Zadaci za vježbu

(uz predavanje 3 – OO i OR baze podataka)

Sa stranica predmeta (Vježbe\ORDBStruct.sql) preuzmite SQL naredbe za kreiranje relacija (i punjenje relacija odgovarajućim sadržajem) potrebnih za ovu vježbu.

Spojite se na PostgreSQL SUBP. Odaberite postojeću ili kreirajte novu bazu podataka te bavite naredbe iz ORDBStruct.sql.

Zadatak 1.

Potrebno je

a) definirati dva tipa podatka:
   - ***GPSPointDeg*** prikladan za pohranu informacija o koordinatama točke na površini Zemlje: *latitude* $\in$ [-180, 180], specificira sjever-jug i *longitude* $\in$ [-90, 90], specificira istok-zapad poziciju na površini Zemlje. Za pohranu decimalnog dijela vrijednosti *latitude* i *longitude* predvidjeti 6 mjesta.
   - ***GPSPointRad*** sličan tipu ***GPSPointDeg*** s razlikom da su koordinate izražene u radijanima umjesto u stupnjevima.

b) napisati funkciju ***DegToRad*** koja kao argument prima podatak tipa ***GPSPointDeg,*** a vraća podatak tipa ***GPSPointRad*** i pretvara vrijednosti latitude i longitude iz stupnjeva u radijane. **Pomoć**: vidi funkciju *radians*.

c) napisati naredbu kojom će se kreirati svi objekti potrebni za obavljanje pretvorbe CAST iz ***GPSPointDeg*** u ***GPSPointRad*** tip podatka. Koristiti funkciju ***DegToRad.***

d) napisati funkciju ***GPSdistance*** koja kao argumente prima dvije vrijednosti podatka tipa ***GPSPointDeg,*** a pomoću Haversinove formule izračunava i vraća najkraću udaljenost između dvije točke na površini Zemlje u km.
   Haversinova formula:

$$\mathrm{a} = \sin^2\left(\frac{lat2-lat1}{2}\right) + \cos(lat1) * \cos(lat2) * \sin^2\left(\frac{long2-long1}{2}\right)$$

$$\mathrm{d} = R * 2 * atan2\left(\sqrt{a}, \sqrt{1-a}\right) \qquad R = 6371\ \mathrm{km}\ \text{prosječna vrijednost radijusa Zemlje}$$

   Pomoć: ako je u funkciji potrebno deklarirati varijable ili koristiti proceduralne naredbe potrebno je koristiti `LANGUAGE 'plpgsql'`. Kod SQL funkcija dozvoljeno je samo obaviti niz SQL naredbi s tim da je povratna vrijednost iz funkcije rezultat zadnje SQL naredbe.

e) pozivom funkcije ***GPSdistance*** izračunati udaljenost između
   Zagreba lat: 45.81497 long: 15.97851
   i
   Splita lat: 43.50692 long:16.44245
   Udaljenost iznosi oko 259km.

Nazive i argumente matematičkih funkcija implementiranih u PostgreSQL-u možete vidjeti ovdje: http://www.postgresql.org/docs/9.4/static/functions-math.html

U zadacima 2., 3. I 4. koriste se relacije čija je shema I sadržaj definiran naredbama u ORDBStruct.sql.

Zadatak 2.

Zadane su relacije ***tramRoute***, ***tramStation***, ***tramRouteStation*** i ***tramRouteSchedule***.

tramRoute		
tramRouteId	**tramRouteAbrev**	**tramRouteName**

tramStation	
tramStationId	**tramStationName**

tramRouteStation		
tramRouteId	**TramStationId**	**tramStationOrd**

tramRouteSchedule	
tramRouteId	**departureTime**

Potrebno je napisati SQL naredbe

a) za kreiranje relacije ***tramRouteOR*** sljedeće sheme:

tramRoute				
tramRouteId	**tramRouteAbrev**	**tramRouteName**	**stationsOnTramRoute**	**departuresOnTramRoute**
1	1	Zapadni kolodvor - Borongaj	{"(1,\"Zapadni kolodvor\",1)", "(2,Talovčeva,2)",...}	{04:16:52, 04:42:00, …}
…	…	…	…	…

Atribut ***stationsOnTramRoute*** je kolekcija čiji je svaki element sastavljen od sljedećih atributa: tramStationId, tramStationName i tramStationOrd.
Atribut ***departuresOnTramRoute*** je kolekcija koja se sastoji od svih vremena polazaka tramvaja na toj liniji.

b) jednu INSERT naredbu za punjenje relacije za kreiranje relacije ***tramRouteOR*** sadržajem u skladu sa sadržajem relacija ***tramRoute***, ***tramStation***, ***tramRouteStation*** i ***tramRouteSchedule***.

Zadatak 3.

Korištenjem isključivo tablice ***tramRouteOR*** ispisati nazive tramvajskih postaja kroz koje prolaze barem 2 linije (od 3 koliko ih je u tablici).

Zadatak 4.

Korištenjem isključivo tablice ***tramRouteOR*** ispisati nazive tramvajskih linija i broj polazaka na liniji u svakom satu. Pored broja polazaka u satu ispisati i kumulativni broj polazaka – npr. broj polazaka od 5-6 sati uključuje i broj polazaka od 4-5 i td.
Pomoć: za izdvajanje sata iz atributa *source* tipa **TIME** moguće je koristiti funkciju EXTRACT(HOUR FROM source).
Rezultat upita treba izgledati kao na donjoj slici:

	tramroutename character varying(120)	hourfromto text	noofdepartures bigint	noofdeparturescumul numeric
1	Črnomerec - Savišće	0-1	2	2
2	Črnomerec - Savišće	4-5	5	7
3	Črnomerec - Savišće	5-6	7	14
4	Črnomerec - Savišće	6-7	8	22

Zadatak 5.

SQL naredbama u nastavku potrebno je kreirati relacije ***person*** i ***student***. Potrebno je kreirati objekte u bazi podataka koji će osigurati jedinstvenu vrijednost atributa ***person.personId*** bez obzira je li n-torka nastala unosom u relaciju person ili relaciju student.

```
CREATE TABLE person(
  personId INTEGER CONSTRAINT pkPerson
                   PRIMARY KEY,
  FName    VARCHAR(25),
  Lname    VARCHAR(25)) ;
```

```
CREATE TABLE student (
  JMBAG      CHAR(20),
  enrolDate  VARCHAR(100))
 INHERITS (person);
```

RJEŠENJA:

Zadatak 1.

a)

```
CREATE TYPE GPSPointDeg AS (
lat DECIMAl (10,6),
long DECIMAl (10,6))

CREATE TYPE GPSPointRad AS (
lat DECIMAl (10,6),
long DECIMAl (10,6))
```

b)

```
CREATE OR REPLACE FUNCTION DegToRad (GPSPointDeg)
RETURNS GPSPointRad AS $$
    SELECT CAST( ROW (radians($1.lat), radians($1.long)) AS GPSPointRad);
$$ LANGUAGE SQL;
```

c)

```
CREATE CAST (GPSPointDeg AS GPSPointRad)
  WITH FUNCTION DegToRad (GPSPointDeg) ;
```

Sve trigonometrijske funkcije koje se koriste u donjoj funkciji kao argumente očekuju vrijednosti u radijanima. S druge strane, uobičajeno je GPS koordinate izražavati u stupnjevima. Zbog toga će nam dobro doći mogućnost pretvorbe GPS koordinate u stupnjevima u GPS koordinatu u radijanima.

d)

```
CREATE OR REPLACE FUNCTION GPSdistance (GPSPointDeg, GPSPointDeg)
RETURNS DECIMAl(10,6) AS $$
   DECLARE
   a DECIMAL(10,6);
   DECLARE
   point1 GPSPointRad;
   DECLARE
   point2 GPSPointRad;

   BEGIN
     point1:= $1::GPSPointRad;
     point2:= $2::GPSPointRad;
     a:= sin((point2.lat-point1.lat)/2)^2 +
cos(point1.lat)*cos(point2.lat)*sin((point2.long-point1.long)/2)^2;

     RETURN 6371 *2*atan2((a ^0.5), ((1-a)^0.5));
   END
 $$ LANGUAGE 'plpgsql';
```

Nije moguće koristiti „običnu" SQL funkciju jer u tom slučaju nije moguće deklarirati varijable niti koristiti naredbe za dodjeljivanje vrijednosti varijablama.

e)

```
select GPSDistance ((45.81497, 15.97851), (43.50692, 16.44245))

Rezultat: 259.279238301364
```

Zadatak 2.

a)

```
CREATE TYPE stationOnTramRoute AS
  (tramStationId          INT,
   tramStationName        VARCHAR(120),
   tramStationOrd         SMALLINT);

CREATE TABLE tramRouteOR
  (tramRouteId            INTEGER CONSTRAINT pkTramRouteOR PRIMARY KEY,
   tramRouteAbrev         CHAR(3) NOT NULL UNIQUE,
   tramRouteName          VARCHAR(120) NOT NULL UNIQUE,
   stationsOnTramRoute    stationOnTramRoute[],
   departuresOnTramRoute  TIME[]
   );
```

b)

```
INSERT INTO tramRouteOR
 SELECT tramRoute.*
      , (SELECT array_agg(CAST( ROW(tramStation.*, tramRouteStation.tramStationOrd)
AS StationOnTramRoute))
           FROM tramRouteStation, tramStation
          WHERE tramRouteStation.tramStationId = tramStation.tramStationId
            AND tramRouteStation.tramRouteId = tramRoute.tramRouteId)
      , (SELECT array_agg(departureTime)
           FROM tramRouteSchedule
          WHERE tramRouteSchedule.tramRouteId = tramRoute.tramRouteId)
FROM tramRoute
```

Promotrimo izraz:

```
array_agg(CAST( ROW(tramStation.*, tramRouteStation.tramStationOrd)  AS
StationOnTramRoute))
```

1. Treba li nam `CAST (…) AS StationOnTramRoute`?

   Pokušaj izvođenja INSERT naredbe bez provedenog CAST-a tj.

   ```
   INSERT INTO tramRouteOR
    SELECT tramRoute.*
         , (SELECT array_agg(ROW(tramStation.*, tramRouteStation.tramStationOrd))
              FROM tramRouteStation, tramStation
                ...
   ```

   završi greškom

   ```
   ERROR:  column "stationsOnTramRoute" is of type stationOnTramRoute[] but expression
   is of type record[]
   LINE 4:       , (SELECT array_agg(ROW(tramStation.*, tramRouteStation.r...
                    ^
   HINT:  You will need to rewrite or cast the expression.
   ```

`array_agg(ROW(tramStation.*, tramRouteStation.tramStationOrd))` je kolekcija čiji su elementi "nepoznatog" tipa tj. rezultat tog izraza je tipa record [ ]. Atribut ***tramRouteOR.stationsOnTramRoute*** je kolekcija čiji su elementi tipa ***stationOnTramRoute.***
Moramo provesti CAST jer u protivnom, zbog nekompatibilnih tipova podataka, INSERT naredbu neće biti moguće obaviti.

2. Treba li nam `ROW(tramStation.*, tramRouteStation.tramStationOrd)`? Možemo li napisati samo
   `CAST( (tramStation.*, tramRouteStation.tramStationOrd)  AS StationOnTramRoute)`

Ne treba - sintaksa je ispravna i bez ROW.

Zadatak 3.

Da bismo mogli grupirati zapise prema nazivu postaje koji se nalazi u kolekciji ***tramRouteOr.stationsOnTramRoute*** moramo odgnijezditi kolekciju. Rezultat upita

```
SELECT tramRouteOR.tramRouteId
     , UNNEST (tramRouteOR.stationsOnTramRoute) AS stationsOnTramRoute
  FROM tramRouteOR
```

je sljedećeg oblika:

	tramrouteid integer	stationsontramroute stationontramroute
1	1	(1,"Zapadni kolodvor",1)
2	1	(2,Talovčeva,2)
3	1	(3,Reljkovićeva,3)
4	1	(4,"Trg dr. Franje Tuđmana",4)
5	1	(5,"Britanski trg",5)
6	1	(6,Frankopanska,6)

Atribut ***stationsOnTramRoute*** je složenog tipa (`TYPE stationOnTramRoute`). Njegovim atributima moguće je pristupiti preko imena. Međutim, treba paziti prilikom pisanja upita kako pristupamo atributima složenog tipa.
Sljedeći upit npr. završi pogreškom:

```
SELECT stationsOnTramRoute.tramStationId,
       stationsOnTramRoute.tramStationName,
       stationsOnTramRoute.tramStationOrd
  FROM (
        SELECT tramRouteOR.tramRouteId
             , UNNEST (tramRouteOR.stationsOnTramRoute) AS stationsOnTramRoute
         FROM tramRouteOR)    AS stationsOnAllTramRoutes
ERROR:  missing FROM-clause entry for table "stationsontramroute"
LINE 1: SELECT stationsOnTramRoute.tramStationId,
               ^
```

jer prema SQL sintaksi ***stationsOnTramRoute*** predstavlja ime tablice (a to je samo složeni atribut u "tablici" ***stationsOnAllTramRoutes).*** Ispravno napisana SELECT lista za gornji upit je:

```
SELECT (stationsOnTramRoute).tramStationId,
       (stationsOnTramRoute).tramStationName,
       (stationsOnTramRoute).tramStationOrd
```

ili (ako treba uključiti I ime tablice)

```
SELECT (stationsOnAllTramRoutes.stationsOnTramRoute).tramStationId,
       (stationsOnAllTramRoutes.stationsOnTramRoute).tramStationName,
       (stationsOnAllTramRoutes.stationsOnTramRoute).tramStationOrd
```

Objekt u zagradama PostgreSQL interpretira kao referencu na objekt ***stationsOnTramRoute***, pa se potom mogu selektirati njegovi atributi.

Rješenje zadatka:

```
SELECT (stationsOnAllTramRoutes).stationsOnTramRoute.tramStationName,
       COUNT((stationsOnAllTramRoutes).tramRouteId)
  FROM (
        SELECT tramRouteOR.tramRouteId
             , UNNEST (tramRouteOR.stationsOnTramRoute) AS stationsOnTramRoute
         FROM tramRouteOR)    AS stationsOnAllTramRoutes
 GROUP BY (stationsOnAllTramRoutes).stationsOnTramRoute.tramStationName
 HAVING COUNT(*) >= 2
```

Zadatak 4.

Da bismo mogli grupirati zapise prema nazivu linije i satu polaska koji se nalazi u kolekciji ***tramRouteOR.departuresOnTramRoute*** moramo odgnijezditi kolekciju. Rezultat upita

```
SELECT tramRouteOR.tramRouteId, tramRouteOR.tramRouteName
            , UNNEST (tramRouteOR.departuresOnTramRoute) AS departuresOnTramRoute
         FROM tramRouteOR
```

je sljedećeg oblika:

	tramrouteid integer	tramroutename character varying(120)	departuresontramroute time without time zone
1	1	Zapadni kolodvor - Borongaj	04:16:52
2	1	Zapadni kolodvor - Borongaj	04:42:00
3	1	Zapadni kolodvor - Borongaj	05:00:22

Za računanje kumulativnog broja polazaka u sati treba

1. stvoriti particiju za svaku liniju (PARTITION BY tramRouteName; može i PARTITION BY tramRouteId ili PARTITION BY tramRouteName, tramRouteId ali onda treba prilagoditi GROUP BY)
2. ispravno poredati međurezultate u particiji `ORDER BY (EXTRACT(HOUR FROM departuresOnTramRoute))`. Budući da se u okviru trenutne n-torke (obzirom na to kako je napisan OVER dio) pored nje same nalaze i sve n-torke koje su u particiji prije nje, bez ispravno napisanog ORDER BY dijela, kumulativne sume neće biti ispravne.

```
SELECT tramRouteName,
        EXTRACT(HOUR FROM departuresOnTramRoute) || '-' ||
        EXTRACT(HOUR FROM departuresOnTramRoute)+1 hourFromTo,
        COUNT(*) noOfdepartures,
        SUM(count(*)) OVER (PARTITION BY tramRouteName
                            ORDER BY (EXTRACT(HOUR FROM departuresOnTramRoute)))
                      AS noOfdeparturesCumul
  FROM (
        SELECT tramRouteOR.tramRouteId, tramRouteOR.tramRouteName
             , UNNEST (tramRouteOR.departuresOnTramRoute) AS departuresOnTramRoute
         FROM tramRouteOR)    AS stationsOnAllTramRoutes
 GROUP BY tramRouteName,
          EXTRACT(HOUR FROM departuresOnTramRoute) || '-' ||
          EXTRACT(HOUR FROM departuresOnTramRoute)+1,
          EXTRACT(HOUR FROM departuresOnTramRoute)
 ORDER BY tramRouteName, EXTRACT(HOUR FROM departuresOnTramRoute)
```

U ovom zadatku je važno ispravno napisati GROUP BY dio. Jasno je da se u GROUP BY mora nalaziti

```
tramRouteName
i
EXTRACT(HOUR FROM departuresOnTramRoute) || '-' || EXTRACT(HOUR FROM
departuresOnTramRoute)+1
```

jer se pored `COUNT(*)` oba izraza nalaze u SELECT listi SQL naredbe.
Manje je očigledno da se u GROUP BY mora nalaziti i `EXTRACT(HOUR FROM departuresOnTramRoute)`. Razlog je taj što se koristi pri izračunavanju `SUM(count(*))` za particiju. Budući da se izrazi koji se računaju nad n-torkama u prozorima evaluiraju tek nakon obavljanja GROUP BY (i eventualnog HAVING, kojeg u ovom upitu nema), ako izostavimo `EXTRACT(HOUR FROM departuresOnTramRoute)` neće se moći niti n-torke u particiji poredati prema toj vrijednosti.

Zadatak 5.

Da integritet ključa nije očuvan može se vidjeti izvođenjem nisa sljedećih SQL naredbi.

```
set DateStyle ='German, DMY';
INSERT INTO person     VALUES (1, 'Ana'   , 'Ban');
INSERT INTO person     VALUES (1, 'Tena'  , 'Pale');
ERROR:  duplicate key value violates unique constraint "pkperson"
DETAIL:  Key (personId)=(1) already exists.

INSERT INTO student   VALUES (1, 'Mia'  , 'Nel', '0036000001', '01.07.2013');
Query returned successfully: one row affected, 12 ms execution time.
```

```
SELECT *
FROM person;
```

	personid integer	fname character varying(25)	lname character varying(25)
1	1	Ana	Ban
2	1	Mia	Nel

U ***person*** više nije očuvan integritet ključa. Pokušajmo očuvati integritet ključa u ***person*** na sljedeći način:

```
ALTER TABLE student ADD CONSTRAINT studentPk PRIMARY KEY (personId);
```

Činjenica da je gornja SQL naredba obavljena bez dojave greške govori nam da nismo zaštitili integritet ključa u person.

Također, iz sljedećeg niza naredbi vidi se da je zaštićena jedinstvenost atributa ***personId*** na razini n-torki unešenih isključivo u tablicu ***student***, ali ne i na razini unije n-torki u ***person*** *i* ***student***.

```
INSERT INTO person    VALUES (2, 'Tena'  , 'Pale');
INSERT INTO student   VALUES (2, 'Ivo'  , 'Puž',  '0036000002', '12.07.2014');
Query returned successfully: one row affected, 12 ms execution time.

INSERT INTO student   VALUES (2, 'Ivo'  , 'Puž',  '0036000002', '12.07.2014');
ERROR:  duplicate key value violates unique constraint "studentpk"
DETAIL:  Key (personId)=(2) already exists.
```

Integritet ključa na razini unije n-torki u ***person*** *i* ***student*** moguće je postići pomoću procedure koja će dojavljivati pogrešku pri pokušaju unosa n-torke s istim ključem u relaciju ***person*** i okidača koji će se aktivirati pri svakom unosu u ***person*** *i* ***student.***

```
CREATE FUNCTION
  personPkChk() RETURNS TRIGGER AS $$
  BEGIN
     IF (EXISTS (SELECT * FROM person
                 WHERE person.personId = NEW.personId)) THEN
        RAISE EXCEPTION 'Postoji zapis sa šifrom %.', NEW.personId;
     END IF;
     RETURN NEW;
   END
 $$ LANGUAGE plpgsql;

CREATE TRIGGER studentPkChk
BEFORE INSERT ON student
  FOR EACH ROW
      EXECUTE PROCEDURE personPkChk();

CREATE TRIGGER personPkChk
BEFORE INSERT ON person
  FOR EACH ROW
      EXECUTE PROCEDURE personPkChk();
```

Sljedećim nizom naredbi moguće je testirati ispravnost rješenja:

```
INSERT INTO person     VALUES (1, 'Ana'   , 'Ban');                                    -
-ok
INSERT INTO student   VALUES (1, 'Mia'  , 'Nel', '0036000001', '01.07.2013');
Already exists record with id 1.

INSERT INTO student   VALUES (2, 'Mia'  , 'Nel', '0036000001', '01.07.2013');  --ok
INSERT INTO person    VALUES (2, 'Tena'  , 'Pale');
Already exists record with id 2.
```