Advanced databases – midterm exam

November 17th. 2014.

Problems **1**, **2** and 4 refer to the object-relational database shown below. The database stores data about persons (relation **person**), real estates (relation **realEstate**) and parcels (relation **parcel**). Information about real estate and parcel property is stored within corresponding attribute **owners** – the attribute contains identifiers of persons that own a part of or whole real estate or parcel (owners are ordered by the size of the ownership share, the owners with greater share come before those with smaller share). A real estate can have at most 20 owners. The attribute **realEstate.address** if of type ROW. Real estate and parcel geospatial data is stored with corresponding attribute **geom** (type POLYGON). Relation **realEstate** contains all real estates that can be house or apartments. For houses, a number of floors is also stored (relation house is not displayed). Relation keys are underlined.

person

personID	***firstname***	***lastname***
1	Slack	David
2	Segel	Amanda
3	Smith	Bob
4	Cardell	Ema
5	Tillman	Joan

realEstate

realEstateID	***area***	***owners***	***address (zip, street, number)***	***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 (…
…	…	…		

parcel

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

Problems 1 and 2 assume the use of SQL standard.

1. (***4 points***) Write a series of SQL statements to create the tables **person**, **realEstate** and **house** as well as all necessary object-relational database objects. A part of the model used to describe real estates must be realized using a table hierarchy. Object identifiers are generated by the system (names for attributes storing object identifiers name arbitrarily).

2. (***3 points***) Write an SQL statements that will print out all persons that own (a whole or a share of) more than one apartment in the street 'Ilica'. For each person print out a first and last name and a total number of real estates owned by that person. **Do not use subqueries**.

3. (**4 points**) A usage of a parcel is a ration of the total area taken by all real estates on the parcel and the area of the parcel itself. Assume that the parcel 7007 has a usage greater than 50%. Write an SQL statement that will print out all of neighboring parcels that the parcel 7007 owner could buy (assume that he doesn't already own any of them and that he will buy only one) such that the total usage of both parcels together (parcel 7007 and the bought neighboring parcel) is less than 50%. Print out parcels according to their area, from the smallest to the largest.

   Keep in mind that neighboring parcels can also have real estate on them.

   You can use the following functions:

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

4. (**3 points**) Explain the difference tween spatial topology operations *overlaps* and *crosses*. Draw an example for each operation for two lines.

5. **(5 points)** Based on the data in the table studyStructure for all of the final parts of the study (which are not superior to any other part of the study) print name, cumulative duration in the semester and cumulative number of credits that a student needs to acquire in order to complete this study. Cumulative duration and cumulative number of ECTS include the duration and ECTS credits of all higher-level components of the study. Take into account that the depth of the hierarchy between the initial and final part of the study may be different.

studyStructure

studyStructureID	***studyStructureName***	***studyStructureSuperiorID***	***durationSem***	***ECTSCredits***
10	Electrical Engineering and Information Technology and Computing		2	60
20	Electrical Engineering and Information Technology	10	2	60
30	Computing	10	2	60
21	Control Engineering and Automation	20	2	60
22	Electrical Power Engineering	20	2	60
31	Software Engineering and Information Systems	30	2	60
32	Computer Science	30	2	60
…	…	…	…	…

Query results should look like:

studyStructureName	***totalDurationSem***	***totalECTSCredits***
Automatika	6	180
Elektroenergetika	6	180
Programsko inženjerstvo i informacijski sustavi	6	180
…	…	…

6. (**4 points**) Explain the role and functioning of the parser and dictionaries in the context of the text search in the PostgreSQL DBMS. Explain how the parser and dictionaries are interconnected and how parametrization of the relationship between parser and dictionaries affects the processing and text retrieval.

7. (**2 points**) Define valid time and transaction time.

Answers:

1. **(4 points)**

```
CREATE TABLE person (personID  INT PRIMARY KEY,
                     firstName  VARCHAR(60) NOT NULL,
                     lastName   VARCHAR(60) NOT NULL);

CREATE TYPE realEstateT AS (realEstateID INT,
                            area         DECIMAL(5,1),
                            owners       INTEGER ARRAY[20],
                            address      ROW (zip   INT,
                                              street CHAR(60),
                                              number  SMALLINT)
                            geom         POLYGON)
                            INSTANTIABLE
                            NOT FINAL
                            REF IS SYSTEM GENERATED;

CREATE TYPE houseT UNDER realEstateT AS (floors SMALLINT)
                   INSTANTIABLE NOT FINAL;

CREATE TABLE realEstate OF realEstateT (PRIMARY KEY (realEstateID),
                                        REF IS realEstateOID SYSTEM GENERATED);
CREATE TABLE house OF houseT UNDER realEstate;
```

2. **(3 points)**

```
SELECT firstName, lastName, COUNT(*)
  FROM ONLY (realEstate), UNNEST(realEstate.owners) AS v(personID), person
 WHERE v.personID = person.personID
   AND address.street = 'Ilica'
 GROUP BY v.personID, firstName, lastName
 HAVING COUNT(*) > 1
```

3. **(4 points)**

```
SELECT p2.number, ST_Area(p2.geom)
  FROM parcel p1, parcel p2
 WHERE p1.number = 7007
   AND ST_Touches(p1.geom, p2.geom)
  AND ((SELECT SUM(ST_Area(r1.geom)) FROM realestate r1
        WHERE ST_Within(r1.geom, p1.geom)
           OR ST_Within(r1.geom, p2.geom)) /
    (ST_Area(p1.geom) + ST_Area(p2.geom))) < 0.5
ORDER BY ST_Area(p2.geom)
```

4. (3 points)

With overlap, dimension of the intersection has to be equal to the dimension of the arguments, while with cross the intersection dimension has to be lower by one than the maximum argument dimension.

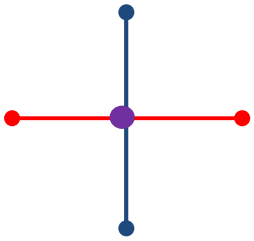
Ccrosses:

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

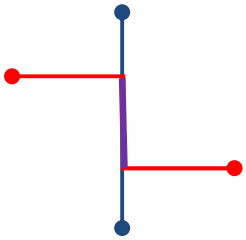
Overlaps:

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

Crosses:



Overlaps:



5. (5 points)

```
// if you want to test queries, you can use the following SQL statements to create the
table and insert the records

CREATE TABLE studyStructure (
studyStructureID integer,
studyStructurename char(100),
studyStructureSuperiorID integer,
durationSem integer,
ECTSCredits integer);

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



WITH RECURSIVE SSRecursive (studyStructureID, studyStructurename,
studyStructureSuperiorID, durationSem, ECTSCredits) AS
(SELECT studyStructureID, studyStructurename, studyStructureSuperiorID, durationSem,
ECTSCredits
   FROM studyStructure
  UNION
  SELECT SSRecursive.studyStructureID, SSRecursive.studyStructurename,
         studyStructure.studyStructureSuperiorID,
         studyStructure.durationSem, studyStructure.ECTSCredits
    FROM SSRecursive, studyStructure
   WHERE SSRecursive.studyStructureSuperiorID = studyStructure.studyStructureID
  )
  SELECT studyStructureID, studyStructurename, SUM(ECTSCredits), SUM(durationSem)
    FROM SSRecursive
    WHERE NOT EXISTS (SELECT *
                        FROM studyStructure ssp
                       WHERE SSRecursive.studyStructureID = ssp.studyStructureSuperiorID)
/* OR
    WHERE studyStructureID NOT IN (SELECT DISTINCT studyStructureSuperiorID
                                   FROM studyStructure
```

```
                                  WHERE studyStructureSuperiorID IS NOT NULL)
*/
GROUP BY studyStructureID, SSRecursive.studyStructurename

OR

WITH RECURSIVE SSRecursive (studyStructureID, studyStructurename,
studyStructureSuperiorID, durationSem, ECTSCredits) AS
(SELECT studyStructureID, studyStructurename, studyStructureSuperiorID, durationSem,
ECTSCredits
   FROM studyStructure
  UNION
  SELECT SSRecursive.studyStructureID, SSRecursive.studyStructurename,
         studyStructure.studyStructureSuperiorID,
         studyStructure.durationSem, studyStructure.ECTSCredits
    FROM SSRecursive, studyStructure
   WHERE SSRecursive.studyStructureSuperiorID = studyStructure.studyStructureID
  )
  SELECT DISTINCt studyStructurename,
         SUM(ECTSCredits) OVER (PARTITION BY studyStructurename),
         SUM(durationSem) OVER (PARTITION BY studyStructurename)
    FROM SSRecursive
    WHERE studyStructureID NOT IN (SELECT DISTINCT studyStructureSuperiorID
                                   FROM studyStructure
                                  WHERE studyStructureSuperiorID IS NOT NULL)

OR

WITH RECURSIVE SSRecursive (studyStructureID, studyStructureSuperiorID) AS
(SELECT studyStructureID, studyStructureSuperiorID
   FROM studyStructure
  UNION
  SELECT SSRecursive.studyStructureID,
         studyStructure.studyStructureSuperiorID
    FROM SSRecursive, studyStructure
   WHERE SSRecursive.studyStructureSuperiorID = studyStructure.studyStructureID
  )
  SELECT SSRecursive.studyStructureID,
         SSnaziv.studyStructurename,
         SUM(SSTrajeIECTS.ECTSCredits),
         SUM(SSTrajeIECTS.durationSem)
    FROM SSRecursive, studyStructure SSnaziv, studyStructure SSTrajeIECTS
   WHERE SSRecursive.studyStructureID = SSnaziv.studyStructureID
     AND (SSRecursive.studyStructureSuperiorID = SSTrajeIECTS.studyStructureID OR
          SSRecursive.studyStructureSuperiorID IS NULL AND
          SSRecursive.studyStructureID = SSTrajeIECTS.studyStructureID)
     AND  SSRecursive.studyStructureID NOT IN (SELECT DISTINCT studyStructureSuperiorID
                                               FROM studyStructure
                                              WHERE studyStructureSuperiorID IS NOT NULL)
     GROUP BY SSRecursive.studyStructureID, SSnaziv.studyStructurename

OR

WITH RECURSIVE SSRecursive (studyStructureID, studyStructurename,
studyStructureSuperiorID, durationSem, ECTSCredits) AS
(SELECT studyStructureID, studyStructurename, studyStructureSuperiorID, durationSem,
ECTSCredits
   FROM studyStructure
  UNION
  SELECT SSRecursive.studyStructureID, SSRecursive.studyStructurename,
         studyStructure.studyStructureSuperiorID,
         SSRecursive.durationSem + studyStructure.durationSem,
         SSRecursive.ECTSCredits + studyStructure.ECTSCredits
    FROM SSRecursive, studyStructure
   WHERE SSRecursive.studyStructureSuperiorID = studyStructure.studyStructureID
  )
  SELECT studyStructurename, durationSem, ECTSCredits
    FROM SSRecursive
```

```
//in the last iteration the root studyStructure element (the one without superior
element) will be reached and at this point values SSRecursive.durationSem and
SSRecursive.ECTSCredits will be final
    WHERE SSRecursive.studyStructureSuperiorID IS NULL
    AND studyStructureID NOT IN (SELECT DISTINCT studyStructureSuperiorID
                                 FROM studyStructure
                                WHERE studyStructureSuperiorID IS NOT NULL)
```

6. (4 points)

Parser

- Breaks document into tokens and corresponding token type
- Token types that the parser recognizes are pre-defined
- Do not modify the original text
- Default PosgreSQL parser recognizes 23 types: asciiword, word, numword,email, protocol, url, host, file, tag, blank,…

Dictionaries

- Used to eliminate words that should not be considered in a search (stop words), and to normalize words so that different derived forms of the same word will match
- Normalization and removal of stop words
- reduce the size of the TSVector representation of a document
- improve search quality

A text search configuration binds a parser with a set of dictionaries to process the parser's output tokens. For each token type, a list of dictionaries and their order is specified

- Dictionaries are being consulted according to stated order
- When consulted, dictionary
    - recognizes word as a known word further dictionaries are not consulted
    - don't recognize the word, token is being passed to next dictionary
- General rule: at first place the most specific dictionary, then the more general dictionaries, finishing with a very general dictionary, like a Snowball stemmer or simple, which recognizes everything.

7. (2 points)

Valid time : The time in the real world when some event happened or in which a certain fact is true, independent of the time when information about that event/fact is stored in a database

Transaction time: The time when certain information was stored in a database or time interval in which a database is in a certain state