



# **Podatkovne baze**

## **Visokošolski študij**

### **Bolonjski program**

Matjaž Kukar  
2009-2010



© Matjaž Kukar, 2010

# Urnik PB 2010 (predavanja in vaje)

	Pon	Tor	Sre	Cet	Pet
7					
8			1, 6, 7		
9			PR-JB		
10					
11		A-Ž			
12					
13	2, 9	5, 10, 4			
14	LRI-JC	PR-JB			
15	3, 8				
16	LRI-JC				
17				11, 12	
18				LRI-JD	
19					
20					

	Vaje (redni študenti)
	Vaje (izredni študenti)
	Predavanja

- 1: A-B
- 2: C-F
- 3: G-Ja
- 4: Je-Kor
- 5: Kos-Lj
- 6: Lo-M
- 7: N-Pop
- 8: Por-Sl
- 9: So-T
- 10: U-Ž
- 11: izredni
- 12: izredni

# Vsebina vaj

1. SUPB
2. 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
3. Indeksiranje na splošno
4. Nadzor nad sočasno uporabo PB
5. Obnavljanje podatkovne baze

# Izpitni red

1. Iz domačih nalog in seminarja morate doseči skupno najmanj 5 točk (pogoj za pristop k pisnemu izpitu).
2. Dosežene točke se upoštevajo pri rezultatu pisnega izpita, vendar samo v tem šolskem letu.
3. Pisni izpit morate za pozitivno oceno pisati najmanj 50%, k čemer se potem prištejejo točke vaj
4. V naslednjem šolskem letu še lahko opravljate izpit, potem pa morate ponovno opraviti vaje (točka 1).

Janez Novak je v šolskem letu 2009/10 pri vajah PB dosegel 8 točk.

Datum	Pisno	Rezultat
10. 6. 2010	45	45
16. 9. 2010	60	68
11. 6. 2011	70	70
17. 9. 2011	66	66
Ponovno opravljanje vaj		
12. 6. 2012	...	...

# Režim izvajanja vaj

- Govorilne ure
  - Četrtek, 10:30, kabinet na Jadranski 21
- Individualno delo (zelo priporočljivo)
  - Oracle (FRI, dobite navodila za priklop odjemalca)
  - MySQL (dobite navodila za inštalacijo)
  - Microsoft SQL Server (kdor želi)
  - Nikakor ne Microsoft Access !!!!

# **SUPB – sistem za upravljanje s podatkovnimi bazami**

- Splošno namenski skupek programske opreme, ki omogoča kreiranje, vzdrževanje in nadzor nad dostopom do podatkov v PB.
- SUPB mora omogočati
  - Upravljanje s podatki
  - Varen dostop do podatkov
  - Sočasen večuporabniški dostop
  - Skladnost (konsistentnost) podatkov
  - Obnavljanje podatkov

# SUPB

- SUPB kot tudi platforma za povpraševanje/programiranje v bazah
- specializirani povpraševalni/programski jeziki, najpopularnejši je SQL
- SQL je neproceduralen jezik (za razliko od npr. Java)
- SQL temelji na **relacijskem podatkovnem modelu** in **relacijski algebri**

# Orodja za dostop do SUPB

- Tronivojska arhitektura  
(odjemalec/aplikacijski strežnik/strežnik)
- Dvonivojska arhitektura (odjemalec/strežnik)
- Priporočljivo:
  - Oracle
  - MySQL
  - Microsoft SQL Server 2008



# Oracle

- Strežnik na FRI: Oracle 10g
  - todo.fri.uni-lj.si, TCP/IP port 1521
  - Uporabniško ime: pb, geslo: pbvaje, baza (SID): vaje
  - Uporabniško ime: pbb<vpisna številka>, geslo: <vpisna številka>, baza (SID): vaje  
npr: pbb12345678, 12345678
- Odjemalec: SQLDeveloper 2.1
  - Konfiguracija na učilnici

# MySQL

- MySQL (lastna inštalacija)
  - Strežnik MySQL 5.1
  - Odjemalec: MySQL Workbench 5.2 (beta)
- Konfiguracija na učilnici

# Microsoft SQL Server 2008

- Lastna inštalacija
- MSDN AA (zastonj)
- Developer verzija ekvivalentna Enterprise
- Možnost strežnika (Microsoft)
- Omejena podpora

# 1. domača naloga

- Opišite, kako si predstavljate razliko pod pojmom "podatkovna baza" (PB) in "sistem za upravljanje s podatkovnimi bazami" (SUPB)
- Pobrskajte po Internetu in naštejte ter v enem odstavku kratko opišite najpomembnejše komercialne in odprtokodne SUPB (Oracle, Microsoft SQL Server, IBM DB2, MySQL, PostGres, ...)
- Izberite in inštalirajte si svoj SUPB in ustrezne programe za dostop (MySQL, Microsoft SQL Server)
- Pripravite in oddajte poročilo dolžine do eno stran

# Relacijski podatkovni model (RPM)

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

# Osnovni koncepti RPM

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

# Relacija

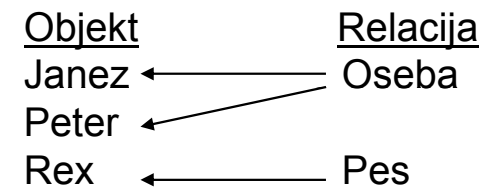
- Preslikava kartezičnega produkta vrednostnih množic

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

- Množica resničnih trditev:

je\_oseba={Janez, Peter}

je\_pes={Rex}



© Matjaž Kukar, 2010

# 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
Janez (v celoti)
Peter (v celoti)

Ime	Priimek	Naslov
Janez	Novak	Ljubljana
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 elementov drugih relacij (tabel) med seboj

zakonec(Janez, Micka)                      Janez <sup>zakonec</sup> ————— Micka)

# 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  $r$  pripada natanko ena relacijska shema, sestavljena iz oznake sheme  $R$  in oznakami imen in domen atributov

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

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

- 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  $r$ , 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  $\{X\} \rightarrow \{Y\}$  ali krajše  $X \rightarrow Y$
- Množico vseh funkcionalnih odvisnosti v shemi  $R$  označimo s  $F(R)$

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

# 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  $\sigma$ , projekcija  $\pi$ , stik  $\bowtie$ , deljenje  $\div$



# 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  $r \cup s$ :

G	H	I
a	b	c
d	a	f
c	b	d
b	g	a
<del>d</del>	<del>a</del>	<del>f</del>

Relacija  $r \cap s$ :

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
a	b	c	d	a	f
d	a	f	d	a	f
c	b	d	d	a	f

## Relacijski operatorji

- Projekcija  $\pi$ : zmanjševanje števila stolpcev
- Selekcija  $\sigma$ : 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 $\pi$

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

A	B
a	b
d	a
c	b

$$\pi_B(r)$$

B
b
a
<del>b</del>

Sintaksa:  $\pi_{A1,A2,\dots,Ak}$  - naštejemo attribute

Včasih se lahko zmanjša tudi število vrstic!

# Selekcija $\sigma$

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

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

$$r \mid \underset{\theta}{\times} \mid s = r \mid \underset{P}{\times} \mid s \equiv \sigma_P(r \times s)$$

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

## Pogojni stik (1. korak)

$$r \quad \left| \begin{array}{c} \times \\ \times \end{array} \right| \quad s =$$

$(B=D) \vee (C=c)$

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



## Pogojni stik (2. korak)

$$r \quad \left| \begin{array}{c} \times \\ \times \end{array} \right| \quad S =$$

$$(B=D) \vee (C=c)$$

A	B	C	D	E	F
a	b	c	b	g	a
<del>d</del>	<del>a</del>	<del>f</del>	<del>b</del>	<del>g</del>	<del>a</del>
c	b	d	b	g	a
a	b	c	d	a	f
<del>d</del>	<del>a</del>	<del>f</del>	<del>d</del>	<del>a</del>	<del>f</del>
<del>c</del>	<del>b</del>	<del>d</del>	<del>d</del>	<del>a</del>	<del>f</del>

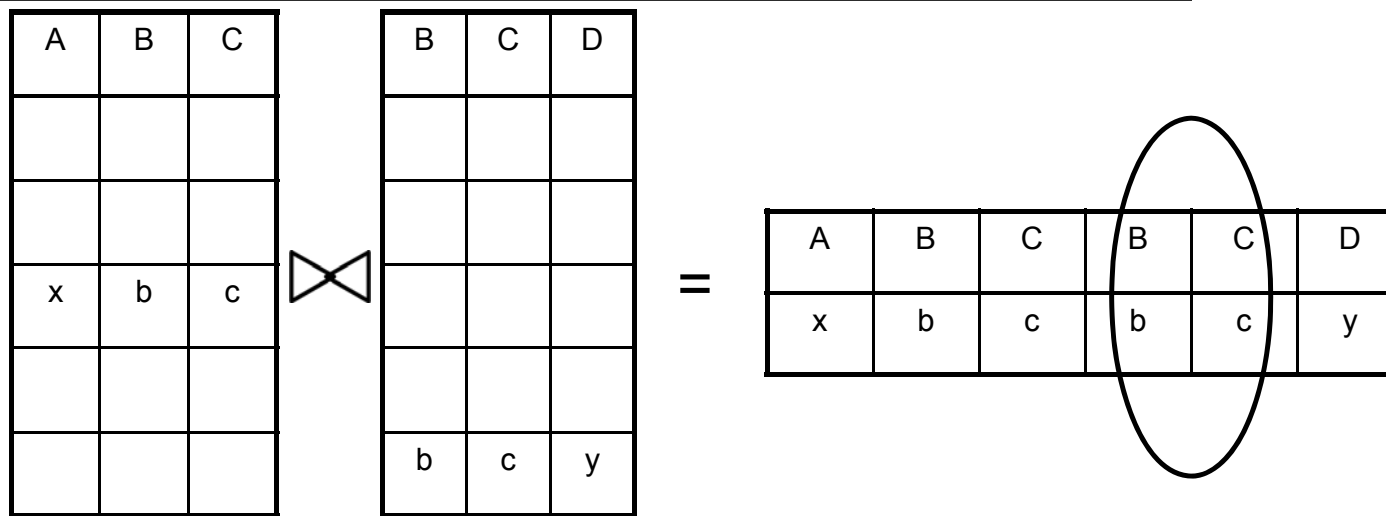
## Ekvistik in naravni stik

- 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

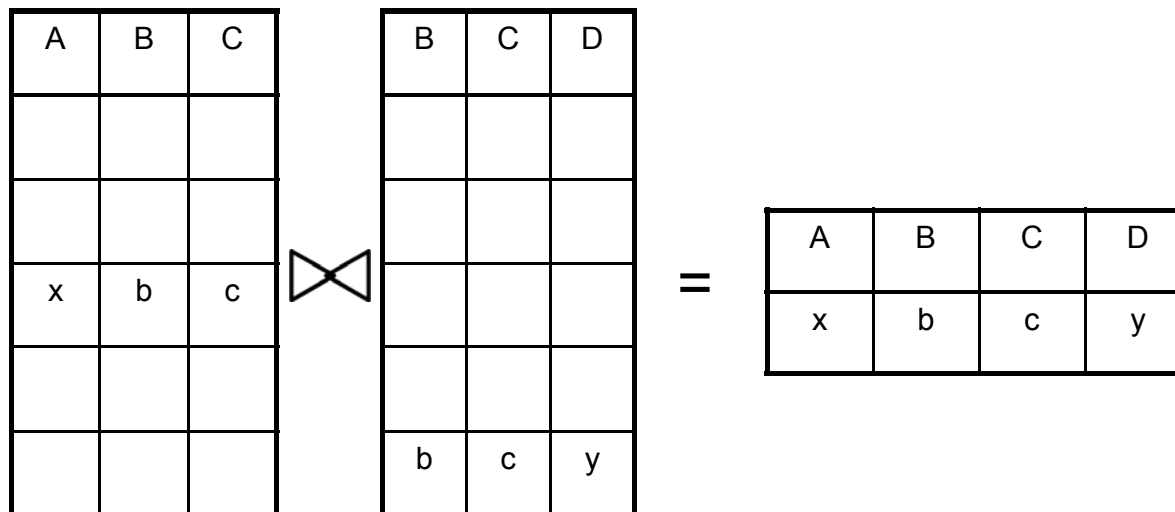
## Naravni stik (1. korak)

A	B	C		B	C	D		A	B	C	B	C	D
x	b	c	$\otimes$				=						
								x	b	c	b	c	y
				b	c	y							

## Naravni stik (2. korak)



## Naravni stik (3. korak)



## 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

## Relacijska algebra - ponovitev

- Tradicionalni operatorji za delo z množicami: unija  $\cup$ , presek  $\cap$ , razlika  $-$ , kartezični produkt  $\times$
- Posebni relacijski operatorji: selekcija  $\sigma$ , projekcija  $\pi$ , stik  $\bowtie$ , deljenje  $\div$

# 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:

<u>jid</u>	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

Coln:

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

Rezervacija:

<u>jid</u>	<u>cid</u>	<u>dan</u>
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

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, starejših od 50 let:
- Poišči barve vseh čolnov krajših od 40 čevljev

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

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

# Stik

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

- Poišči vse pare ime jadrancev in čolnov, kjer je jadranec rezerviral ustrezen čoln

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

$$\pi_{\substack{\text{jadranec.ime,} \\ \text{coln.ime}}}(\text{jadranec} \mid \times \mid \text{rezervacija} \mid \times \mid \text{coln})$$

$\substack{\text{jadranec.jid=} \\ \text{rezervacija.jid}} \qquad \substack{\text{rezervacija.cid=} \\ \text{coln.cid}}$

$$\pi_{\substack{\text{jadranec.ime,} \\ \text{coln.ime}}}(\text{jadranec} \mid \times \mid \text{rezervacija} \mid \times \mid \text{coln})$$

$\substack{\text{jid} \qquad \text{cid}}$

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

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

$\substack{\text{jid} \qquad \text{cid}}$

Namig: {vsi} – {tisti, ki so si že kaj izposodili}

## Naloga

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

- Napišite izraz v relacijski algebri, ki izpiše imena tistih jadralcev, ki si še nikoli niso izposodili nobenega čolnov.

$$\overbrace{\pi_{jid}(\text{jadralec})}^{\text{vsi}} - \overbrace{\pi_{jid}(\text{rezervacija})}^{\text{izposodili}}$$

$$\underbrace{\pi_{ime}(\text{jadralec} \mid \times \mid \pi_{jid}((\text{jadralec}) - \pi_{jid}(\text{rezervacija})))}_{\text{dobimo imena}}$$

# Structured Query Language - SQL

- Rezultat projektov v IBM (1974-77)
- Vsak proizvajalec ga po svoje razširja
- Standardi:
  - SQL-87 (1986, 1987): ANSI SQL
  - SQL-89 (1989): ANSI SQL, FIPS popravki
  - SQL-92 (1992): SQL2, ANSI/ISO SQL
  - SQL:1999 (1999): SQL3, objekti, rekurzija, dogodki, regularni izrazi
  - SQL:2003 (2003): podpora XML, avtomatsko generiranje polj, delo s sekvencami
  - SQL:2006, SQL:2008: dodatna podpora delu z XML, integracija XQuery, drugi manjši popravki
- V praksi: ni 100% podpore standardom

© Matjaž Kukar, 2010

# Structured Query Language - SQL

- SQL
  - Beginning SQL. Paul Wilton and John W. Colby. Wrox, 2005.
- SQL in relacijska algebra, teorija o PB
  - R. Ramakrishnan, J. Gehrke: Database Management Systems, 3. izdaja, McGraw-Hill, 2002

# 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

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

## Tabele za vaje

- Oracle na FRI jih že ima
- MySQL:
  - Z učilnice prenesite datoteko jadralci.sql
  - Odprite jo v MySQL Workbenchu (File->Open SQL Script...)
  - Poženite (samo prvič – torej samo enkrat!)

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

## Primeri tabel

Jadralec:

<u>jid</u>	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

Coln:

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

Rezervacija:

<u>jid</u>	<u>cid</u>	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

# Osnovni SELECT stavek

SELECT A1, A2, ..., Ak	$\pi_{A1,A2,...,Ak}$
FROM T1, T2, ..., Tn	$T1 \times T2 \times \dots \times Tn$
WHERE P;	$\sigma_P$
	$\pi_{A1,A2,...,Ak}(\sigma_P(T1 \times T2 \times \dots \times Tn))$

- 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, starejših od 50 let:

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

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

- Poišči barve vseh čolnov krajših od 40 čevljev

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

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

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

## Stik (1)

- Poišči vse pare imen jadrancev in čolnov, kjer je jadranec rezerviral ustrezen čoln

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

$\pi_{\substack{\text{jadranec.ime,} \\ \text{coln.ime}}}(\text{jadranec} \mid \times_{\text{jid}} \mid \text{rezervacija} \mid \times_{\text{cid}} \mid \text{coln})$

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

© Matjaž Kukar, 2010

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

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

```
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



# SQL

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

- Poišči vse pare ime jadrancev 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

© Matjaž Kukar, 2010

# SQL

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

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

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

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

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

```
SELECT jadralec.ime      -- Ekvistik  
FROM jadralec JOIN rezervacija USING(jid)  
WHERE rezervacija.cid = 103;
```

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

```
SELECT jadralec.ime      -- Pogojni stik s sestavljenim pogojem  
FROM jadralec JOIN rezervacija ON (rezervacija.jid = jadralec.jid AND  
                                   rezervacija.cid = 103);
```

**Obstaja še več  
različic, niso pa  
vedno vse  
implementirane,  
zato pozor!**

© Matjaž Kukar, 2010

# 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
```

# SQL

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

- Poišči imena jadralcev, ki so rezervirali rdeč čoln.

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

```
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';
```

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

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

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

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

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

## Vaje naslednji teden

- V ponedeljek, 5. aprila vaje odpadejo (velikonočni ponedeljek)
- Nadomeščanje:  
torek, 6. april 15-17h PR-JB ali izjemoma ostali termini
- Rok oddaje domače naloge podaljšan do 2. 4.
- Naslednji teden: domača naloga SQL

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.  
 $\rho$ : operator preimenovanja

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

```
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.  
 $\rho$ : operator preimenovanja

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

```
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.  
 $\rho$ : operator preimenovanja

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

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?

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

© Matjaž Kukar, 2010

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.  
 $\rho$ : operator preimenovanja

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

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

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

© Matjaž Kukar, 2010

# Operatorji v SQL (WHERE vrstica)

- =
- LIKE: približna primerjava nizov znakov
- SIMILAR TO vzorec [ESCAPE znak]:  
regularni izrazi (SQL:1999)
- != ali <>
- <=, >=, <, >
- BETWEEN x AND y:  $x \leq \text{vrednost} \leq y$
- AND, OR, NOT
- IS [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|
+-----+
|      21 |
|      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

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

## SQL: regularni izrazi

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

```
SELECT j.starost
FROM jadralec j
WHERE j.ime REGEXP '^b.....*$';           -- MySQL
-- ali (REGEXP = RLIKE)
WHERE j.ime RLIKE '^b[a-z]{4}[a-z]*$';      -- MySQL
-- ali
WHERE REGEXP_LIKE (j.ime, '^B[a-z]{4}[a-z]*$'); -- Oracle
```

- ^ označuje začetek, \$ pa konec niza (sicer se išče poljuben podniz)
- . (pika) ustreza natanko eni poljubnemu znaku
- [a-z] je katera koli črka med a in z
- \* pomeni 0 ali več ponovitev predhodnega znaka
- SQL:1999: operator SIMILAR TO z regularnimi izrazi (redko implementirano)

© Matjaž Kukar, 2010

# SQL

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

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

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

```
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.

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

```
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 c.barva='zelen');
```

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

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



# Operatorji za delo z množicami

- UNION: unija  $\cup$   
UNION ALL: unija  $\cup$  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): (ne)praznost množice
- UNIQUE (tabela): enoličnost elementov v tabeli
- Operatorji IN, NOT IN in EXISTS so 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.

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

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

```
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.

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

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

```
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.

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

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

```
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='zeleni';
```

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

© Matjaž Kukar, 2010

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.

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

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

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

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

© Matjaž Kukar, 2010

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
+-----+

© Matjaž Kukar, 2010

## 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) zaokroži 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) zaokroži 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 jadrancev 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 številka 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

- Preveri veljavnost kvantificiranega ( $\exists, \forall$ ) logičnega pogoja nad celotno množico skalarnih (posameznih) vrednosti atributa
- Kvantifikatorja:
  - 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 $x < \text{ANY}(\text{SELECT } y \dots);$	$\exists y : x < y$	
WHERE $x = \text{ANY}(\text{SELECT } y \dots);$	$x \in \{y \mid \dots\}$	Isto kot IN
WHERE $x <> \text{ANY}(\text{SELECT } y \dots);$	$\exists y : x \neq y$	
WHERE $x < \text{ALL}(\text{SELECT } y \dots);$	$\forall y : x < y$	
WHERE $x = \text{ALL}(\text{SELECT } y \dots);$	$\forall y : x = y$	
WHERE $x <> \text{ALL}(\text{SELECT } y \dots);$	$\forall y : x \neq y$	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.

```
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)

## 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.

```
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.

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

© Matjaž Kukar, 2010

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

## Razlika množic z operatorjem

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

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

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

```
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='zeleni';
```

```
+-----+
|  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.

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

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

```
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
  (SELECT c.cid
   FROM coln c
   MINUS
   SELECT r.cid
   FROM rezervacija r
   WHERE r.jid = j.jid);
```

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

-- Vsi - Rezervirani = prazna mnozica

-- Vsi colni

-- Rezervirani colni

-- za vsakega jadralca

-- posebej (korelirana)

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

# 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(DISTINCT ime) |
+-----+
|                      9 |
+-----+
```

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

## Povprečje (AVG)

- Izračunaj povprečno starost jadralcev!

```
SELECT AVG(starost)      -- povpreči 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)
FROM jadralec
WHERE rating = 10;
```

```
-- povpreči vrednosti atributa
-- starost v tabeli jadralcev
-- vendar le za tiste z ratingom 10
```

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

© Matjaž Kukar, 2010



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

## Minimum (MIN) in maksimum (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, WHERE ali ORDER BY 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 attribute, 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!

© Matjaž Kukar, 2010

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 jadrancev 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 jadrancev, starejših od najstarejšega jadrarca z ratingom 10!

```
SELECT ime
FROM jadrlec
WHERE starost > (SELECT MAX(starost)
                 FROM jadrlec
                 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

[DISTINCT | ALL]

SELECT 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 Ai, ..., Aj -- urejanje po atributih

[ASC|DESC]

© Matjaž Kukar, 2010

## Razširjeni SELECT stavek

- GROUP BY  $x$ : razdeli množico iz SELECT-FROM-WHERE na podmnožice glede na enake vrednosti atributa  $x$
- GROUP BY  $x_1, x_2, \dots, x_n$ : skupine imajo enake vrednosti vseh  $n$  atributov (torej je število možnih – vendar ne nujno dejanskih – skupin enako moči kartezičnega produkta vseh  $n$  atributov)
- Vsak osnovni 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 pogoj  $P$
- 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 attribute?
- 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 jadranci

## 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 attribute moramo poimenovati
```

rating	Povp_star
3	44.5
7	40
8	40.5
10	25.5

© Matjaž Kukar, 2010

# 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 ));
```

© Matjaž Kukar, 2010

# 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);`

© Matjaž Kukar, 2010

# 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;

# Pogledi (views) v SQL

- Pogled (VIEW) je tabela, katere 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

© Matjaž Kukar, 2010

## 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 attribute

## 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

© Matjaž Kukar, 2010

## 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

© Matjaž Kukar, 2010

## 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;
```

©Matjaž Kukar, 2010

## Primer indeksiranja

- Indeksiraj čolne po barvi!  
`CREATE INDEX po_barvi  
ON coln(barva);`
- Indeksiraj rezervacije ločeno po datumih ter šifrah jadrancev 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).

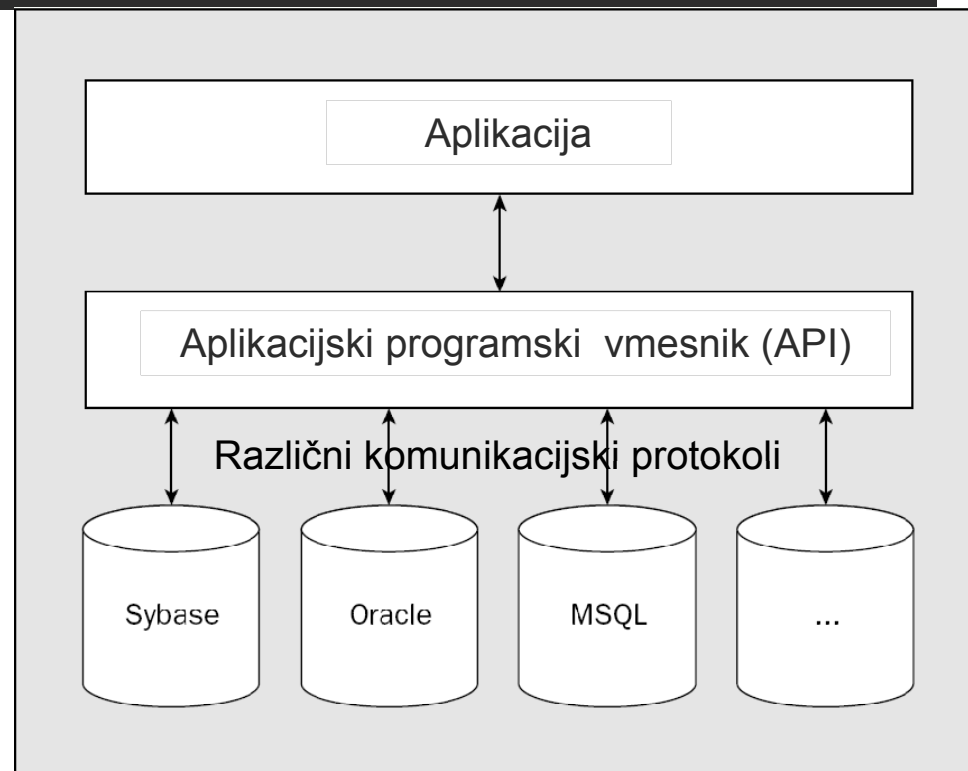
```
SELECT *  
FROM jadralec  
USE INDEX(jad_index1)           -- uporabi indeks  
ORDER BY ime, jid;             -- po jid in imenu
```

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

© Matjaz Koncan, 2010

## Komunikacija med aplikacijo in SUPB

Podatkovna  
baza





## Nastanek standardnih programskih vmesnikov

- Različni proizvajalci podatkovnih baz uporabljajo različne protokole in programske vmesnike (API)
- Težavno programiranje aplikacij
- Leta 1992 se pojavi vmesnik ODBC (open data base connectivity), ki skuša poenotiti programski dostop
- Aplikacije prenosljive na različne platforme, vendar je njihova funkcionalnost in učinkovitost nekoliko okrnjena v primerjavi z uporabo originalnih programskih vmesnikov

## ODBC - open data base connectivity

- Nastal je leta 1992 v sodelovanju Microsofta s podjetjem Simba Technologies
- Sloni na različnih standardnih Call Level Interface (CLI) specifikacijah iz SQL Access Group, X/Open in ISO/IEC
- Leta 1995 je ODBC 3.0 postal del standarda ISO/IEC 9075-3 -- Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI).

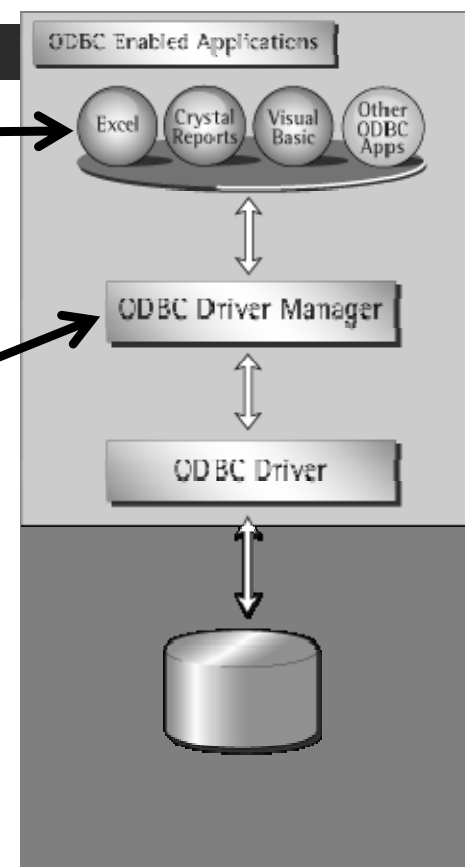
## Kaj nam ODBC ponuja

- Gonilnike, ki omogočajo poenoten dostop do PB
- Knjižnico funkcij, ki omogoča aplikaciji povezavo s SUPB, izvajanje SQL stavkov in dostop do rezultatov in statusa izvajanja
- Standarden način za prijavo in odjavo na SUPB
- Standardno (a omejeno) predstavitev podatkovnih tipov
- Standarden nabor sporočil o napakah
- Podporo SQL sintaksi po X/Open in ISO CLI specifikacijah

- 164 -

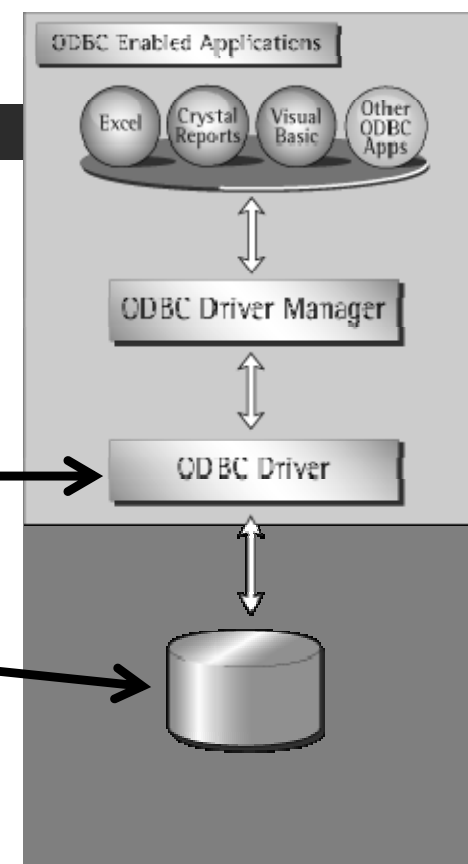
# Arhitektura ODBC

- Aplikacije:
  - procesiranje podatkov,
  - klici ODBC funkcij za posredovanje poizvedb in rezultatov
- ODBC upravljalec gonilnikov:
  - Nalaga gonilnike glede na potrebe aplikacij
  - Procesira klice ODBC funkcij in jih posreduje gonilniku



# Arhitektura ODBC

- ODBC gonilnik:
  - Prevzema klice ODBC funkcij, jih po potrebi preoblikuje in posreduje SUPB
  - Omogoča manjkajočo funkcionalnost glede na implementiran ODBC standard
- Podatkovni vir:
  - SUPB
  - tekstovne datoteke
  - preglednice
  - ...



# ODBC in standardni SQL

- Minimalni SQL
  - Data Definition Language (DDL): CREATE TABLE in DROP TABLE
  - Data Manipulation Language (DML): enostavni SELECT, INSERT, UPDATE, SEARCHED, in DELETE z iskalnim pogojem
  - Preprosti izrazi: (npr. as  $A > B + C$ )
  - Samo znakovni podatkovni tipi: CHAR, VARCHAR, LONG VARCHAR

# ODBC in standardni SQL

- Standardni SQL
  - Vsebuje minimalni SQL
  - Data Definition Language (DDL): ALTER TABLE, CREATE INDEX, DROP INDEX, CREATE VIEW, DROP VIEW, GRANT, in REVOKE
  - Data Manipulation Language (DML): polni SELECT stavek
  - Izrazi: gnezdene poizvedbe, skupinski operantorji (npr. SUM, MIN, ...)
  - Podatkovni tipo: DECIMAL, NUMERIC, SMALLINT, INTEGER, REAL, FLOAT, DOUBLE PRECISION

# ODBC in standardni SQL

- Razširjeni SQL
  - Minimalni in osnovni SQL
  - Data Manipulation Language (DML): zunanji stiki, pozicijski UPDATE, pozicijski DELETE, SELECT FOR UPDATE, unije
  - Izrazi: skalarne funkcije (npr.SUBSTRING, ABS), določila za deklaracijo konstant DATE, TIME in TIMESTAMP
  - Podatkovni tipi: BIT, TINYINT, BIGINT, BINARY, VARBINARY, LONG VARBINARY, DATE, TIME, TIMESTAMP
  - Paketi SQL stavkov
  - Podpora shranjenim proceduram (klicanje)



## pyodbc – implementacija ODBC za Python

- pyodbc je modul za Python ki omogoča dostop do poljubnega SUPB (ki podpora ODBC)
- implementira Python Database API Specification v2.0 z dodatki, ki poenostavljajo delo s podatkovno bazo
- pyodbc je odprtokoden, uporablja MIT licenco, in ga lahko zastonj uporabljamo tako v pridobitne, kot nepridobitne namene (vključno z izvorno kodo)
- domača stran in dokumentacija:  
<http://code.google.com/p/pyodbc/>

Poglejte si!!!

# Predpriprava na uporabo pyodbc

- SUPB s podatki
- Python (2.6) in pyodbc (za Python 2.6)
- Delovno okolje: Pythonwin ali Idle
- ODBC gonilnik za izbrani OS in SUPB
  - MySQL: Connector/ODBC (na učilnici: instalirajte)
- Oracle:
  - Microsoft ODBC for Oracle (z Windowsi): na Windows XP deluje, na Windows Vista ali Windows 7 pa običajno ne ali
  - Oracle Instant Client (na učilnici): odpakirajte in poženite odbc\_install.exe

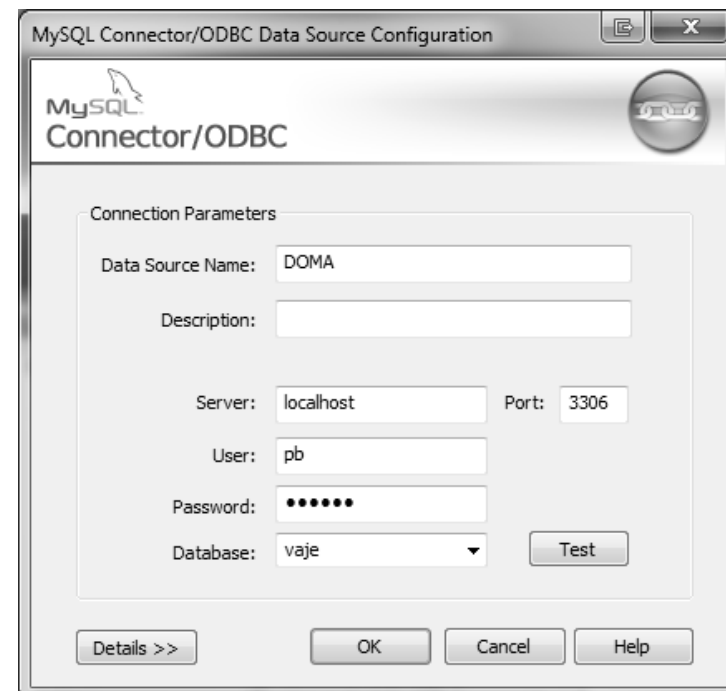
# Priprava podatkovnega vira (Oracle)

- Odprite Control Panel->Administrative tools  
->Data Sources (ODBC)
- V zavihku User DSN izberite Add in gonilnik:
  - Microsoft ODBC for Oracle ali
  - Oracle in instantclient\_11\_1
  - Vnesite vrednosti s slike: DSN je lahko poljuben.
  - User ID je lahko: ime/geslo

The screenshot shows the 'Oracle ODBC Driver Configuration' dialog box. It has a title bar with a standard Windows icon. The dialog is divided into two main sections. The top section contains four input fields: 'Data Source Name' with the value 'FRI', 'Description' (empty), 'TNS Service Name' with a dropdown menu showing 'todo.fri.uni-lj.si/vaje', and 'User ID' (empty). To the right of these fields are four buttons: 'OK', 'Cancel', 'Help', and 'Test Connection'. The bottom section is titled 'Application' and has three tabs: 'Oracle' (selected), 'Workarounds', and 'SQLServer Migration'. Under the 'Oracle' tab, there are several options: 'Enable Result Sets' (checked), 'Enable Query Timeout' (checked), 'Read-Only Connection' (unchecked), 'Enable Closing Cursors' (unchecked), 'Enable Thread Safety' (checked), 'Batch Autocommit Mode' (dropdown menu showing 'Commit only if all statements succeed'), and 'Numeric Settings' (dropdown menu showing 'Use Oracle NLS settings').

# Priprava podatkovnega vira (MySQL)

- Odprite Control Panel->Administrative tools  
->Data Sources (ODBC)
- V zavihku User DSN izberite Add in nato določite ODBC gonilnik:
  - MySQL ODBC 5.1 driver
  - Vnesite vrednosti s slike: DSN je lahko poljuben.
  - Lahko vnesete uporab. ime in geslo



# Osnovni gradniki pyodbc

- Uvoz modula:  
`import pyodbc`
- Najpomembnejši razredi:
  - Povezava (connection)
  - Kurzor (cursor)
  - Podatkovni tipi in njihovi konstruktorji
  - Obravnava napak

## pyodbc: povezava

- Povezavo c ustvarimo z ukazom:  
`c = pyodbc.connect(ConnectionString)`
- ConnectionString določa povezavo, npr.  
`ConnectionString = 'DSN=FRI;UID=pb;PWD=pbvaje'`  
ali  
`ConnectionString = 'DSN=DOMA;UID=pb;PWD=pbvaje'`
- ConnectionString bi lahko napisali tudi brez definiranega DSN:  
`ConnectionString = 'DRIVER={MySQL ODBC 5.1 driver};  
SERVER=localhost;DATABASE=vaje;UID=pb;PWD=pbvaje;  
CHARSET=UTF8'`

## pyodbc: povezava

- Nekatere metode:
  - close(): zapri povezavo, enako pri destruktorku objekta
  - commit(): uveljavi transakcijo (če SUPB podpira)
  - rollback(): razveljavi transakcijo (če SUPB podpira)
  - cursor(): vrne nov kurzorski objekt, ki uporablja povezavo
- Kurzorji: izvajajo SQL stavke in omogočajo iteracijo po vrsticah rezultata

## pyodbc: kurzor

- Kurzor x ustvarimo z ukazom:  
`x = c.cursor()` # c je povezava
- Nekateri atributi:
  - `description`: opis stolpcev rezultata (shema)
  - `rowcount`: število vrstic rezultata
- Nekatere metode:
  - `execute(ukaz, [parametri])`: izvede ukaz z opsijskimi parametri
  - `fetchall()`: prenese vse vrstice rezultata
  - `fetchone()`, `fetchmany(size)`: preneseta eno ali več vrstic



## pyodbc: kurzor

- Po kurzorju lahko iteriramo, vendar samo enkrat:  
x.execute(SQLukaz)  
for r in x: print r
- Več iteracij: v = x.fetchall(), nato iteriramo po v
- Parametri v SQL ukazih:
  - Primer: SELECT \* FROM jadralec
  - SQL ukaz kot niz znakov: Pythonov način parametrizacije
    - x.execute('SELECT %s FROM %s' % ('\*', 'jadralec'))
  - pyodbc prenos parametrov v metodi execute:
    - ? označuje parameter
    - seznam parametrov za ukazom
    - x.execute('SELECT ? FROM ?', '\*', 'jadralec')
    - x.execute('SELECT ? FROM ?', ('\*', 'jadralec'))

## pyodbc: obravnava napak

- Razredi pyodbc ob napakah javljajo naslednje izjeme:
  - DatabaseError
  - DataError
  - OperationalError
  - IntegrityError
  - InternalError
  - ProgrammingError
  - NotSupportedError

## pyodbc: obravnava napak

```
try:
    x.execute ( SQLLukaz)
    ...
except pyodbc.DataError:
    -- obravnava napake
    pass

...
except pyodbc.DatabaseError:
    -- obravnava napake
    pass

except:
    -- obravnava ostalih napak
    pass
```

## pyodbc: preslikava med ODBC/SQL in Pythonovimi podatkovnimi tipi

ODBC	Python
char varchar longvarchar GUID	string
wchar wvarchar wlongvarchar	unicode
smallint integer tinyint	int
bigint	long
decimal numeric	decimal
real float double	double
date	datetime.date
time	datetime.time
timestamp	datetime.datetime
bit	bool
binary varbinary longvarbinary	buffer
SQL Server XML type	unicode

## Primer programa

- Naloga: postaraj jadralce za eno leto
- Izvedba:
  - ustvari novo tabelo: postarani
  - v vsaki vrstici povečaj starost za 1
- Vse to lahko naredimo direktno v SQL-u. Kako?

# Primer programa

```
import pyodbc
cnxn = pyodbc.connect('DSN=FRI;UID=pb;PWD=pbvaje')
cursor = cnxn.cursor()
updater= cnxn.cursor()
try:
    cursor.execute("DROP TABLE postarani")
except pyodbc.DatabaseError:
    pass
cursor.execute("CREATE TABLE postarani AS SELECT * FROM jadralec")
cnxn.commit()
```

Izbriši, če že  
obstaja tabela.

# Primer programa

```
cursor.execute("SELECT * from postarani")
print "PRED"
for r in cursor:
    print r
cursor.execute("SELECT * from postarani")
for r in cursor:
    updater.execute("UPDATE postarani SET starost =? WHERE jid =?",
                    r.STAROST + 1, r.JID)
cursor.execute("SELECT * from postarani")
print "PO"
for r in cursor:
    print r
cnxn.commit()
```

Zakaj še en  
SELECT?

Pazite na  
velikost črk  
pri imenih  
atributov

COMMIT zares  
zapiše v bazo

# NALOGA

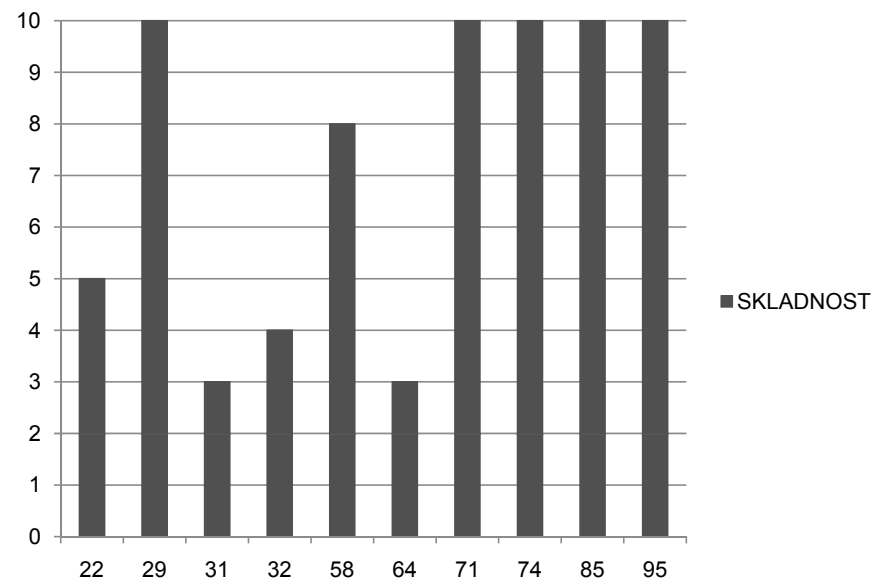
- V Pythonu napišite program, ki za vsakega jadralca izpiše šifro najbolj skladnega čolna.
- Rezultate zapisite v tabelo `optimal(jid, cid, skladnost)`
- Najbolj skladen čoln *cid* za jadralca *jid* je tisti, ki po formuli  $(jid + cid) \% 11$  daje najvišjo vrednost.



# NALOGA V EXCELU

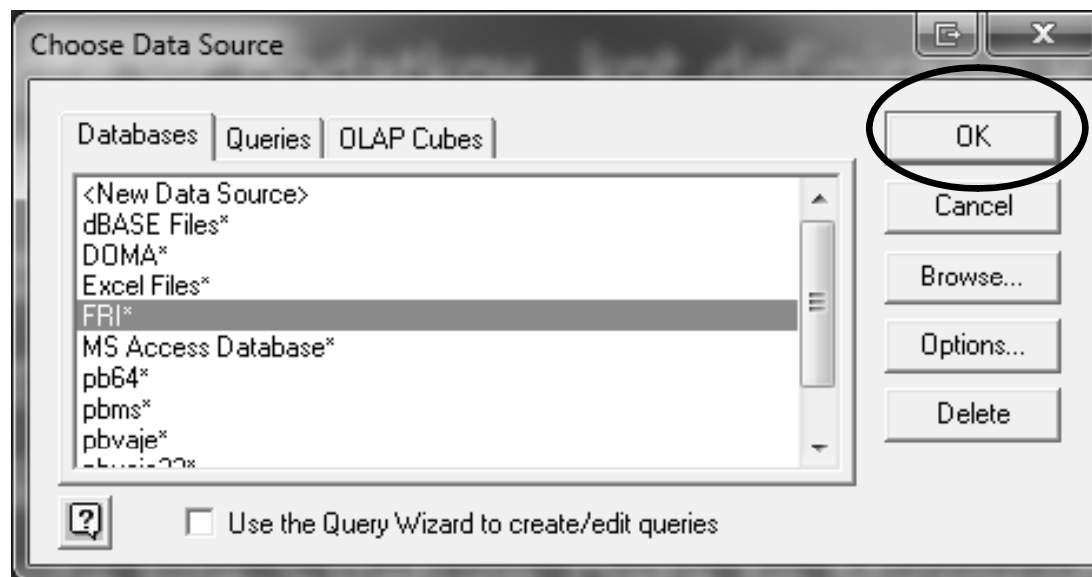
- Povežite se na bazo in prenesite tabelo optimal
- Narišite graf te tabele, kot ga kaže slika.

JID	CID	SKLADNOST
22	104	5
29	102	10
31	104	3
32	104	4
58	104	8
64	104	3
71	104	10
74	101	10
85	101	10
95	102	10



# Microsoft Excel in ODBC

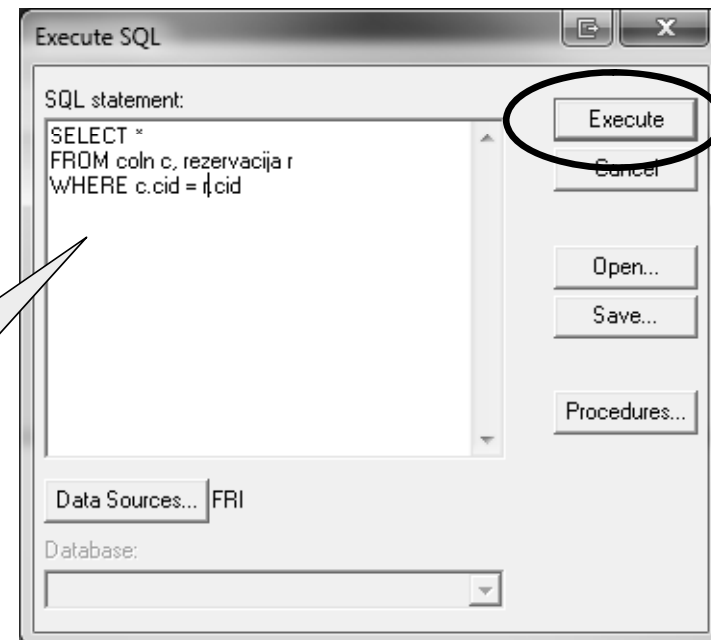
- Izberite  
Data->From Other Sources->From Microsoft Query
- Izberite vir podatkov, kot definirano v ODBC Data Sources (npr. FRI)



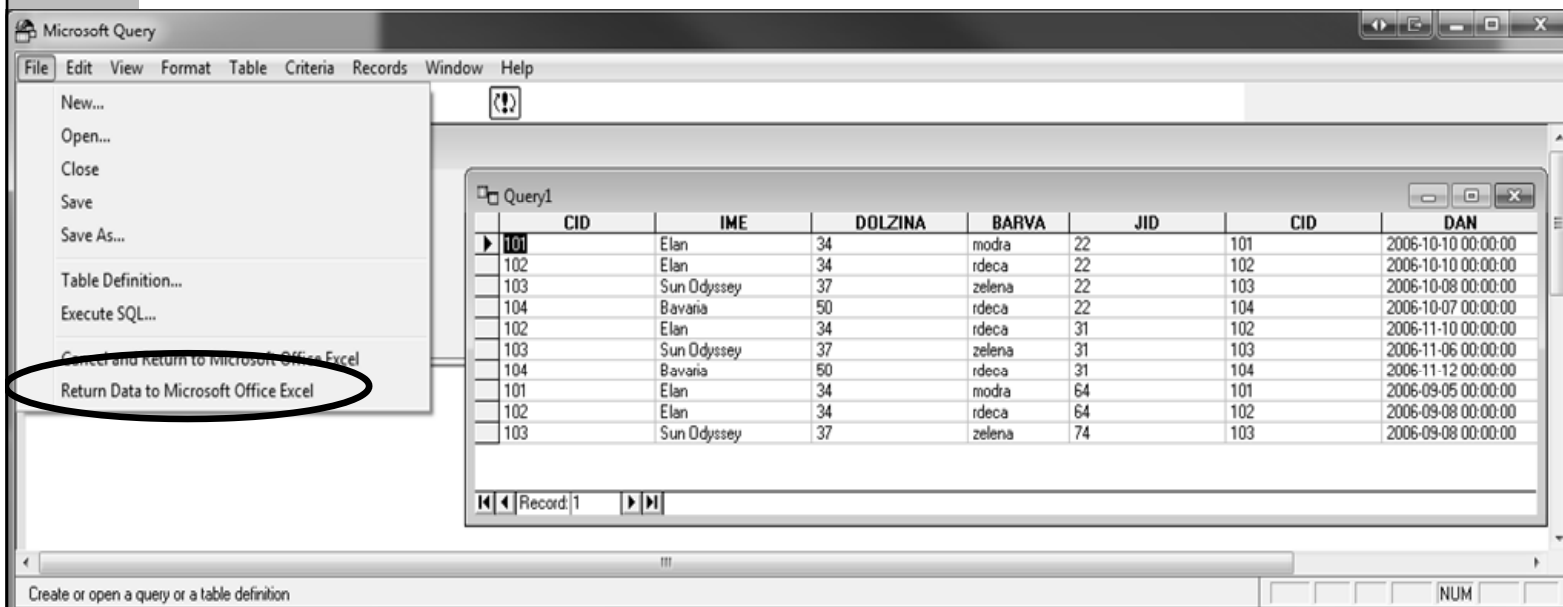
# Microsoft Excel in ODBC

- Ne izberite nobene tabele (gumb Close)
- Izberite File->Execute SQL
- Vnesite SQL poizvedbo in pritisnite gumb Execute

SQL poizvedbo je smiselno napisati in preveriti v za to namenjenem okolju (SQL Developer, MySQL Workbench) ob upoštevanju ODBC omejitev.

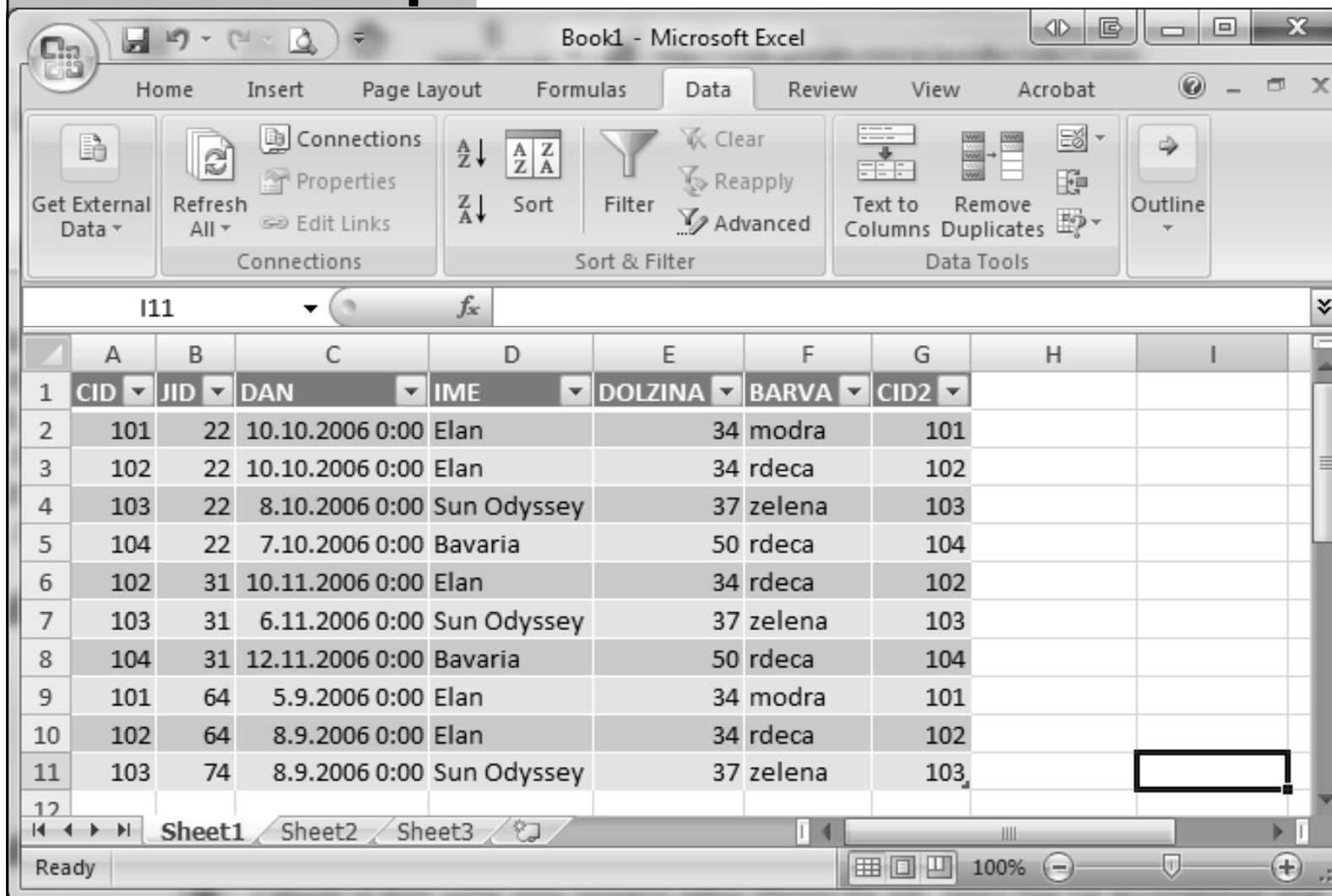


# Microsoft Excel in ODBC



- Izberite File->Return Data to Microsoft Excel
- V Excelu dobite tabelo z rezultatom
- Odvisno od definicije DSN (z ali brez gesla) je občasno potrebno vnesti ime in geslo za dostop do podatkovne baze

# Rezultat poizvedbe v Excelu

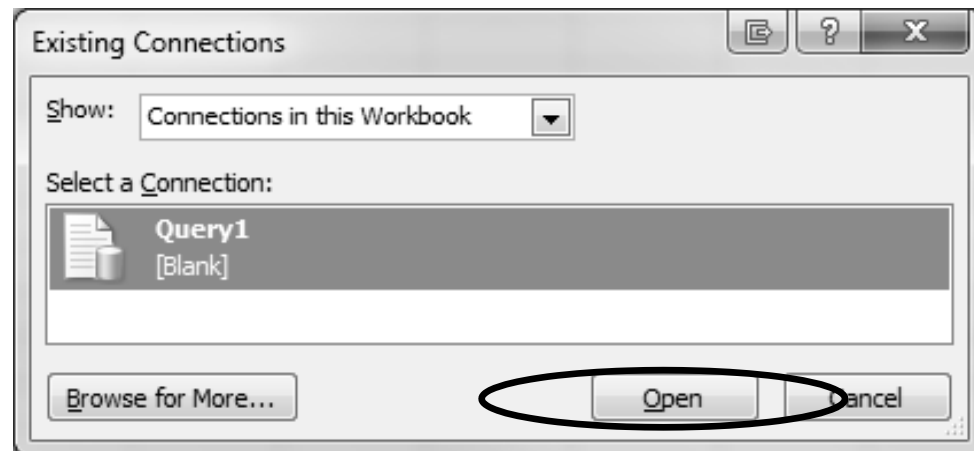


The screenshot shows the Microsoft Excel interface with the 'Data' tab selected. The data table is displayed in the following format:

	A	B	C	D	E	F	G	H	I
	CID	JID	DAN	IME	DOLZINA	BARVA	CID2		
2	101	22	10.10.2006 0:00	Elan	34	modra	101		
3	102	22	10.10.2006 0:00	Elan	34	rdeca	102		
4	103	22	8.10.2006 0:00	Sun Odyssey	37	zelena	103		
5	104	22	7.10.2006 0:00	Bavaria	50	rdeca	104		
6	102	31	10.11.2006 0:00	Elan	34	rdeca	102		
7	103	31	6.11.2006 0:00	Sun Odyssey	37	zelena	103		
8	104	31	12.11.2006 0:00	Bavaria	50	rdeca	104		
9	101	64	5.9.2006 0:00	Elan	34	modra	101		
10	102	64	8.9.2006 0:00	Elan	34	rdeca	102		
11	103	74	8.9.2006 0:00	Sun Odyssey	37	zelena	103		

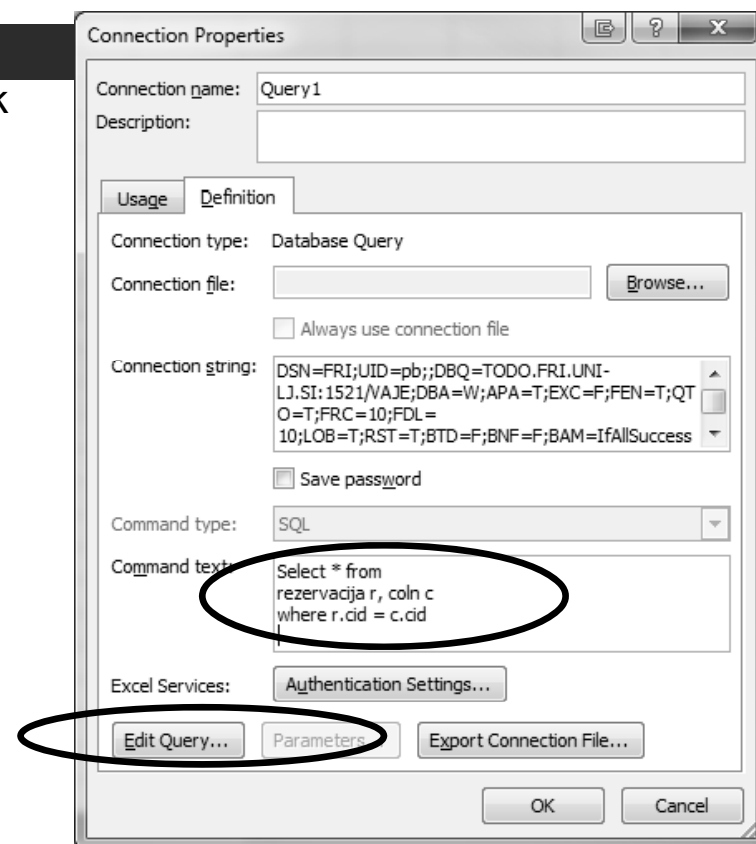
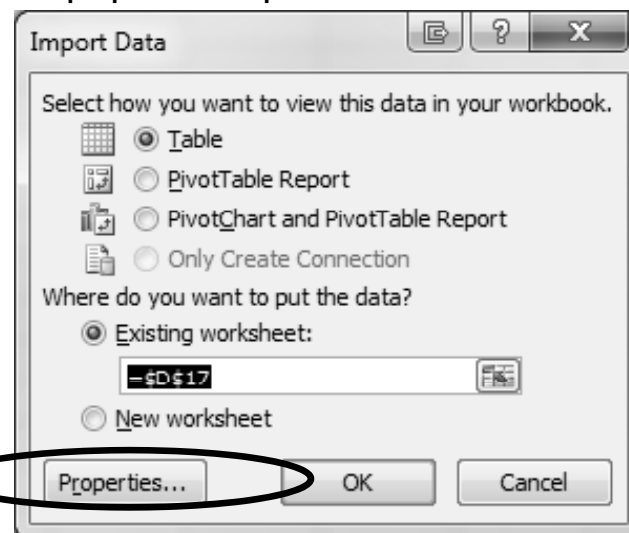
# Microsoft Excel in ODBC

- Povezave s podatkovno bazo so "žive"
  - S pritiskom na Data-> Refresh All osvežite vsebino rezultata poizvedbe
- Urejanje poizvedb
  - Pritisnite Data->Existing Connections
  - Izberite ustrezno poizvedbo in pritisnite Open



# Microsoft Excel in ODBC

- Izberite Properties in nato zavihek Definition
- V okencu Command text ali s klikom na Edit Query lahko popravimo poizvedbo



# Normalizacija relacij (tabel)

- Relacijski podatkovni model
- Relacija, atribut, relacijska shema
- Odvisnosti med atributi relacije:
  - Funkcionalne
  - Večvrednostne
  - Stične
- Ažurirne anomalije
- Normalne oblike relacij (1, 2, 3, BCNO, 4, 5) in postopki za normalizacijo
- Kako določiti ključ relacije na podlagi funkcionalnih odvisnosti?



# Relacijski podatkovni model

- Svet modeliramo z relacijami (množicami resničnih trditev)
- Atribut  $A_i$ : opisuje določeno lastnost
- $D_i$ : domena (vrednostna množica) atributa  $A_i$
- Relacija  $r$ :  $D_1 \times D_2 \times \dots \times D_n \rightarrow \{\text{res}, \text{ni\_res}\}$
- Shema relacije (glava tabele):  
 $\text{Sh}(r) = R(A_1:D_1, \dots, A_n:D_n)$
- Vsaka relacija ima natanko eno shemo, vsaki shemi pa lahko pripada več različnih relacij

## Odvisnosti med atributi relacije

- Relacija se podreja integritetnim omejitvam iz realnega sveta, ki omogočajo, le določene kombinacije vrednosti atributov.
- Integritetne omejitve v modelu določimo s pomočjo funkcionalnih in drugih odvisnosti.
- Odvisnosti so sredstvo, s katerim lahko v relacijskem modelu povemo, katere vrstice relacij (kombinacije vrednosti atributov) so oziroma bi lahko bile veljavne in katere sploh ne morejo obstajati.

# Upoštevanje odvisnosti

- Pri ažuriranju relacij (tabel) je treba odvisnosti upoštevati, sicer pride do ažurirnih anomalij.
- Več možnosti za upoštevanje:
  - Uporabnik se zaveda vseh odvisnosti in jih upošteva
  - SUPB se preverja vse odvisnosti (časovno zahtevno)
  - Preoblikovanje relacij na način, da do ažurirnih anomalij sploh ne more priti (normalizacija)

# Funkcionalne odvisnosti

- Funkcionalne odvisnosti veljajo na nivoju relacijske sheme, torej za vse relacije, ki pripadajo isti shemi.
- Imejmo relacijsko shemo  $R$  s podmnožicama atributov,  $X$  in  $Y$ .
- V relacijski shemi  $R$  velja  $X \rightarrow Y$  ( $X$  funkcionalno določa  $Y$  oziroma  $Y$  je funkcionalno odvisen od  $X$ ), če v nobeni relaciji, ki pripada shemi  $R$ , ne obstajata dve  $n$ -terici, ki bi se ujemali v vrednostih atributov  $X$  in se ne bi ujemali v vrednostih atributov  $Y$ .
- Preprosto povedano, obstaja neka funkcija, s pomočjo katere lahko iz vrednosti  $X$  izračunamo vrednosti  $Y$ .

## Ažurirne anomalije

- Relacije, ki vsebujejo odvečne podatke lahko povzročajo ažurirne anomalije pri operacijah nad podatki.
- Poznamo več vrst anomalij:
  - Anomalije pri dodajanju n-teric v relacijo
  - Anomalije pri brisanju n-teric iz relacije
  - Anomalije pri spreminjanju n-teric

## Anomalije pri dodajanju vrstic

- Dodajanje novih članov oddelka: ponovno moramo (pravilno) vpisati naslov oddelka
- Dodajanje novega oddelka: za podatke o članu vpišemo NULL

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6
Janko	Jankovič	1A	Tržaška 52
NULL	NULL	3A	Celovška 12

## Anomalije pri brisanju vrstic

- Brisanje edinega člana oddelka: izgubimo tudi vse informacije o tem oddelku (šifra oddelka, naslov)

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
<del>Marija</del>	<del>Kovač</del>	<del>2A</del>	<del>Dunajska 6</del>

## Anomalije pri spreminjanju vrstic

- Oddelek 1A se preseli na Jadransko 21. Naslov je treba pravilno popraviti pri vseh članih oddelka!

Ime	Priimek	Oddelek	Naslov
Janez	Novak	1A	Tržaška 25
Peter	Klepec	1A	Tržaška 25
Marija	Kovač	2A	Dunajska 6



# Primarni ključ

- Imejmo relacijsko shemo  $R$  s podmnožico atributov  $X$ .
- $X$  je ključ relacijske sheme  $R$ , če velja
  1.  $X \rightarrow R$
  2. Za noben atribut  $A$  iz  $X$  ne velja  $(X-A) \rightarrow R$  (minimalnost)
- Shema ima lahko več ključev, izberemo enega najprimernejšega, ki mu pravimo primarni ključ. Ostalim pravimo alternativni ključi.
- Specifikacija primarnih in alternativnih ključev omogoča dosledno spoštovanje nekaterih omejitev!

# Normalizacija

- Normalizacija je postopek, s katerim pridemo do množice primerno strukturiranih relacij, ki ustrezajo kriteriju normalne oblike.
- Lastnosti primernih relacij:
  - Relacije imajo minimalen nabor atributov
  - Atributi, ki so logično povezani, so zajeti v isti relaciji
  - Med atributi relacij je minimalna redundanca, vsak atribut (razen tujih ključev) je predstavljen samo enkrat.

# Prva normalna oblika

- Relacija je v prvi normalni obliki, če:
  - Nima večvrednostnih atributov, kar pomeni, da ima vsak atribut lahko le eno vrednost (torej vrednost ne more biti množica). Primer: vzdevek
  - Nima sestavljenih atributov (torej vrednost ne more biti relacija). Primer: naslov
  - Ima definiran ključ in določene funkcionalne odvisnosti
- Koraki:
  - Eliminiranje ponavljajočih skupin (večvrednostnih sestavljenih atributov)
  - Določitev funkcionalnih odvisnosti
  - Določitev ključa

## Primer normalizacije v 1. NO

Voznik (ime, priimek, stdov, (datum, znesek, davčna))  
Prekršek

- Odpravimo ponavljajočo skupino:  
Voznik (ime, priimek, stdov)  
Prekršek(datum, znesek, davčna)
- Določimo ključe:  
Voznik (ime, priimek, stdov)  
Prekršek(#stdov, datum, znesek, davčna)  
Relacijska shema Prekršek vključuje ključ originalne sheme.
- Določimo funkcionalne odvisnosti:  
stdov → ime, stdov → priimek,  
stdov, datum → znesek, stdov → davčna, davčna → stdov

## Druga normalna oblika

Shema: ABCDE

$ABC \rightarrow D$

$ABC \rightarrow E$

$B \rightarrow E$  parcialna

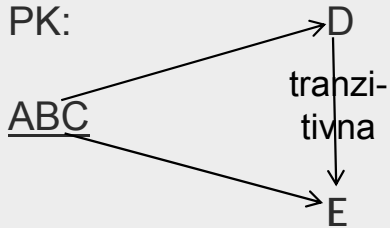
- Relacija je v drugi normalni obliki:
  - Če je v prvi normalni obliki
  - Ne vsebuje parcialnih odvisnosti: noben atribut ni funkcionalno odvisen le od dela primarnega ključa, temveč od celotnega ključa
- Nekaj pogostih primerov:
  - Relacija, katere primarni ključ je sestavljen le iz enega atributa, je v drugi normalni obliki
  - Relacija, katere primarni ključ je sestavljen iz vseh atributov, je v drugi normalni obliki
- 2. NO je definirana kot pomožna NO za definicijo 3. NO

## Primer normalizacije v 2. NO

- Voznik (ime, priimek, stdov)  
stdov → ime, stdov → priimek ✓
- Prekršek(stdov, datum, znesek, davčna)  
stdov, datum → znesek, stdov → davčna, davčna → stdov ✗
- Postopek normalizacije: problematične neosnovne attribute (tiste, ki niso del ključa) in so delno odvisni od njega prenesemo v novo tabelo in dodamo še dele ključa, od katerih so odvisni
- Prekršek(stdov, datum, znesek)  
stdov, datum → znesek, ✓
- Davek(stdov, davčna)  
stdov → davčna, davčna → stdov ✓

\_\_\_\_\_

- funkcionalnih odvisnosti: ni  
ed atributi, ki niso del primarnega ključa  
del primarnega ključa, ki bi bil  
rugega atributa, ki ravno tako ni del



## Tretja normalna oblika

- Relacija je v 3. NO (formalna definicija), če za vsako odvisnost  $X \rightarrow A \in F$  velja vsaj eden izmed pogojev:
  1.  $X \rightarrow A$  je trivialna odvisnost ( $A \subseteq X$ )
  2.  $X$  je nadključ sheme  $R$
  3.  $A$  je osnovni atribut (del nekega ključa)
- Normalizacija v 3. NO je neizgubna; s stikom dobljenih relacij lahko dobimo nazaj originalno relacijo (obstoj neizgubnega stika).



## Postopek normalizacije v 3. NO

- Dekompozicija relacijske sheme  $R$  v  $\rho$ 
  - Določimo  $F$  (še bolje: izračunamo  $F_{\min}$ )
  - Vsaki problematični odvisnosti  $X \rightarrow A \in F$  priredimo novo relacijsko shemo  $XA$  v  $\rho$ , razen v primeru, če že obstaja kakšna shema, ki  $XA$  vključuje kot podmnožico. Desno stran odvisnosti ( $A$ ) izločimo iz originalne sheme.
  - Kar ostane od originalne relacijske sheme dodamo v  $\rho$ , razen če v  $\rho$  že obstaja kakšna shema, ki jo vsebuje

## Primer normalizacije v 3. NO

R=ABCDEFGG

F={A → D, AG → B, B → G, B → E, E → B, E → F}

Ključ: ACG

1.  $\rho = \{\}$
2. A → D:  $\rho = \rho \cup \{AD\}$
3. AG → B:  $\rho = \{AD\} \cup \{AGB\}$
4. B → G: ni problematična
5. B → E:  $\rho = \{AD, AGB\} \cup \{BE\}$
6. E → B:  $\{EB\} \subseteq \{BE\}$
7. E → F:  $\rho = \{AD, AGB, BE\} \cup \{EF\}$
8. Končamo:  $\rho = \{AD, AGB, BE, EF\} \cup \{ACG\}$

$\rho = \{AD, AGB, BE, EF, ACG\}$

## Ali so spodnje relacije v 3. NO?

- Voznik (ime, priimek, stdov)  
stdov → ime, stdov → priimek
- Prekršek(stdov, datum, znesek)  
stdov, datum → znesek,
- Davek(stdov, davčna)  
stdov → davčna, davčna → stdov
- PrekršekDavek (stdov, datum, znesek, davčna)  
stdov, datum → znesek, stdov → davčna, davčna → stdov

## Pomožni pojmi

- Imejmo shemo R v kateri velja množica funkcionalnih odvisnosti F
- Osnovni atribut: del nekega (ne nujno primarnega) ključa
- Kanonična oblika funkcionalne odvisnosti: na desni strani je največ en atribut
- Logična izpeljava odvisnosti:  $F \Rightarrow X \rightarrow Y$
- Zaprtje množice odvisnosti:  $F^+ = \{X \rightarrow Y : F \Rightarrow X \rightarrow Y\}$  (vse možne izpeljane odvisnosti)

## Zaprtje (closure) množice atributov

- Zaprtje množice atributov  $X$  glede na  $F$   
 $X^+ = \{A: X \rightarrow A \in F^+\}$

- Postopek za izračun  $X^+$

Vhod:  $X, F$

Izhod:  $X^+$

$X^+ = X$

ponavljaaj

    stari  $X^+ = X^+$

    za vsako odvisnost  $Y \rightarrow Z \in F$  naredi

        če  $Y \subseteq X$  potem

$X^+ = X^+ \cup Z$

dokler ni stari  $X^+ = X^+$

## Primer izračuna zaprtja

$R = ABCDEFG$

$F = \{A \rightarrow B, BE \rightarrow G, EF \rightarrow A, D \rightarrow AC\}$

Iščemo  $\{EF\}^+$ :

1.  $\{EF\}^+ = EFA$
2.  $\{EF\}^+ = EFAB$
3.  $\{EF\}^+ = EFABG$
4.  $\{EF\}^+ = EFABG$

## Minimalno pokritje množice odvisnosti

- $F$  pokriva  $E$ :  $\forall f \in E: F \Rightarrow f$  oziroma  $E^+ \subseteq F^+$
- Minimalno pokritje  $F_{\min}$ : zahtevamo  $F_{\min}^+ = F^+$ , vendar ohranimo samo neredundantne odvisnosti
  - Kanonična oblika (en atribut na desni)
  - Minimalnost (ne moremo odstraniti nobene odvisnosti, da bi še vedno veljalo  $F_{\min}^+ = F^+$ )
  - $\forall F_{\min}$  ne moremo zamenjati nobene  $X \rightarrow A$  z  $Y \rightarrow A$ ,  $Y \subset X$ , da bi še vedno veljalo  $F_{\min}^+ = F^+$ )
- Postopek v praksi: narišemo graf odvisnosti, kjer odstranimimo tranzitivne povezave. Rezultat ni nujno enoličen!
- Primer:  $R = ABC$ ,  $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C, C \rightarrow A\}$

## Izpeljevanje funkcionalnih odvisnosti in določanje ključev

- Armstrongovi aksiomi
- Izpeljana pravila sklepanja
- Postopki za določanje kandidatov za ključe na osnovi funkcionalnih odvisnosti:
  - Elmasri-Navathe
  - Edmondson
  - Saiedian-Spencer



## Iskanje ključev relacije na podlagi funkcionalnih odvisnosti

- Splošni veljavne resnice
  - Atribut, ki ne nastope na desni strani nobene funkcionalne odvisnosti, mora biti vsebovan v vsakem ključu
  - Atribut, ki nastopa na desni strani neke funkcionalne odvisnosti in ne nastopa na levi strani nobene funkcionalne odvisnosti, ne more biti vsebovan v nobenem ključu
  - Dobri kandidati za ključje so leve strani funkcionalnih odvisnosti in njihove unije

## Elmasri-Navathe algoritem za določanje enega ključa

- Vhod: relacijska shema  $R$ , množica funkcionalnih odvisnosti  $F$ 
  1. Postavi  $K$  = začetni kandidat, npr.  $R$  (vsi atributi)
  2. Za vsak atribut  $X \in K$ 
    - Izračunaj  $\{K-X\}^+$  glede na  $F$
    - Če  $\{K-X\}^+$  vsebuje vse attribute  $R$  (poenostavljeno  $K$ , če začnemo z  $R$ ) postavi  $K = K - \{X\}$
  3. Kar ostane v  $K$  je ključ.
- Problem: vrne samo en ključ, odvisen od vrstnega reda pregledovanja atributov

## Primer (Elmasri-Navathe):

**R=ABCDEFGG**

**F={A → D, AG → B, B → G, B → E, E → B, E → F}**

- K=ABCDEFGG, X=A  
K-X= BCDEFG, (K-X)<sup>+</sup>= BCDEFG
- K=ABCDEFGG, X=B  
K-X= ACDEFG, (K-X)<sup>+</sup>= ABCDEFG (AG→B)
- K=ACDEFG, X=C  
K-X= ADEFG, (K-X)<sup>+</sup>= ADEFG
- K=ACDEFG, X=D  
K-X= ACEFG, (K-X)<sup>+</sup>= ACDEFG (A→D)
- K=ACEFG, X=E  
K-X= ACFG, (K-X)<sup>+</sup>= ACEFG (AG → B, B → E)
- K=ACFG, X=F  
K-X= ACG, (K-X)<sup>+</sup>= ACFG (AG → B, B → E, E → F)
- K=ACG, X=G  
K-X= AC, (K-X)<sup>+</sup>= AC
- Ključ je ACG

## Saiedian-Spencer algoritem za določanje vseh ključev

- Vhod: relacijska shema  $R$ ,  
min. pokritje funkcionalnih odvisnosti  $F_{\min}$
- 1. Poišči množice  $\mathcal{L}$  (atributi samo na levi strani odvisnosti in atributi ki ne nastopajo v nobeni odvisnosti),  $\mathcal{R}$  (atributi samo na desni strani odvisnosti) in  $\mathcal{B}$  (atributi na levi in desni strani odvisnosti)
- 2. Preveri množico  $\mathcal{L}$ . Če  $\mathcal{L}^+ = R$ , je edini ključ in lahko končaš, sicer nadaljuj na koraku 3.
- 3. Preveri množico  $\mathcal{B}$  tako da v  $\mathcal{L}$  vstavljaš po vrsti vse možne kombinacije atributov  $X$  iz  $\mathcal{B}$ , začenši s posameznimi atributi. Kadar dobimo  $\{\mathcal{L} \cup X\}^+ = R$ , smo našli ključ. N-teric, ki vsebujejo  $X$ , dalje ne obravnavamo več.

## Primer (Saiedian-Spencer):

$R=ABCDEFGG$

$F=\{A \rightarrow D, AG \rightarrow B, B \rightarrow G, B \rightarrow E, E \rightarrow B, E \rightarrow F\}$

1.  $\mathcal{L}=CAGBE=CA$   
 $\mathcal{R}=DBGEF=DF$   
 $\mathcal{B}=BGE$
2.  $\mathcal{L}^+=CAD \subseteq R$
3.  $X=B \mathcal{L}=CAB$   
 $\mathcal{L}^+=CABDGEF = R$
4.  $X=G \mathcal{L}=CAG$   
 $\mathcal{L}^+=CAGDBEF = R$
5.  $X=E \mathcal{L}=CAE$   
 $\mathcal{L}^+=CAEDBFG = R$

Ključ: ABC, ACG, ACE

## Primer izpitne naloge

- Za relacijsko shemo  $R$  s funkcionalnimi odvisnostmi  $F$  poiščite oba ključa, določite minimalno pokritje  $F_{\min}$  in ugotovite, v kateri najvišji normalni obliki se nahaja! Odgovore utemeljite!

$R =$  ABCD

$F =$  {ACD $\rightarrow$ B, BCD $\rightarrow$ A, ACD $\rightarrow$ AB, BD $\rightarrow$ D,  
AC $\rightarrow$ C , AB $\rightarrow$ B}