |
| Henrik |
| Anze   |
| Bine   |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (množice)

- Poišči imena jadralcev, z ratingi 1, 3 ali 7.

$$\pi_{ime}(\sigma_{\text{rating}\in\{1,3,7\}}(\text{Jadralec}))$$

```
SELECT ime
FROM jadralec
WHERE rating IN (1,3,7);
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Borut  |
| Henrik |
| Anze   |
| Bine   |
+--------+
```

Množica v SQL. Namesto seznama imamo lahko tudi tabelo ali rezultat neke poizvedbe.

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (gnezdene poizvedbe)

- Poišči imena jadralcev ki so rezervirali čoln s šifro 103.

```
SELECT ime
FROM jadralec
WHERE jid IN
   (SELECT jid
    FROM rezervacija -- Mnozica rezervacij
    WHERE cid=103);  -- colna 103
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (gnezdene poizvedbe)

- Poišči imena jadralcev ki so rezervirali kakšen rdeč čoln.

```
SELECT ime
FROM jadralec
WHERE jid IN        -- Mnozica rezervacij
   (SELECT jid      -- rdecih colnov
    FROM rezervacija
    WHERE cid IN    -- Mnozica rdecih colnov
        (SELECT cid
         FROM coln
         WHERE barva='rdeca'));
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Operatorji in množice v SQL - ponovitev

- Operatorji v SQL: =, !=, <>, <,>,<=,>=, LIKE, BETWEEN, AND, OR, NOT, NULL
- Operator AS: preimenovanje atributov, izračun novih atributov
- Operatorji za delo z množicami: UNION, UNION ALL, INTERSECT, MINUS, EXISTS, IN, UNIQUE, ALL, ANY
- Gnezdenje poizvedb: znotraj IN ali EXISTS operatorja, lahko tudi v FROM vrstici kot začasna tabela

Urejanje izpisa SELECT stavka

- SELECT stavku dodamo vrstico:
  ORDER BY ime_atributa [ASC ali DESC]
  ali
  ORDER BY številka_atributa [ASC ali DESC]

- Lahko urejamo tudi po izrazu ali na novo izračunanem atributu, ki ga ustrezno poimenujemo

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Urejanje izpisa

- Izpiši imena jadralcev urejena po količniku med ratingom in starostjo

  ROUND(stevilo, mest) zaokrozi rezultat na dano stevilo mest

```
SELECT ime, ROUND(rating/starost,2)
FROM jadralec;
```

```
+---------+--------------------------+
| ime     | ROUND(rating/starost,2) |
+---------+--------------------------+
| Darko   |                     0.16 |
| Borut   |                     0.03 |
| Lojze   |                     0.14 |
| Andrej  |                     0.31 |
| Rajko   |                     0.29 |
| Henrik  |                     0.20 |
| Zdravko |                     0.62 |
| Henrik  |                     0.26 |
| Anze    |                     0.12 |
| Bine    |                     0.05 |
+---------+--------------------------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Urejanje izpisa

- Izpiši imena jadralcev urejena po količniku med ratingom in starostjo

  ROUND(stevilo, mest) zaokrozi rezultat na dano stevilo mest

```
SELECT ime, ROUND(rating/starost,2)
FROM jadralec
ORDER BY 2 DESC;
```

```
+---------+--------------------------+
| ime     | ROUND(rating/starost,2) |
+---------+--------------------------+
| Zdravko |                     0.62 |
| Andrej  |                     0.31 |
| Rajko   |                     0.29 |
| Henrik  |                     0.26 |
| Henrik  |                     0.20 |
| Darko   |                     0.16 |
| Lojze   |                     0.14 |
| Anze    |                     0.12 |
| Bine    |                     0.05 |
| Borut   |                     0.03 |
+---------+--------------------------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Urejanje izpisa

- Izpiši imena jadralcev urejena po količniku med ratingom in starostjo

```
SELECT ime, ROUND(rating/starost,2)
            AS kolicnik
FROM jadralec
ORDER BY kolicnik DESC;
```

```
+---------+----------+
| ime     | kolicnik |
+---------+----------+
| Zdravko |     0.62 |
| Andrej  |     0.31 |
| Rajko   |     0.29 |
| Henrik  |     0.26 |
| Henrik  |     0.20 |
| Darko   |     0.16 |
| Lojze   |     0.14 |
| Anze    |     0.12 |
| Bine    |     0.05 |
| Borut   |     0.03 |
+---------+----------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (gnezdene poizvedbe)

- Poišči imena jadralcev ki niso rezervirali rdečega čoln.
  Kvantificirano: nobenega!

```
SELECT ime
FROM jadralec
WHERE jid NOT IN   -- Mnozica rezervacij
   (SELECT jid     -- rdecih colnov
    FROM rezervacija
    WHERE cid IN   -- Mnozica rdecih colnov
        (SELECT cid
         FROM coln
         WHERE barva='rdeca'));
```

```
+---------+
| ime     |
+---------+
| Borut   |
| Andrej  |
| Rajko   |
| Zdravko |
| Henrik  |
| Anze    |
| Bine    |
+---------+
```

```
Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)
```

SQL (gnezdene poizvedbe)

- Poišči imena jadralcev ki vsaj enkrat niso rezervirali rdečega čoln.

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva <> 'rdeca';
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

- Zakaj manj imen v rezultatu?
- Lahko so kdaj rezervirali rdeč čoln
- V prejšni poizvedbi tudi tisti, ki niso še nič rezervirali!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

SQL (Korelirane gnezdene poizvedbe)

- Poišči imena jadralcev ki so rezervirali čoln številna 103.

```
SELECT j.ime
FROM jadralec j
WHERE EXISTS                -- Neprazna mnozica
   (SELECT *                -- rezervacij colna
    FROM rezervacija r      -- 103 za vsakega
    WHERE r.cid = 103 AND   -- jadralca
          r.jid = j.jid);   -- posebej
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

- Problem: neučinkovitost, zato se jim izognemo, kadar je le mogoče.

Kvantifikatorji v SQL

- ANY (ali SOME): eksistenčni
- ALL: univerzalni
- Sintaksa (v WHERE vrstici):

```
WHERE atribut operator ANY ali ALL (mnozica)
    npr.
WHERE x<ANY(SELECT ...);
```

Pomen kvantifikatorjev

WHERE x<ANY(SELECT y ...); $\exists y : x < y$

WHERE x=ANY(SELECT y ...); $x \in \{y \mid ...\}$ Isto kot IN

WHERE x<>ANY(SELECT y ...); $\exists y : x \neq y$

WHERE x<ALL(SELECT y ...); $\forall y : x < y$

WHERE x=ALL(SELECT y ...); $\forall y : x = y$

WHERE x<>ALL(SELECT y ...); $\forall y : x \neq y$ Isto kot NOT IN

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Kvantifikatorji v SQL

- Poišči šifre jadralcev ki imajo ratinge višje kot jadralec z imenom Henrik. Opomba: Henrika sta dva!

```
SELECT j.jid
FROM jadralec j
WHERE j.rating > ANY
   (SELECT j2.rating
    FROM jadralec j2
    WHERE j2.ime='Henrik');
```

```
+-----+
| jid |
+-----+
|  31 |
|  32 |
|  58 |
|  71 |
|  74 |
+-----+
```

- Rating mora biti višji od vsaj enega Henrika!

```
Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)
```

Kvantifikatorji v SQL

- Poišči šifre jadralcev ki imajo najvišji rating!
  Opomba: lahko jih je več.

```
+-----+
| jid |
+-----+
|  58 |
|  71 |
+-----+
```

```
SELECT j.jid
FROM jadralec j
WHERE j.rating >= ALL
   (SELECT j2.rating
    FROM jadralec j2);
```

- Rating mora biti višji ali enak od vseh ratingov, torej tudi od lastnega!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Kvantifikatorji v SQL

- Poišči šifre jadralcev ki nimajo najnižjega ratinga!
  Opomba: lahko jih je več.

```
SELECT j.jid
FROM jadralec j
WHERE j.rating > ANY
   (SELECT j2.rating
    FROM jadralec j2);
```

```
+-----+
| jid |
+-----+
|  22 |
|  31 |
|  32 |
|  58 |
|  64 |
|  71 |
|  74 |
|  85 |
|  95 |
+-----+
```

- Rating mora biti strogo višji od vsaj enega ratinga.

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Presek z operatorjem

- Poišči imena jadralcev, ki so rezervirali rdeč IN zelen čoln.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec}) \cap$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
INTERSECT
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='zelena';
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Presek z uporabo gnezdenja

- Poišči imena jadralcev, ki so rezervirali rdeč IN zelen čoln.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec}) \cap$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times\mid_{cid} \text{Rezervacija} \mid\times\mid_{jid} \text{Jadralec})$$

```
SELECT DISTINCT j.ime                -- Prva mnozica
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
      AND j.ime IN (                 -- Druga mnozica
          SELECT DISTINCT j.ime
          FROM jadralec j, rezervacija r, coln c
          WHERE j.jid=r.jid AND r.cid=c.cid AND
                c.barva='zelena');
```

```
+--------+
| ime    |
+--------+
| Darko  |
| Lojze  |
| Henrik |
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Razlika množic z operatorjem

- Poišči imena jadralcev, ki so rezervirali rdeč čoln, vendar nikoli zelenega.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec}) -$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
MINUS           -- ali EXCEPT
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='zelena';
```

```
+--------+
| ime    |
+--------+
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Razlika množic z gnezdenjem

- Poišči imena jadralcev, ki so rezervirali rdeč čoln, vendar nikoli zelenega.

$$\pi_{jadralec.ime}(\sigma_{\text{barva=rdeca}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec}) -$$

$$\pi_{jadralec.ime}(\sigma_{\text{barva=zelena}}(\text{Coln}) \mid\times_{cid}\mid \text{Rezervacija} \mid\times_{jid}\mid \text{Jadralec})$$

```
SELECT DISTINCT j.ime
FROM jadralec j, rezervacija r, coln c
WHERE j.jid=r.jid AND r.cid=c.cid AND
      c.barva='rdeca'
      AND j.ime NOT IN (
          SELECT DISTINCT j.ime
          FROM jadralec j, rezervacija r, coln c
          WHERE j.jid=r.jid AND r.cid=c.cid AND
                c.barva='zelena');
```

```
+--------+
| ime    |
+--------+
+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Implementacija deljenja

- Poišči imena jadralcev, ki so rezervirali vse čolne.

```
SELECT j.ime
FROM jadralec j
WHERE NOT EXISTS              -- Vsi - Rezervirani = prazna mnozica
   (SELECT c.cid              -- Vsi colni
    FROM coln c
    MINUS
    SELECT r.cid              -- Rezervirani colni
    FROM rezervacija r        -- (korelirana)
    WHERE r.jid = j.jid);
```

$$\pi_{ime}(\pi_{jid,cid}(\text{Rezervacija})/\pi_{cid}(\text{Coln}) \rtimes_{jid} \text{Jadralec})$$

```
+--------+
| ime    |
+--------+
| Darko  |
+--------+
```

SQL - ponovitev

- urejanje izpisa (ORDER BY)
- nekorelirane in korelirane gnezdene poizvedbe
- implementacija preseka in razlike z gnezdenjem
- kvantifikatorji v SQL
  - ANY ali SOME
  - ALL
- deljenje

Skupinski operatorji v SQL

- Običajni operatorji delujejo nad posameznimi vrsticami kartezičnega produkta
- Skupinski operatorji in funkcije delujejo nad skupinami (množicami), torej nad več vrsticami istočasno
- Rezultat (izračunana vrednost) skupinskega operatorja postane skupinski atribut, ki ga ne smemo mešati z navadnimi atributi

Skupinski operatorji

- Sintaksa:
  `OPERATOR ([DISTINCT] ime_atributa)`

- COUNT(): prešteje (različne) vrstice
- SUM(): sešteje (različne) vrednosti
- AVG (): povprečje (različnih)
- MIN(): minimum
- MAX(): maksimum

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Štetje (COUNT)

- Preštej, koliko je vseh jadralcev!

```
SELECT COUNT(*)  -- presteje stevilo vrstic
FROM jadralec;   -- v tabeli jadralcev
```

```
+----------+
| COUNT(*) |
+----------+
|       10 |
+----------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Štetje (COUNT)

- Preštej, koliko je jadralcev z različnimi imeni!

```
SELECT COUNT(DISTINCT ime)  -- presteje razlicnih vrednosti
FROM jadralec;              -- atributa ime v tabeli jadralcev
```

Tipična uporaba operatorja COUNT:

- COUNT(*)
- COUNT(DISTINCT ime_atributa)

```
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Povprečje (AVG)

- Izračunaj povprečno starost jadralcev!

```
SELECT AVG(starost)          -- sesteje vrednosti atributa
FROM jadralec;               -- starost v tabeli jadralcev
```

```
+--------------+
| AVG(starost) |
+--------------+
|         36.9 |
+--------------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Povprečje (AVG)

- Izračunaj povprečno starost jadralcev z ratingom 10!

```
SELECT AVG(starost)          -- sesteje vrednosti atributa
FROM jadralec                -- starost v tabeli jadralcev
WHERE rating = 10;           -- vendar le za tiste z ratingom 10
```

```
+--------------+
| AVG(starost) |
+--------------+
|         25.5 |
+--------------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Minimum (MIN) in maksimim (MAX)

- Poišči minimalno in maksimalno starost jadralcev!

```
SELECT MIN(starost)
FROM jadralec;
```

```
+--------------+
| MIN(starost) |
+--------------+
|           16 |
+--------------+
```

```
SELECT MAX(starost)
FROM jadralec;
```

```
+--------------+
| MAX(starost) |
+--------------+
|         63.5 |
+--------------+
```

Skupinski operatorji

- Lahko nastopajo v SELECT ali WHERE vrstici
- V SELECT ali WHERE vrstici se v dani poizvedbi lahko nahajajo samo navadni ali samo skupinski atributi (ne smemo jih mešati)
- V primeru da potrebujemo oboje atribute, uporabimo gnezdene poizvedbe

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Skupinski operatorji

- Poišči imena in starost najstarejših jadralcev!

```
SELECT ime, MAX(starost)
FROM jadralec;
```

```
SELECT ime, starost
FROM jadralec
WHERE starost = MAX(starost);
```

```
SELECT ime, starost
FROM jadralec
WHERE starost = ( SELECT MAX(starost)
                  FROM jadralec );
```

```
+------+---------+
| ime  | starost |
+------+---------+
| Bine |    63.5 |
+------+---------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Skupinski operatorji namesto kvantifikatorjev

- Poišči šifre jadralcev ki imajo najvišji rating!
  Opomba: lahko jih je več.

```
SELECT j.jid
FROM jadralec j
WHERE j.rating >= ALL
   (SELECT j2.rating
    FROM jadralec j2);
```

```
+-----+
| jid |
+-----+
|  58 |
|  71 |
+-----+
```

- Rating mora biti višji ali enak od vseh ratingov, torej tudi od lastnega!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Skupinski operatorji namesto kvantifikatorjev

- Poišči šifre jadralcev ki imajo najvišji rating!
  Opomba: lahko jih je več.

```
SELECT jid
FROM jadralec
WHERE rating = (SELECT MAX(rating)
                FROM jadralec);
```

```
+-----+
| jid |
+-----+
|  58 |
|  71 |
+-----+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Skupinski operatorji namesto kvantifikatorjev

- Poišči šifre jadralcev ki nimajo najnižjega ratinga!
  Opomba: lahko jih je več.

```
SELECT j.jid
FROM jadralec j
WHERE j.rating > ANY
   (SELECT j2.rating
    FROM jadralec j2);
```

```
+-----+
| jid |
+-----+
|  22 |
|  31 |
|  32 |
|  58 |
|  64 |
|  71 |
|  74 |
|  85 |
|  95 |
+-----+
```

- Rating mora biti strogo višji od vsaj enega ratinga.

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Skupinski operatorji namesto kvantifikatorjev

- Poišči šifre jadralcev ki nimajo najnižjega ratinga! Opomba: lahko jih je več.

```
SELECT jid
FROM jadralec
WHERE rating > (SELECT MIN(rating)
                FROM jadralec);
```

```
+-----+
| jid |
+-----+
|  22 |
|  31 |
|  32 |
|  58 |
|  64 |
|  71 |
|  74 |
|  85 |
|  95 |
+-----+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Skupinski operatorji

- Poišči imena jadralcev, starejših od najstarejšega jadralca z ratingom 10!

```
SELECT ime
FROM jadralec
WHERE starost > (SELECT MAX(starost)
                 FROM jadralec
                 WHERE rating = 10);
```

```
+-------+
| ime   |
+-------+
| Darko |
| Lojze |
| Bine  |
+-------+
```

Delo nad skupinami

- Skupinski operatorji znotraj ene poizvedbe delujejo le nad eno skupino (množico)
- Če želimo istočasno dobiti rezultate nad več skupinamo moramo razširiti SELECT stavek z novimi vrsticami, ki omogočajo uporabo skupinskih operatorjev nad skupinami vrstic
- Primer naloge: za vsak rating v tabeli jadralcev izpiši starost najmlajšega jadralca s tem ratingom.

Razširjeni SELECT stavek

```
SELECT [DISTINCT] A1, ..., Ak     -- projekcija

FROM T1, T2, ..., Tn              -- kartezicni produkt

WHERE P1                          -- selekcija po vrsticah

GROUP BY A1, A2, .. Am            -- grupiranje po atributih

HAVING P2                         -- selekcija po skupinah

ORDER BY A1 [ASC ali DESC]; -- urejanje
```

Razširjeni SELECT stavek

- GROUP BY x: razdeli množico iz SELECT-FROM-WHERE na podmnožice glede na enake vrednosti atributa x
- GROUP BY x_1, x_2, ..., x_n: skupine imajo enake vrednosti vseh n atributov (torej je število skupin enako moči kartezičnega produkta n atributov)
- Vsak atribut, ki se nahaja v SELECT vrstici, se mora nahajati tudi v GROUP BY vrstici
- S pogojem HAVING P ohranimo samo tiste skupine, ki izpolnjujejo ta pogoj
- V HAVING vrstici se lahko nahajajo le skupinski atributi in operatorji

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rating v tabeli jadralcev izpiši starost najmlajšega jadralca s tem ratingom.

```
SELECT MIN(starost)
FROM jadralec
WHERE rating = i;  -- za i = 1, 2, ... 10
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rating v tabeli jadralcev izpiši starost najmlajšega jadralca s tem ratingom.

```
SELECT rating, MIN(starost)
FROM jadralec
GROUP BY rating;
```

```
+--------+--------------+
| rating | MIN(starost) |
+--------+--------------+
|      1 |           33 |
|      3 |         25.5 |
|      7 |           35 |
|      8 |         25.5 |
|      9 |           35 |
|     10 |           16 |
+--------+--------------+
```

- Ali s tem mešamo navadne in skupinske atribute?
- Ne, ker po grupiranju rating postane skupinski atribut!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rating v tabeli jadralcev izpiši starost najmlajšega polnoletnega jadralca s tem ratingom, vendar samo za tiste ratinge, ki jih imata vsaj dva jadralca!

```
SELECT rating, MIN(starost)
               AS najmlajsi
FROM jadralec
WHERE starost >= 18
GROUP BY rating
HAVING COUNT(*) > 1;
```

```
+--------+--------------+
| rating |    najmlajsi |
+--------+--------------+
|      3 |         25.5 |
|      7 |           35 |
|      8 |         25.5 |
+--------+--------------+
```

Kako deluje ta poizvedba (1)?

jid	ime	rating	starost
22	Darko	7	45
29	Borut	1	33
31	Lojze	8	55.5
32	Andrej	8	25.5
58	Rajko	10	35
64	Henrik	7	35
71	Zdravko	10	16
74	Henrik	9	35
85	Anze	3	25.5
95	Bine	3	63.5

1. korak:
vsi jadralci

Kako deluje ta poizvedba (2)?

jid	ime	rating	starost
22	Darko	7	45
29	Borut	1	33
31	Lojze	8	55.5
32	Andrej	8	25.5
58	Rajko	10	35
64	Henrik	7	35
74	Henrik	9	35
85	Anze	3	25.5
95	Bine	3	63.5

2. korak: selekcija
WHERE starost>=18

Kako deluje ta poizvedba (3)?

rating	starost
7	45
1	33
8	55.5
8	25.5
10	35
7	35
9	35
3	25.5
3	63.5

3. korak: eliminacija nepotrebnih atributov
samo atributi iz SELECT in WHERE vrstice so potrebni za nadaljnje delo

Kako deluje ta poizvedba (4)?

rating	starost
1	33
3	25.5
3	63.5
7	45
7	35
8	55.5
8	25.5
9	35
10	35

4. korak: grupiranje po vrednosti atributa rating

Kako deluje ta poizvedba (5)?

rating	starost
3	25.5
3	63.5
7	45
7	35
8	55.5
8	25.5

5. korak: eliminacija odvečnih skupin. Ohranimo samo tiste, za katere velja HAVING COUNT(*) > 1

Kako deluje ta poizvedba (6)?

rating	starost
3	25.5
7	35
8	25.5

6. korak: izvajanje skupinskega operatorja (v naše primeru MIN) na vsaki posamezni skupini

Opomba: če bi naša SELECT vrstica vsebovala DISTINCT, bi se podvojene vrstice izločile šele po 6. koraku!

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rdeč čoln izpišite število rezervacij!

```
SELECT c.cid, COUNT(*) AS St_rez
FROM coln c, rezervacija r
WHERE c.cid = r.cid AND barva='rdeca'
GROUP BY c.cid;
```

```
+-----+--------+
| cid | St_rez |
+-----+--------+
| 102 |      3 |
| 104 |      2 |
+-----+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rdeč čoln izpišite število rezervacij!

```
SELECT c.cid, COUNT(*) AS St_rez
FROM coln c, rezervacija r
WHERE c.cid = r.cid
GROUP BY c.cid
HAVING c.barva='rdeca';
```

```
SELECT c.cid, COUNT(*) AS St_rez
FROM coln c, rezervacija r
WHERE c.cid = r.cid
GROUP BY c.cid, c.barva
HAVING c.barva='rdeca';
```

```
+-----+--------+
| cid | St_rez |
+-----+--------+
| 102 |      3 |
| 104 |      2 |
+-----+--------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rating z najmanj dvema jadralcema izpišite povprečno starost jadralcev!

```
SELECT j.rating, AVG(j.starost) as Povp_star
FROM jadralec j
GROUP BY j.rating
HAVING COUNT(*) > 1;
```

```
+--------+-----------+
| rating | Povp_star |
+--------+-----------+
|      3 |      44.5 |
|      7 |        40 |
|      8 |      40.5 |
|     10 |      25.5 |
+--------+-----------+
```

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rating z najmanj dvema jadralcema izpišite povprečno starost jadralcev!
  Alternativna rešitev z gnezdenjem v HAVING vrstici.

```
SELECT j.rating, AVG(j.starost) as Povp_star
FROM jadralec j
GROUP BY j.rating
HAVING 1 < (SELECT COUNT(*)
            FROM jadralec j2
            WHERE j.rating = j2.rating);
-- korelirano gnezdenje
-- j.rating je skupinski atribut in
-- zato lahko nastopa v HAVING vrsici
-- (tudi v gnezdenem delu)
```

```
+--------+-----------+
| rating | Povp_star |
+--------+-----------+
|      3 |      44.5 |
|      7 |        40 |
|      8 |      40.5 |
|     10 |      25.5 |
+--------+-----------+
```

Gnezdenje v FROM vrstici

- Pozvedbe lahko gnezdimo tudi v FROM vrstici, pri čemer se rezultat poizvedbe naprej obravnava kot (začasna) tabela in ga je zato potrebno ustrezno poimenovati
- Vsi atributi v SELECT vrstici gnezdene poizvedbe morajo imeti eksplicitno določena imena

```
SELECT stik.*
FROM (SELECT j.jid, r.cid, j.starost
      FROM jadralec j, rezervacija r
      WHERE j.jid = r.jid) AS stik
WHERE stik.starost > 40;



-- Zacasna tabela stik ---------->
```

jid	cid	starost
22	101	45
22	102	45
22	103	45
22	104	45
31	102	55.5
31	103	55.5
31	104	55.5
64	101	35
64	102	35
74	103	35

Jadralec(jid, ime, rating, starost)
Coln(cid, ime, dolzina, barva)
Rezervacija(jid, cid, dan)

Delo s skupinami

- Za vsak rating z najmanj dvema jadralcema izpišite povprečno starost jadralcev!
  Alternativna rešitev z gnezdenjem v FROM vrstici.

```
SELECT t.rating, t.Povp_star
FROM ( SELECT j.rating, AVG(j.starost) AS Povp_star,
                        COUNT(*) AS St_ratingov
       FROM jadralec j
       GROUP BY j.rating ) AS t
WHERE t.St_ratingov > 1;

-- rezultat gnezdene poizvede se
-- uporablja kot zacasna tabela t
-- vse nove atribute moramo poimenovati
```

rating	Povp_star
3	44.5
7	40
8	40.5
10	25.5

Skupinski operatorji - ponovitev

- Omogočajo delo nad skupinami več vrstic hkrati
- Razširitev SELECT stavka omogoča skupinske operacije nad več skupinami:
- GROUP BY A1, A2, ... določa možne skupine
- HAVING P določa katere od možnih skupin ohranimo

Pogledi (views) v SQL

- Pogled (VIEW) je tabela, katare vrstice NISO shranjene podatkovni bazi, ampak se sproti računajo na podlagi *definicije pogleda*.
- Uporaba pogledov: pogosto uporabljane poizvedbe, omejitev dostopa do nekaterih stolpcev, izločevanje nepotrebnih detajlov
- Pogledi so definirani s SELECT stavki
- Vsaka sprememba v bazi se pozna v pogledu in obratno: vsaka sprememba v pogledu se pozna v bazi

Kreiranje in brisanje pogledov

- Sintaksa za kreiranje:

```
CREATE VIEW ime_pogleda(imena atributov)
AS SELECT stavek;
```

- Imena atributov lahko izpustimo; v tem primeru so v pogledu vsi atributi rezultata poizvedbe
- Paziti moramo na morebitna podvojena imena atributov in jih po potrebi preimenovati
- Sintaksa za brisanje:

```
DROP VIEW ime_pogleda;
```

Pogledi: rezervacija z barvo

```
CREATE VIEW barv_rez
AS SELECT r.*, c.barva
   FROM coln c, rezervacija r
   WHERE c.cid = r.cid;

SELECT *
FROM barv_rez;
```

jid	cid	dan	barva
22	101	2006-10-10	modra
64	101	2006-09-05	modra
22	102	2006-10-10	rdeca
31	102	2006-11-10	rdeca
64	102	2006-09-08	rdeca
22	103	2006-10-08	zelena
31	103	2006-11-06	zelena
74	103	2006-09-08	zelena
22	104	2006-10-07	rdeca
31	104	2006-11-12	rdeca

Pogledi: coln z rezervacijo (1)

```
CREATE VIEW coln_rez
AS SELECT *
   FROM coln c, rezervacija r
   WHERE c.cid = r.cid;
```

- Problem: dva stolpca z istim imenom (cid)
- Lahko rešimo na tri načine:
  - preimenujemo v SELECT stavku
  - preimenujemo v CREATE VIEW stavku
  - izločimo podvojene atribute

Pogledi: coln z rezervacijo (2)

```
CREATE VIEW coln_rez
AS SELECT r.*, c.cid AS ccid, c.ime, c.barva, c.dolzina
   FROM coln c, rezervacija r
   WHERE c.cid = r.cid;
SELECT * FROM coln_rez;
```

jid	cid	dan	ccid	ime	barva	dolzina
22	101	2006-10-10	101	Elan	modra	34
64	101	2006-09-05	101	Elan	modra	34
22	102	2006-10-10	102	Elan	rdeca	34
31	102	2006-11-10	102	Elan	rdeca	34
64	102	2006-09-08	102	Elan	rdeca	34
22	103	2006-10-08	103	Sun Odyssey	zelena	37
31	103	2006-11-06	103	Sun Odyssey	zelena	37
74	103	2006-09-08	103	Sun Odyssey	zelena	37
22	104	2006-10-07	104	Bavaria	rdeca	50
31	104	2006-11-12	104	Bavaria	rdeca	50

Pogledi: coln z rezervacijo (3)

```
CREATE VIEW coln_rez(jid,cid,dan,ccid,ime,barva,dolzina)
AS SELECT *
   FROM coln c, rezervacija r
   WHERE c.cid = r.cid;
SELECT * FROM coln_rez;
```

jid	cid	dan	ccid	ime	barva	dolzina
22	101	2006-10-10	101	Elan	modra	34
64	101	2006-09-05	101	Elan	modra	34
22	102	2006-10-10	102	Elan	rdeca	34
31	102	2006-11-10	102	Elan	rdeca	34
64	102	2006-09-08	102	Elan	rdeca	34
22	103	2006-10-08	103	Sun Odyssey	zelena	37
31	103	2006-11-06	103	Sun Odyssey	zelena	37
74	103	2006-09-08	103	Sun Odyssey	zelena	37
22	104	2006-10-07	104	Bavaria	rdeca	50
31	104	2006-11-12	104	Bavaria	rdeca	50

Materializirani pogledi (ORACLE)

- Posnetek rezultatov poizvedbe, ki definira pogled
- Problem: ažuriranje posnetka
- Zahteva posebne privilegije
- Lahko ga indeksiramo

ORACLE:

```
CREATE MATERIALIZED VIEW ...
```

Indeksiranje v SQL

- Indeks je za uporabnika nevidna podatkovna struktura, ki bistveno pospeši dostop do vrstic tabele; preiskovanje *n* vrstic: s $t_1=O(n)$ na $t_2=O(\log n)$;
- pri n=1 milijon je $t_1=O(100000)$, $t_2=O(6)$
- Indeksiramo po enem ali več stolpcih
- Zakaj vedno ne indeksiramo celotne tabele:
  - za *k* atributov je možnih 2^k indeksov (vse podmnožice)
  - vsak indeks zahteva prostor na disku in čas za njegovo gradnjo in posodabljanje ob spremembah tabele

Kdaj zgraditi indeks na podmnožici atributov?

- Ključi in ostali enolični (UNIQUE) atributi: pogosto avtomatska generacija
- Pogostost preiskovanja in urejanja
- Velikost tabele
- Porazdelitev podatkov
- Prostorsko-časovne omejitve: prostor na voljo v PB, pogostost spreminjanja tabele

Kreiranje in brisanje indeksov

- Kreiranje indeksov:

```
CREATE [UNIQUE] INDEX ime_indeksa
ON ime_tabele (ime_atributa1 [ASC|DESC],
               ime_atributa2 [ASC|DESC],
               ... );
```

- Indeks se gradi po kombinaciji vrednosti atributov; za vsako kombinacijo atributov potrebujemo svoj indeks
- Možna specifikacija tipa indeksa (npr. BTREE)
- Brisanje nepotrebnih indeksov:

```
DROP INDEX ime_indeksa ON ime_tabele;
```

Primer indeksiranja

- Indeksiraj čolne po barvi!

```
CREATE INDEX po_barvi
ON coln(barva);
```

- Indeksiraj rezervacije ločeno po datumih ter šifrah jadralcev in čolnov skupaj!

```
CREATE INDEX po_dnevih
ON rezervacija(dan);

CREATE INDEX po_jid_cid
ON rezervacija(jid,cid);
```

Uporaba indeksov

- Indeksi se uporabljajo avtomatsko, ko jih enkrat kreiramo; sistem izbere, katerega od potencialno več možnih obstoječih bo uporabil
- Eksplicitna (ne)uporaba indeksov: dosežemo s specialnimi komentarji - namigi (hints)
- Zakaj namigi? Ker vnaprej vemo več kot sistem o tem, kako se bodo podatki uporabljali
- Namigi so NESTANDARDNA razširitev SQL

Namigi za indeksiranje v Oraclu

- Namig: dodan komentar k SELECT stavku
- Sintaksa: **`/*+ NAMIG(parametri) */`**
- Namigi za indeksiranje so naslednji:
  - INDEX(atributi): uporabi indeks po specificiranih atributih
  - INDEX_ASC(atributi): uporabi naraščajoči indeks
  - INDEX_DESC(atributi): uporabi padajoči indeks
  - NO_INDEX(atributi): ne uporabi speciranega indeksa
  - FULL(tabela): ne uporabi nobenega indeksa tabele

Primeri nekaterih namigov v Oraclu

```
SELECT /*+ INDEX(jid, ime) */ *  -- uporabi indeks
FROM jadralec                    -- po jid in imenu
ORDER BY ime, jid;

SELECT /*+ FULL(jadralec) */  *  -- full search
FROM jadralec           -- po tabeli jadralec: ne
ORDER BY ime, jid;      -- uporabi nobenega indeksa
```

Namigi za indeksiranje v MySQL

- Namig: dodana ključna beseda v SELECT stavku

```
USE INDEX(ime_indeksa1, ime_indeksa2, ...)
IGNORE INDEX(ime_indeksa1, ime_indeksa2, ...)
```

Primeri nekaterih namigov v MySQL

- Denimo, da smo vnaprej kreirali indekse **jad_index1(jid,ime), jad_index2(jid), jad_index3(ime).**

```
SELECT *
FROM jadralec
USE INDEX(jad_index1)        -- uporabi indeks
ORDER BY ime, jid;           -- po jid in imenu

SELECT *
FROM jadralec
IGNORE INDEX(jad_index1, jad_index2, jad_index3)
ORDER BY ime, jid;  -- ne uporabi nobenega indeksa
```

Data definition language - DDL

- Tipi atributov
- Kreiranje in spreminjanje tabel
- Polnjenje tabel

Tipi atributov

- Numerični tipi: NUMBER, INTEGER, FLOAT, DOUBLE, DECIMAL, ...
- Znakovni tipi: CHAR, VARCHAR, TEXT, ...
- Datumski tip: DATE
- Netipizirani tip: BLOB (binary large object)

Kreiranje tabel

- Sintaksa:

```
CREATE TABLE ime_tabele
    (atributi in omejitve)
    druge opcije;
```

- Primer:

```
CREATE TABLE Jadralec
( jid    INTEGER,          -- atributi
  ime    VARCHAR(10),
  rating INTEGER,
  starost REAL,
  PRIMARY KEY (jid),       -- omejitve
  CHECK ( rating >= 1 AND rating <= 10 ));
```

Polnjenje tabel

- Sintaksa:
  INSERT INTO ime_tabele VALUES(v1, ..., vn);
  INSERT INTO ime_tabele VALUES(ime1=v1, ..., imek=vk);
- V drugem primeru lahko nekatere vrednosti manjkajo in dobijo vrednost NULL
- Primer:

```
insert into Jadralec
       values( 22, 'Darko', 7, 45.0);
insert into Jadralec
       values( jid=29, ime='Borut',
               rating=1, starost=33.0);
```

Spreminjanje in brisanje tabel

- Spreminjanje tabel:
  - ALTER TABLE ime_tabele opcije;
  - dodajanje, brisanje, preimenovanje in spreminjanje atributov
  - dodajanje, brisanje in spreminjanje omejitev, indeksov, ...
- Brisanje tabel:
  - DROP TABLE ime_tabele;

SQL*Plus: nekaj uporabnih napotkov

SKRIPTA ZA BRISANJE VSEH INDEKSOV:

```
set linesize 132
set pagesize 0
set heading off
set verify off
set echo off
set feedback off
set termout off
spool drop_indexxx.sql
prompt spool drop_indexx.lis

SELECT ' DROP INDEX '|| index_name ||';'
FROM user_indexes;

prompt spool off
spool off
@drop_indexxx.sql
prompt Check drop_indexx.lis
```

SKRIPTA ZA BRISANJE VSEH POGLEDOV:

```
set linesize 132
set pagesize 0
set heading off
set verify off
set echo off
set feedback off
set termout off
spool drop_viewsss.sql
prompt spool drop_viewsss.lis

SELECT ' DROP VIEW '|| view_name ||';'
FROM user_views;

prompt spool off
spool off
@drop_viewsss.sql
prompt Check drop_viewsss.lis
```

Merjenje časa v SQL*Plus: SET TIMING ON

Indeksiranje na splošno

- Literatura:
  - R. Ramakrishnan, J. Gehrke: *Database Management Systems*, 2. izdaja, poglavji 8 in 9
- Splošni pojmi:
  - Indeks je podatkovna struktura, ki omogoča učinkovito delo (preiskovanje, spreminjanje, dodajanje, brisanje) nad dano tabelo (datoteko) z atributi, ki sestavljajo iskalni ključ
  - Iskalni ključ je lahko poljubna podmnožica atributov dane tabele. Iskalni ključ je tipično mnogo manjši od celotne vrstice v tabeli.

Indeksiranje na splošno

- Ločimo:
  - Ključ tabele (minimalna množica atributov, ki enolično določa vsako vrstico v tabeli)
  - Iskalni ključ (množica atributov, po katerih iščemo)
- Alternative indeksiranju:
  - Urejena datoteka (dobro za urejene izpise, iskanje v razponu vrednosti)
  - Dostop s pomočjo razpršene (hash) tabele (dobro za iskanje enakosti)

Analiza časovne zahtevnosti za iskanje enega elementa

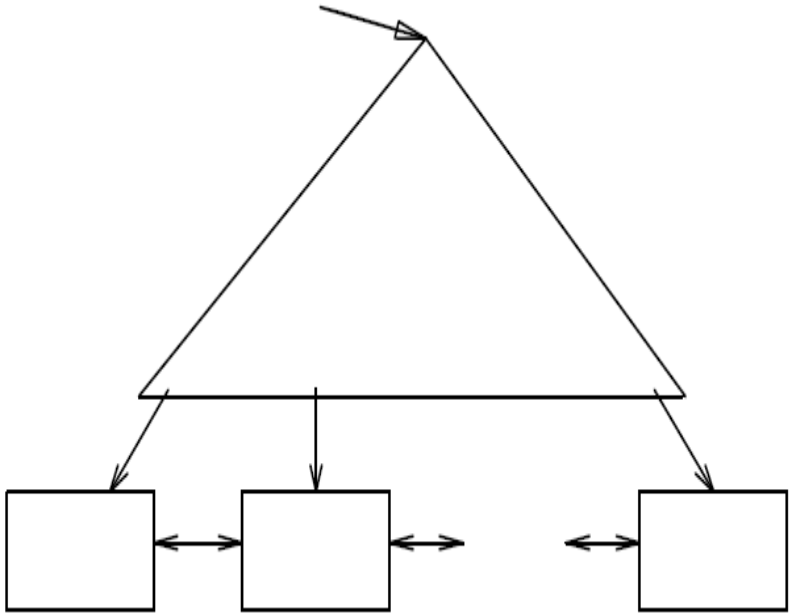
- Naključno urejena datoteka: *O(n)*
- Urejena datoteka: *O(log n)*, vendar ob spremembi *O(n log n)* za ponovno urejanje
- Hash tabela: idealno *O(1)*, v najslabšem primeru *O(n)*
- B indeks (najpopularnejši): tipično *O(log n)*, ob spremembi *O(log n)*

Vrste indeksov

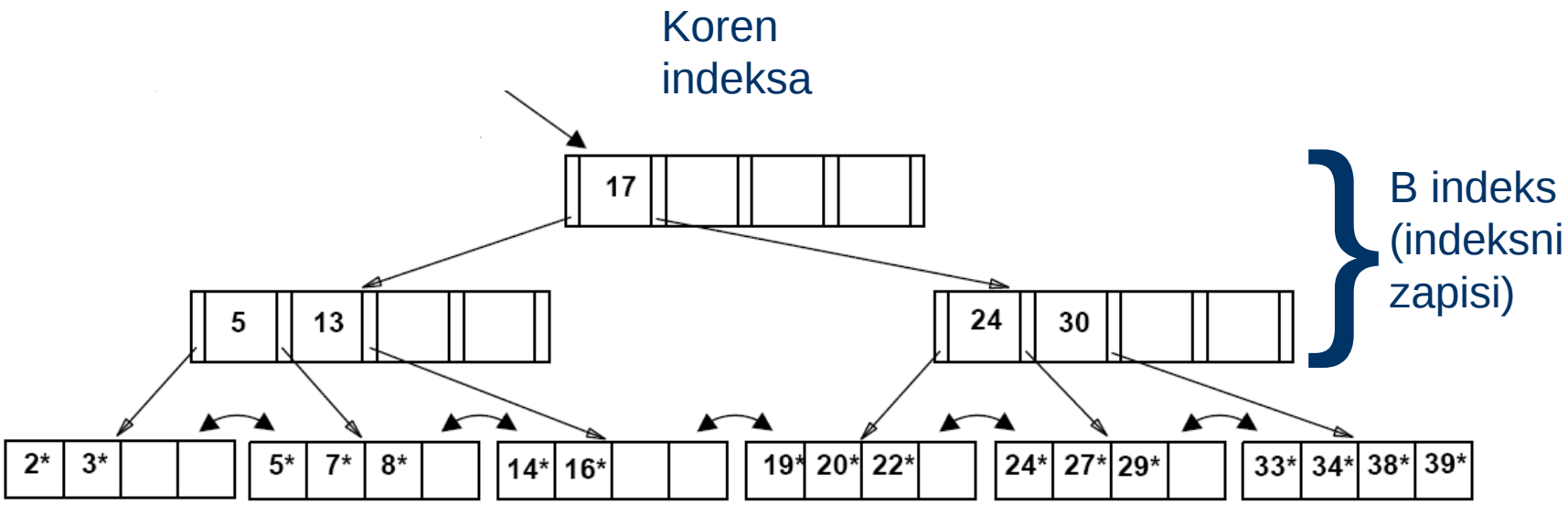
- Primarni in sekundarni:
  primarni vsebuje ključ tabele, sekundarni pa ne
- Gosti in redki:
  gosti indeks ima zapis za vsako vrstico iz dane tabele, redki pa ne

B in B+ drevesno indeksiranje

- V vozliščih podatkovne strukture – drevesa – se nahajajo zapisi, ki pospešujejo operacije iskanja, ...
- Predhodnik: ISAM – statična drevesna struktura
- B/B+: dinamična drevesna struktura, prilagaja se dodajanju in brisanju elementov (vrstic v originalni tabeli)

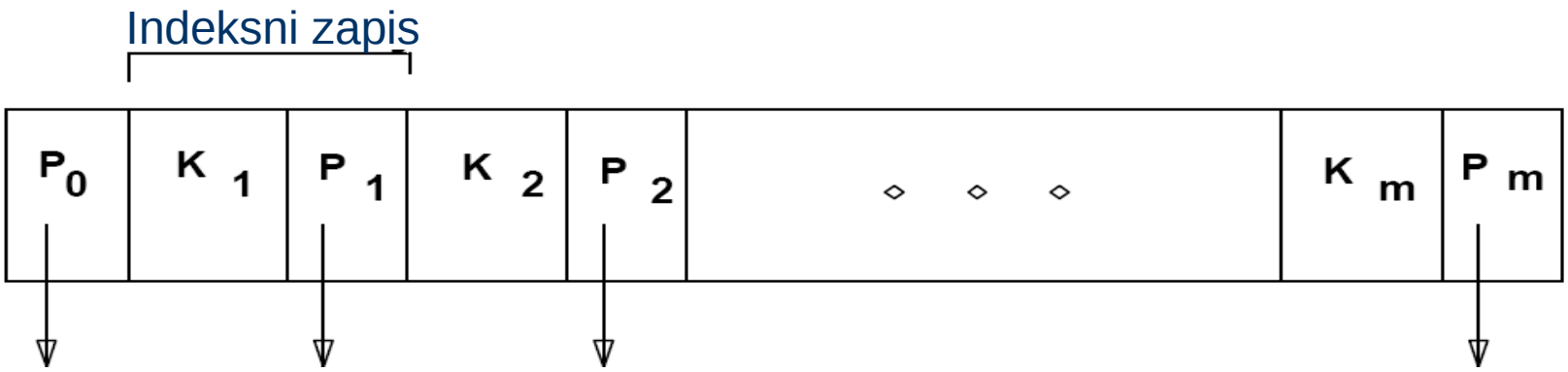


Primer B indeksa (red *d*=2, globina *g*=2)



Listi indeksa: podatkovni zapisi ali dodatni gosti indeks

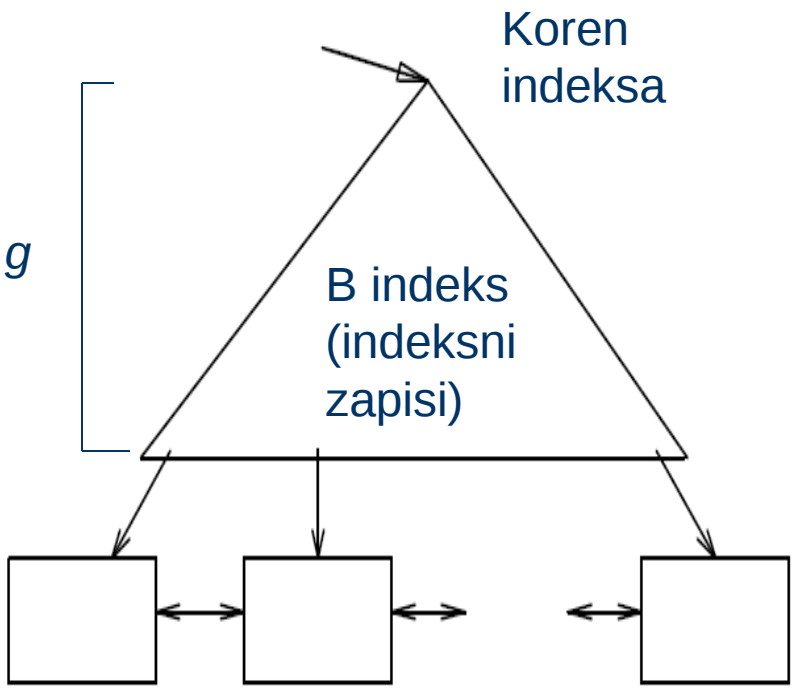
Struktura B in B+ indeksnega vozlišča



- P_i: kazalci na druga indeksna vozlišča
- K_i: iskalni ključi
- Ob iskanju zapisa v vrednostjo iskalnega ključa X se premaknemo v vozlišče s kazalcem P_i, za katerega velja:

  $K_i \leq X < K_{i+1}$

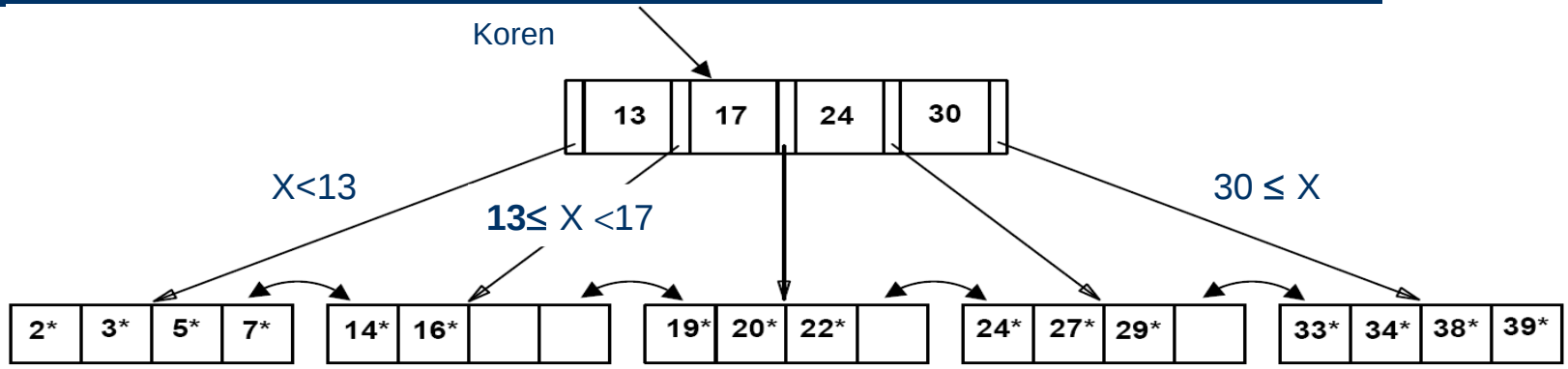
Zgradba B/B+ indeksa



Listi indeksa: podatkovni zapisi
ali dodatni gosti indeks

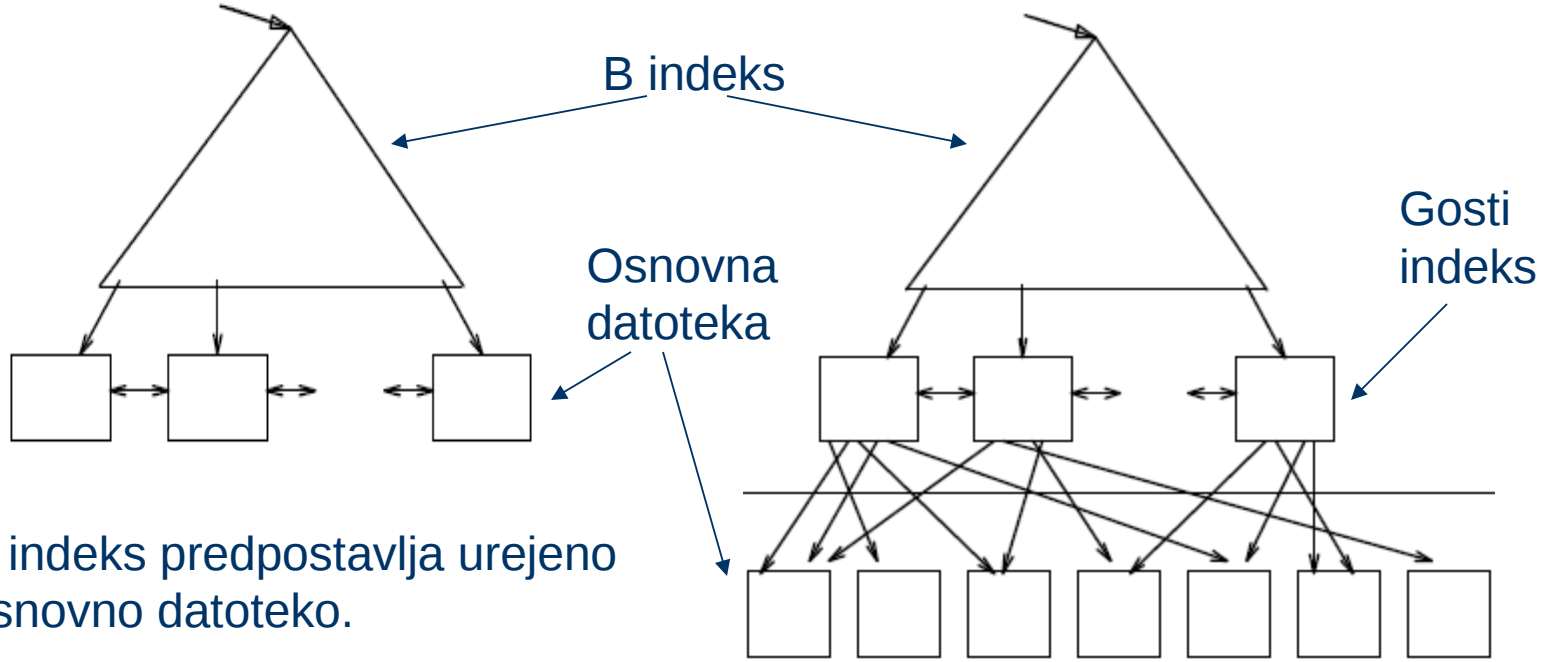
- Vsako indeksno vozlišče (razen eventuelno korena) vsebuje $d \leq m \leq 2d$ indeksnih zapisov.
- Vsako vozlišče ima kapaciteto *2d* in je torej vedno vsaj na pol polno.
- Parametru *d* pravimo red indeksa
- Globina indeksa *g*: razdalja (število indeksnih nivojev) od korena do listov (nivoja z listi ne štejemo)

Primeri iskanja po B indeksu



- Poiščimo zapis z iskalnim ključem X=5
- Poiščimo zapis z iskalnim ključem X=15
- Poiščimo zapis z iskalnim ključem X= 22
- Poiščimo vse zapise z iskalnim ključem X≥ 24

Razlika med B in B+ indeksom

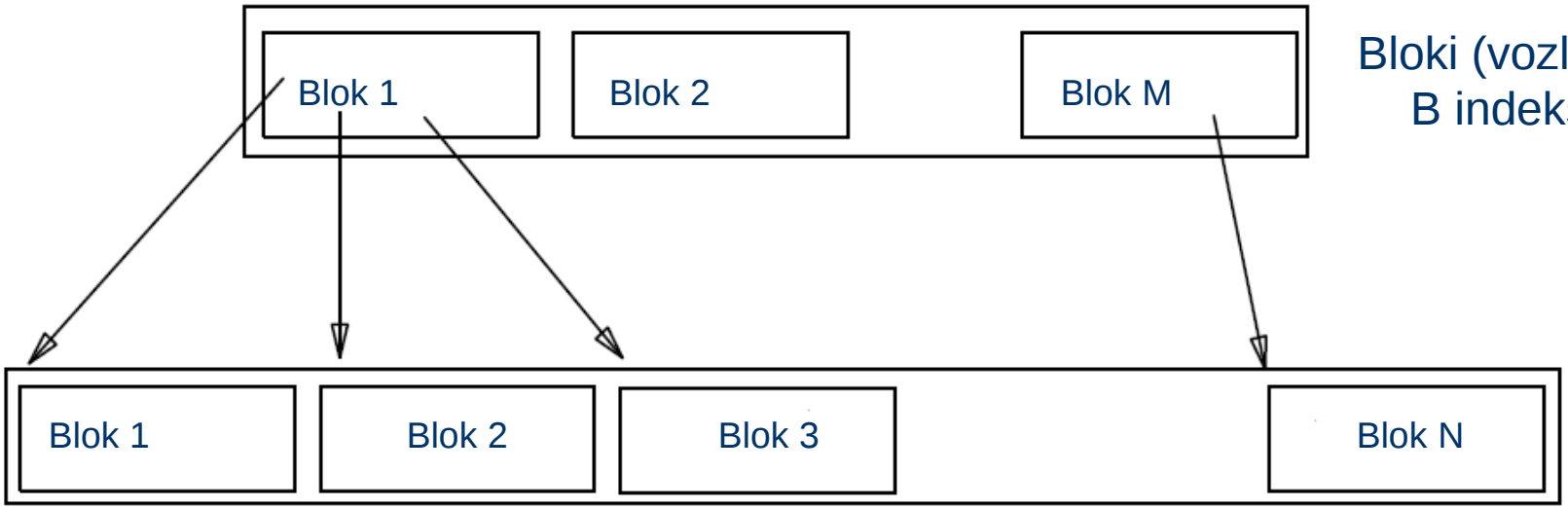


B indeks predpostavlja urejeno osnovno datoteko.

B+ indeks vsebuje dodatni nivo – gosti indeks, ki nam nudi urejen pogled na osnovno datoteko.

Torej:
B+ indeks =
B indeks + gosti indeks

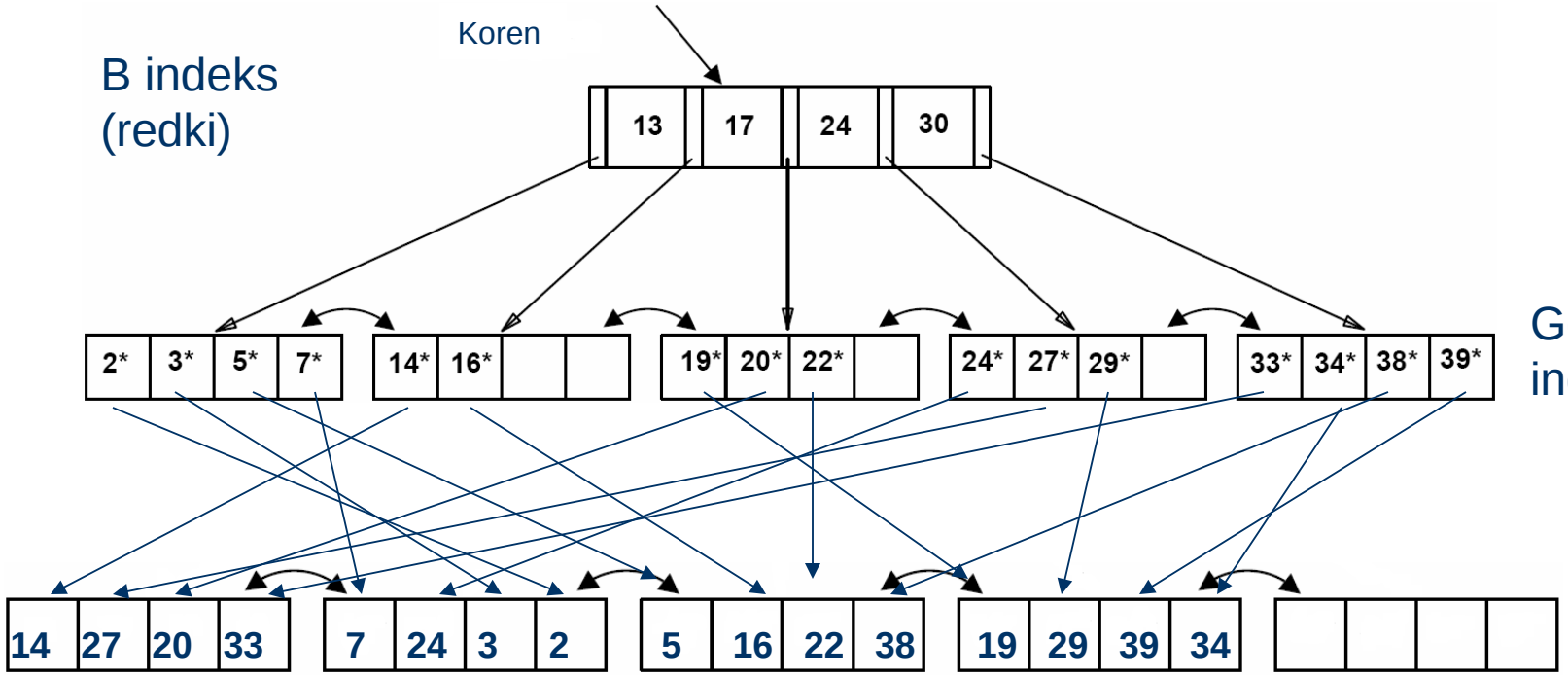
Gosti indeks v B+ indeksu



Bloki gostega indeksa

Bloki odražajo datotečno strukturu sistema, kjer teče PB. Npr. blok ima velikost 4096 bytov. Vsak blok gostega indeksa ima *k* indeksnih zapisov (ključev in kazalcev na neurejeno osnovno datoteko).

Primer B+ indeksa



Osnovna - neurejena datoteka (tabela)

Nekaj podatkov o B indeksih (1)

- Vozlišče reda *d* vsebuje:
  - $d \leq m \leq 2d$ indeksnih zapisov – parov (kazalec, ključ)
  - $d+1 \leq m \leq 2d+1$ kazalcev
- Indeks reda *d* z globino *g* vsebuje:
  - $(d+1)^{g-1} \leq n \leq (2d+1)^{g-1}$ vozlišč na zadnjem nivoju indeksa (nivo pred listi)
  - indeksira lahko $(d+1)^{g} \leq N \leq (2d+1)^{g}$ listov (blokov osnovne datoteke ali gostega indeksa)

Nekaj podatkov o B indeksih (2)

- Z uporabo gostega indeksa, ki vsebuje *k* indeksnih zapisov na blok, se število indeksiranih elementov pomnoži s *k*: $k(d+1)^g \leq N \leq k(2d+1)^g$
- Poraba prostora: indeks reda *d* z globino *g* vsebuje *M* indeksnih vozlišč:

$$\frac{(d+1)^g - 1}{d} \leq M \leq \frac{(2d+1)^g - 1}{2d}$$

Primer izračuna

- Imamo B+ indeks reda d=2, v blokih gostega indeksa pa imamo k=5 indeksnih zapisov. Kakšna bo globina g indeksa, če želimo indeksirati N=1000 zapisov?

$\left\lceil \frac{N}{k} \right\rceil$ je število blokov gostega indeksa (te indeksiramo)

$g_{\max} = \left\lceil \log_{d+1} \left\lceil \frac{N}{k} \right\rceil \right\rceil$ maksimalna globina pri minimalno polnih vozliščih

$g_{\min} = \left\lceil \log_{2d+1} \left\lceil \frac{N}{k} \right\rceil \right\rceil$ minimalna globina pri maksimalno polnih vozliščih

$g_{\max} = \log_3 200 \approx 4.8 = 5$

$g_{\min} = \log_5 200 \approx 3.3 = 4$

B+ indeksi v praksi

- Kriterij zapolnjenosti se (zaradi spremenljivih dolžin iskalnih ključev) običajno nanaša na zapolnjen prostor in ne na zapolnjene zapise
- Tipično so reda *d=100*, zapolnjeni *66.7%,* kar nam da povprečno število naslednikov posameznega vozlišča *200*66.7%=133*
- Tipične kapacitete:
  - Globina *3: 133^3=2,352,637* zapisov
  - Globina *4: 133^4=312,900,700* zapisov

Nadzor nad sočasno uporabo PB

- Literatura: Tomaž Mohorič, *Uvod v podatkovne baze*, 9. poglavje
- Sočasnost uporabe PB s strani mnogo uporabnikov, ki lahko dostopajo do istih podatkov (naslovov v bazi)
- Zagotavljanje konsistentnosti podatkov s pomočjo transakcij
- Transakcija je program (zaporedje običajnih programskih konstruktov in specializiranih transakcijskih ukazov, ki skrbijo za dostop do baze), ki se mora izvršiti v celoti, da je konsistentnost PB zagotovljena

Transakcijski ukazi

- Dogovor:
  - X (poljubna velika črka) je naslov v PB
  - x (poljubna mala črka) je lokalna spremenljivka transakcijskega programa
- Transakcijski ukazi:
  - Začetek: začetek transakcije; pogosto ga tudi izpustimo
  - Pomni: uspešen zaključek transakcije
  - Pozabi: neuspešen zaključek transakcije
  - PoiščiPreberi(X,x) ali krajše PP(X,x). Pomen: x = X
  - Dodaj(X,x)
  - Briši(X)
  - Spremeni(X,x)

} Nadomestimo z Ažuriraj(X,x) ali krajše A(X,x). Pomen: X = x

Ažuriranje podatkovne baze

- Neposredno ažuriranje
  - Ažuriranja se vršijo neposredno v bazo
  - Ob uspešnem zaključku transakcije ni potrebno narediti ničesar
  - Ob neuspešnem zaključku transakcije se spremembe razveljavijo in povrnejo v stanje pred začetkom izvajanja neuspešnega ažuriranja
- Odloženo ažuriranje
  - Ažuriranja se vršijo v pomožni strukturi (npr. dnevniku)
  - Ob uspešnem zaključku transakcije je potrebno spremembe uveljaviti (prepisati iz pomožne strukture v bazo)
  - Ob neuspešnem zaključku transakcije ni potrebno narediti ničesar, saj v bazi ni sprememb
- Če ne bo drugače zapisano, v nadaljevanju uporabljamo izključno neposredno ažuriranje

Primer transakcijskega programa

Transakcijski program v banki, ki tolarje preračuna v evre. Banka si za svoj trud vzame 1% zneska. Levo originalni, desno krajši zapis ukazov.

- R: račun stranke (parameter programa)
- B: račun banke (konstanta)
- t, p, e: lokalne spremenljivka

```
Program Evro(R):
        Začetek
        PoiščiPreberi(R,t)
        p = t*0.01
        e = (t-p)/239.64
        Ažuriraj(R,e)
        PoiščiPreberi(B,b)
        b = b + p
        Ažuriraj(B,b)
        Pomni
```

```
Program Evro(R):
        Začetek
        PP(R,t)
        p = t*0.01
        e = (t-p)/239.64
        A(R,e)
        PP(B,b)
        b = b + p
        A(B,b)
        Pomni
```

Primer transakcijskega programa

- V SUPB lahko istočasno teče več transakcijskih programov
- Transakcijski programi se lahko izvajajo zaporedno ali izmenično.
- Zaporedno: najprek se do konca izvrši en transakcijski program, nato drugi
- Izmenično: ukazi transakcijskih programov se prepletajo

Zaporedno izvajanje transakcij (1)

```
T1: Začetek
PoišČiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
Ažuriraj(X,e)
PoišČiPreberi(B,b)
b = b + p
Ažuriraj(B,b)
Pomni
```

```
T2: Začetek
PoišČiPreberi(Y,t)
p = t*0.01
e = (t-p)/239.64
Ažuriraj(Y,e)
PoišČiPreberi(B,b)
b = b + p
Ažuriraj(B,b)
Pomni
```

- Primer zaporednega izvajanja

  T1: Evro(X)
  T2: Evro(Y)

- Kakšen je rezultat, če je na začetku X=10.000, Y=20.000 in B=1000?

T1: X=41.31 Y= 20000 B=1100
T2: X=41.31 Y= 82.62 B=1300

Izmenično izvajanje transakcij (2)

```
T1: Začetek
PoiščiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    PoiščiPreberi(Y,t)

Ažuriraj(X,e)
PoiščiPreberi(B,b)
b = b + p
Ažuriraj(B,b)
                    p = t*0.01
                    e = (t-p)/239.64
                    Ažuriraj(Y,e)
                    PoiščiPreberi(B,b)
                    b = b + p
                    Ažuriraj(B,b)

Pomni
                    Pomni
```

T1: X=41.31 Y= 82.62 B=1300
T2: X=41.31 Y= 82.62 B=1300

- Primer izmeničnega izvajanja

  T1: Evro(X)
  T2: Evro(Y)

- Kakšen je rezultat, če je na začetku X=10.000, Y=20.000 in B=1000?

Izmenično izvajanje transakcij (3)

```
T1: Začetek
PoišČiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    PoišČiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
PoišČiPreberi(B,b)
                    Ažuriraj(Y,e)
                    PoišČiPreberi(B,b)
b = b + p
Ažuriraj(B,b)
                    b = b + p
Pomni
                    Ažuriraj(B,b)
                    Pomni
```

T1: X=41.31 Y= 82.62 B=1100
T2: X=41.31 Y= 82.62 B=1200

- Kaj pa takšno izmenično izvajanje?

  T1: Evro(X)
  T2: Evro(Y)

- Kakšen je rezultat, če je na začetku X=10.000, Y=20.000 in B=1000?

Izgubljeno ažuriranje naslova B v transakciji T1!

Izmenično izvajanje transakcij (4)

```
T1: Začetek
PoiščiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    PoiščiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
PoiščiPreberi(B,b)
                    Ažuriraj(Y,e)
b = b + p
Ažuriraj(B,b)
                    PoiščiPreberi(B,b)
                    b = b + p
Pozabi
                    Ažuriraj(B,b)
                    Pomni
```

T1: X=10.000 Y= 82.62 B=1000
T2: X=10.000 Y= 82.62 B=1300

- Kaj pa takšno izmenično izvajanje?

  T1: Evro(X)
  T2: Evro(Y)

- Kakšen je rezultat, če je na začetku X=10.000, Y=20.000 in B=1000?

Branje nelegalne vrednosti z naslova B v transakciji T2!

Izmenično izvajanje transakcij (5)

```
T1: Začetek
PoišČiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    PoišČiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
PoišČiPreberi(B,b)
                    Ažuriraj(Y,e)
b = b + p
Ažuriraj(B,b)
                    PoišČiPreberi(B,b)
                    b = b + p
                    Ažuriraj(B,b)
Pozabi
                    Pomni
```

T1: X=10.000 Y= 82.62 B=1000
T2: X=10.000 Y= 82.62 B=1000

- Kaj pa takšno izmenično izvajanje?

  T1: Evro(X)
  T2: Evro(Y)

- Kakšen je rezultat, če je na začetku X=10.000, Y=20.000 in B=1000?

Branje nelegalne vrednosti z naslova B in izgubljeno ažuriranje naslova B v transakciji T2!

Zaporedniški razpored ukazov

- Izmenično razporedu, katerega rezultat vedno ustreza nekemu zaporednemu razporedu transakcij, pravimo zaporedniški (serializable)
- V SUPB se splošni izmenični razporedi ukazov pretvorijo v zaporedniške z uporabo posebnih protokolov, ki temeljijo na zaseganju ali časovnem označevanju podatkov.

Zaseganje (zaklepanje) podatkov

- Novi transakcijski ukazi
  - E-PoiščiPreberi (E-PP): ekskluzivno zaseganje
  - D-PoiščiPreberi (D-PP): deljeno zaseganje

		Zahteva po zaseženju		
		Ekskluzivno	Deljeno	-
Trenutno	Ekskluzivno	ne	ne	da
zaseženje	Deljeno	ne	da	da
	-	da	da	da

Protokol PXC

- Temelji na ekskluzivnem zaseganju
  1. Transakcija, ki želi ažurirati podatek ali ga prebrati brez nevarnosti sočasnega ažuriranja, ga mora najprej ekskluzivno zaseči.
  2. Če transakciji zahteva po zaseženju ne more biti takoj odobrena, preide v stanje čakanja na odobritev, njeno izvajanje pa se nadaljuje po odobritvi zaseženja.
  3. Vsa zaseženja podatkov se smejo sprostiti šele po zaključku transakcije (po izvedeni uveljavitvi ali razveljavitvi ažuriranj).
  4. Transakcija, ki želi le prebrati podatek in ji ni mar za sočasno ažuriranje tega podatka s strani kake druge transakcije, ga sme prebrati ne glede na to, ali je podatek zasežen ali ne.

- Transakcij iz (4) je le malo in dajejo približno točne rezultate

Protokol PSC

- Temelji na ekskluzivnem in deljenem zaseganju
  1. Transakcija, ki želi ažurirati podatek, ga mora najprej ekskluzivno zaseči.
  2. Če transakciji zahteva po zaseženju ne more biti takoj odobrena, preide v stanje čakanja na odobritev, njeno izvajanje pa se nadaljuje po odobritvi zaseženja.
  3. Vsa zaseženja podatkov se smejo sprostiti šele po zaključku transakcije (po izvedeni uveljavitvi ali razveljavitvi ažuriranj).
  4. Transakcija, ki želi le prebrati podatek in ji ni mar za sočasno ažuriranje tega podatka s strani kake druge transakcije, ga sme prebrati ne glede na to, ali je podatek zasežen ali ne.
  5. Transakcija, ki želi ekskluzivno zaseči podatek, mora imeti pred tem odobreno njegovo deljeno zaseženje;

Protokola PXC in PSC

- Protokol PSC dopušča sočasno izvajanje dveh zgolj povpraševalnih transakcij, ki zahtevata zaščito pred ažuriranjem, protokol PXC pa ne.
- Protokol PSC veliko bolj dopušča nastop neskončne (mrtve) zanke kot PXC
- S sproščanjem vseh zaseženj šele ob zaključku transakcije je zagotovljena konsistentnost povpraševanj tudi v primerih, ko se transakcija zaključi neuspešno in se njena ažuriranja razveljavijo - izključi se možnost branja neveljavnega podatka.

Primer (3) s protokolom PXC

```
T1: Začetek
PoiščiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    PoiščiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
PoiščiPreberi(B,b)
                    Ažuriraj(Y,e)
                    PoiščiPreberi(B,b)
b = b + p
Ažuriraj(B,b)
                    b = b + p
Pomni
                    Ažuriraj(B,b)
                    Pomni
```

```
T1: Začetek
E-PoiščiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    E-PoiščiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
E-PoiščiPreberi(B,b)
                    Ažuriraj(Y,e)
                    E-PoiščiPreberi(B,b)
                    (čaka na odobritev)
b = b + p
Ažuriraj(B,b)
Pomni
                    (zaseženje odobreno)
                    b = b + p
                    Ažuriraj(B,b)
                    Pomni
```

Primer (5) s protokolom PXC

```
T1: Začetek
PoiščiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    PoiščiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
PoiščiPreberi(B,b)
                    Ažuriraj(Y,e)
b = b + p
Ažuriraj(B,b)
                    PoiščiPreberi(B,b)
                    b = b + p
                    Ažuriraj(B,b)
Pozabi
                    Pomni
```

```
T1: Začetek
E-PoiščiPreberi(X,t)
p = t*0.01
e = (t-p)/239.64
                    T2: Začetek
                    E-PoiščiPreberi(Y,t)
                    p = t*0.01
                    e = (t-p)/239.64
Ažuriraj(X,e)
E-PoiščiPreberi(B,b)
                    Ažuriraj(Y,e)
b = b + p
Ažuriraj(B,b)
                    E-PoiščiPreberi(B,b)
                    (čaka na odobritev)
Pozabi
                    (zasežene odobreno)
                    b = b + p
                    Ažuriraj(B,b)
                    Pomni
```

Pojav mrtve zanke (deadlock)

```
T1: Začetek
E-PoiščiPreberi(X,x)
                        T2: Začetek
                        E-PoiščiPreberi(Y,y)
E-PoiščiPreberi(Y,y)
(čaka na odobritev)
                        E-PoiščiPreberi(X,x)
                        (čaka na odobritev)
```

Mrtva zanka!

- Mrtva zanka: transakciji v neskončnost čakata druga na drugo
- Možne rešitve:
  1. Urnik izvajanja transakcij
  2. Vnaprejšnje zaseganje podatkov, ki jih bo transakcija ažurirala
  3. Ureditev objektov zaseganja
  4. Odločitev transakcijskega programa
  5. Dodatni protokoli za prekinjanje transakcij

Protokoli za preprečevanje mrtve zanke v povezavi s PXC ali PSC

- Situacija: transakcija T_A zahteva zaseženje podatka, ki ga je pred tem zasedla že T_B
  - Protokol **Čakaj ali izdihni**:
    če je transakcija T_A starejša od T_B, preide T_A v stanje čakanja na odobritev, če je mlajša, pa se izvajanje T_A prekine, transakcija T_A se razveljavi in posreduje transakcijskemu programu v ponovno izvajanje;
  - Protokol **Rani ali čakaj**:
    če je transakcija T_A starejša od T_B, se prekine, razveljavi in vrne v ponovno izvajanje transakcija T_B (po njeni razveljavitvi se transakciji T_A odobri zaseženje podatka), če je T_A mlajša, pa T_A preide v stanje čakanja.
- Starost transakcije je čas od njenega prvega začetka (sicer bi bila prekinjena transakcija vedno najmlajša).

Mrtva zanka (rani ali čakaj)

```
T1: Začetek
E-PoišČiPreberi(X,x)
                        T2: Začetek
                        E-PoišČiPreberi(Y,y)
E-PoišČiPreberi(Y,y)
(čaka na odobritev)
                        E-PoišČiPreberi(X,x)
                        (čaka na odobritev)
```

Transakcija T_A zahteva zaseženje podatka, ki ga je pred tem zasegla že T_B

Protokol **Rani ali čakaj**: če je transakcija T_A starejša od T_B, se prekine, razveljavi in vrne v ponovno izvajanje transakcija T_B (po njeni razveljavitvi se transakciji T_A odobri zaseženje podatka), če je T_A mlajša, pa T_A preide v stanje čakanja.

```
T1: Začetek
E-PoišČiPreberi(X,x)
                        T2: Začetek
                        E-PoišČiPreberi(Y,y)
E-PoišČiPreberi(Y,y)
(T1 je starejša od
 T2, zato prekine
 transakcijo T2)
```

Pomni

```
                        (ponovno izvajanje)
                        T2: Začetek
                        E-PoišČiPreberi(Y,y)
                        E-PoišČiPreberi(X,x)

                        Pomni
```

$T_1 = T_A$, $T_2 = T_B$

Mrtva zanka (čakaj ali izdihni)

```
T1: Začetek
E-PoišČiPreberi(X,x)
                        T2: Začetek
                        E-PoišČiPreberi(Y,y)
E-PoišČiPreberi(Y,y)
(čaka na odobritev)
                        E-PoišČiPreberi(X,x)
                        (čaka na odobritev)
```

Transakcija T_A zahteva zaseženje podatka, ki ga je pred tem zasegla že T_B

Protokol **Čakaj ali izdihni**: če je transakcija T_A starejša od T_B, preide T_A v stanje čakanja na odobritev, če je mlajša, pa se izvajanje T_A prekine, transakcija T_A se razveljavi in posreduje transakcijskemu programu v ponovno izvajanje;

```
T1: Začetek
E-PoišČiPreberi(X,x)
                        T2: Začetek
                        E-PoišČiPreberi(Y,y)
E-PoišČiPreberi(Y,y)
(T1 je starejša od
 T2, zato čaka na
 odobritev)                                    (a)
                        E-PoišČiPreberi(X,x)
                        (prekinitev in ponovno
                        izvajanje)               (b)
(zaseženje odobreno)
Pomni
                        T2: Začetek
                        E-PoišČiPreberi(Y,y)
                        E-PoišČiPreberi(X,x)
                        Pomni
```

(a) $T_1 = T_A$, $T_2 = T_B$

(b) $T_1 = T_B$, $T_2 = T_A$

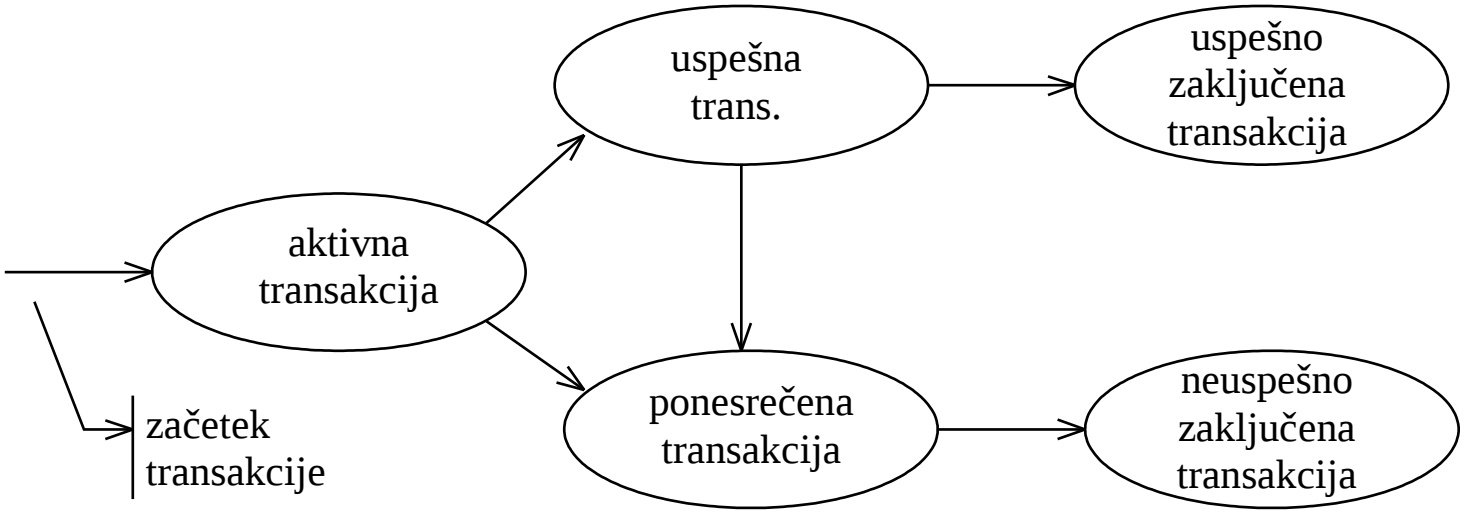
Obnavljanje podatkovne baze

- Skrb za varnost v PB
  - Uporabniški pogled (dostopna dovoljenja)
  - Zagotavljanje konsistentnosti in integritete podatkov
- Konsistentnost in integriteto podatkov ogrožajo podatkovne nesreče
- Mehanizmi za zagotavljanje integritete

Transakcija

- Osnovni element za zagotavljanje integritete podatkov
- Transakcija je program (zaporedje običajnih programskih konstruktov in specializiranih transakcijskih ukazov, ki skrbijo za dostop do baze), ki se mora izvršiti v celoti, da je integriteta PB zagotovljena

Življenjski cikel transakcije



- Na uspešnost oz. neuspešnost transakcij vplivajo podatkovne nesreče

Podatkovne nesreče

- transakcijske nesreče
  - odkrijejo uporabniški programi sami: nepravilni vhodni podatki
  - odkrijeta SUPB ali pa operacijski sistem: konflikt pri dostopu do podatkov, deljenje z 0
- sistemske nesreče
  - okvara na strežniku (CPU, pomnilnik), prekinitev napajanja
- diskovne nesreče
  - izguba ali uničenje podatkov na disku

Odpravljanje posledic pod. nesreč

- Po pričetku ponovnega delovanja računalniškega sistema je potrebno obnoviti podatkovno bazo v zadnje veljavno stanje pred nesrečo in ponoviti izvajanje prekinjenih transakcij
- Kako to dosežemo?

Mehanizmi za obnavljanje PB

- Redundatni podatki se generirajo občasno ali pa sočasno z izvajanjem operacij v okviru posameznih transakcij.
  - kopija podatkovne baze
  - vhodni podatki, na osnovi katerih se izvajajo transakcije
  - vrednosti podatkov v podatkovni bazi pred njihovim ažuriranjem (stare vrednosti)
  - vrednosti podatkov v podatkovni bazi po njihovem ažuriranju (nove vrednosti)
- Obnavljanje temelji na vseh naštetih redundantnih podatkih ali pa le na nekaterih izmed njih

Podatki za obnavljanje PB

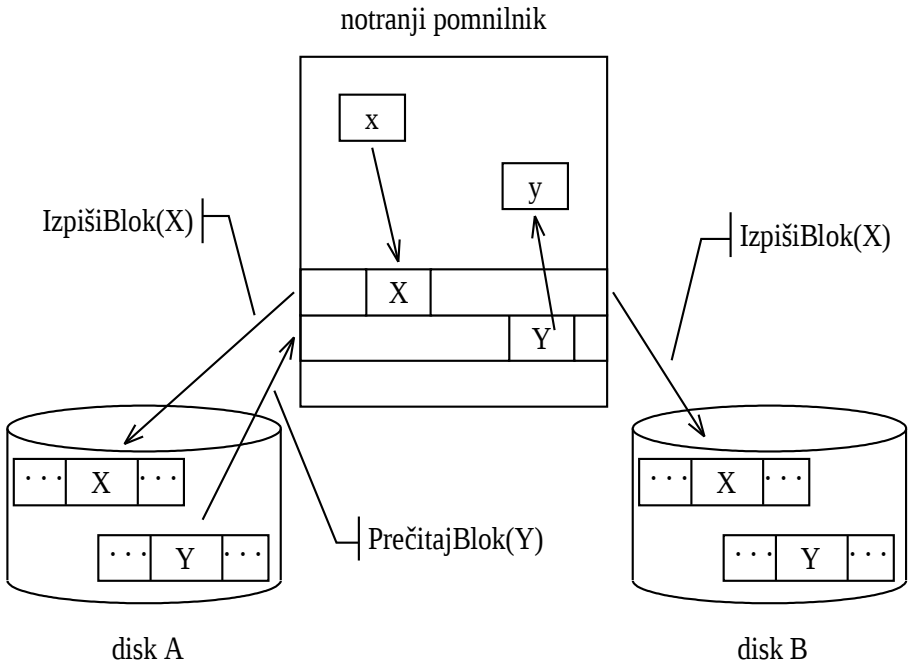
- Periodično **generiranje kopije** podatkovne baze omogoča njeno obnovitev v veljavno stanje ob času kopiranja
- Vhodni transakcijski podatki omogočajo **ponovno izvajanje transakcij**, ki so bile zaradi podatkovne nesreče prekinjene.
- Beleženje **starih** vrednosti podatkov (pred ažuriranjem) omogoča **razveljavitev ažuriranj** neuspešno zaključenih transakcij in vrnitev podatkovne baze v veljavno stanje pred pričetkom izvajanja posamezne transakcije.
- Beleženje **novih** vrednosti podatkov (po ažuriranju) omogoča **ponovitev ažuriranj** uspešno izvedenih transakcij, ki so se izgubila v podatkovni nesreči brez ponovnega izvajanja transakcijskih programov.

Načini obnavljanja PB

- Več različnih načinov, ki temeljijo na podmnožicah redundantnih podatkov
  - Dvojna podatkovna baza
  - Obnavljanje s senčnimi stranmi
  - Obnavljanje z dnevnikom in kopijo
- Razlike med njimi glede na
  - hitrost obnovitve podatkovne baze
  - pogostnost in količino beleženja redundantnih podatkov
  - poslabšanje odzivnih časov zaradi beleženja redundantnih podatkov

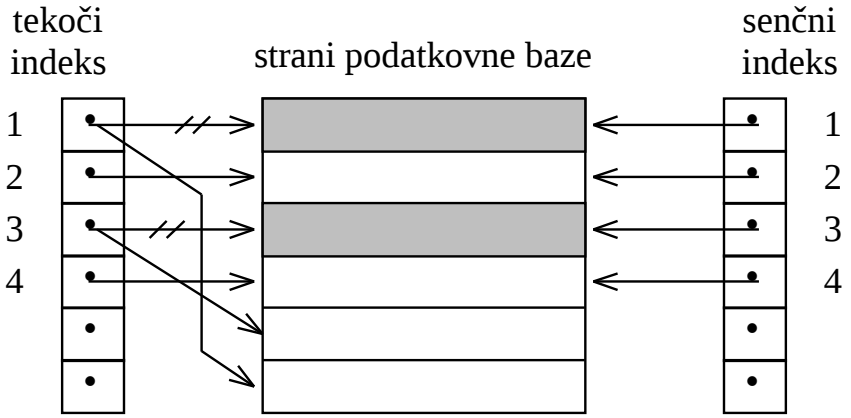
Dvojna podatkovna baza

- Podvojeni diski
- Podvojeni kontrolerji in diski
- Periodične kopije na zunanji pomnilniški medij
- Dobro sistemske in diskovne nesreče; potrebuje dodatne mehanizme za obravnavo transakcijskih nesreč



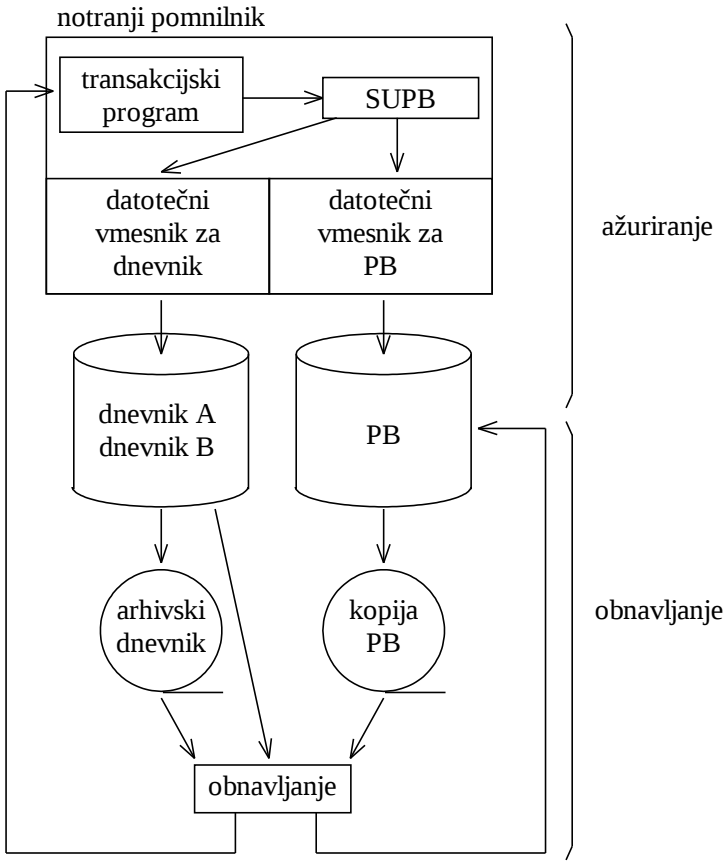
Obnavljanje s senčnimi stranmi

- *shadow paging*
- učinkovito predvsem pri transakcijskih podatkovnih nesrečah, ko je potrebno razveljaviti že izvedena ažuriranja podatkovne baze.
- Izpis ažurirane vsebine ne direktno v podatkovno bazo, pač pa na nezasedene sektorje na disku.
- Če se transakcija uspešno zaključi, se v podatkovno bazo vključijo ti novi bloki namesto starih - neažuriranih.



Obnavljanje z dnevnikom in kopijo

- Občasna izdelava kopije PB, s katero lahko obnovimo veljavno stanje
- Pisanje dnevnika, v katerega se zapisujejo podatki, s katerimi je možno s kopijo obnovljeno podatkovno bazo obnoviti v zadnje veljavno stanje tik pred podatkovno nesrečo.
- Dnevnik vsebuje tudi podatke, s katerimi je možno vse v trenutku nesreče prekinjene transakcije ponovno izvesti.
- Obstajata dve vrsti obnavljanja podatkovne baze z dnevnikom v odvisnosti vrste ažuriranja podatkovne baze
  - neposredno ažuriranje
  - odloženo ažuriranje



Neposredno ažuriranje

- Vsa ažuriranja se sproti izvajajo v podatkovni bazi, v dnevnik pa se beležijo tisti podatki, ki so potrebni pri morebitnem obnavljanju podatkovne baze.
  1. vsak zapis v dnevniku je opremljen z individualno oznako transakcije $\mathbf{T}_i$ in s časom generiranja zapisa **t**
  2. zapis ob začetku izvajanja transakcije vsebuje oznako **Začetek**, oznako transakcijskega programa $\mathbf{P}_j$, za vsak vhodni podatek par (ImePodatka, VrednostPodatka)
  3. zapis ob ukazu za ažuriranje, ki vsebuje oznako **StaraVrednost**, vrsto operacije **Dodaj**, **Izbriši**, **Spremeni,** ter par (NaslovPodatka, StaraVrednostPodatka) in
     zapis ob ukazu za ažuriranje, ki vsebujeoznako **NovaVrednost**, vrsto operacije **Dodaj**, **Izbriši**, **Spremeni,** ter par (NaslovPodatka, NovaVrednostPodatka)
  4. zapis ob prehodu transakcije v stanje Uspešna transakcija vsebuje oznako **Pomni**
  5. zapis ob prehodu transakcije v stanje Neuspešna transakcija vsebuje oznako **Pozabi**.

Odloženo ažuriranje

- Vsi podatki o ažuriranjih v okviru transakcije se najprej shranijo v dnevnik, ob uspešnem zaključku transakcije pa se izvede njihovo uveljavljanje
  1. vsak zapis v dnevniku je opremljen z individualno oznako transakcije $\mathbf{T}_i$ in s časom generiranja zapisa **t**
  2. zapis ob začetku izvajanja transakcije vsebuje oznako **Začetek**, oznako transakcijskega programa $\mathbf{P}_j$, za vsak vhodni podatek par (ImePodatka, VrednostPodatka)
  3. zapis ob ukazu za ažuriranje vsebujeoznako **NovaVrednost**, vrsto operacije **Dodaj**, **Izbriši**, **Spremeni,** ter par (NaslovPodatka, NovaVrednostPodatka)
  4. zapis ob prehodu transakcije v stanje Uspešna transakcija vsebuje oznako **Pomni**
  5. zapis ob prehodu transakcije v stanje Neuspešna transakcija vsebuje oznako **Pozabi**.

Primer vodenja dnevnika

- Primer dnevnika:

```
<T0, t0, Začetek, Odštej,(v1,1),(v2,4),(v3,10)>
<T0, t1, StaraVrednost, Spremeni, (P1, 20)>
<T0, t2, NovaVrednost,  Spremeni, (P1, 10)>
<T0, t3, StaraVrednost, Spremeni, (P2, 50)>
<T0, t4, NovaVrednost,  Spremeni, (P2, 40)>
<T0, t5, StaraVrednost, Spremeni, (P3, 70)>
<T0, t6, NovaVrednost,  Spremeni, (P3, 60)>
<T0, t7, StaraVrednost, Spremeni, (P4, 30)>
<T0, t8, NovaVrednost,  Spremeni, (P4, 20)>
<T0, t9, Pomni>
```

- Časi so lahko simbolični (kot v primeru) ali dejanski
- Obnavljanje z dnevnikom in kopijo
- Neposredno ažuriranje

- *Vhodni podatki:*
  v1 = 1, v2 = 4 in v3 = 10.
  Vrednosti podatkov v PB:
  P1 = 20, P2 = 50, P3 = 70,
  P4 = 30.

```
program Odštej(v1, v2, v3)


za i := v1 do v2 izvedi
   PoiščiPreberi(Pi,x)
   če podatek ne obstaja
   potem
      Pozabi
   sicer
      x := x - v3
      Spremeni(Pi,x)
      Pomni
```

Primer vodenja dnevnika

- Primer dnevnika:

```
<T0, t0, Začetek, Odštej,(v1,1),(v2,4),(v3,10)>
<T0, t1, StaraVrednost, Spremeni, (P1, 20)>
<T0, t2, NovaVrednost,  Spremeni, (P1, 10)>
<T0, t3, StaraVrednost, Spremeni, (P2, 50)>
<T0, t4, NovaVrednost,  Spremeni, (P2, 40)>
<T0, t5, StaraVrednost, Spremeni, (P3, 70)>
<T0, t6, NovaVrednost,  Spremeni, (P3, 60)>
<T0, t7, StaraVrednost, Spremeni, (P4, 30)>
<T0, t8, NovaVrednost,  Spremeni, (P4, 20)>
<T0, t9, Pomni>
```

- Časi so lahko simbolični (kot v primeru) ali dejanski
- Obnavljanje z dnevnikom in kopijo
- Odloženo ažuriranje

- *Vhodni podatki:*
  v1 = 1, v2 = 4 in v3 = 10.
  Vrednosti podatkov v PB:
  P1 = 20, P2 = 50, P3 = 70,
  P4 = 30.

```
program Odštej(v1, v2, v3)

za i := v1 do v2 izvedi
   PoiščiPreberi(Pi,x)
   če podatek ne obstaja
   potem
      Pozabi
   sicer
      x := x - v3
      Spremeni(Pi,x)
      Pomni
```

Primer vodenja dnevnika

Razpored transakcijskih ukazov dveh izvajanj programa Evro; neposredno ažuriranje

```
1  T1: Začetek
2  E-PoišČiPreberi(X,t)
3  p = t*0.01
4  e = (t-p)/239.64
5                          T2: Začetek
6                          E-PoišČiPreberi(Y,t)
7                          p = t*0.01
8                          e = (t-p)/239.64
9  Ažuriraj(X,e)
10 E-PoišČiPreberi(B,b)
11                         Ažuriraj(Y,e)
12 b = b + p
13 Ažuriraj(B,b)
14                         E-PoišČiPreberi(B,b)
                           (čaka na odobritev)
15 Pomni
16                         (zasežene odobreno)
                           b = b + p
17                         Ažuriraj(B,b)
18                         Pomni
```

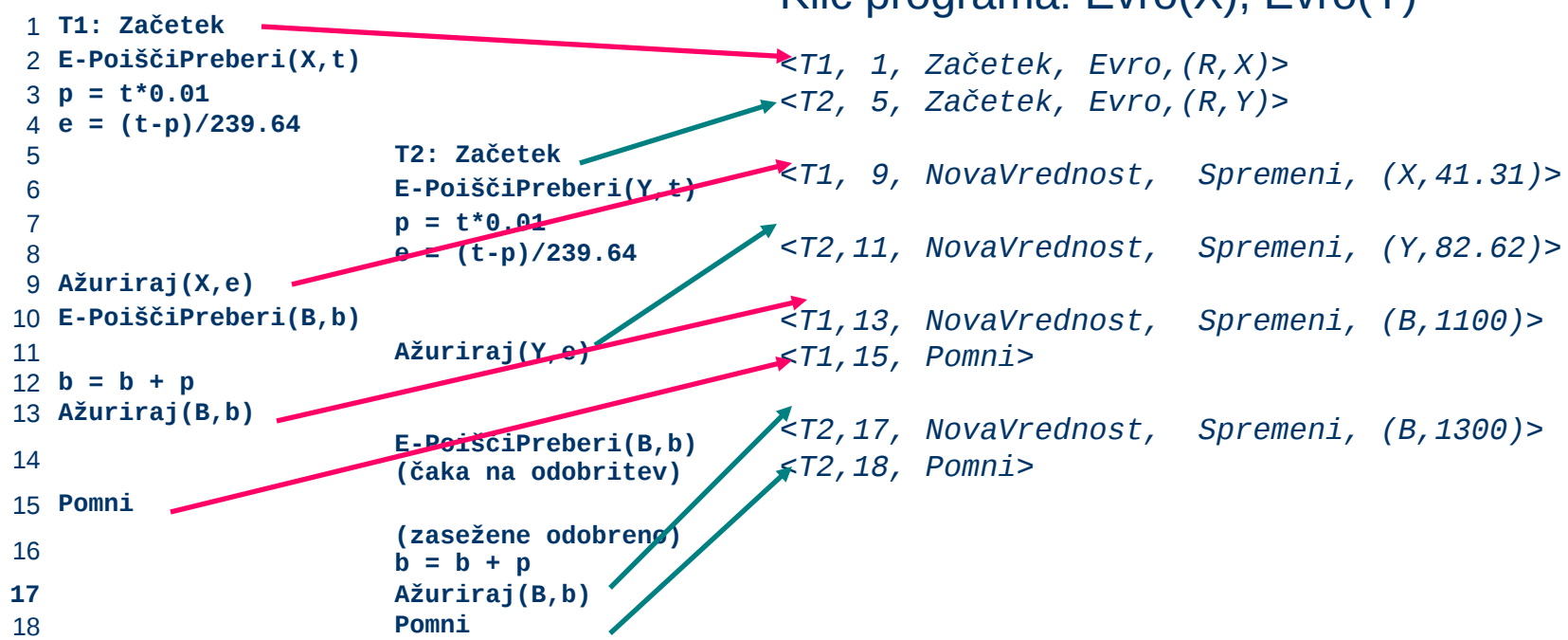
Začetne vrednosti v PB:
X=10.000, Y=20.000 in B=1000?
Klic programa: Evro(X), Evro(Y)

```
<T1, 1, Začetek, Evro,(R,X)>
<T2, 5, Začetek, Evro,(R,Y)>
<T1, 9, StaraVrednost, Spremeni, (X,10000)>
<T1, 9, NovaVrednost,  Spremeni, (X,41.31)>
<T2,11, StaraVrednost, Spremeni, (Y,20000)>
<T2,11, NovaVrednost,  Spremeni, (Y,82.62)>
<T1,13, StaraVrednost, Spremeni, (B,1000)>
<T1,13, NovaVrednost,  Spremeni, (B,1100)>
<T1,15, Pomni>
<T2,17, StaraVrednost, Spremeni, (B,1100)>
<T2,17, NovaVrednost,  Spremeni, (B,1300)>
<T2,18, Pomni>
```

Primer vodenja dnevnika

Razpored transakcijskih ukazov dveh izvajanj programa Evro; odloženo ažuriranje

Začetne vrednosti v PB:
X=10.000, Y=20.000 in B=1000?
Klic programa: Evro(X), Evro(Y)



Uporaba dnevnika za obnavljanje (neposredno in odloženo ažuriranje)

```
<T1, 1, Začetek, Evro,(R,X)>
<T2, 5, Začetek, Evro,(R,Y)>
<T1, 9, StaraVrednost, Spremeni, (X,10000)>
<T1, 9, NovaVrednost,  Spremeni, (X,41.31)>
<T2,11, StaraVrednost, Spremeni, (Y,20000)>
<T2,11, NovaVrednost,  Spremeni, (Y,82.62)>
<T1,13, StaraVrednost, Spremeni, (B,1000)>
<T1,13, NovaVrednost,  Spremeni, (B,1100)>
<T1,15, Pomni>
<T2,17, StaraVrednost, Spremeni, (B,1100)>
<T2,17, NovaVrednost,  Spremeni, (B,1300)>
<T2,18, Pomni>
```

t	X	Y	B
0	10000	20000	1000
9	41.31		
11		82.62	
13			1100
17			1300

- Kopija PB narajena v t=0
- Vrednosti v PB: X=10.000, Y=20.000 in B=1000
- Obnovimo do trenutka t=19:
- Od časa t=1do t=18 po dnevniku iščemo zapise z oznako NovaVrednost
- Nove vrednosti prepišemo na ustrezne naslove

Uporaba dnevnika za razveljavljanje (neposredno ažuriranje)

```
<T1, 1, Začetek, Evro,(R,X)>
<T2, 5, Začetek, Evro,(R,Y)>
<T1, 9, StaraVrednost, Spremeni, (X,10000)>
<T1, 9, NovaVrednost,  Spremeni, (X,41.31)>
<T2,11, StaraVrednost, Spremeni, (Y,20000)>
<T2,11, NovaVrednost,  Spremeni, (Y,82.62)>
<T1,13, StaraVrednost, Spremeni, (B,1000)>
<T1,13, NovaVrednost,  Spremeni, (B,1100)>
<T1,15, Pomni>
<T2,17, StaraVrednost, Spremeni, (B,1100)>
<T2,17, NovaVrednost,  Spremeni, (B,1300)>
<T2,18, Pomni>
```

t	X	Y	B
18	41.31	82.62	1300
17			1100
13			1000
11		20.000	
9	10.000		

- Razveljavimo transakciji T1 in T2
- Od zaključka do začetka vsake transakcije po dnevniku iščemo zapise z oznako StaraVrednost
- Stare vrednosti prepišemo na ustrezne naslove

Uporaba dnevnika za razveljavljanje (odloženo ažuriranje)

```
<T1, 1, Začetek, Evro,(R,X)>
<T2, 5, Začetek, Evro,(R,Y)>
<T1, 9, StaraVrednost, Spremeni, (X,10000)>
<T1, 9, NovaVrednost,  Spremeni, (X,41.31)>
<T2,11, StaraVrednost, Spremeni, (Y,20000)>
<T2,11, NovaVrednost,  Spremeni, (Y,82.62)>
<T1,13, StaraVrednost, Spremeni, (B,1000)>
<T1,13, NovaVrednost,  Spremeni, (B,1100)>
<T1,15, Pomni>
<T2,17, StaraVrednost, Spremeni, (B,1100)>
<T2,17, NovaVrednost,  Spremeni, (B,1300)>
<T2,18, Pomni>
```

- Razveljavimo transakciji T1 in T2
- Pri odloženem ažuriranju ni potrebno narediti ničesar, ker se spremembe ne uveljavijo v PB do ukaza Pomni!

V praksi: ARIES

- *Algorithms for Recovery and Isolation Exploiting Semantics*
- Najpopularnejša družina algoritmov za implementacijo obnavljanja z dnevnikom in kopijo
- IBM DB2, Microsoft SQL Server in mnogi drugi SUPB