SQL I

BAZE PODATAKA I

doc. dr. sc. Goran Oreški
Fakultet informatike,
Sveučilište Jurja Dobrile, Pula

Sadržaj

- ponavljanje prethodnih predavanja
  - proširene operacije relacijske algebre
  - *null* vrijednost
  - promjene podataka u bazi
- SQL osnove
- stvaranje tablica
- tipovi domena
- umetanje redova
- `SELECT` operacija
- dijelovi `SELECT` operacije
- primjeri upita
- oblikovanje SQL koda

Ponavljanje

- proširene operacije relacijske algebre
  - generalizirana projekcija – koristeći operaciju mogu se specificirati atributi (projekcija), kao i funkcije na atributima
    - mogu se dodijeliti imena izračunatim vrijednostima ili preimenovati atributi
  - funkcije agregacije – različite funkcije, npr.?
    - korak grupiranja, korak primjene funkcije
  - dodatne join operacije
    - theta join
    - outer joins
      - left, right, full
- *null* vrijednost – nepoznata ili nepostojeća vrijednost

Ponavljanje

- podaci se često mijenjaju u bazi podataka
- za promjenu podataka se koristi operator dodjeljivanja ⟵
- promjene podataka u bazi

  $r \longleftarrow r \cup E$
  - umetanje novih n-torki u relaciju

  $r \longleftarrow r - E$
  - brisanje n-torki iz relacije

  $r \longleftarrow \Pi(r)$
  - promjena n-torki koje se nalaze u relaciji

SQL

- SQL = Structured Query Language
- originalni jezik naziva “SEQUEL”
  - IBM System R project (rane 70-te)
  - “Structured English Query Language”
- vrlo dobro prihvaćen
  - jednostavan, deklarativni jezik za pisanje upita
  - uključuje mnoge druge značajke
- standardiziran ANSI/ISO
  - SQL:2003, SQL:2008, SQL:2011
  - većina implementacija donekle prati standardizaciju (oprez prenošenje koda!)

SQL značajke

- Data Definition Language (DDL)
  - specificiranje shema relacija
  - definiranje ograničenja
  - ograničavanje pristupa podacima
- Data Manipulation Language (DML)
  - bazirano na relacijskoj algebri
  - podržava upite, insert, update i delete podataka
  - napredne značajke za multi-table upite
- drugi korisni alati
  - definiranje pogleda, transakcija…

SQL osnove

- SQL jezik nije case-sensitive
  - vrijedi za ključne riječi i identifikatore
- SQL izraz završava s dvotočkom (;)
- SQL komentari imaju dva oblika
  - komentar jednog reda

```
-- This is a SQL comment.
```

  - komentari bloka

```
/*
This is a SQL comment.
*/
```

SQL baze podataka

- SQL relacije su sadržane unutar baza podataka
  - aplikacija može raditi u okviru svoje baze podataka
  - više aplikacije može dijeliti jednu bazu podataka
- primjer iz *MySql* baze podataka

```
CREATE DATABASE sveuciliste;
USE sveuciliste;
```

  - stvara se prazna baza podataka pod nazivom *sveuciliste*
  - USE naredba čini *sveuciliste* zadanom bazom podataka za trenutnu konekciju
  - DDL i DML naredbe će biti provedene u kontekstu zadane baze podataka trenutne konekcije

Stvaranje SQL tablice

- u SQL-u relacija se naziva tablica
- sintaksa:

```
CREATE TABLE t (
attr1 domain1,
attr2 domain2,
... ,
attrN domainN
);
```

  - *t* je naziv relacije (tablice)
  - *attr1*,… je naziv atributa (stupca)
  - *domain1*,… je naziv domene (tipa) atributa

SQL imena

- tablice, kolone.. trebaju imena
- valjana imena razlikuju se od implementacije
  - ovisno o korištenom sustavu potrebno je proučiti dokumentaciju
- dobra, univerzalna pravila:
  - prvi znak treba biti iz alfabetskog skup znakova (ne sadrži brojke (znamenke))
  - ostali znakovi mogu biti iz alfanumeričkog skupa znakova ili podvlaka ”_”(*engl. underscore*)
  - potrebno je koristiti isti case u DML koji se koristi u DDL
  - ne koristiti hrvatske znakove (č, ć, š, đ, ž)

Domene atributa

- uobičajeni SQL tipovi domena:

  `CHAR(N)`
  - polje znakova, ograničeno na veličinu N
  - skraćeno od `CHARACTER(N)`

  `VARCHAR(N)`
  - polje znakova varijabilne veličine, s maksimalnom veličinom N
  - skraćeno od `CHARACTER VARYING(N)`

  `INT`
  - signed cijeli broj (tipično 32 bita)
  - skraćeno do `INTEGER`
  - postoje `TINYINT(8 bita)`, `SMALLINT(16 bita)`, `BIGINT(64 bita)`…
  - postoje i nestandardne unsigned varijante

CHAR vs VARCHAR

- oboje CHAR i VARCHAR imaju limitiranu veličinu
- CHAR je polje znakova fiksne veličine
  - mogu se pohraniti znakovni nizovi manje veličine, sustav prilikom pohrane popuni ostatak prostora
- VARCHAR je polje znakova varijabilne veličine
  - sustav ne popunjava ostatak prostora prilikom spremanja kraćih znakovnih nizova
  - dužina znakovnog niza se mora pohraniti za svaku vrijednost
- CHAR se koristi kada su nizovi otprilike ili točno iste dužine dok VARCHAR za nizove koji jako variraju u dužini

Cjelobrojne vrijednosti

TIP	BAJTOVA	MIN. VRIJEDNOST	MAX. VRIJENDOST
		Signed/Unsigned	Signed/Unsigned
`tinyint`	1	-128	127
		0	255
`smallint`	2	-32768	32767
		0	65535
`mediumint`	3	-8388608	8388607
		0	16777215
`int`	4	-2147483648	2147483647
		0	4294967295
`bigint`	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Domene atributa

- ostali uobičajeni SQL tipovi domena:

  `NUMERIC(P,D)`

  - fixed point broj s korisnički specificiranom preciznosti
  - P je ukupan broj znamenki; D je broj znamenki na desno od decimalne točke
  - egzaktna pohrana brojeva

  `DOUBLE PRECISION`

  - folating-point vrijednost
  - to je aproksimacija; ne koristiti za novac
  - `REAL` je ponekad sinonim

  `FLOAT(N)`

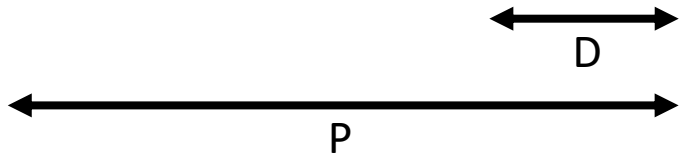
  - folating-point vrijednost s najmanje N bitova preciznosti

Egzaktna reprezentacija i aproksimacija

- definiramo atribut za pohranu salda

```
…
saldo numeric(8,2)
…
```

187456.17




```
…
saldo double precision
…
```

1.8745617 x 10^5

Domene atributa

- još SQL tipova domene

  `DATE`, `TIME`, `TIMESTAMP`

  - za pohranu vremenskih podataka

  - velika binarna/tekstualna polja podataka

  `BLOB`, `CLOB`, `TEXT`

  - Binary Large Objects, Character Large Objects
  - velika tekstualna polja
  - kada `CHAR` i `VARCHAR` nisu dovoljni

  drugi tipovi

  - enumeracije, geometrijski ili prostorni tipovi podataka
  - korisnički definirani tipovi (na temelju postojećih tipova)

Odabir pravog tipa

- potrebno je dobro razumjeti podatke da bi se ispravno odredio tip podataka (domena)
- primjer: na sveučilištu trebamo pohraniti poštanske brojeve u okviru prikupljanja adresa studenata!
  - poštanski broj u Hrvatskoj se sastoji od 5 znamenki (10000 Zagreb, 51000 Rijeka…)
- da li `INTEGER` ima smisla?
  - postoji li mogućnost da strani student dođe studirati?
  - različiti oblici poštanski brojeva, neki uključuju:
    - znakove
    - vodeće nule…

Odabir pravog tipa

- bolji izbor tipa za poštanske adrese?
  - `CHAR` ili `VARCHAR`?
- obzirom da ne znamo format poštanskih brojeva svih zemalja iz kojih studenti mogu doći, sigurnija opcija bi bila npr. `VARCHAR(10)`
- drugi primjer, podaci vezani uz novčane iznose
  - floating-point reprezentacije ne mogu uvjek točno pohraniti broj
  - npr. 0.1 je beskonačno ponavljajuća binarna decimalna vrijednost
  - za novčane iznose uvjek se koristi `NUMERIC`

Primjer SQL sheme

- stvaranje relacije nastavnik:

```
CREATE TABLE nastavnik (
    nastavnik_sifra CHAR(10),
    prezime         VARCHAR(20),
    primanja        NUMERIC(10,2)
);
```

  - šifra nastavnika ne može biti veća od 10 znakova
  - prezime ne može biti veće od 20 znakova, koristimo varijabilni znakovni niz
  - primanja imaju 8 znamenki lijevo od decimalne točke i 2 znamenke desno
    - koristi se fixed-point reprezentacija broja

Umetanje redova

- tablice su stvaraju prazne
- podatke je potrebno dodati koristeći `INSERT` naredbu:

```
INSERT INTO nastavnik
   VALUES('R-34532', 'Kovačević', 8876);
INSERT INTO nastavnik
   VALUES('I-337', 'Marić', 2876.10);
```

  - za navođenje stringa koriste se jednostruki navodnici
  - dvostruki navodnici u SQL se koriste za naziv stupaca
  - vrijednosti se navode u istom redoslijedu kako su definirani atributi

Umetanje redova

- atributi se mogu specificirati unutar naredbe `INSERT`

```
INSERT INTO nastavnik(nastavnik_sifra, prezime, primanja)
   VALUES('R-34532', 'Kovačević', 8876);
```

  - atributi se mogu navoditi u različitom poretku
  - atributi se mogu i isključiti ako imaju postavljene zadane vrijednosti

- problem: mogu se dodati nastavnici s istom šifrom

```
INSERT INTO nastavnik
   VALUES('R-34532', 'Kovačević', 8876);
INSERT INTO nastavnik
   VALUES('R-34532', 'Marić', 2876.10);
```

Ograničenje primarnog ključa

- naredba CREATE TABLE također dopušta definiranje ograničenja primarnog ključa
  - pravilo je da se definira nakon svih atributa
- ograničenje primarnog ključa

```
CREATE TABLE nastavnik (
   nastavnik_sifra CHAR(10),
   prezime         VARCHAR(20),
   primanja        NUMERIC(10,2),
   PRIMARY KEY (nastavnik_sifra)
);
```

  - baza podataka neće dopustiti unos iste vrijednosti primarnog ključa

Ograničenje primarnog ključa

- primarni ključ može imati više atributa

```
CREATE TABLE predaje (
    nastavnik_sifra CHAR(10),
    kolegij_naziv   VARCHAR(40),
    PRIMARY KEY (nastavnik_sifra, kolegij_naziv)
);
```

  - potrebno zato što su SQL tablice multiskupovi
- tablica ne može imati više primarnih ključeva
  - zašto?

Brisanje redova, tablica…

- redovi iz tablice se brišu naredbom DELETE
  - brišemo nastavnika sa šifrom R-34532

```
DELETE FROM nastavnik WHERE nastavnik_sifra = 'R-34532';
```

  - brišemo sve nastanike

```
DELETE FROM nastavnik;
```

- tablice i baze podataka se brišu naredbom DROP
  - brisanje tablice nastavnik

```
DROP TABLE nastavnik;
```

  - brisanje cijele baze podataka, zajedno sa svim tablicama

```
DROP DATABASE sveuciliste;
```

Izmjena tablica

- za izmjenu strukture tablice tj. dodavanje i brisanje redova koristi se naredba `ALTER TABLE`

```
ALTER TABLE r ADD att_name att_type;
```

  - *r* je naziv relacije kojoj se dodaje atribut
  - *att_name* je naziv novog atributa koji se dodaje
  - *att_type* je domena (tip) novog atributa

vrijednost novog atributa za n-torke koje se nalaze unutar relacije je NULL!

- brisanje atributa (*ne dopuštaju svi sustavi baza podataka*)

```
ALTER TABLE r DROP att_name;
```

  - *r* je naziv relacije iz koje se briše atribut
  - *att_name* atribut koji se briše

SQL upiti

- SQL upiti se zadaju `SELECT` naredbom
- glavni aspekt SQL jezika
- općenita forma `SELECT` naredbe:

```
SELECT A1, A2, …
FROM r1, r2, …
WHERE P;
```

  - r_i su relacije (tablice)
  - A_i su atributi (stupci)
  - P je predikat selekcije

SELECT operacija

- **SELECT A_1, A_2, …**
  - odgovara operaciji projekcije u relacijskoj algebri $\prod_{A_1,A_2,\ldots}(\ldots)$
  - u ovom slučaju SQL terminologija ne odgovara onoj iz relacijske algebre
  - postoji još razlika koje će biti naglašene u nastavku
- **FROM r_1, r_2, …**
  - odgovara Kartezijevom produktu relacija r_1, r_2, …
  - $r_1 \times r_2 \times \ldots$
  - navode se sve relacije koje sadrže podatke do kojih želimo doći u rezultatu upita

SELECT operacija

- **WHERE P;**
  - odgovara operaciji selekcije iz relacijske algebre $\sigma_P(\dots)$
  - može se izostaviti
  - ako nije navedena *P = true*
- kada se svi dijelovi operacije SELECT spoje

```
SELECT A1, A2, …
FROM r1, r2, …
WHERE P;
```

  - ekvivalentno izrazu: $\Pi_{A_1,A_2,\dots}(\sigma_P(r_1 \times r_2 \times \cdots))$

SQL i duplikati

- najveća razlika između relacijske algebre i SQL jezika je korištenje multiskupova
  - u SQL-u, relacija je multiskup n-torki, ne skup
- najveći razlog je praktičan
  - uklanjanje duplikata je vremenski zahtjevno
- iz navedenog razloga potrebno je ponovno sagledati neke od operacija relacijske algebre
  - najviše utječe na operacije sa skupovima: unija, presjek, razlika
- SQL omogućava načine za uklanjanje duplikata za sve operacije

Primjeri upita

- prikažite sve podatke iz relacije nastavnik

```
SELECT nastavnik_sifra, prezime, primanja FROM nastavnik;
```

  - isto kao:

```
SELECT * FROM nastavnik;
```

```
+----------------+------------+----------+
| nastavnik_sifra | prezime     | primanja |
+----------------+------------+----------+
| I-337           | Marić       |  2876.10 |
| R-34532         | Kovačević   |  8876.00 |
| R-67432         | Marić       |  6784.20 |
+----------------+------------+----------+
```

- korištenje * u naredbi `SELECT` se koristi kada se dohvaćaju **svi** atributi relacije

Primjeri upita

- pronađite prezimena svih nastavnika na sveučilištu

```
SELECT prezime FROM nastavnik;
```

  - isto kao:

```
SELECT ALL prezime FROM nastavnik;
```

```
+-------------+
| prezime     |
+-------------+
| Marić       |
| Kovačević   |
| Marić       |
+-------------+
```

- da bi se eliminirali duplikati

```
SELECT DISTINCT prezime FROM nastavnik;
```

```
+-------------+
| prezime     |
+-------------+
| Marić       |
| Kovačević   |
+-------------+
```

Primjeri upita

- moguće je specificirati bilo koji podskup atributa relacije
  - pronađite sve nastavnike i njihova primanja na fakultetu

```
SELECT nastavnik_sifra, primanja FROM nastavnik;
```

```
+-----------------+----------+
| nastavnik_sifra | primanja |
+-----------------+----------+
| I-337           |  2876.10 |
| R-34532         |  8876.00 |
| R-67432         |  6784.20 |
+-----------------+----------+
```

- za dohvat svih atributa koristi se *

Računanje rezultata

- naredba `SELECT` u SQL-u se može koristiti kao operacija generalizirane projekcije
  - mogu se računati nove vrijednosti na temelju atributa

```
SELECT nastavnik_sifra, primanja + 1000
    FROM nastavnik;
```

  - nova kolona nema ime
    - neki DBMS sustavi generiraju sami ime
  - imena kolonama se mogu dodati ili izmjeniti

```
SELECT nastavnik_sifra,
       primanja + 1000 AS povisica
FROM nastavnik;
```

WHERE ključna riječ

- WHERE ključna riječ specificira predikat selekcije
  - mogu se koristiti operatori usporedbe:

    =, <>, != jednako ili različito

    <, <= manje od, manje ili jednako

    >, >= veće od, veće ili jednako

  - može se odnositi na bilo koji atribut iz relacija navedenim u FROM dijelu SELECT-a
  - mogu se koristiti aritmetički izrazi u usporedbi

WHERE primjeri

- pronađite šifre svih profesora na sveučilištu koji se prezivaju Marić

```
SELECT nastavnik_sifra FROM nastavnik
   WHERE prezime = 'Marić';
```

```
+-----------------+
| nastavnik_sifra |
+-----------------+
| I-337           |
| R-67432         |
+-----------------+
```

- pronađite sve profesore na sveučilištu koji imaju primanja veća od 5000.00

```
SELECT * FROM nastavnik
   WHERE primanja > 5000.00;
```

```
+-----------------+------------+----------+
| nastavnik_sifra | prezime    | primanja |
+-----------------+------------+----------+
| R-34532         | Kovačević  |  8876.00 |
| R-67432         | Marić      |  6784.20 |
+-----------------+------------+----------+
```

Složeniji predikati

- unutar WHERE klauzule mogu se koristiti AND, OR i NOT

```
SELECT nastavnik_sifra, prezime FROM nastavnik
   WHERE prezime = 'Marić' AND primanja > 5000.00;
```

  - tražimo nastavnika po prezimenu i primanjima

```
SELECT nastavnik_sifra, prezime FROM nastavnik
   WHERE primanja >= 2500.00 AND primanja <= 5000.00;
```

- u SQL se mogu koristiti BETWEEN i NOT BETWEEN ključne riječi

```
SELECT * FROM nastavnik
   WHERE primanja BETWEEN 2500.00 AND 5000.00;
```

  - BETWEEN obuhvaća i krajnje točke intervala

Usporedba znakovnog niza (string)

- string vrijednosti se mogu uspoređivati
  - leksikografska usporedba
  - često je zadano da se ne gleda *case* znakova

```
SELECT 'DOBAR DAN' = 'dobar dan';   --istina
```

- moguće je tražiti uzorak unutar stringa s `LIKE` ključnom riječi

```
string_attr LIKE pattern
```

  - % (postotak) mijenja substring
  - _ mijenja jedan znak
  - traženje znakova % ili _ se označava s \ ispred znaka
  - `LIKE` koristi case sensitive usporedbu?!

`LIKE` primjer

- pronađite prezimena svih nastavnika koji u prezimenu sadrže '*ić*'

```
SELECT * FROM nastavnik WHERE prezime LIKE '%ić%';
```

```
+-----------------+------------+----------+
| nastavnik_sifra | prezime    | primanja |
+-----------------+------------+----------+
| I-337           | Marić      |  2876.10 |
| R-34532         | Kovačević  |  8876.00 |
| R-67432         | Marić      |  6784.20 |
+-----------------+------------+----------+
```

- mala izmjena u izrazu

```
SELECT * FROM nastavnik WHERE prezime LIKE '%ić_';
```

  - % predstavlja 0 ili više znakova, _ točno jedan znak

FROM ključna riječ

```
CREATE TABLE predaje (
    nastavnik_sifra CHAR(10),
    kolegij_naziv   VARCHAR(40),
    PRIMARY KEY (nastavnik_sifra, kolegij_naziv));
INSERT INTO predaje VALUES('R-34532', 'NTP');
INSERT INTO predaje VALUES('R-34532', 'SPI');
INSERT INTO predaje VALUES('R-67432', 'BP1');
```

- unutar `FROM-a` se može specificirati veći broj tablica
- ukoliko se koristi na više tablica:
  - rezultat je Kartezijev produkt
  - stvara se red za svaku kombinaciju redova iz tablica

```
SELECT * FROM nastavnik, predaje;
```

nastavnik_sifra	prezime	primanja	nastavnik_sifra	kolegij_naziv
I-337	Marić	2876.10	R-34532	NTP
R-34532	Kovačević	8876.00	R-34532	NTP
R-67432	Marić	6784.20	R-34532	NTP
I-337	Marić	2876.10	R-34532	SPI
R-34532	Kovačević	8876.00	R-34532	SPI
R-67432	Marić	6784.20	R-34532	SPI
I-337	Marić	2876.10	R-67432	BP1
R-34532	Kovačević	8876.00	R-67432	BP1
R-67432	Marić	6784.20	R-67432	BP1

FROM ključna riječ

- ukoliko tablice imaju nazive stupaca s istim imenom tada je potrebno koristiti `table_name.att_name` da bi se isti mogli razlikovati

```
SELECT * FROM nastavnik, predaje
WHERE nastavnik.nastavnik_sifra = predaje.nastavnik_sifra;
```

```
+----------------+------------+---------+----------------+---------------+
| nastavnik_sifra | prezime     | primanja | nastavnik_sifra | kolegij_naziv |
+----------------+------------+---------+----------------+---------------+
| R-34532         | Kovačević   |  8876.00 | R-34532         | NTP           |
| R-34532         | Kovačević   |  8876.00 | R-34532         | SPI           |
| R-67432         | Marić       |  6784.20 | R-67432         | BP1           |
+----------------+------------+---------+----------------+---------------+
```

  - `table_name.att_name` se može koristiti za sve stupce
  - takva vrsta join-a se zove equijoin
  - baze podataka su optimizirane za efikasno izvršavanje equijoin

Preimenovanje tablica

- u `FROM` dijelu se može specificirati drugo ime za tablicu
  - sintaksa *`table AS name`*
  - `AS` ključna riječ se može izostaviti
- primjer:

  pronađite nastavnika s najvišim primanjima!
  - prvi korak je pronaći nastavnike s primanjima koja su manja od drugih iznosa primanja
  - za to koristimo Kartezijev produkt i operaciju preimenovanja

```
SELECT DISTINCT nastavnik.nastavnik_sifra
      FROM nastavnik, nastavnik AS prof
      WHERE nastavnik.primanja < prof.primanja;
```

Preimenovanje tablice

- kada se tablica preimenuje u `FROM` dijelu, novo ime se može koristiti u `SELECT` i `WHERE` dijelu naredbe
- promjena imena se odnosi samo na izraz u kojem je navedena
- korištenjem preimenovanja može se smanjiti veličina izraza

```
SELECT DISTINCT n.nastavnik_sifra
     FROM nastavnik AS n, nastavnik AS p
     WHERE n.primanja < p.primanja;
```

Operacije sa skupovima

- SQL omogućava operacije skupova, kao u relacijskoj algebri
- operacije na ulazu primaju dva upita i kao rezultat daju novu relaciju
- unija:

  $select_1$ **UNION** $select_2$;

- presjek:

  $select_1$ **INTERSECTION** $select_2$;

- razlika:

  $select_1$ **EXCEPT** $select_2$;

- $select_i$ je zasebni `SELECT` izrazi

Dodavanje tablice student

- dodajmo tablicu student

```
CREATE TABLE student(
    student_sifra CHAR(6),
    prezime VARCHAR(30),
    godina INT,
    PRIMARY KEY (student_sifra)
);
```

- i studente:

```
INSERT INTO student VALUES ('000001', 'Marković', 1);
INSERT INTO student VALUES ('000002', 'Modrić', 3);
INSERT INTO student VALUES ('000003', 'Horvat', 1);
INSERT INTO student VALUES ('000004', 'Marić', 4);
```

Operacije sa skupovima primjer

- pronađite imena svih osoba vezanih uz sveučilište

```
SELECT prezime FROM nastavnik UNION
SELECT prezime FROM student;
```

  - baza podataka briše duplikate

- pronađite nastavnike čije prezime nema niti jedan student

```
SELECT prezime FROM nastavnik EXCEPT
SELECT prezime FROM student;
```

  - radi čitljivosti mogu se koristiti zagrade

```
(SELECT prezime FROM nastavnik) EXCEPT
(SELECT prezime FROM student);
```

MYSQL ne podržava EXCEPT!

Operacije na skupovima i duplikati

- operacije na skupovima SQL jezika uklanjaju duplikate n-torki iz rezultata
  - u suprotnosti s ponašanjem naredbe `SELECT`
- duplikati se mogu zadržati ako se doda ključna riječ `ALL` svakoj operaciji:

```
select1 UNION ALL select2;
select1 INTERSECTION ALL select2;
select1 EXCEPT ALL select2;
```

Koliko duplikata?

- definirajmo ponašanje operacija na skupovima na multiskupovima
- ako su zadane dvije multiset relacije r_1 i r_2
  - r_1 i r_2 imaju istu shemu
  - neka n-torka t se pojavljuje c_1 puta u r_1 i c_2 puta u r_2

$r_1 \cup_{ALL} r_2$
?

$r_1 \cap_{ALL} r_2$
?

$r_1 -_{ALL} r_2$
?

Koliko duplikata?

- definirajmo ponašanje operacija na skupovima na multiskupovima
- ako su zadane dvije multiset relacije r_1 i r_2
  - r_1 i r_2 imaju istu shemu
  - neka n-torka t se pojavljuje c_1 puta u r_1 i c_2 puta u r_2

$r_1 \cup_{ALL} r_2$
sadrži $c_1 + c_2$ kopija t

$r_1 \cap_{ALL} r_2$
sadrži $min(c_1, c_2)$ kopija t

$r_1 -_{ALL} r_2$
sadrži $max(c_1 - c_2, 0)$ kopija t

Oblikovanje SQL koda

- vrlo je bitno, radi čitljivosti, koristiti jedan stilova oblikovanja koda
- ne postoji jedinstveno pravilo imenovanja i oblikovanja, najvažnije je konzistentno se držati odabranog stila
- neke preporuke:
  - koristiti malo početno slovo za tablice, stupce, …
  - koristiti jedninu prilikom imenovanja tablica
  - što češće dodavati komentare u kod
  - pisati SQL ključne riječi velikim tiskanim slovima
  - pratiti neku do shema uvlaka koda
  - ne pisati dugačke linije koda

Oblikovanje SQL koda

- primjer dobro oblikovanog koda
  - pisanje SQL ključnih riječi velikim slovima
  - tri razmaka nakon SELECT-a
  - svi elementi naredbe osim ključnih riječi poravnati prema tom razmaku
  - preimenovati tablice u kratke nazive
  - stvoriti logiku stvaranja kratkih imena tablica i pratiti istu
  - id kolona u tablici ne sadrži ime tablice
  - stanji ključ sadrži i naziv tablice

```
SELECT    ss.id 'Id'
,         cy.name 'Retailer country'
,         od.name 'Order method'
,         re.name 'Retailer type'
,         pe.name 'Product type'
,         pt.name 'Product'
,         ss.year 'Year'
,         ss.quarter 'Quarter'
,         ss.revenue 'Revenue'
,         ss.quantity 'Quantity'
,         ss.gross_margin 'Gross margin'
FROM      pentaho.product pt
,         pentaho.product_type pe
,         pentaho.sales ss
,         pentaho.order_method od
,         pentaho.retailer_type re
,         pentaho.country cy
WHERE     pt.product_type_id = pe.id
AND       ss.product_id = pt.id
AND       ss.order_method_id = od.id
AND       ss.retailer_type_id = re.id
AND       ss.country_id = cy.id
ORDER BY  ss.id ASC
LIMIT     100;
```

lakša kontrola ispravnosti

Zadatak

- dana je baza podataka osiguravajućeg društva za auto-odgvornost sa slijedećom shemom *[DSC] – 3.10*.

```
person (driver_id, name, address)
car (license, model, year)
accident (report_number, date, location)
owns (driver_id, license)
participated (report_number, license, driver_id, damage_amount)
```

- na temelju zadane sheme riješite sljedeće zadatke

Zadatak

- ispišite ime i adresu svih osoba u bazi podatka

```
SELECT name, address FROM person;
```

- pronađite sva različita imena osoba u bazi podataka

```
SELECT DISTINCT name FROM person;
```

- pronađite sve osobe s prezimenom Grgić

```
SELECT * FROM person WHERE name = 'Grgić';
```

  - različitim imenom od Grgić

    <> ili !=

Zadatak

- pronađite sve štete veće od 5000.00

```
SELECT * FROM participated WHERE damage_amount > 5000.00;
```

- pronađite sve štete manje od 3000.00 i veće od 8000.00

```
SELECT * FROM participated
   WHERE damage_amount < 3000.00 AND damage_amount > 8000.00;
```

- pronađite sve štete manje od 3000.00 i veće od 8000.00 u kojima je sudjelovala osoba s *vozačkim id*-em 0445678

```
SELECT * FROM participated
   WHERE damage_amount BETWEEN 3000.00 AND 8000.00
   AND driver_id = '0445678';
```

Zadatak

- pronađite sve osobe čiji ime počinje s 'Ma'

```
SELECT * FROM person WHERE name LIKE 'Ma%';
```

- pronađite sve osobe čije je treće solo u imenu 'z'

```
SELECT * FROM person WHERE name LIKE '__z%';
```

- pronađite id osobe čije ime počinje s 'Ma' i završava s 'ić'

```
SELECT * FROM person
   WHERE name LIKE 'Ma%' AND name LIKE '%ić';
```

Zadatak

- ispišite sve moguće kombinacije imena osoba i modela auta koji su pohranjeni u bazi podataka

```
SELECT person.name, car.model FROM person, car;
```

- ispišite imena osoba i njihovih automobila pohranjenih u bazi podataka

```
SELECT person.name, car.model FROM person, car, owns
   WHERE person.driver_id = owns.driver_id
   AND car.license = owns.license;
```

Zadatak

- ispišite imena osoba, model automobila i iznos štete svih nesreća koje su spremljene u bazi

```
SELECT person.name, car.model, participated.damage_amount
    FROM person, car, participated
    WHERE person.driver_id = participated.driver_id
    AND car.license = participated.license;
```

- napišite upit koristeći skraćena imena tablica (alias)

```
SELECT pn.name, cr.model, pd.damage_amount AS 'amount'
    FROM person AS pn, car AS cr, participated AS pd
    WHERE pn.driver_id = pd.driver_id
    AND cr.license = pd.license;
```

Zadatak

- ispišite imena vlasnika automobila (koji je sudjelovao u nesreći), model automobila i iznos štete svih nesreća koje su spremljene u bazi

```
SELECT pn.name, cr.model, pd.damage_amount
   FROM person AS pn, car AS cr, participated AS pd, owns AS os
   WHERE pn.driver_id = os.driver_id
   AND cr.license = os.license
   AND cr.licnese = pd.license;
```

  - u prijašnjem upitu je vozač mogao voziti posuđeni automobil
  - stvara se veza između osobe i automobila preko tablice vlasništva
  - razlika ako tražimo imena vlasnika automobila koji je sudjelovao u nesreći?

Zadatak

- pronađite sve lokacije na kojima je osoba s prezimenom 'Ivanković' imala prometnu nesreću
  - atribut *name* sadrži ime i prezime osobe

```
SELECT DISTINCT at.location
    FROM person AS pn, participated AS pd, accident AS at
    WHERE pn.driver_id = pd.driver_id
    AND pd.report_number = at.report_number
    AND pn.name LIKE '%Ivanković';
```

  - koristi se `DISTINCT` jer želimo izbaciti duplikate iz rezultata

Zadatak

- objasnite upite

```
(SELECT course id FROM section
   WHERE semester = 'Fall' AND year= 2009)
INTERSECT
(SELECT course id FROM section
   WHERE semester = 'Spring' AND year= 2010);
```

  - skup svih kolegija koji su se podučavali u jesenskom semestru 2009 i proljetnom 2010-te

```
(SELECT course id FROM section
   WHERE semester = 'Fall' AND year= 2009)
EXCEPT
(SELECT course id FROM section
   WHERE semester = 'Spring' AND year= 2010);
```

  - kolegiji koji su se podučavali u jesenskom semestru 2009 ali ne u proljetnom 2010-te

Slijedeće predavanje

- sortiranje rezultata
- grupiranje i agregirane funkcije
- ugniježđeni upiti i druge operacije na skupovima
- update SQL baza podataka

Literatura

- Pročitati
  - [DSC] poglavlje 3.1 – 3.5.
  - Caltech CS121 – P4
- Slijedeće predavanje
  - [DSC] poglavlje 3.6. – 3.10.
  - [DSC] poglavlje 4.1. – 4.2.
  - Caltech CS121 – P5, P6