Napredni modeli i baze podataka - međuispit 01.12.2015.

U svim zadacima pretpostavlja se korištenje PostgreSQL SUBP-a i baze podataka sa slike 1.

Baza podataka je namijenjena pohrani podataka vezanih uz izbor zastupnika u Hrvatski sabor (prošlih i budućih). Pohranjuju se podaci o političkim strankama (***party***) i njihovim članovima (***person)*** te rezultatima izbora: glasovi (***vote***) za kandidate se evidentiraju po biračkim mjestima (***electPlace***) koja pripadaju izbornim jedinicama (***electUnit***) i gradovima (***city***). Grad ne pripada nužno samo jednoj izbornoj jedinici, npr. Zagreb je podijeljen na četiri izborne jedinice.
Rezultati izbora se objavljuju prema izbornim jedinicama i kandidacijskim listama (***electList).*** Kandidacijske liste u nekim slučajevima objedinjavaju više političkih stranaka.

electUnit

unitId	***unitName***	***UnitShort Name***
1	I. izborna jedinica	I
2	II. izborna jedinica	II
…	…	

city

cityId	***cityName***	***population***
1	Zadar	71471
2	Split	167121
…	…	…

electPlace

placeId	***placeName***	***unitId***	***cityId***
1001	OŠ Tina Ujevića	7	10
1002	Mjesna zajednica Kruge	2	10
…	…	…	

electList

listId	***listName***
1	SDP + HNS + HSU + SR+ HSS + ZS
2	HDZ + HSS + HSP AS + BUZ +…
…	…

party

partyId	***partyName***	***shortName***	***listId***	***…***
88	Socijaldemokratska partija Hrvatske	SDP	1	…
245	Most nezavisnih lista	MOST	3	…
…	…	…		…

person

personId	***fName***	***lName***	***partyId***	***…***
100	Zoran	Milanović	88	…
104	Drago	Prgomet	245	…
…	…	…	…	…

vote

voteId	***placeId***	***personId***	***electDate***
100001	1001	104	7.11.2015
100002	1001	101	7.11.2015
…	…	…	

slika 1

1. (**2 boda**) Rezultate izbora je potrebno redundantno pohraniti u relaciju ***electResultOR***.

electResultOR

electDate	***unitShort Name***	***listId***	***listName***	***listMembers***
7.11.2015	I	1	SDP + HNS + HSU + SR+ HSS + ZS	{(1, Zoran, Milanović, 49379), (2, Vesna, Pusić, 5870), … }
7.11.2015	I	3	Most nezavisnih lista	{(1, Drago, Prgomet, 21438), (2, Gordana, Rusak, 1209), … }
…		…	…	…

Napisati SQL naredbu(e) za kreiranje relacije ***electResultOR.*** Svaki element kolekcije ***listMembers*** se sastoji od:

- rednog broja kandidata na listi u izbornoj jedinici,
- imena kandidata,
- prezimena kandidata,
- osvojenog broja glasova u toj izbornoj jedinici i kandidacijskoj listi

Tipove podataka odaberite proizvoljno.

2. (**4 boda**) Napisati SQL naredbu kojom će se ispisati rezultati izbora obavljenih '7.11.2015' u sljedećem obliku:

electDate	***unitShortName***	***listId***	***ordinal***	***personId***	***FName***	***LName***	***noOfVotes***
7.11.2015	I	1	1	100	Zoran	Milanović	49379
7.11.2015	I	1	2	102	Vesna	Pusić	5870
		…	…		…	…	…

Redni broj kandidata na listi u izbornoj jedinici (***ordinal)*** se odredi temeljem broja osvojenih glasova. Pretpostavite da dva kandidata na istoj kandidacijskoj listi u istoj izbornoj jedinici neće imati jednak broj glasova.

3. (**4 boda**) Pretpostavite da su rezultati upita iz zadatka 2 pohranjeni u privremenu relaciju ***electResultTemp*** (bez obzira na točnost Vašeg rješenja)**.** Temeljem sadržaja relacija sa slike 1 i relacije ***electResultTemp*** napunite sadržajem relaciju ***electResultOR***.

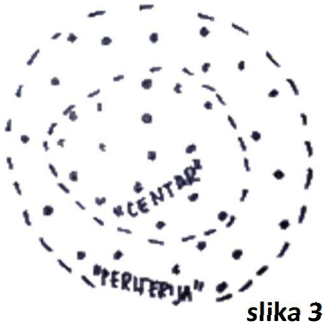
4. (**4 boda**) Potrebno je odrediti sličnost između znakovnih nizova: MAJKA i MAČKA metodom Q-grama. Možete se, ali i ne morate, rukovoditi implementacijom ove metode u PostgreSQL SUBP. Odredite sličnost za:
   a) Q=2
   b) Q=3
   Koji Q smatrate boljim izborom u ovom konkretnom slučaju? Obrazložite odgovor.

5. (**4 bodova**) Objasnite pojmove vrijeme valjanosti i relacija vremena valjanosti.
   Relacija ***party*** trenutno ne podržava činjenicu da se političke stranke na različitim izborima ne pojavljuju uvijek na istim kandidacijskim listam odnosno, ona nije relacija vremena valjanosti. Opišite što biste promijenili u shemi relacije ***party*** sa slike 1 tako da postane relacija vremena valjanosti.
   Obavezno naznačite ključ u novoj shemi. Komentirajte kako se štiti integritet ključa pri unosu nove n-torke u relaciju.
   Napišite SQL naredbe za promjenu sheme relacije ***party.*** Trenutno ograničenje ključa je imenovano s ***pkPartyMember***.

6. Pretpostavimo da je vaš zadatak analizirati i vizualizirati rezultate izbora. Ovdje nećemo doista vizualizirati već analizirati kako organizirati i prirediti podatke odgovarajućeg tipa potrebne za vizualizaciju. Rješenja pišite u formi pseudokoda i SQL naredbi, te komentara, pri čemu **navedite GIS funkcije koje planirate koristiti**. Možete smatrati da su vam na raspolaganju svi mogući dodatni podatci.
   Potrebno je:
   (a) (**3 boda**) Za jednu stranku (bilo koju), prikazati **uspjeh** te **stranke** u nijansama neke boje **po izbornim jedinicama** (npr. slika 2). Uzmite u obzir da nemaju sve izborne jedinice isti broj glasača te stoga odaberite prikladnu mjeru uspješnosti.
   Potrebno je opisati potencijalne izmjene na bazi u pogledu dodatnih podataka, te SQL naredbe kojima se pripremaju podatci potrebni za vizualizaciju.

slika 2

   (b) (**4 boda**) Za grad Zagreb je potrebno vizualizirati izborne rezultate s obzirom na udaljenost **glasačkih mjesta** od „centra“ Zagreba – potrebno je razdijeliti glasačka mjesta na dva skupa – „centar“ i „periferija“ koji sadrže približno jednak broj izbornih mjesta (skica na slici 3). Opišite eventualne izmjene na bazi koje je potrebno napraviti, te opišite kako biste razvrstali glasačka mjesta u dva navedena skupa, te pritom dobili odgovarajući geometrijski tip podatka koji može poslužiti za vizualizaciju.



slika 3

Rješenja:

1. (2 boda)

```
CREATE TYPE ListMember AS
  (ordinal          INTEGER,
  fName             VARCHAR(25),
  LName             VARCHAR(25),
  noOfVotes         INTEGER
 );

CREATE TABLE electResultOR(
   electDate      DATE,
   unitShortName  VARCHAR(10),
   listId         INT REFERENCES electList(listId),
   listName       VARCHAR(300) NOT NULL,
   listMembers    listMember[],
PRIMARY KEY (electDate, unitShortName, listId)
)
```

2. (4 boda)

```
SELECT vote.electDate, electUnit.unitShortName, party.listId,
       rank() OVER (PARTITION BY party.listId, electPlace.unitId
                    ORDER BY COUNT(*) DESC) ordinal
     , person.personId, FName, LName, COUNT(*) noOfVotes
  -- INTO temp table electResultTemp
     FROM vote, person, electPlace, party, electUnit
     WHERE vote.personId = person.personId
     AND vote.placeId = electPlace.placeId
     AND person.partyId = party.partyId
     AND electPlace.unitId = electUnit.unitId
     AND vote.electDate = '07.11.2015'
    GROUP BY vote.electDate, electUnit.unitShortName, party.listId,
             electPlace.unitId,
             FName, LName, person.personId
```

3. (4 boda)

```
INSERT INTO electResultOR
SELECT electDate, unitShortName, electResultTemp.listId, electList.listName,
       array_agg((ordinal, FName, LName, noOfVotes)::ListMember)
  FROM electResultTemp, electList
 WHERE electResultTemp.listId = electList.listId
GROUP BY electDate, unitShortName, electResultTemp.listId, electList.listName
```

4. (4 boda)

```
Za Q=3
   a) ako se ne dodaju blankovi niti na kraj niti na početak znakovnog niza
      MAJKA = {MAJ, AJK, JKA}
      MAČKA = {MAČ, AČK, ČKA}
      sličnost = 0/6=0
   b) ako se dodaju 2 blanka na početak i 1 na kraj znakovnog niza (tako
      Postgres radi):
      MAJKA = {_ _M, _MA, MAJ, AJK, JKA, KA_}
      MAČKA = {_ _M, _MA, MAČ, AČK, ČKA, KA_}
      sličnost = 3/(6+6-3)=3/9=0.333
Za Q=2
   a) ako se ne dodaju blankovi niti na kraj niti na početak znakovnog niza
      MAJKA = {MA, AJ, JK, KA}
```

```
   MAČKA = {MA, AČ, ČK, KA}
   sličnost = 2/(4+4-2)=2/6= 0.333
b) ako se doda 1 blank na početak i 1 na kraj znakovnog niza:
   MAJKA = {_M, MA, AJ, JK, KA, A_}
   MAČKA = {_M, MA, AČ, ČK, KA, A_}
   sličnost = 4/(6+6-4)=4/8=0.5
```

Budući da metoda Q-grama ne vodi računa o značenju i semantici riječi Q=3 +a) nikako nije dobar izbor jer se ta dva niza razlikuju samo u jednom slovu, a sličnost je 0.
Q=3 + b) i Q=2 + a) daju jednak rezultat s tim da u je drugom slučaju postupak brži (manje je dvoslovčanih podskupova od troslovčanih).

Odabrala bih (samo u ovom konkretnom slučaju) postupak Q=2 + b) jer je tu izračunata sličnost 0.5 što mi se čini primjerenije od 0.333

5. (**4 bodova**)
Vrijeme valjanosti je vrijeme u stvarnom svijetu kada se neki događaj dogodio ili period u kojem je neka činjenica važeća, nezavisno od trenutka kada je informacija o tom događaju/činjenici zapisana u bazu podataka.

Relacije vremena valjanosti su relacije koje prate promjene u podacima duž osi vremena valjanosti. One ne sadrže samo trenutno stanje i trenutno važeće n-torke nego i njihovo stanje

Da bi ***party*** postala relacija vremena valjanosti treba:

1. Dodati u shemu atribut ***dateFromTo*** dateRange - period datuma
2. Uništiti stari primarni ključ
3. Postaviti ***dateFromTo*** na neku vrijednost jer PK ne može imati NULL vrijednost atributa
4. Kreirati novi PK = {***partyId, dateFromTo***}
5. Kreirati EXCLUDE ograničenje koje će štiti integritet ključa

Kako se štiti integritet ključa pri unosu nove n-torke: INSERT nije dozvoljen ako postoji n-torka s jednakom vrijednošću atributa ***partyId*** i periodom valjanosti ***dateFromTo*** koji se preklapa barem u jednom vremenskom trenutku (a to znači datumu) s periodom n-torke koju želimo INSERT-irati.

```
ALTER TABLE party DROP CONSTRAINT pkPartyMember
ALTER TABLE party ADD dateFromTo dateRange;
UPDATE party SET dateFromTo = '[dd.mm.yyyy, dd.mm.yyyy)';
ALTER TABLE party ADD CONSTRAINT pkPartyMemeber
PRIMARY KEY (partyId, dateFromTo) ;
```

PRIMARY KEY ograničenje neće očuvati temporalni primarni ključ na način kako je gore opisano. U PostgreSQL-u se takvo ograničenje implementira deklarativno pomoću tzv EXCLUDE ograničenja:

```
CREATE EXTENSION bTree_gist;
ALTER TABLE party ADD CONSTRAINT noOverlapParty EXCLUDE USING gist (partyId WITH
=, dateFromTo WITH &&) ;
```

6. a) (3 boda)
Potrebno je dodati geometry tip u izbornu jedinicu electUnit i naravno ažurirati podatke. Uspjeh u izbornoj jedinici definiramo kao broj glasača za stranku dijeljeno broj glasača. Ne koristiti populaciju – nije dobro (npr. što ako 10% ljudi izađe na izbore i svi glasaju za jednu stranku – uspjeh bi bio 10%)!
Sa sljedećim upitom dohvatiti podatke, za npr. partyId = **<XX>**:

```
SELECT electUnit.unitId, UnitShortName,
     1. * SUM(CASE WHEN partyId = <XX> THEN 1 ELSE 0 END) / COUNT(*) as perc
 FROM vote
  JOIN electPlace ON vote.placeId = electPlace.placeId
  JOIN electUnit ON electPlace.unitId = electUnit.unitId
  JOIN person ON vote.personId = person.personId
```

```
-- nisam stavio WHERE partyId=<XX> jer mi trebaju svi, da dobijem postotak
GROUP BY electUnit.unitId, UnitShortName
```

Gornji upit spojiti s electUnit, kako bi se dobili podatci (relacija): **geom, unitshortname, perc**. Ti podatci su onda ulaz u bilo koji GIS alat koji to može vizualizirati.

Napomena: u gornji upit se ne može staviti geometrijski tip podatka jer se po njemu ne može grupirati (iako bi se i takvo rješenje priznalo). Alternative su privremene tablice, view ili čak dinamički spojiti grupiranu tablicu s normalnom tablicom itd., sve se priznaje.

Postotak se mogao dobiti i na složeniji način, koristeći particioniranje, npr.:

```
SELECT electPlace.UnitId, person.partyId, COUNT(*) /
 SUM(COUNT(*)) OVER (PARTITION BY electPlace.UnitId)
  FROM vote
  JOIN electPlace ON vote.placeId = electPlace.placeId
  JOIN person ON vote.personId = person.personId
GROUP BY electPlace.UnitId, person.partyId
```

b) (4 boda) Potrebno je dodati point geomtrijski tip podatka u electPlace, te potom napraviti st_convexhull() od svih biračkih mjesta koja spadaju pod Zagreb.

Označimo je s:

```
CXZagreb = st_convexhull(  -- U PostGisu ovdje treba ići ST_collect, ali nema veze
    SELECT geom FROM electPlace
       JOIN city ON electPlace.cityId = city.cityId
       WHERE cityName = 'Zagreb'
      )
```

Označimo broj izbornih mjesta u Zagrebu s:

```
NZ =  SELECT COUNT(*)
            FROM electPlace
            JOIN city ON electPlace.cityId = city.cityId
            WHERE cityName = 'Zagreb'
```

Novonastalu geometriju je potrebno postupno smanjivati s **st_buffer** i negativnim radijusom, dok ne pronađemo otprilike pola glasačkih mjesta:

```
R = -1;
Preth = NZ;
while (1) {
   Centar = ST_Buffer(CXZagreb, R)
   NC =  SELECT COUNT(*) FROM electPlace WHERE st_within(geom, Centar);
   if (abs(NC - NZ/2) > abs(Preth - NZ/2)) {
        R = R + 1;
        break;
   }
   Preth = NC;
   R = R - 1;
}
// Na koncu periferiju izračunamo kao razliku ukupnog područja i centra:
Centar = ST_Buffer(CXZagreb, R);
Periferija = ST_Difference(CXZagreb, Centar);
```