Napredni modeli i baze podataka - međuispit 17.11.2014.

U zadacima 1, 2, 3, 5 i 6 pretpostavlja se korištenje PostgreSQL SUBP-a.

Zadaci **1, 2** i **3** se odnose na objektno-relacijsku bazu podataka prikazanu na donjoj slici. U bazi se pohranjuju podaci o osobama (relacija ***osoba***), nekretninama (relacija ***nekretnina***) i parcelama (relacija **parcela**). Informacija o vlasništvu osoba nad nekretninom i parcelom pohranjena je u odgovarajućem atributu ***vlasnici***. Geoprostorni podaci o nekretninama i parcelama pohranjeni su u odgovarajućem atributu **geom** (tipa POLYGON). U relaciji ***nekretnina*** evidentirane su sve nekretnine, koje mogu biti ili kuće ili stanovi. Za kuće se dodatno evidentira i broj katova (relacija ***kuca*** nije prikazana na slici). Atributi koji čine ključeve relacija su podcrtani.

osoba

sifOsoba	***imeOsoba***	***prezOsoba***
1	Slack	David
2	Segel	Amanda
3	Smith	Bob
4	Cardell	Ema
5	Tillman	Joan

nekretnina

sifNekretnina	***povrsina***	***vlasnici***	***adresa (pbr, ulica,broj)***	***geom***
1	45.6	{13, 22,5}	(10000, "Ilica", 12)	Polygon (…
2	380.4	{555}	(10000, "Vinogradska", 14)	Polygon (…
3	33.5	{22}	(10000, "Ilica", 12)	Polygon (…
4	1087.3	{22,13}	(10000, "Grada Vukovara", 12)	Polygon (…
5	234.5	{1}	(10000, "Grada Vukovara", 345)	Polygon (…
6	22.1	{5}	(44000, "Vinogradska", 12)	Polygon (…
…	…	…		

parcela

brParcela	***vlasnici***	***geom***
101	{1}	Polygon (…
102/1	{5}	Polygon (…
102/2	{22, 13}	Polygon (…

1. (**4 boda**) Napisati niz SQL naredbi kojima će se kreirati tablice ***osoba***, ***nekretnina*** i ***kuca***, te svi potrebni objekti objektno-relacijske baze podataka. Segment modela kojim su opisane nekretnine mora biti realiziran pomoću hijerarhijske strukture tablica. Niti jedan atribut ne smije poprimiti NULL vrijednost.
   - Osigurati da je za nekretninu naveden barem jedan vlasnik. Pretpostaviti da u SUBP-u postoji **intarray** proširenje.
   - Nije potrebno osigurati da vlasnik nekretnine bude evidentiran kao osoba.
   - SQL naredbe za kreiranje objekata potrebnih za očuvanje jedinstvenosti atributa *sifNekretnina* u hijerarhiji tablica nije potrebno pisati - dovoljno je navesti i opisati te objekte i objasniti na koji način oni rješavaju problem.

2. (**3 boda**) Napisati SQL naredbu kojom će se dohvatiti osobe koje u vlasništvu (bilo kao jedini vlasnik bilo kao suvlasnik) imaju više od jednog stana u ulici 'Ilica'. Za svaku osobu ispisati ime, prezime i polje koje sadrži šifre stanova u ulici 'Ilica' u vlasništvu te osobe. Zadatak riješiti bez korištenja podupita.

3. (**4 boda**) Izgrađenost parcele računa se kao omjer ukupne površine svih nekretnina na parceli i površine same parcele. Pretpostavite da parcela s brojem 7007 ima izgrađenost veću od 50% (taj podatak ne treba provjeravati). Napišite upit koji će ispisati sve susjedne parcele koje bi vlasnik parcele 7007 mogao kupiti (možete pretpostaviti da ne posjeduje niti jednu od njih i da kupuje samo jednu) pa da ukupna izgrađenost njegovog vlasništva (parcele 7007 i kupljene susjedne parcele) bude manja od 50%. Ispisane parcele treba poredati uzlazno prema njihovoj površini.

   Vodite računa da i susjedne parcele mogu imati izgrađene nekretnine koje treba uračunati.

   Na raspolaganju su sljedeće prostorne funkcije:

```
ST_Area(geometry)
ST_Touches(geometry, geometry)
ST_Within(geometry, geometry)
ST_Contains(geometry, geometry)
ST_Union(geometry, geometry)
```

4. (**3 boda**) Objasniti razliku između prostornih topoloških operacija preklapanja (*overlaps*) i presijecanja (*crosses*). Nacrtajte primjer obje operacije za dvije linije

5. **(5 bodova)** Temeljem podataka u tablici ***strukturaStudij*** za sve završne dijelove studija (koji nisu nadređeni nijednom drugom dijelu studija) ispisati naziv, kumulativno trajanje u semestrima i kumulativan broj ECTS bodova koje student mora osvojiti da bi završio taj studij. Kumulativno trajanje i kumulativan broj ECTS-a uključuju trajanje i ECTS-e svih nadređenih dijelova studija. Uzeti u obzir da dubina hijerarhije između početnog i završnog dijela studija može biti različita.

strukturaStudij

SifDioStudij	***nazivDioStudij***	***sifNadDioStudij***	***trajeSem***	***ECTSBod***
10	Elektrotehnika i informacijska tehnologija i Računarstvo		2	60
20	Elektrotehnika i informacijska tehnologija	10	2	60
30	Računarstvo	10	2	60
21	Automatika	20	2	60
22	Elektroenergetika	20	2	60
31	Programsko inženjerstvo i informacijski sustavi	30	2	60
32	Računarska znanost	30	2	60
…	…	…	…	…

Izgled rezultata upita:

nazivDioStudij	***ukupnoTrajanje***	***potrebnoECTS***
Automatika	6	180
Elektroenergetika	6	180
Programsko inženjerstvo i informacijski sustavi	6	180
…	…	…

6. **(4 boda)** Objasniti ulogu i način funkcioniranja parsera i rječnika u kontekstu pretraživanju teksta u PostgreSQL SUBP. Objasniti kako su parser i rječnici međusobno povezani te na koji način promjena parametara povezivanja utječe na obradu i pretraživanje teksta.

7. (**2 boda**) Objasniti što je to vrijeme valjanosti, a što je transakcijsko vrijeme.

Rješenja:

1. (4 bodova)

```
CREATE TABLE osoba (sifOsoba  INT PRIMARY KEY,
                    imeOsoba  VARCHAR(60) NOT NULL,
                    prezOsoba VARCHAR(60) NOT NULL);

CREATE TYPE adresaT AS (pbr   INT,
                        ulica VARCHAR(60),
                        broj  SMALLINT);

CREATE TABLE nekretnina (sifNekretnina INT PRIMARY KEY,
                         povrsina       DECIMAL(5,1) NOT NULL,
                         vlasnici       INT[] NOT NULL CHECK (icount (vlasnici)>0),
                         adresa         adresaT NOT NULL,
                         geom           POLYGON);

CREATE TABLE kuca (brKat SMALLINT) INHERITS (nekretnina);
```

Očuvanje jedinstvenosti atributa *sifNekretnina*: nije nužno navoditi naredbe već opisati rješenje.
Slajd 106 predavanja *Objektno orijentirane i objektno-relacijske baze podataka*.

2. (**3 boda**)

```
SELECT imeOsoba, prezOsoba, array_agg(sifNekretnina)
    FROM ONLY (nekretnina), UNNEST(nekretnina.vlasnici) AS v(sifOsoba), osoba
 WHERE v.sifOsoba = osoba.sifOsoba
   AND (adresa).ulica = 'Ilica'
    GROUP BY v.sifOsoba, imeOsoba, prezOsoba
 HAVING COUNT(*) > 1
```

3. **(3 boda)**

```
SELECT p2.broj, ST_Area(p2.geom)
  FROM parcela p1, parcela p2
 WHERE p1. broj = 7007
   AND ST_Touches(p1.geom, p2.geom)
  AND ((SELECT SUM(ST_Area(n1.geom)) FROM stambenazgrada n1
         WHERE ST_Within(n1.geom, p1.geom)
            OR ST_Within(n1.geom, p2.geom)) /
    (ST_Area(p1.geom) + ST_Area(p2.geom))) < 0.5
ORDER BY ST_Area(p2.geom)
```

4. (3 boda)
Kod preklapanja dimenzija presjeka mora biti jednaka dimenziji argumenata, dok kod presijecanja mora biti za jedan manja od maksimalne dimenzije.

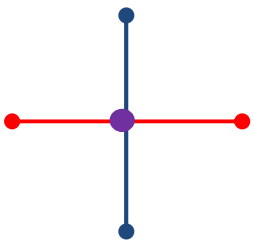
Presijecanje (crosses):

$$\langle \lambda_1, cross, \lambda_2 \rangle \Leftrightarrow \left(\dim\left(\lambda_1^0 \cap \lambda_2^0\right) = \max\left(\dim\left(\lambda_1^0\right), \dim\left(\lambda_2^0\right)\right) - 1\right) \wedge \left(\lambda_1 \cap \lambda_2 \neq \lambda_1\right) \wedge \left(\lambda_1 \cap \lambda_2 \neq \lambda_2\right)$$

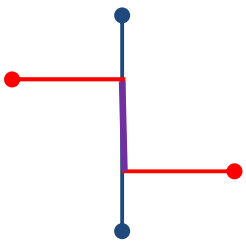
Preklapanje (overlaps):

$$\langle \lambda_1, overlap, \lambda_2 \rangle \Leftrightarrow \left(\dim\left(\lambda_1^0\right) = \dim\left(\lambda_2^0\right) = \dim\left(\lambda_1^0 \cap \lambda_2^0\right)\right) \wedge \left(\lambda_1 \cap \lambda_2 \neq \lambda_1\right) \wedge \left(\lambda_1 \cap \lambda_2 \neq \lambda_2\right)$$

Presijecanje:



Preklapanje:



5. (5 bodova)

```
//ako želite testirati upite možete pomoću sljedećih SQL naredbi kreirati relaciju i
napuniti je zapisima
CREATE TABLE strukturaStudij (
sifDioStudij integer,
nazivDioStudij char(100),
sifNadDioStudij integer,
trajeSem integer,
ECTSBod integer);

INSERT INTO strukturaStudij VALUES (10,'Elektrotehnika i informacijska tehnologija i
Računarstvo', null, 2, 60);
INSERT INTO strukturaStudij VALUES (20,'Elektrotehnika i informacijska tehnologija',
10,2, 60);
INSERT INTO strukturaStudij VALUES (30,'Računarstvo', 10,2, 60);
INSERT INTO strukturaStudij VALUES (21,'Automatika',20 , 2, 60);
INSERT INTO strukturaStudij VALUES (22,'Elektroenergetika',      20, 2, 60);
INSERT INTO strukturaStudij VALUES (31,'Programsko inženjerstvo i informacijski
sustavi',    30 , 2, 60);
INSERT INTO strukturaStudij VALUES (32,'Računarska znanost',     30, 2, 60);



WITH RECURSIVE SSRecursive (sifDioStudij, nazivDioStudij, sifNadDioStudij, trajeSem,
ECTSBod) AS
(SELECT sifDioStudij, nazivDioStudij, sifNadDioStudij, trajeSem, ECTSBod
   FROM strukturaStudij
  UNION
  SELECT SSRecursive.sifDioStudij, SSRecursive.nazivDioStudij,
         strukturaStudij.sifNadDioStudij,
         strukturaStudij.trajeSem, strukturaStudij.ECTSBod
    FROM SSRecursive, strukturaStudij
   WHERE SSRecursive.sifNadDioStudij = strukturaStudij.sifDioStudij
  )
  SELECT sifDioStudij, nazivDioStudij, SUM(ECTSBod), SUM(trajeSem)
    FROM SSRecursive
    WHERE NOT EXISTS (SELECT *
                        FROM strukturaStudij ssp
                       WHERE SSRecursive.sifDioStudij = ssp.sifNadDiostudij)
/* ili
    WHERE sifDioStudij NOT IN (SELECT DISTINCT sifNadDiostudij
                                 FROM strukturaStudij
                                WHERE sifNadDioStudij IS NOT NULL)
*/
GROUP BY sifDioStudij, SSRecursive.nazivDioStudij

ili

WITH RECURSIVE SSRecursive (sifDioStudij, nazivDioStudij, sifNadDioStudij, trajeSem,
ECTSBod) AS
(SELECT sifDioStudij, nazivDioStudij, sifNadDioStudij, trajeSem, ECTSBod
   FROM strukturaStudij
  UNION
  SELECT SSRecursive.sifDioStudij, SSRecursive.nazivDioStudij,
         strukturaStudij.sifNadDioStudij,
         strukturaStudij.trajeSem, strukturaStudij.ECTSBod
    FROM SSRecursive, strukturaStudij
   WHERE SSRecursive.sifNadDioStudij = strukturaStudij.sifDioStudij
  )
  SELECT DISTINCt nazivDioStudij,
         SUM(ECTSBod) OVER (PARTITION BY nazivDioStudij),
         SUM(trajeSem) OVER (PARTITION BY nazivDioStudij)
    FROM SSRecursive
    WHERE sifDioStudij NOT IN (SELECT DISTINCT sifNadDiostudij
                                 FROM strukturaStudij
                                WHERE sifNadDioStudij IS NOT NULL)
```

ili

```
WITH RECURSIVE SSRecursive (sifDioStudij, sifNadDioStudij) AS
(SELECT sifDioStudij, sifNadDioStudij
   FROM strukturaStudij
  UNION
  SELECT SSRecursive.sifDioStudij,
         strukturaStudij.sifNadDioStudij
    FROM SSRecursive, strukturaStudij
   WHERE SSRecursive.sifNadDioStudij = strukturaStudij.sifDioStudij
  )
  SELECT SSRecursive.sifDioStudij,
         SSnaziv.nazivDioStudij,
         SUM(SSTrajeIECTS.ECTSBod),
         SUM(SSTrajeIECTS.trajeSem)
    FROM SSRecursive, strukturaStudij SSnaziv, strukturaStudij SSTrajeIECTS
   WHERE SSRecursive.sifDioStudij = SSnaziv.sifDioStudij
     AND (SSRecursive.sifNadDioStudij = SSTrajeIECTS.sifDioStudij OR
          SSRecursive.sifNadDioStudij IS NULL AND
          SSRecursive.sifDioStudij = SSTrajeIECTS.sifDioStudij)
     AND  SSRecursive.sifDioStudij NOT IN (SELECT DISTINCT sifNadDiostudij
                                             FROM strukturaStudij
                                            WHERE sifNadDioStudij IS NOT NULL)
    GROUP BY SSRecursive.sifDioStudij, SSnaziv.nazivDioStudij
```

ili

```
WITH RECURSIVE SSRecursive (sifDioStudij, nazivDioStudij, sifNadDioStudij, trajeSem,
ECTSBod) AS
(SELECT sifDioStudij, nazivDioStudij, sifNadDioStudij, trajeSem, ECTSBod
   FROM strukturaStudij
  UNION
  SELECT SSRecursive.sifDioStudij, SSRecursive.nazivDioStudij,
         strukturaStudij.sifNadDioStudij,
         SSRecursive.trajeSem + strukturaStudij.trajeSem,
         SSRecursive.ECTSBod + strukturaStudij.ECTSBod
    FROM SSRecursive, strukturaStudij
   WHERE SSRecursive.sifNadDioStudij = strukturaStudij.sifDioStudij
  )
  SELECT nazivDioStudij, trajeSem, ectsBod
    FROM SSRecursive
//u zadnjoj iteraciji će doći do korijenskog dijela studija koji nema nadređeni i
//u tom trenutku će vrijednosti SSRecursive.trajeSem i SSRecursive.ectsBod biti konačne
    WHERE SSRecursive.sifNadDioStudij IS NULL
    AND sifDioStudij NOT IN (SELECT DISTINCT sifNadDiostudij
                               FROM strukturaStudij
                              WHERE sifNadDioStudij IS NOT NULL)
```

6. **(4 boda)**

- Parser je program koji u ulaznom izvornom tekstu identificira tokene i utvrđuje tip tokena.
- Ulazni tekst ne modificira čak ne mijenja velika slova u mala i obratno.
- Tipovi tokena koje parser prepoznaje unaprijed su definirani.
- Ugrađeni parser PostgreSQL-a razlikuje 23 tokena: asciiword, word, numword,email, protocol, url, host, file, tag, blank,…
-
- **Rječnici** su programi koji dalje obrađuju rezultat parsiranja.
- Koriste za se normalizaciju riječi koja omogućuje prepoznavanje riječi s jednakim normaliziranim oblikom (leksemom). Time se reducira veličina (tsvector) reprezentacije dokumenta i postižu dobra svojstva pretraživanja cijelog teksta
- Rječnik omogućava definiranje:
  - stop riječi - riječi koje se vrlo često pojavljuju, gotovo u svakom dokumentu i najčešće se ignoriraju pri pretrazi teksta
  - sinonima
  - stvaranje veza između fraza i pojedinih riječi
  - pronalaženje
  -
- Parser se povezuje sa skupom rječnika pomoću konfiguracijskih parametara. Oni utječu na način na koji se rezultat parsiranja dalje normalizira.

- Za svaki tip tokena definira se lista rječnika i redoslijed kojim se obilaze pri normalizaciji tokena
- Rječnici se pozivaju navedenim redoslijedom
    - Ako rječnik prepozna token, rječnici nakon njega se preskaču
    - Ako rječnik ne prepozna token (vrati NULL), token se prosljeđuje sljedećem rječniku
- Obično se na početak liste stavlja najspecifičniji rječnik, potom općenitiji rječnici a na kraj liste najopćenitij rječnik (kao Snowball) koji prepoznaje svaku riječ

7. (2 boda)

Vrijeme valjanosti : Vrijeme u stvarnom svijetu kada se neki događaj dogodio ili kada je neka činjenica važeća, nezavisno od trenutka kada je informacija o tom događaju/činjenici zapisana u bazu podataka

Transakcijsko vrijeme: Vrijeme kada je određena promjena zabilježena u bazi podataka ili vremenski interval tijekom kojeg se baza podataka nalazi u određenom stanju