Uvod u SQL kroz primjere

1. Vrsta podataka

Prilikom definisanja relacije treba specificirati vrstu ili tip atributa. Atribut može biti datum, broj, neko ime, tekst, internet broj računara, logička veličina tipa istina/laž, ili nešto drugo. Navedimo nekoliko važnijih i češće korištenih tipova atributa:

INT
: Cijeli broj - tipično 4 byte-a, ali to zavisi i o DBMSu.

BIGINT
: Cijeli broj spremljen u 8 byte-ova.

SMALLINT
: Cijeli broj spremljen u 2 byte-ova.

REAL
: Decimalni broj spremljen u 4 byte-a

NUMERIC (p,s)
: Decimalni proizvoljne preciznosti

BOOLEAN
: Istina/laž (TRUE/FALSE, t/n, y/n, 1/0, ...)

CHAR(n)
: String - niz slova/brojeva/znakova fiksne dužine n.

VARCHAR(n)
: String - niz slova/brojeva/znakova maksimalne dužine n.

TEXT
: String proizvoljne dužine (MEMO)

DATE
: Datum, npr. 2002-10-21

TIME
: Vrijeme, npr. 04:05:06

TIMESTAMP
: Datum+Vrijeme (1999-01-08 04:05:06)

MONEY
: Novac

Veličine kao što su datumi, vremena, nizovi znakova, tj sve nebrojčane veličine omeđuju se sa jednostrukim navodnim znacima, npr. 'Ovo je string', '1999-01-08', itd.
SQL naredbe mogu ići preko nekoliko linija a završavaju/terminiraju se s tačka-zarezom ili s sa \g

2. Definisanje relacije

Relacija se definiše sa SQL naredbom CREATE TABLE, iza koje slijedi ime relacije, te u zagradama navođenjem liste definicija atributa međusobno odvojenih zarezom. Atributi se definišu s imenom atributa iza čega slijedi specifikacija vrste atributa i ostalih svojstava atributa.
Primjer definisanja relacije student:

```
CREATE TABLE student (
        prezime VARCHAR(50),
        ime     VARCHAR(50),
        indeks  VARCHAR(10),
        godina  INT,
        smjer   VARCHAR(10),
        PRIMARY KEY (indeks)
        );
```

1. Primjer definisanja relacije *student*

Na kraju relacije definisana je primarnih lista koji čine ključ u relaciji. U drugom primjeru definišemo relaciju koja sadrži podatke o kolegijima:

```
CREATE TABLE kolegij (
        ime     VARCHAR(50),
        prezime VARCHAR(50),
        kid     INT,
        naslov  VARCHAR(100),
        sati1   INT,
        vjezbi1 INT,
        sati2   INT,
        vjezbi2 INT,
        PRIMARY KEY(kid)
        );
```

2. Primjer još jedna definicija relacije

U trećem primjeru navodimo primjer definisanja relacije predavanje u koji upisujemo podatke o studentima koji slušaju ili su slušali predavanja iz nekog kolegija.

```
CREATE TABLE predavanje (
        kid     INT REFERENCES kolegij (kid),
        indeks  VARCHAR(10) REFERENCES student (indeks),
        ak_godina VARCHAR(10) NOT NULL,
        iid     INT REFERENCES ispit (iid),
        PRIMARY KEY(kid,indeks)
        );
```

3. Primjer definisanja relacije *predavanje* i upotrebe reference na druge relacije

Atributi kid, indeks i iid **referenciraju** se na druge prethodno definisane relacije i atribute u njima definisane. SQL instrukcija REFERENCES kaže da navedeni atribut može imati samo vrijednost koja se pojavljuje u referentnoj relaciji ili biti nedefinisan (NULL). Svrha ove instrukcije je čuvanje konzistencije podataka. Ne bi imalo smisla upisivati podatke u relaciju za nepostojećeg studenta ili nepostojeći kolegij.
Relacija se briše iz baze podataka naredbom DROP TABLE <ime_relacije>.

3. Upis podataka
Novi podaci se upisuju u relaciju SQL naredbom INSERT INTO, iza koje slijedi ime relacije te opcionalno lista imena atributa u zagradama, a zatim riječ VALUES iza koje slijedi lista vrijednosti atributa u zagradama Pogledajmo primjer upisa studenata u relaciju student:

```
INSERT INTO student VALUES ('PETROVIĆ','ANA','F-1961',2,'pfi');
INSERT INTO student VALUES ('MARIĆ','VLATKA','F-1761',2,'pfi');
INSERT INTO student VALUES ('BABIĆ','KOSTA','F-2523',1,'pfi');
....
```

4. Primjer upisa podataka u relaciju *student*

ili

```
INSERT INTO student (prezime,ime,indeks,godina,smjer)
                          VALUES ('PETROVIĆ','ANA','F-1961',2,'pfi');
INSERT INTO student (prezime,ime,indeks,godina,smjer)
                          VALUES ('MARIĆ','VLATKA','F-1761',2,'pfi');
INSERT INTO student (prezime,ime,indeks,godina,smjer)
                          VALUES ('BABIĆ','KOSTA','F-2523',1,'pfi');
....
```

5. Primjer upisa podataka s navođenjem imena atributa

Listu imena atributa nije potrebno navoditi ako lista vrijednost koja slijedi ide točno po slijedu definicije relacije. Tekstualne vrijednosti su omeđene jednostrukim navodnim znakom za razliku od brojčanih vrijednosti. Ako vrijednost za neki atribut nije navedena, atribut poprima pre definisanu vrijednost koja se specificira kod definicije relacije, ili NULL vrijednost (prazno).

4. Ažuriranje/mijenjanje podataka
Podaci se mijenjaju SQL naredbom UPDATE. Primjer:

Problem: *Promijeniti ime prof. K.Furiću koji predaje kolegij pod brojem 2362 u "Krešimir"!*

```
UPDATE kolegij SET ime='Krešimir' WHERE kid=2362;
```

6. Primjer izmjenje postojećih podataka

Iza riječi SET dolazi lista atributa koji se mijenjaju međusobno odvojeni zarezima.

5. Brisanje podataka
Podaci se brišu naredbom DELETE FROM. Primjer:

Problem: *Izbrišimo izborni kolegij "Informatika" (2404) iz studija PFI!*

```
DELETE FROM studij_pfi WHERE kid=2404;
```

7. Primjer brisanja upisanih podataka

6. Transakcije

Nekad je potrebno učiniti više vezanih izmjena/nadopuna podataka, a njih nije moguće napraviti s jednom UPDATE naredbom. Npr. treba izmjeniti podatke u više različitih relacija. Pri tome, jedna izmjena nema smisla, i treba je poništiti ako nije moguće sve promjene učiniti. Takav niz vezanih SQL komandi zove se transakcija. Transakcija počinje s SQL naredbom BEGIN, te nakon toga slijedi niz SQL naredbi s kojima se mijenjaju ili pregledavaju podaci.
Taj niz naredbi treba završiti s SQL naredbom COMMIT ili ako se želi poništiti prethodni niz naredbi, onda ROLLBACK.

```
BEGIN TRANSACTION;
  UPDATE student SET indeks='F-3342' WHERE indeks='F-3343';
  UPDATE predavanje SET indeks='F-3342' WHERE indeks='F-3343';
ROLLBACK TRANSACTION;

ili

BEGIN TRANSACTION;
  DELETE FROM student WHERE indeks='F-3343';
  INSERT INTO student VALUES ('Nikić','Nikša','F-3342',4,'pfi'),
  UPDATE predavanje SET indeks='F-3342' WHERE indeks='F-3343';
COMMIT TRANSACTION;
```

8. Primjer transakcije

7. Postavljanje upita

Upit se postavlja naredbom SELECT. (Naglasimo da operacija selekcije u relacijskoj algebri ne odgovara u potpunosti SQL naredbi SELECT; ona je bliže operaciji projekcije koja bira određene atribute koji će biti prikazani.) Rezultat upita uopšteno nije relacija već tablica, jer n-torke u rezultatu ne moraju biti sve različite.
Da bi se dobile sve različite n-torke treba koristiti SELECT DISTINCT naredbu.

Problem: *Pronađi brojeve indeksa i imena svih studenata prve godine!*

```
SELECT indeks,ime,prezime FROM student WHERE godina=1;
```

9. Primjer jednostavnog upita

ili ako se želi dobiti sortirani ispis:

```
SELECT indeks,ime,prezime FROM student WHERE godina=1
      ORDER BY prezime,ime;


 indeks  |     ime      |   prezime
---------+--------------+-------------
 F-2523  | KOSTA       | BABIĆ
 F-2506  | KREŠIMIR     | BAČIĆ
 F-2271  | DAMIR        | BAKMAZ
 F-2143  | TIBOR        | BALI
 F-2144  | IVAN         | BEDNAJNEC
 F-2356  | BRUNO        | BLAŽINČ
 .....
 F-2561  | DEJAN        | ŽIKOVIĆ
 (69 rows)
```

10. Primjer sa sortiranjem rezultata

Problem: *Pronađi brojeve indeksa i imena studenata koji su upisali kolegij broj 1224!*

```
 SELECT student.ime,student.prezime,student.indeks
       FROM student,predavanje
       WHERE student.indeks=predavanje.indeks AND predavanje.kid=1224
       ORDER BY student.prezime;


   ime    |   prezime    | indeks
----------+--------------+--------
 AMIR     | EL-OCH       | F-2025
 IVAN     | GLADOVIĆ     | F-1823
 IVA      | BAŠIĆ        | F-2291
 MARKO    | JOVIČIĆ      | F-2288
 ANA      | JURIĆ        | F-1937
 NENAD    | KARLOVČEC    | F-1872
 MARKO    | KLOKOČKI     | F-1851
 MARIN    | KOSOVIĆ      | F-1830
 VEDRAN   | KRALJ        | F-1972
 ...
 ZRINKA   | ŠUMANOVAC    | F-1789
 KREŠIMIR | BASTA       | F-2023
(17 rows)
```

11. Primjer složenijeg upita koji kombinira dvije relacije

Vidimo da smo morali koristiti **proširena** imena atributa koja sadrže i imena relacija u kojima se pojavljuju. Naredba je SELECT napravila spajanje dviju relacija, te je iz nove relacije izdvojila tražene n-torke.
U stvari, nanizane relacije u FROM dijelu upita predstavljaju *virtuelnu* relaciju koja je njihov Kartezijev produkt. Ako WHERE dijela upita ne bi bilo, dobili bi sve moguće kombinacije n-torki iz prve relacije s n-torkama iz druge relacije.

Ali do traženog rezultata se je moglo doći i drugačije:

```
 SELECT ime,prezime,indeks FROM student
       WHERE indeks IN (SELECT indeks FROM predavanje WHERE kid=1224)
       ORDER BY prezime;
```

Primjer 8a. alternativnog upita iz prethodnog primjera uz upotrebu podupita

Ovdje se koristi konstrukcija WHERE *atribut* IN (....), gdje se u zagradi navodi lista vrijednosti koje navedeni atribut mora imati. IN znači skupovno pripadanje.
Umjesto eksplicitnog listanja mogućih vrijednosti moguće je koristiti drugu SELECT naredbu koja će tu listu stvoriti. Dakle radi se o ugnježdenoj SELECT neredbi - SELECT unutar drugog SELECTa.

Postoji još jedna alternativa

```
 SELECT ime,prezime,indeks FROM student NATURAL JOIN predavanje
        WHERE kid=1224 ORDER BY student.prezime;
```

Primjer 8b. još jedna alternativa uz upotrebu *prirodnog spoja*

uz upotrebu **prirodnog spajanja** (spoja) dviju relacija.

Problem: *Pronađi brojeve indeksa i imena studenata koji su upisali bar jedan kolegij koji predaje profesor Androić!*
Kao i u prethodnom primjeru imamo više alternativa, jednu s Kartezijevim produktom triju relacija te drugu s ugnježdenim SELECT naredbama:

```
SELECT student.ime,student.prezime,student.indeks
   FROM student,predavanje,kolegij
   WHERE student.indeks=predavanje.indeks
   AND predavanje.kid=kolegij.kid AND kolegij.prezime='Androić';

ili

SELECT ime,prezime,indeks FROM student
   WHERE indeks IN ( SELECT indeks FROM predavanje
            WHERE kid IN (SELECT kid FROM kolegij WHERE prezime='Androić')
                  );


   ime    |   prezime    | indeks
----------+--------------+--------
 ANA      | PETROVIĆ     | F-1961
 VLATKA   | MARIĆ    | F-1761
 IVANA    | BILIĆ        | F-1813
 HRVOJE   | BOGNAR       | F-1884
 ZVONIMIR | BREZOVEC     | F-2175
 ....
 MARIJAN  | VRBANČIĆ     | F-1948
 KREŠIMIR | BASTA       | F-2023
 HANA     | ZLONOGA      | F-2278
 ANA      | ZELENIKA       | F-1862
 (48 rows)
```

12. Primjer složenog upita s koji kombinira tri relacije

Problem: *Pronađi sve parove brojeve indeksa studenata koji su na istoj godini!*

```
SELECT t1.indeks,t2.indeks FROM student t1,student t2
   WHERE t1.indeks < t2.indeks AND t1.godina=t2.godina;


 indeks  | indeks
---------+---------
 F-2523  | F-2556
 F-2523  | F-2571
 F-2523  | F-2531
 F-2523  | F-2561
 F-2506  | F-2523
 F-2506  | F-2522
 F-2506  | F-2556
....
 F-2023  | F-2025
 F-2023  | F-2288
 F-2023  | F-2284
(2947 rows)
```

13. Primjer upotrebe *aliasa*

Ovdje smo uveli *aliase* ili tz. druga imena, t1 i t2, za relaciju student. Upotreba aliasa je bila potrebna jer se radi o Kartezijevom produktu relacije student sa samom sobom kada može doći do konfuzije oko imena atributa. Ispred imena aliasa može se po volji dodati riječ AS.

Problem: *Pronađi sve podatke o studentima koji **nisu** upisali kolegij broj 1224!*

```
SELECT * FROM student WHERE indeks NOT IN
   (SELECT indeks FROM predavanje where kid=1224);


   prezime   |     ime      | indeks  | godina | smjer
-------------+--------------+---------+--------+-------
 PETROVIĆ     | ANA         | F-1961  |      2 | pfi
 MARIĆ  | VLATKA       | F-1761  |      2 | pfi
 BABIĆ        | KOSTA       | F-2523  |      1 | pfi
 BAČIĆ        | KREŠIMIR     | F-2506  |      1 | pfi
 ....
 ZUBČIĆ       | IVAN         | F-2345a |      1 | pfi
 ŽIKOVIĆ      | DEJAN        | F-2561  |      1 | pfi
 ZELENIKA      | ANA          | F-1862  |      2 | pfi
(100 rows)
```

14. Primjer upotrebe zvjezdice umjesto navođenja liste atributa

Svi podaci znači svi atributi, a umjesto navođenja svih imena atributa može se koristiti zvjezdica, *****.

Problem: *Pronađi brojeve indeksa onih studenata koji su upisali sve kolegije koje je upisao i student F-1862!*

Ovdje je očito potrebno naći rezultat dijeljenja relacije predavanje[indeks,kid] s relacijom koja sadrži listu brojeva kolegija koje je upisao student s indeksom F-1862. Neka je S relaciju koja se traži, relacija s listom kolegija koje je upisao student F-1862 neka je T, te relaciju s listom upisanih kao P. Tada je:

```
Tada je S [indeks] = P [indeks,kid] devide by T [kid].
```

S druge strane djeljenje se može prikazati preko drugih operacija (Kartezijev produkt i razlika):

```
 S[indeks] = P[indeks] minus ((P[indeks] x T[kid]) minus
P[indeks,kid])[indeks]
```

Pa se SQL upit može prikazati ovako:

```
SELECT DISTINCT indeks FROM predavanje                    -- P[indeks]
   EXCEPT (SELECT DISTINCT t3.indeks                      -- minus (
         FROM (SELECT t1.indeks,t2.kid                    -- (P[indeks] x
T[kid]
                  FROM predavanje t1,
                       (SELECT kid FROM predavanje WHERE indeks='F-1862')
t2
                  EXCEPT                                  -- minus
                     (SELECT indeks,kid FROM predavanje) -- P[indeks,kid])
              ) t3                                        -- ) [indeks]
          );

 indeks
--------
 F-1761
 F-1789
 F-1813
 F-1823
 ....
 F-2570
```

```
 F-2577
 F-2579
 F-2580
(48 rows)

ili

SELECT indeks FROM (SELECT DISTINCT indeks FROM predavanje) t1
       WHERE NOT EXISTS (
          (SELECT kid FROM predavanje t2 WHERE t2.indeks='F-1862')
           EXCEPT
          (SELECT kid FROM predavanje t3 WHERE t3.indeks=t1.indeks)
                        );
```

15. Primjer upotrebe EXCEPT (očekivati sporiji odgovor računara!)

Ovaj se problem može riješiti drukčije. Naime, može se napraviti razlika između skupa kolegija koje je upisao student 'F-1862' i skupa kolegija koje je upisao neki drugi student. Ako je razultat prazni sklup onda to znači da je taj student upisao sve kolegije koje je upisao i student s indeksom 'F-1862' (i možda još neke). To smo iskoristili u ovom drugom rješenju. Pri tome smo koristili konstrukciju EXISTS koja je istina uvijek za skup s barem jednim elementom (n-torkom), odnosno NOT EXISTS za prazan skup.

Problem: *Naći ocjenu iz kolegija Osnove fizike 3 koju je dobio student BASTA!*

```
SELECT ocjena FROM ispit WHERE iid IN
  (SELECT iid FROM predavanje
     WHERE kid IN (SELECT kid FROM kolegij WHERE naslov='Osnove fizike 3')
     AND indeks IN (SELECT indeks FROM student WHERE prezime='BASTA'));


 ocjena
--------
      4
(1 row)

ili

SELECT student.ime,student.prezime,ispit.ocjena
   FROM ispit,predavanje,kolegij,student
     WHERE student.indeks=predavanje.indeks
     AND ispit.iid=predavanje.iid AND predavanje.kid=kolegij.kid
     AND kolegij.naslov='Osnove fizike 3' AND student.prezime='BASTA';
```

14. Primjer kombiniranja upita s četiri relacija

Problem: *Naći imena studenata koji su iz kolegija "Osnove fizike 3" dobili ocjenu veću od 3!*

```
SELECT student.ime,student.prezime,ispit.ocjena
   FROM ispit,predavanje,kolegij,student
   WHERE student.indeks=predavanje.indeks
   AND ispit.iid=predavanje.iid AND predavanje.kid=kolegij.kid
   AND kolegij.naslov='Osnove fizike 3' AND ispit.ocjena>3;


   ime    |  prezime   | ocjena
----------+------------+--------
 AMIR     | EL-OCH     |      5
 IVA      | BAŠIĆ      |      5
 MARKO    | JOVIČIĆ    |      4
 MARKO    | KLOKOČKI   |      4
 VEDRAN   | KRALJ      |      5
 IVAN     | LESIĆ      |      4
 GORAN    | MAŠIĆ |      5
 ANA      | PARTALO    |      5
 KREŠIMIR | BASTA     |      4
(9 rows)
```

15. Primjer još jedan složeni slučaj upita

Ako već znamo kako doći do ocjena dobivenim na ispitu, zašto ne izračunati srednju ocjenu ? Za to se služimo AVG funkcijim. AVG spada u tz. agregatne funkcije (ili funkcije skupljanja) koje operišu na cijeloj relaciji, ili njenom dijelu, te daju **jedan** završni izraz ili broj za skup svih n-torki ili pojedine grupe n-torki. (Druge agregatne funkcije su maksimalna i minimalna vrijednost, MAX i MIN.)

Problem: *Naći srednju ocjenu iz kolegija Osnove fizike 3 koju su dobili studenti treće godine!*

```
SELECT AVG(ocjena),MAX(ocjena),MIN(ocjena),COUNT(*)
   FROM (SELECT ocjena FROM ispit
            WHERE iid IN (SELECT iid FROM predavanje
                    WHERE kid IN (SELECT kid FROM kolegij
                                    WHERE naslov='Osnove fizike 3')
                    AND indeks IN (SELECT indeks FROM student
                                    WHERE godina=3))) t;


     avg      | max | min | count
--------------+-----+-----+-------
 3.5294117647 |   5 |   2 |    17
(1 row)

ili

SELECT AVG(ocjena),MAX(ocjena),MIN(ocjena),COUNT(*)
   FROM (SELECT student.ime,student.prezime,ispit.ocjena
        FROM ispit,predavanje,kolegij,student
        WHERE student.indeks=predavanje.indeks
        AND ispit.iid=predavanje.iid AND predavanje.kid=kolegij.kid
        AND kolegij.naslov='Osnove fizike 3' AND student.godina=3) t;
```

16. Primjer upotrebe funkcija u upitu

COUNT funkcija izračunava broj n-torki koje se uzete u proračun. Zvjezdica znači uračunati sve n-torke, dok bi COUNT(ocjena) uračunati samo one n-torke gdje ocjena nije jednaka NULL (nije definisana, štoje različito od numeričke vrijednosti nula). Napomena: Agregatne funkcije se ne mogu

rabiti u WHERE dijelu SQL upita, jer upravo WHERE dio služi za izdvajanje n-torki koje će ući u proračun funkcije.

Problem: *Naći imena i godinu studenata kojima prezime počinje s slovom 'P'!*

Da bi se ovaj problem mogao riješiti treba koristiti tz. *pravilni/regularni izraz* ili engleski *regular expression*. Regulari izraz je niz znakova slaganih po određenom pravilu koji zamjenjuju grupe slova, riječi ili cijele tekstove. SQL regularni izrazi, mogu sadržavati dva znaka, znak procenta % ili podvučenu crtu _.
Znak procenta zamjenjuje od nula pa naviše proizvoljnih slova ili brojki, dok podvočena crta zamjenjuje samo jedno slovo, brojku ili znak. Unix regularni izrazi su dosta bogatiji ali i složeniji u upotrebi.

```
SELECT ime,prezime,godina FROM student WHERE prezime ~ '^P';   -- unix
(posix)

ili

SELECT ime,prezime,godina FROM student WHERE prezime LIKE 'P%'; -- SQL


   ime    | prezime  | godina
----------+----------+--------
 ANA      | PARTALO  |      3
 LUCIJA   | PAVIĆ    |      1
 TOMISLAV | PAVKOVIĆ |      1
 MARKO    | PAVLINEK |      2
 AMALIJA  | PERKOVIĆ |      1
 HRVOJE   | PETKOVIĆ |      1
 BRUNO    | PLANČIĆ  |      2
 DEJAN    | POPOVIĆ  |      2
 ŽELJKO   | PUŠEC    |      1
(9 rows)
```

17. Primjer upotrebe unix i SQL *regular expression* u upitu

Kapica ispred slova 'P' u posix regularnom izrazu znači početak vrijednosti atributa. Kombinirano '^P' znači slovo P na početku atributa. O ostatku se ništa ne kaže pa može biti bilo što. U SQL pravilnom izrazu taj ostatak treba nadomjestiti znakom procenta. Jer se u LIKE poređenje uspoređuje iznos cijelog atributa.

Evo jednog primjera gdje se to još jasnije pokazano.

Problem: *Naći imena studenata koji imaju u svom imenu slovo 'Š'!*

```
SELECT ime,prezime,godina FROM student WHERE prezime LIKE '%Š%';

ili

SELECT ime,prezime,godina FROM student WHERE prezime ~ 'Š';


   ime    |   prezime   | godina
----------+-------------+--------
 EUGEN    | OREŠKOVIĆ   |      1
 ŽELJKO   | PUŠEC       |      1
 MARTINA  | REBERNIŠĆAK |      1
 TOMISLAV | ŠANTEK      |      1
```

```
 ANA      | ŠIMUNDŽA    |      1
 DALIBOR  | ŠIMUNIĆ     |      2
 IVAN     | ŠTAMBAK     |      1
 MARKO    | ŠTEFANEC    |      2
 ZRINKA   | ŠUMANOVAC   |      3
 DAMIR    | ŠVELEC      |      1
 DEJAN    | VRANEŠEVIĆ  |      1
(11 rows)
```

18. Primjer još jedan primjer korištenja *regular expression*

Poblem: *Naći imena studenata kojima su brojevi indeksa između 1500 i 1799!*

```
SELECT ime,prezime,indeks FROM student WHERE indeks LIKE 'F-15__'
                                       OR    indeks LIKE 'F-16__'
                                       OR    indeks LIKE 'F-17__';

ili

SELECT ime,prezime,indeks FROM student WHERE indeks ~ 'F-1[5-7]';

ili

SELECT ime,prezime,indeks FROM student WHERE indeks ~ 'F-1[5-7][0-9]{2}';


  ime   |  prezime   | indeks
--------+------------+--------
 VLATKA | MARIĆ | F-1761
 ZRINKA | ŠUMANOVAC  | F-1789
(2 rows)
```

19. Primjer još jedan primjer korištenja *regular expression*

Skript jezici Perl i PHP imaju još bogatija i složenija pravila za pravilne izraze.

8. Kontrola sigurnosti

Naredbe GRANT and REVOKE služe za kontrolu pristupa podacima od strane korisnika baze podataka. Pristup se definišena nivou pojedinih relacije za razne vrste operacija kao sto su SELECT, INSERT, UPDATE, DELETE itd bilo pojedinačnom korisniku ili grupi. Primjeri:

Problem: *Dopustiti SELECT svima za relaciju kolegij!*

```
GRANT SELECT ON kolegij TO PUBLIC;
```

20. Primjer davanja prava na upit svima

Problem: *Dopustiti korisniku peri select, insert i update za relaciju adresa!*

```
GRANT SELECT, INSERT, UPDATE ON adresa TO pero;
```

21. Primjer davanja prava na ažuriranje podataka jednom korisniku

Problem: *Dopustiti korisniku peri sve moguće operacije!*

```
GRANT ALL PRIVILEGES ON adresa TO pero;
```

22. Primjer davanja **svih** prava jednom korisniku

Problem: *Poništimo pravo za sve operacije grupi student za relaciju adresa!*

```
REVOKE ALL PRIVILEGES ON adresa TO GROUP student;
```

23. Primjer poništenje prava danih grupi korisnika

24,
Šta je rezultat upita

```
SELECT * FROM Sellers
WHERE Deposit BETWEEN 200 AND 300;
```

primjenjenog na relaciju-tabelu Sellers:

SellerID	SellerName	City	Country	Withdrawal	Deposit
1	Caroline Diana	Dallas	USA	100	235.45
2	Jitinder Jeetendra	Mumbai	India	110	275.67
3	Kanti Kapila	Mumbai	India	120	645.43
4	Lachlan James	Mumbai	India	105	561.69
5	Cooper Alexander	Perth	India	350	745.31
6	Ryan Max	Quensland	Australia	120	361.24
7	Joshua Jayden	Perth	Australia	130	538.12
8	Benjamin Oscar	Queensland	Australia	110	543.16
15	Aiguo An	Shanghai	China	60	102.29
26	Brian Federico	Mendoza	Argentina	80	282.11

odgovor

SellerID	SellerName	City	Country	Withdrawal	Deposit
1	Caroline Diana	Dallas	USA	100	235.45
2	Jitinder Jeetendra	Mumbai	India	110	275.67
26	Brian Federico	Mendoza	Argentina	80	282.11

25
Šta je rezultat upita

```
SELECT * FROM Buyers
WHERE BuyerName='David Clinton' AND BuyerAddress='Blue Lake 123'
```

primjenjenog na relaciju-tabelu Buyers:

BuyerID	BuyerName	BuyerAddress	City	Country	Deposit
1	David Clinton	Blue Lake 123	New York	USA	435.45
2	Shea Roxanne	Santa Rose 123	Liverpool	England	234.12
3	Bryan de Lima	Corcovado Street	Rio de Janeiro	Brazil	534.22
4	Tjen Chun Yen	Zhonghua Street 456	Shanghai	China	234.12
5	Sheila White	Santa San Marine 456	New York	USA	234.12
6	Bryan Clinton	Blue Lake 123	Sidney	Australia	534.22
7	David Clinton	Blue Lake 123	Sidney	Australia	4361.87
8	Tjen Chun Yen	Blue Lake 123	Singapore City	Singapore	308.33
9	Sheila White	Santa San Marine 456	Sidney	Australia	435.45

odgovor

BuyerID	BuyerName	BuyerAddress	City	Country	Deposit
1	David Clinton	Blue Lake 123	New York	USA	435.45
7	David Clinton	Blue Lake 123	Sidney	Australia	4361.87

Šta je rezultat upita

```
a) SELECT *
       FROM Sellers
       WHERE City LIKE 'S%';

b) SELECT *
      FROM Sellers
        WHERE City LIKE '%da%';
```

primjenjenog na relaciju-tabelu Sellers:

SellerID	SellerName	City	Country	Withdrawal	Deposit
1	Caroline Diana	Dallas	USA	100	235.45
2	Jitinder Jeetendra	Mumbai	India	110	275.67
3	Kanti Kapila	Mumbai	India	120	645.43
4	Lachlan James	Mumbai	India	105	561.69
5	Cooper Alexander	Perth	India	350	745.31
6	Ryan Max	Quensland	Australia	120	361.24
7	Joshua Jayden	Perth	Australia	130	538.12
8	Benjamin Oscar	Queensland	Australia	110	543.16
15	Aiguo An	Shanghai	China	60	102.29
26	Brian Federico	Mendoza	Argentina	80	282.11

odgovor:
a)

SellerID	SellerName	City	Country	Withdrawal	Deposit
15	Aiguo An	Shanghai	China	60	102.29

b)

SellerID	SellerName	City	Country	Withdrawal	Deposit
1	Caroline Diana	Dallas	USA	100	235.45

---

primjer 27
Ako su date sljedeće relacione šeme:

Grad(Naziv, Drzava, BrojStanovnika)
Dvorana(DID, Kapacitet, Naziv *references* Grad)
Koncert(KID, JMBG *references* Izvodjac, Trajanje, DID *refences* Dvorana)
Izvodjac(JMBG, Ime, Adresa, Starost)
Ulaznica(UID, KID *references* Koncert, Cijena, Tip)

Napomena. Primarni ključevi su podvučeni. Atribut Ime u relacionoj šemi Dvorana je spoljnji ključ na relacionu šemu Grad i označava u kom gradu se nalazi dvorana. Atribut JMBG u relacionoj šemi Koncert je spoljnji ključ na relacionu šemu Izvođač i označava koji izvođač je nastupao na koncertu. U relacionoj šemi Koncert atribut DID je spoljnji ključ na relacionu šemu Dvorana i označava u kojoj dvorani je održan koncert.

a) **Izlistati ime izvođača, kao i broj različitih gradova u kojima je izvođač održao koncerte pod uslovom da je taj broj veći od 10.**

```
SELECT i.Ime, COUNT(DISTINCT d.Naziv) as BrGradova FROM Izvodjac i, Koncert k, Dvorana d
WHERE k.JMBG = i.JMBG AND k.DID = d.DID
GROUP BY i.Ime
HAVING BrGradova > 10
```

b) **Naći koncerte za koji je prodato više VIP ulaznica tipa nego ostalih karata. Napomena. VIP ulaznice su one karte čiji je Tip jednak VIP**.

```
SELECT k.* FROM Koncert k
WHERE (SELECT COUNT(*) FROM ULAZNICA u1
WHERE u1.KID = k.KID AND u1.Tip = 'VIP') > (SELECT COUNT(*) FROM Ulaznica u2
WHERE u2.KID = k.KID and u2.Tip != 'VIP')
```

c) **Naći izvođače koji nastupaju isključivo u dvoranama koje imaju kapacitet veći od kapaciteta najveće dvorane u Podgorici**.

```
SELECT i.* FROM Izvodjac i, Koncert k, Dvorana d
WHERE NOT EXISTS(SELECT * FROM Koncert k, Dvorana d
WHERE k.JMBG = i.JMBG AND k.DID = d.DID AND d.Kapacitet < (SELECT MAX(d1.Kapacitet)
FROM Dvorana d1
WHERE d1.Grad = 'Podgorica'))
```

d) **Naći parove JMBG, Naziv tako da izvođač koji je identifikovan atributom JMBG nikada nije održao koncert u gradu koji je identifikovan atributom Naziv**.

```
SELECT i.JMBG, g.Naziv FROM Izvodjac i, Grad g
WHERE NOT EXISTS(SELECT * FROM Koncert k, Dvorana d
WHERE k.JMBG = i.JMBG AND k.DID = d.DID AND d.Naziv = g.Naziv)
```

e) **Naći ime grada u kome je nastupao svaki izvođač**.

```
SELECT g.Naziv FROM Grad g
WHERE NOT EXISTS (SELECT * FROM Izvodjac i
WHERE NOT EXISTS (SELECT * FROM Koncert k, Dvorana d
WHERE k.JMBG = i.JMBG AND k.DID = d.DID AND d.Naziv = g.Naziv))
```

f) **Naći koncert za koji je prodato najviše ulazica.**

```
SELECT u.KID, COUNT(*) as ProdateUlaznice FROM Ulaznica u
GROUP BY u.KID
HAVING ProdateUlazince = (SELECT MAX(PrUlazn) FROM (SELECT COUNT(*) as PrUlazn FROM
Ulaznica u1
GROUP BY u1.KID))
```

g) **Naći države u kojima je nastupao izvođač koji je do sada imao najveći broj koncerata.**

```
SELECT g.Drzava FROM Grad g, Izvodjac i, Koncert k, Dvorana d
WHERE k.JMBG = i.JMBG AND k.DID = d.DID AND d.Naziv = g.Naziv AND (SELECT COUNT(*)
FROM Koncert k1
WHERE k1.JMBG = i.JMBG) >= ALL (SELECT BrKoncerata FROM (SELECT k2.JMBG, COUNT(*) as
BrKoncerata FROM Koncert k2 GROUP BY k2.JMBG)
```

h) **Naći grad takav da je u njemu održano više od deset koncerata i nijedan izvođač nije nastupao dva puta u tom gradu**.

```
SELECT d.Naziv, COUNT(*) FROM Dvorana d, Koncert k,
WHERE k.DID = d.DID AND NOT EXIST (SELECT i.JMBG, COUNT(*) FROM Izvodjac i, Koncert k1,
Dvorana d1
WHERE k1.JMBG = i.JMBG AND k1.DID = d1.DID AND d1.Naziv = d.Naziv
GROUP BY i.JMBG
HAVING COUNT(*) >= 2)
GROUP BY d.Naziv
HAVING COUNT(*) > 10
```

i) **Naći izvođača koji je imao najviše rasprodatih koncerata. Napomena. Koncert je rasprodat ako je broj prodatih ulaznica jednak kapacitetu dvorane u kojoj je održan koncert.**

```
SELECT k.JMBG, COUNT(*) from Koncert k, Dvorana d
WHERE k.DID = d.DID AND (SELECT COUNT(*) FROM Ulaznica u WHERE u.KID = k.KID) =
d.Kapacitet
GROUP BY k.JMBG
HAVING COUNT(*) = (SELECT MAX(COUNT(*)) FROM (SELECT k1.JMBG, COUNT(*) FROM
Koncert k1, Dvorana d1
WHERE k1.DID = d1.DID AND (SELECT COUNT(*) FROM Ulaznica u1 WHERE u1.KID = k1.KID) =
d1.Kapacitet)
GROUP BY k1.JMBG)
```

j) **Naći izvođača koji je u svakoj dvorani imao makar dva koncerta.**

```
SELECT i.* FROM Izvodjac i
WHERE NOT EXISTS (SELECT d.DID, COUNT(*) FROM Dvorana d, Koncert k WHERE d.DID = k.DID
AND k.JMBG = i.JMBG
GROUP BY d.DID HAVING COUNT(*) < 2) AND NOT EXISTS (SELECT d1.DID FROM Dvorana d1
WHERE NOT EXISTS(SELECT * FROM Koncert k1 WHERE k1.DID = d1.DID AND k1.JMBG =
i.JMBG ))
```