Napredni modeli i baze podataka - međuispit 21.11.2016.

U zadacima 1. – 4. pretpostavlja se korištenje PostgreSQL SUBP-a i baze podataka sa slike 1.

Baza podataka je namijenjena pohrani podataka vezanih uz poslovanje autobusnog kolodvora. Evidentiraju se podaci o autobusnim postajama i linijama te dionicama pojedinih linija. Pojedine dionice mogu biti zajedničke za više linija(***dionicaLinija***). Vozni red se evidentira za svaku dionicu linije. Autobusna karta se može kupiti za bilo koju odlaznu i dolaznu postaju kroz koje linija prolazi.
U tablicama ***zupanija*** i ***suma*** su pohranjeni geografski podaci o Hrvatskoj.

postaja

postajaId	***nazivPostaja***	***geom***
1	Zagreb	<point>
2	Karlovac	<point>
3	Slunj	<point>
…	…	…
50	Split	<point>

linija

linijaId	***nazivLinija***
1	Zagreb-Split
2	Karlovac-Makarska
3	Varaždin-Zadar
…	…
50	Trogir - Split

dionica

dionicaId	***postajaOdId***	***postajaDoId***	***geom***
1	1	2	<polyline>
2	2	3	< polyline >
3	3	4	< polyline >
	…		…

dionicaLinija

linijaId	***dionicaId***
1	2
2	3
3	4
…	

redVoznje

linijaId	***dionicaId***	***vrijemePolazak***	***cijena***
1	1	2016-11-19 08:00.000	25
1	1	2016-11-20 08:00.000	30
…		…	…
120		2016-11-19 16:15.000	42

kupnja

kupnjaId	***postajaOdId***	***postajaDoId***	***linijaId***	***vrijemePolazak***
1023568	1	50	1	2016-11-19 08:00:00.0
1023568	2	3	2	2016-11-19 08:00:00.0
…				

zupanija

zupId	***nazZup***	***geom***
1	Grad Zagreb	<polygon>
2	Zadarska	<polygon>
3	Splitska	< polygon >
…	…	…

suma

sumaId	***geom***
1	<polygon>
2	<polygon>
3	< polygon >
…	…

Slika 1

1. (**3 boda**) Napisati SQL naredbu kojom će se ispisati podaci o broju kupljenih autobusnih karata u sljedećem obliku:

postajaOd	***postajaDo***	***polazakOdDo***	***brKupnji***	***brKupnjiKumul***
…	…	…	…	…
Zagreb	Split	7:00 -7:59	150	400
Zagreb	Split	8:00 -8:59	300	700
…	…	…	…	…

brKupnji je broj kupljenih autobusnih karata (bez obzira na datum) za navedenu polaznu i dolaznu postaju na bilo kojoj liniji s polaskom u navedenom periodu (***polazakOdDo***).
brKupnjiKumul je kumulativan broj kupljenih karata za navedenu polaznu i dolaznu postaju na bilo kojoj liniji s polaskom od početka dana do kraja promatranog perioda.
Pomoć: EXTRACT (HOUR FROM vrijemePolazak) će izdvojiti sat iz TIMESTAMP atributa vrijemePolazak u PostgreSQL-u.

2. (**5 bodova**) Za osobu koja želi kupiti autobusnu kartu za putovanje od autobusne postaje 'Zagreb' do postaje 'Split' na dan 21.11.2016. potrebno je ispisati sve moguće autobusne rute zajedno s datumom i satom polaska te cijenom karte u sljedećem obliku:

datumPolazak	***vrijeme***	***cijena***	***planPuta***
21.11.2016	9:00	180	Zagreb-->Karlovac-->Plitvička jezera-->Korenica…Trogir-->Split
21.11.2016	12:00	220	Zagreb--> Plitvička jezera-->Split
21.11.2016	9:15	149	Zagreb--> Split
…	…	…	…

Cijena se izračuna zbrajanjem cijena pojedinih dionica.
Pomoć: EXTRACT (MINUTE FROM vrijemePolazak) će izdvojiti minute iz TIMESTAMP atributa vrijemePolazak.

3. (**3 boda**) Napisati SQL naredbu kojom će se, za svaku županiju, ispisati koliko ukupno kilometara cesta sadrži. Pretpostaviti da se dionice ne ponavljaju u tablici ***dionica.***

4. (**5 bodova**) Potrebno je na karti Hrvatske vizualizirati sve raspoložive autobusne linije, pri čemu dionice treba obojiti nijansama zelene boje s obzirom na to koliko je dionica udaljena od neke šume, npr. dionice bliže šumi prikazati svjetlijim nijansama zelene, a udaljenije tamnijim. Za udaljenost dionice od šume uzmite najjednostavniju mjeru – minimalnu udaljenost (a ne, npr. prosječnu).
   Dodatno:
   a. potrebno je naznačiti kojim linijama pripadaju dionice (debela crna crta na donjoj slici)
   b. osigurati vidljivost dionica neovisno o alatu koji će se koristiti za vizualizaciju, za QGis npr. smatrati da nijedna od mogućih debljina linije nije dovoljna za uočljivu vizualizaciju

   Rješenje napišite u formi pseudokoda i SQL naredbi, te komentara, pri čemu **navedite GIS funkcije koje planirate koristiti**.
   Opišite kako biste priredili podatke za tu vizualizaciju (možete mijenjati tablice ili dodavati nove tablice, atribute i sl.), te što je potrebno s tim podatcima napraviti u alatu za vizualizaciju (QGis).
   Donja slika, za ilustraciju, prikazuje ručno napravljenu vizualizaciju s dvije autobusne linije (Zg-Ri i Zg-Gospić) sastavljene od više dionica u raznim nijansama zelene boje. Pojedine dionice su zajedničke.

5. **(3 boda)** Navedite nedostatke standardnog SQL-a (npr. operatori LIKE, NOT LIKE, SIMILAR TO,…, indeksiranje pomoću B-stabla) u pretraživanju teksta temeljem morfologije, sintakse i semantike jezika. Objasnite što podrazumijeva kvalitetna priprema tekstualnih sadržaja za kasnije efikasno pretraživanje teksta temeljem morfologije, sintakse i semantike jezika.

6. (**2 boda**) Neformalno opišite 2PC *(two-phase commit)* protokol, čemu služi i kada se koristi, te koja su mu svojstva (ne morate reproducirati dijagram).

7. (**2 boda**) Što je shema distribucije kod DSUBP-a i od čega se sastoji. Objasnite na primjeru.

8. (**2 boda**) Objasnite svojstvo D iz akronima ACID, te objasnite kako se ono ostvaruje u relacijskim bazama podataka?

Rješenja:

1.

```
SELECT postajaOd.nazivPostaja, postajaDo.nazivPostaja,
       EXTRACT (HOUR FROM vrijemePolazak) || ':00 -' ||
       EXTRACT (HOUR FROM vrijemePolazak) || ':59',
       COUNT(*),
       SUM(COUNT(*)) OVER (PARTITION BY postajaOd.postajaId, postajaDo.postajaId
                           ORDER BY EXTRACT (HOUR FROM vrijemePolazak))
  FROM kupnja, postaja postajaOd, postaja postajaDo
 WHERE kupnja.postajaOdId = postajaOd.postajaId
   AND kupnja.postajaDoId = postajaDo.postajaId
GROUP BY postajaOd.postajaId, postajaOd.nazivPostaja, postajaDo.postajaId,
postajaDo.nazivPostaja, extract (hour from vrijemePolazak)
order by postajaOd.nazivPostaja, postajaDo.nazivPostaja, extract (hour from
vrijemePolazak);
```

Rješenja koja NISU ispravna:

Krivo je u gornjem upitu ***brKupnji*** računati pomoću:

a) `COUNT(*) OVER (PARTITION BY postajaOd.postajaId, postajaDo.postajaId)`
zbog toga što će za svaki izlazni redak koji pripada istoj particiji (jednake `postajaOd.postajaId, postajaDo.postajaId`) rezultirati jednakim COUNT-om, a on je (zbog `GROUP BY postajaOd.postajaId, postajaOd.nazivPostaja, postajaDo.postajaId, postajaDo.nazivPostaja, EXTRACT (HOUR FROM vrijemePolazak)`) jednak broju grupa s jednakim vrijednostima `postajaOd.postajaId, postajaOd.nazivPostaja, postajaDo.postajaId, postajaDo.nazivPostaja,`

b) `COUNT(*) OVER (PARTITION BY postajaOd.postajaId, postajaDo.postajaId, EXTRACT (HOUR FROM vrijemePolazak)),`
zbog toga što taj COUNT rezultira s vrijednošću 1 za redak koji pripada toj particiji (jednake `postajaOd.postajaId, postajaDo.postajaId, EXTRACT (HOUR FROM vrijemePolazak)`)

Krivo je u gornjem upitu ***brKupnjiKumul*** računati pomoću:

a) `COUNT(*) OVER (PARTITION BY postajaOd.postajaId, postajaDo.postajaId ORDER BY EXTRACT (HOUR FROM vrijemePolazak))`
zato što će za svaki izlazni redak koji pripada istoj particiji (jednake `postajaOd.postajaId, postajaDo.postajaId`) rezultirati jednakim COUNT-om, a on je (zbog `GROUP BY postajaOd.postajaId, postajaOd.nazivPostaja, postajaDo.postajaId, postajaDo.nazivPostaja, EXTRACT (HOUR FROM vrijemePolazak)`) jednak broju grupa s jednakim vrijednostima `postajaOd.postajaId, postajaOd.nazivPostaja, postajaDo.postajaId, postajaDo.nazivPostaja,`

2. Budući da putnik kupuje kartu na istoj liniji za 1 ili više dionica u donjem upitu treba voditi računa da dionice pripadaju istoj liniji i da se u skladu s tim (linijaId+dionicaId) odredi cijena dionice (za liniju) koja ulazi u ukupnu cijenu autobusne karte.

```
WITH RECURSIVE sveDionice
     (linijaId, postajaOdId, postajaDoId, vrijemePolazak, planPuta, ukCijena)

  AS
   ( (SELECT dionicaLinija.linijaId,
             NULL::INTEGER,
             dionica.postajaOdId,
             NULL::timestamp,
             CAST (postaja.nazivPostaja AS VARCHAR(600)),
             0::float,
             FROM dionica, dionicaLinija, postaja
            WHERE dionicaLinija.dionicaId = dionica.dionicaId
              AND dionica.postajaOdId = postaja.postajaId
     )
     UNION
    (SELECT sveDionice.linijaId,
             sveDionice.postajaDoId,
             postaja.postajaId,
             redVoznje.vrijemePolazak ::timestamp,
             CAST (sveDionice.planPuta || '-->' ||
             CAST (SUBSTR(postaja.nazivPostaja, 1,20)  AS CHAR(20))
```

```
              AS VARCHAR(600)),
            sveDionice.ukCijena + redVoznje.cijena,
       FROM sveDionice, dionicaLinija, dionica, postaja, redVoznje
      WHERE dionicaLinija.linijaId = sveDionice.linijaId
        AND dionicaLinija.dionicaId = dionica.dionicaId

        AND redVoznje.linijaId = dionicaLinija.linijaId
        AND redVoznje.dionicaId = dionica.dionicaId

        AND sveDionice.postajaDoId = dionica.postajaOdId
        AND dionica.postajaDoId = postaja.postajaId
      )
    )
SELECT vrijemePolazak::DATE, EXTRACT(HOUR FROM vrijemePolazak):: CHAR(2) || ':' || EXTRACT
(MINUTE FROM vrijemePolazak)::Char(2), ukCijena, ukKM, planPuta
  FROM sveDionice
 WHERE sveDionice.planPuta LIKE Zagreb%Split
   AND sveDionice.vrijemePolazak::DATE = '19.11.2016'
```

3.

```
SELECT nazZup, SUM(st_length(st_intersection(zupanija.geom, dionica.geom))) / 1000
FROM zupanija join dionica
       On st_intersects(zupanija.geom, dionica.geom)  -- zapravo nije nužan uvjet, može i
cross join
Group by nazZup
```

4.

-- Dodajmo u tablicu dionica minimalnu udaljenost do prve šume:

```
Alter table dionica add dist_suma decimal(10,2)
Alter table dionica add dist_suma_rel decimal(10,2)

Update dionica
Set dist_suma =
 (select min(st_distance(suma.geom, dionica.geom))) from suma
```

--postavljamo relativni faktor, pri čemu je 0 najudaljenija dionica(tamno), a 1 ona koja (ako) dira šumu

```
Update dionica
 dist_suma_rel = 1 - dist_suma/(select max (dist_suma) from dionica)
```

--linijama dodajemo geomeriju koja je unija svih dionica:

```
Alter table linija add geom <polyline>
Update linija
   SET geom = st_union(ARRAY(SELECT geom
                          FROM dionicaLinija, dionica
                         WHERE dionicaLinija.dionica.ID = dionica.dionicaId
                           AND dionicaLinija.linijaId = linija.linijaId))
```

i konačno, u QGIsu prikazati:

- bafere od dionica (npr. 20 metara šire) u nijansama zeleno/crne boje, kao što smo radili na projektu:

```
SELECT dionicaId, st_buffer(geom, 20), dist_suma_rel from dionica
```

- a za linije ću isto iskoristiti buffer, ali veći od ovog kojem ću ukloniti unutrašnjost:

```
SELECT linijaId, ST_Difference(st_buffer(geom, 30), st_buffer(geom, 25)) from linija
```

5. – 8.

vidi predavanja