

Napredni modeli i baze podataka

By Vegeta



Full text search

2. (6 bodova)

- a) Na primjeru relacije **book** objasnite zbog čega standardni indeks (kao što je **contentIdx**) nije pogodan za pretraživanje teksta u RSUBP.

```
CREATE TABLE book (
    bookId INTEGER PRIMARY KEY,
    title VARCHAR (300) NOT NULL,
    content TEXT);
CREATE INDEX contentIdx ON
    book (content);
```

bookId	title	content
88	Beauty and the Beast	Once upon a time, there was a girl named...
563	Snow White	Long ago, there lived a girl ...

- b) Uvezši u obzir funkcionalnosti PostgreSQL SUBP predložite izmijene sheme relacije **book** tako da nova shema omogući efikasno pretraživanje sadržaja knjiga (**content**) temeljem morfologije, sintakse i semantike engleskog jezika. Nije potrebno pisati SQL naredbe, dovoljno je riječima opisati promjene.

RJEŠENJE

2.

- a) Zbog algoritma pretraživanja B-stabla, pomoću indeksa `bookContentIdx`, moguće je obavljati samo upite koji traže zapise s točno jednakim sadržajem atributa `content` ili zapise koji imaju na početku atributa traženi uzorak. Npr.:

```
SELECT * FROM book WHERE content LIKE 'Once%';  
SELECT * FROM book WHERE content LIKE '%Once %';
```

koristi indeks

NE koristi indeks

Takvi upiti nisu realni pa ni indeks ne pomaže.

Pomoću ovakvog indeksa nije moguće obaviti pretraživanje koje bi vodilo računa o:

stop riječima,
različitim pojavnim oblicima riječi s istim korijenom/osnovom,
različitim riječima jednakog značenja,

kao niti pretraživanje koje bi primjenjivalo neku od tehnika približne pretrage teksta kao što je npr. pretraga pomoću Q-gram algoritama.

b)

1. dodati u relaciju `book` novi atribut `content_tsv` tipa `TSVECTOR` koji bi čuvaо `content` u normaliziranom obliku
2. kreirati trigger za `INSERT` i `UPDATE` on `book` koji bi održavao ažurnim vrijednost tog atributa koja se može se dobiti pomoću funkcije `to_tsvector('english', content)`
3. Kreirati invertirani indeks nad atributom `content_tsv`:

```
CREATE INDEX contentIdx2 ON book USING gin(bookContent_tsv)
```

1. **(5 bodova)** Objasnite pojam približnog pretraživanja teksta (Fuzzy Text Search) te navedite barem tri vrste algoritama koje se koriste pri približnom pretraživanju teksta.

RJEŠENJE

Rješenja:

1. Tehnika pronalaženja znakovnog niza koji se približno podudara s traženim uzorkom

Kvaliteta podudaranja se mjeri brojem operacija koje je potrebno obaviti nad znakovnim nizom da bi se u potpunosti podudario s traženim uzorkom

Vrste algoritama:

- algoritmi temeljeni na udaljenosti između znakovnih nizova
 - Hamming,
 - Levenshtein – PostgreSQL: postoji f-ja Levenshtein
 - Q-Gram algoritmi
 - slični znakovni nizovi imaju mnogo zajedničkih Q-Gram-ova (podskupovi znakovnog niza duljine Q) – PostgreSQL: podržano, tekst rastavlja na trigrame similarity(text, text), operator %
 - algoritmi koji traže riječi koje se slično izgovaraju
 - Soundex, Metaphone algoritam – PostgreSQL postoje i soundex i metaphone funkcije
-

5. (4 boda) Objasnite zbog čega standardni indeks koji koristi strukturu balansiranog stabla nije pogodan za pretraživanje teksta (full text search) u RSUBP. Objasnite na koji način funkcioniра invertirani indeks i zbog čega predstavlja bolje rješenje od standardnog indeksa u kontekstu pretraživanja teksta.

RJEŠENJE

5. (4 boda)

Pomoću standardnog indeksa koji koristi strukturu B stabla, indeksa, zbog ugrađenih algoritma pretraživanja moguće je obavljati samo upite koji traže zapise s točno jednakim sadržajem atributa nad kojim je indeks izgrađen ili zapise koji imaju na početku atributa traženi uzorak. Npr.:

```
SELECT * FROM book WHERE content LIKE 'Once%';
```

koristi indeks

```
SELECT * FROM book WHERE content LIKE '%Once %';
```

NE koristi indeks

Takvi upiti nisu realni pa ni indeks ne pomaže. Dodatno, ako promotrimo gornji primjer, atribut s `book.content` će vrlo vjerojatno biti različit za svaku n-torku (koliko ima knjiga identičnog sadržaja?). To znači da će taj neupotrebljivi indeks vjerojatno biti UNIQUE i zauzimati puno prostora.

Pomoću ovakvog indeksa nije moguće obaviti pretraživanje koje bi vodilo računa o:

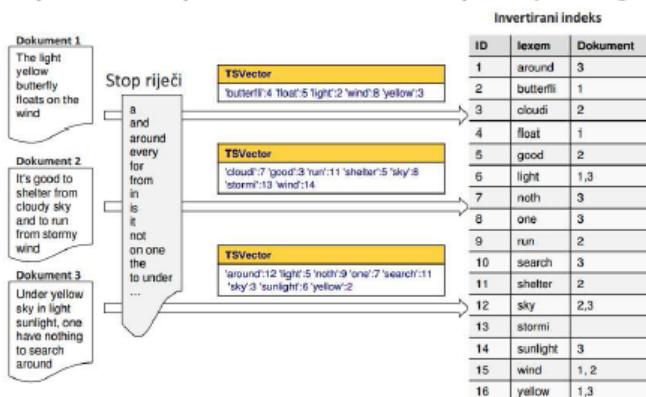
- stop riječima,

- različitim pojavnim oblicima riječi s istim korijenom/osnovom,

- različitim riječima jednakog značenja,

kao niti pretraživanje koje bi primjenjivalo neku od tehnika približne pretrage teksta kao što je npr. pretraga pomoću Q-gram algoritama.

Invertirani indeks s logičkog stanovišta, za svaku riječ koja se pojavljuje u svim vrijednostima atributa nad kojim je definiran, sadrži identifikatore n-torki koji tu riječ sadrže. To omogućuje brzo pronađenje zapisa koji sadrže kombinaciju nekih riječi tj. određeni uzorak. RSUBP kao što je Postgres koji imaju ugrađenu potporu za normalizaciju teksta, omogućavaju izgradnju indeksa nad normaliziranom verzijom sadržaja atributa pa time invertirani indeks postaje upotrebljiv za pretragu koja vodi računa o stop riječima, različitim pojavnim oblicima riječi s istim korijenom/osnovom, različitim riječima jednakog značenja i td.



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.

RJEŠENJE

6. (4 boda)

- Parser je program koji u ulaznom izvornom tekstu identificira tokene i utvrđuje tip tokena.
- Ulazni tekst ne modifcira 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 postiže 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 proslijeđuje sljedećem rječniku
- Obično se na početak liste stavlja najspecifičniji rječnik, potom općenitiji rječnici a na kraj liste najopćenitij rječnik (kao Snowball) koji prepozna svaku riječ

3. (3 boda) Objasnite što podrazumijeva kvalitetna priprema tekstualnih sadržaja za kasnije efikasno pretraživanje cijelog teksta (Full Text Search). Na primjeru PostgreSQL SUBP, objasnite ulogu različitih rječnika pri pretraživanju cijelog teksta.

RJEŠENJE

3. (4 boda)

Efikasna pretraga cijelog teksta podrazumijeva prethodnu obradu teksta (dokumenta):

- parsiranje teksta i rastavljanje na tokene (riječi, brojevi, tagovi, razmak, url,...)
- konvertiranje tokena u lekseme – za to se koristi algoritam prilagođen konkretnom jeziku koji uvažava pravila jezika.
- pohrana obrađenog teksta optimiranog za pretragu
- postojanje indeksa prikladnih za pretragu cijelog teksta (npr. invertirani indeks)

Rječnici trebaju omogućiti:

- definiranje stop riječi, koje ne treba indeksirati
- definiranje sinonima
- stvaranje veza između fraza i pojedinih riječi

PostgreSQL ima nekoliko tipova rječnika (*naravno da studenti ne moraju navesti kako se koji rječnik zove – treba samo navesti što u globalu kolekcija rječnika radi*):

- Simple Dictionary Uklanja stop riječi i velika slova svodi na mala.
- Synonym Dictionary Različite riječi jednakog značenja zamjenjuje reprezentantnom rječju.
- Thesaurus Dictionary Omogućuje prepoznavanje fraza.
- iSpell Dictionary Svodi riječi na normalizirani oblik.
- Snowball Dictionary Algoritamski svodi riječi na korijenski oblik (stemming) i uklanja stop riječi

3. (3 boda) Relacija contract sa shemom CONTRACT (`contractId`, `contractContent`, `contractDate`,...) sadrži $100 \cdot 10^6$ zapisa. Atribut `contractContent` je tipa VARCHAR(5000). Vrijeme obavljanja donjeg upita je neprihvatljivo dugo.

```
SELECT *
  FROM contract
 WHERE TO_tsVector('english', contractContent) @@
      TO_tsquery('english', 'European & independency & I & am');
```

Objasnite zbog čega se upiti gornjeg tipa dugo izvode. Predložite promjene u shemi relacije `contract` s ciljem ubrzanja izvođenja upita. Ako promjene sheme relacije zahtijevaju drugačiji upit, napišite izmjenjeni upit.

RJEŠENJE

3. (4 boda)

Upiti se sporo izvode jer se za 100 miliona zapisa „on thefly“ izračunava vrijednost funkcije `TO_tsVector` – tj određuje sortirana lista leksema kao prezentacija dokumenta nakon provedenog postupka „normalizacije“ teksta sadržanog u atributu `contractContent`.

Upit se može ubrzati tako da se:

1. shema relacije `contract` proširi atributom npr `contractContentTSV` tipa TSVECTOR, koji će sadržavati reprezentaciju teksta u normaliziranom obliku:

```
ALTER TABLE contract ADD contractContentTSV TSVECTOR;
i sadržaj tog atributa ažurira i održava ažurnim pomoću okidača:
CREATE TRIGGER contract_InsUpd_Trigg BEFORE INSERT OR UPDATE ON contract
FOR EACH ROW EXECUTE PROCEDURE tsvector_update_trigger
(contractContentTSV, 'english', contractContent);
```

2. kreira invertirani indeks nad atributom `contractContentTSV`:

```
CREATE INDEX contractContentTSVIdx ON contract
  USING gin(contractContentTSV); //ili gist
```

3. U upitima koristiti `contractContentTSV` umjesto `contractContent`:

```
SELECT *
  FROM contract
 WHERE TO_tsVector('english', contractBody) @@
      TO_tsquery('english', 'European&independency& I & am');
```

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.

RJEŠENJE

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 trošlovčanih).

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

3. (5 bodova)

Navedite i objasnite barem tri kategorije algoritama koje se koriste pri približnom pretraživanju teksta (Fuzzy Text Search). Koje od navedenih kategorija algoritama su primjenjivi bez obzira na jezik i domenu? Naznačite koji od navedenih kategorija algoritama su podržani u PostgreSQL SUBP i na koji način (navedite funkcije/operatore/...).

RJEŠENJE

1. algoritmi temeljeni na udaljenosti znakovnih nizova

- mjera udaljenosti između znakovnih nizova s_1 i s_2 je minimalan broj operacija potrebnih da se jedan niz transformira u drugi. Moguće operacije su: izmjena, umetanje, brisanje znaka.
Problem se svodi na pronalaženje slijeda navedenih operacija kojim će se jedan niz transformirati u drugi uz minimalan trošak.
- PostgreSQL: funkcija Levenshtein

2. Q-Gram algoritmi

- slični znakovni nizovi imaju više zajedničkih Q-Gram-ova (podskupovi znakovnog niza duljine Q)
- primjenjiv bez obzira na jezik i domenu
- PostgreSQL: uspoređuje riječi temeljem podskupova duljine $Q=3$ znaka.
operator %, funkcija similarity i td.
mogućnost kreiranja invertiranog indeksa za tekstualni dokument:
`CREATE INDEX idxName ON tableName USING gist (attributeName gist_trgm_ops);`

3. Soundex, Metaphone algoritam

- Aktualno u jezicima u kojima se izgovor riječi razlikuje od zapisa (ne pretjerano korisno za hrvatski)
- Ideja je dovesti u vezu riječi koje se jednak ili slično izgovaraju ali drugačije zapisuju
- Algoritam riječ predstavlja znakovnim nizom koji prezentira izgovor (zvučanje) riječi
- PostgreSQL: Soundex i Metaphone

Primjenjiv bez obzira na jezik i domenu su algoritmi koji pripadaju kategorijama pod 1 i 2.

Pivotiranje i napredni SQL

1. (6 bodova) Prikazani segment modela podataka trgovacki lanac koristi za praćenje prodaje proizvoda po prodavaonicama. Pretpostavka je da trgovacki lanac trenutno ima 99 prodavaonica.

proizvod		prodaja						...
sifProizv	nazivProizv	...	datum	sifProd	sifProizv	kolicina	iznos	...
15	Clarins maskara		15/01/2015	1	15	1	200	
25	Chanel smokey eyes sjenilo		15/01/2015	1	15	2	400	
35	Shiseido BB krema		16/01/2015	1	35	2	300	
...	...		16/01/2015	1	35	2	300	
			15/01/2015	2	35	2	300	
			15/01/2015	2	15	1	200	
			16/01/2015	2	25	2	200	
			16/01/2015	2	25	3	300	
		
prodavaonica	
sifProd	nazivProd
1	Store 1							
2	Store 2							
...	...							
854	Store 99							

Korištenjem mogućnosti PostgreSQL sustava baza podataka i ugrađenih funkcija za pivotiranje u tom sustavu, napišite SQL upit pomoću kojeg će se ispisati ukupan prihod po proizvodima i poslovnicama u sljedećem obliku:

nazivProizv	Store1	Store2		Store99
Chanel smokey eyes sjenilo		500.00
Clarins maskara	600.00	200.00
Shiseido BB krema	600.00	300.00
...

Bi li napisani (Vaš) upit vraćao ispravne podatke u slučaju:

- da se pojavi jedan (ili više) novih proizvoda
- da se otvoru još jedna (ili više) prodavaonica

Ako upit u nekom od dva gornja slučaja ne bi vraćao ispravne podatke, opišite što bi trebalo promijeniti da vraća ispravne podatke.

RJEŠENJE

```
1) SELECT * FROM crosstab (
    'SELECT CAST(nazivProizv AS char(50)) AS nazivProizv
     , CAST(sifProd AS int) AS sifProd
     , CAST(SUM(iznos) AS DECIMAL(10,2)) AS prihod
     FROM prodaja, proizvod
     WHERE prodaja.sifProizv = proizvod.sifProizv
     GROUP BY prodaja.sifProd, nazivProizv
     ORDER BY nazivProizv, sifProd'
    ,
    'SELECT sifProd FROM prodavaonica ORDER BY sifProd'
)
AS pivotTable (nazivProizv CHAR(50),
               Store1 DECIMAL(10,2),
               Store2 DECIMAL(10,2),
               ...
               Store99 DECIMAL(10,2))
ORDER BY nazivProizv
```

Dodavanjem novih proizvoda upit ispravno radi i nije ga potrebno mijenjati.

Dodavanjem nove prodavaonice upit neće dobro raditi. Sada upit koji su argumenti crosstab funkciji vraćaju podatke za jednu dodatnu prodavaonicu:

upit koji je prvi argument funkcije crosstab vraća podatke za jednu prodavaonicu više
'SELECT sifProd FROM prodavaonica ORDER BY sifProd' vraća jednu n-torku više

i više nisu u skladu sa shemom pivotTable koja očekuje podatke za 99 prodavaonica. Potrebno je promijeniti shemu pivotTable na:

```
AS pivotTable (nazivProizv CHAR(50),
               CosmeticsHygieneNo1 DECIMAL(10,2),
               ...
               CosmeticsHygieneNo99 DECIMAL(10,2),
               CosmeticsHygieneNo100 DECIMAL(10,2))
```

studij		upisGodine				
Sif Studij	nazivStudij	akGod	sifStudent	nastGod	sifStudij	ECTSBod
10	Elektrotehnika i informacijska tehnologija i Računarstvo	2012/13	800	1	10	48
21	Elektrotehnika i informacijska tehnologija	2013/14	100	1	10	56
22	Računarstvo	2013/14	675	1	10	59
31	Programsko inženjerstvo i informacijski sustavi	2013/14	985	2	21	38
...	...	2013/14	155	2	21	60
		2013/14	125	2	22	60
		2013/14	675	3	31	55
		2012/13	800	1	10	48
	

Slika 1

1. **(4 boda)** Ispisati podatke o broju studenata upisanih određene akademske i nastavne godine (**broj1**). Dodatno, ispisati i kumulativni broj upisanih studenata u akademskoj godini (**broj2**). Npr. kumulativni broj studenata upisanih u 2. nastavnu godinu 2013/14 akademske godine uključuje i broj studenata upisanih u 1. nastavnu godinu 2013/14.

akGod	nastGod	broj1	broj2
2012/13	1	1	1
2013/14	1	2	2
2013/14	2	3	5
2013/14	3	1	6

RJEŠENJE

```
1.
SELECT DISTINCT akGodina, nastGodina
    , COUNT(*) OVER (PARTITION BY akGodina, nastGodina)
    , COUNT(*) OVER (PARTITION BY akGodina ORDER BY nastGodina)
FROM upisGodine
ORDER BY akGodina, nastGodina
```

2. **(6 bodova)** Ispisati rang listu uspješnosti temeljem broja osvojenih ECTS bodova u sljedećem obliku:

akGod	nastGod	nazivStudij	projECTS	rang1	rang2	rang3
2013/14	2	Računarstvo	60.000	1	1	1
2013/14	1	Elektrotehnika i informacijska tehnologija i Računarstvo	57.500	1	2	2
2013/14	3	Programsko inženjerstvo i informacijski sustavi	55.000	1	3	3
2013/14	2	Elektrotehnika i informacijska tehnologija	49.000	2	4	4
2012/13	1	Elektrotehnika i informacijska tehnologija i Računarstvo	48.000	1	1	5

ProjECTS je prosječan broj ECTS bodova koje su osvojili studenti određenog studija određene akademske i nastavne godine. **Rang1**, **rang2** i **rang3** se određuju temeljem vrijednosti **projECTS** na sljedeći način:

- **rang1** – rangiranje se provodi unutar akademske godine po nastavnim godinama
- **rang2** – rangiranje se provodi unutar akademske godine (bez obzira na nastavnu godinu)
- **rang3** – rangiranje se provodi globalno (bez obzira na akademsku i nastavnu godinu).

RJEŠENJE

```
2.
SELECT akGodina, nastGodina, nazivStudij, AVG(osvojioECTS)
    , rank() OVER (PARTITION BY akGodina, nastGodina
                    ORDER BY AVG(osvojioECTS) DESC) rang1
    , rank() OVER (PARTITION BY akGodina
                    ORDER BY AVG(osvojioECTS) DESC) rang2
    , rank() OVER (ORDER BY AVG(osvojioECTS) DESC) rang3
FROM upisGodine JOIN studij
    ON upisGodine.sifStudij = studij.sifStudij
GROUP BY akGodina, nastGodina, upisGodine.sifStudij, nazivStudij
ORDER BY rang3 --AVG(osvojioECTS) DESC, akGodina, nastGodina, nazivStudij
```

Baza podataka prikazana na slici 1 služi za evidenciju podataka o rezultatima referendumu koji se održavaju u Ujedinjenom Kraljevstvu Velike Britanije i Sjeverne Irske. Rezultati referendumu se bilježe u relaciji **result**. Osobe koje mogu pristupiti određenom referendumu (zajedno s detaljima kao što su predviđeno glasačko mjesto) evidentirane su u relaciji **personRef**. Teritorijalna podjela države, te glasačka mjesta u teritorijalnim jedinicama evidentirani su u relacijama **terUnit** i **poolPlace**. Ključevi relacija su podrtani.

terUnit					poolPlace				
terUnitId	terUnitName	...	geom	supTerUnitId	poolPlaceId	poolPlaceName	...	terUnitId	
53	England		<polygon>	NULL	184	Islington Town Hall		856	
32	Scotland		<polygon>	NULL	132	Olive Morris House		437	
5478	East England		<polygon>	53	11	York House		976	
9056	London		<polygon>	53	
437	City of Edinburg		<polygon>	32					
856	West London		<polygon>	9056					
876	East London		<polygon>	9056					
...					

personRef				result				
refDate	personId	poolPlaceId	voted	refDate	poolPlaceId	ordinal	voteFor	valid
18.09.2014	234567	132	1	18.09.2014	132	1	1	1
23.06.2016	38954	184	0	23.06.2016	11	2	0	1
23.06.2016	6903423	11	1	23.06.2016	132	105	1	1
23.06.2016	5689045	184	1
...					

person			
personId	FName	LName	...
...

Slika 1.

1. (4 boda) Za teritorijalne jedinice koje nemaju nadređenu jedinicu ispisati naziv teritorijalne jedinice, ukupan broj glasova za i ukupan broj glasova protiv referendumskog pitanja. Glasovima teritorijalne jedinice se pribrajaju glasovi svih teritorijalnih jedinica u njenom sastavu. U obzir uzeti samo važeće glasove.

RJEŠENJE

1. (5 bodova)

```
(SELECT terUnit.terUnitId, subservTerUnitId.terUnitId
   FROM terUnit, terUnitsubservTerUnitId
  WHERE terUnit.terUnitId = subservTerUnitId.supTerUnitId
    AND terUnit.supTerUnitId IS NULL
UNION
SELECT terUnitResuts.terUnitId, subservTerUnitId.terUnitId
   FROM terUnitResuts, terUnitsubservTerUnitId
  WHERE terUnitResuts.terUnitResutsId = subservTerUnitId.supTerUnitId
)
SELECT terUnit.terUnitName, SUM(result.voteFor), SUM(1-result.voteFor)
   FROM terUnitResuts, poolPlace, result, terUnit
  WHERE terUnitResuts.terUnitResutsID = poolPlace.terUnitId
    AND result.poolPlaceId = poolPlace.poolPlaceId
    AND terUnitResuts.terUnitId = terUnit.terUnitId
    AND valid = 1
 GROUP BY terUnit.terUnitName, terUnit.terUnitId
```

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

<i>SifDioStudij</i>	<i>nazivDioStudij</i>	<i>sifNadDioStudij</i>	<i>trajeSem</i>	<i>ECTSBod</i>
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:

<i>nazivDioStudij</i>	<i>ukupnoTrajanje</i>	<i>potrebitnoECTS</i>
Automatika	6	180
Elektroenergetika	6	180
Programsko inženjerstvo i informacijski sustavi	6	180
...

RJEŠENJE

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.sifDioStudij)
/*
 ili
 WHERE sifDioStudij NOT IN (SELECT DISTINCT sifNadDioStudij
                             FROM strukturaStudij
                             WHERE sifNadDioStudij IS NOT NULL)
*/
GROUP BY sifDioStudij, SSRecursive.nazivDioStudij
```

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 objavljaju prema izbornim jedinicama i kandidacijskim listama (**electList**). Kandidacijske liste u nekim slučajevima objedinjavaju više političkih stranaka.

electUnit			city		electPlace				
unitId	unitName	UnitShortName	cityId	cityName	population	placeId	placeName	unitId	cityId
1	I. izborna jedinica	I	1	Zadar	71471	1001	OŠ Tina Ujevića	7	10
2	II. izborna jedinica	II	2	Split	167121	1002	Mjesna zajednica Kruge	2	10
...

electList		party		
listId	listName	partyId	partyName	shortName
1	SDP + HNS + HSU + SR+ HSS + ZS	88	Socijaldemokratska partija Hrvatske	SDP
2	HDZ + HSS + HSP AS + BUZ + ...	245	Most nezavisnih lista	MOST
...

person				
personId	fName	lName	partyId	...
100	Zoran	Milanović	88	...
104	Drago	Prgomet	245	...
...

vote			
votId	placeId	personId	electDate
100001	1001	104	7.11.2015
100002	1001	101	7.11.2015
...

slika 1

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. Prepostavite da dva kandidata na istoj kandidacijskoj listi u istoj izbornoj jedinici neće imati jednak broj glasova.

RJEŠENJE

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

```

Baza podataka je namijenjena pohrani podataka o restoranima na području RH. Za restoran se evidentira naziv, lokacija i prosječna ocjena restorana. Osim restorana, baza sadrži i sva mjesta i sve ceste, odnosno dionice cesta u RH. Npr. A1 autocesta se vodi kao niz dionica (Zagreb-Donja Zdenčina, Donja Zdenčina – Jastrebarsko, Jastrebarsko-Karlovac, Karlovac-Bosiljevo, ...). Restaurant, road i place imaju geoprostorne atributе čiji tipovi se vide iz primjera.

restaurant			
restId	restName	avgGrade	geom
1	Cassandra	3.3	<point>
2	Batelina	4.8	<point>
...

road					
roadId	roadName	placeFrom	placeTo	idRType	Geom
1	A1 Zg-Ka	1	2	1	<polyline>
2	A1 Ka-Bo	2	3	1	<polyline>
...	...				

place			
placeId	placeName	population	geom
1	Zagreb	790017	<point>
2	Karlovac	59395	<point>
3	Bosiljevo	1486	<point>
...

roadType	
rTypeId	rtypeName
1	Highway
2	Motorway
...	...

slika 1

- (5 bodova)** Korištenjem funkcionalnosti PostgreSQL SUBP napisati SQL naredbu kojom će se ispisati nazivi svih mesta s više od 50 000 stanovnika u koja se može stići autoputom iz Zagreba u sljedećem obliku:

PlaceName	distance
Karlovac	43
Rijeka	148
Zadar	258
...	...

distance uzima u obzir samo duljinu autoputa u kilometrima od Zagreba do konkretnog mesta.

RJEŠENJE

```

1.
WITH RECURSIVE roadRoutes (placeFrom, placeTo, totalM)
AS
( (SELECT placeFrom,
           placeTo,
           st_length(road.geom)
      FROM road
     JOIN place ON road.placeFrom = place.placeId
    WHERE place.placeName = 'Zagreb'
      AND road.rdTId = 1
   )
UNION ALL
(SELECT roadChild.placeFrom,
           roadChild.placeTo,
           roadParent.totalM + ST_length (roadChild.geom)
      FROM roadRoutes roadParent
     JOIN road roadChild ON roadParent.placeTo = roadChild.placeFrom
    WHERE roadChild.rdTId = 1
   )
SELECT place.placeName,
           roadRoutes.totalM/1000
      FROM roadRoutes
     JOIN place ON place.placeId = roadRoutes.placeTo
    WHERE place.population >50000

```

Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o studijskim boravcima studenata na inozemnim visokim učilištima u okviru Erasmus+ programa. Relacija **visokoUciliste** sadrži sva sveučilišta i njima podređene visokoobrazovne institucije (fakultete, visoke škole,...) koje sudjeluju u programu.

Sveučilišta nemaju nadređeno visoko učilište. Za studijske boravke se bilježi broj mjeseci koje je student boravio na inozemnoj instituciji (**erasmusBoravak.brMjes**).

U relaciji **ugovorAkGodina** su pohranjeni podaci o stranim visokim učilištima s kojima visoka učilišta imaju sklopljene ugovore o razmjeni studenata u određenoj akademskoj godini. Iznos mjesечne potpore/stipendije (relacija **potpora**) koja se studentu dodjeljuje, ovisi o državi u kojoj student boravi za vrijeme razmjene i vremenom se mijenja. Ključevi relacija su podrtani.

Slika 1.

drzava			visokoUciliste		
Ozn Drzava	nazivDrzava	geom	sifVU	nazivVU	Ozn Drzava
HR	Republika Hrvatska	<polygon>	10	Sveučilište u Zagrebu	HR
SE	Kraljevina Švedska	<polygon>	15	Sveučilište u Splitu	HR
PL	Poljska	<polygon>	53	Fakultet elektrotehnike i računarstva	HR
...	...		54	Ekonomski fakultet	HR
			112	Kraljevsko tehničko sveučilište	SE
		

erasmusBoravak					potpora		ugovorAkGodina				
JMBAG	AK Godina	sifVU Odl	sifVU Dol	brMjes	ak Godina	oznDrz Odl	oznDrz Dol	mjIzn Eur	ak Godina	sifVU Odl	sifVUDol
...	2015	53	112	5	2015	HR	SE	400	2015	53	[112, 345, ...]
...	2015	54	117	10	2016	HR	SE	460	2015	54	[113, 114, ...]
...	2015	SI	SE	500

1. **(4 boda)** Za visoka učilišta čiji studenti su sudjelovali u Erasmus+ programu, po akademskim godinama u kojima su boravci realizirani, ispisati ukupan i kumulativan broj studenata te ukupan i kumulativan iznos dodijeljenih potpora/stipendija.

Npr. kumulativni broj studenata za FER u 2015./2016. akademskoj godini pored broja studenata u 2015./2016. uključuje i broj studenata za godinu 2014./2015. te brojceve studenata u svim godinama prije 2015./2016. Slično vrijedi za kumulativan iznos.

akGodina	nazivVU	broStud	kumulBroStud	uklznosEur	kumulUklznosEur
...
2015	Fakultet elektrotehnike i računarstva	50	200	10000	50000
2016	Fakultet elektrotehnike i računarstva	60	260	12000	62000
...

RJEŠENJE

1. (4 boda)

```

SELECT erasmusBoravak.akGodina, VUOdlazno.nazivVU
, COUNT(erasmusBoravak.JMBAG)
, SUM(COUNT(erasmusBoravak.JMBAG)) OVER (PARTITION BY VUOdlazno.nazivVU
ORDER BY erasmusBoravak.akGodina
RANGE BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW)

//prethodna 2 retka se mogu izostaviti jer je to i defaultna definicija granica okvira
, SUM(brMjes*mjIznosEur)
, SUM(SUM(brMjes*mjIznosEur)) OVER (PARTITION BY VUOdlazno.nazivVU
ORDER BY erasmusBoravak.akGodina
RANGE BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW)

//prethodna 2 retka se mogu izostaviti
FROM erasmusBoravak
, visokoUciliste VUOdlazno
, visokoUciliste VUDolazno
, potpora
WHERE erasmusBoravak.sifVUOdl = VUOdlazno.sifVU
AND erasmusBoravak.sifVUDol = VUDolazno.sifVU
AND VUOdlazno.oznDrzava = potpora.oznDrzavaOdl
AND VUDolazno.oznDrzava = potpora.oznDrzavaDol
AND erasmusBoravak.akGodina = potpora.akGodina
GROUP BY erasmusBoravak.akGodina, VUOdlazno.nazivVU

```

- 2. (3 boda)** Ispisati nazive svih stranih visokih učilišta (bez ponavljanja) s kojima, u akademskoj godini 2015./2016, bilo koje hrvatsko visoko učilište ima sklopljen ugovor o razmjeni studenata.

RJEŠENJE

2. (3 boda)

```
SELECT DISTINCT stranoVU.nazivVU
FROM ugovorAkGodina,
     UNNEST(sifVuDol) AS sifStranoVU,
     visokoUciliste hrVU,
     visokoUciliste stranoVU
WHERE sifStranoVU = stranoVU.sifVU
AND ugovorAkGodina.akGodina = 2015
AND ugovorAkGodina.sifVuOdl = hrVU.sifVU
AND hrVU.oznDrzava = 'HR' -- ili spojiti još s državom, ali to nije važno
ili

SELECT DISTINCT stranoVU.nazivVU
FROM ugovorAkGodina,
     visokoUciliste hrVU,
     visokoUciliste stranoVU
WHERE ugovorAkGodina.akGodina = 2015
AND ugovorAkGodina.sifVuOdl = hrVU.sifVU
AND hrVU.oznDrzava = 'HR'
AND sifVUDol @> ARRAY[stranoVU.sifvu]
```

Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o putovanjima autobusom u ponudi turističke agencije. U relaciji **putLok** su opisane dionice konkretnog putovanja. Atribut **putLok.geom** predstavlja dionicu ceste između početne i završne lokacije te dionice. Atribut **putTermin.cijenaHRK** predstavlja cijenu putovanja za jednu osobu. Ključevi relacija su podrtani.

lokacija				putovanje			drzava		
sifLok	nazivLok	oznDrz	geom	sifPut	nazivPut	...	oznDrz	nazivDrz	geom
11	Split	HR	<point>	20	Ljepote Jadrana		HR	Hrvatska	<polygon>
12	Plitvice	HR	<point>	21	Vikend u Dubrovniku		AT	Austrija	<polygon>
13	Dubrovnik	HR	<point>	23	Austrijska avantura		PL	Poљska	<polygon>
...	24	Finske bijele noći

putLok					putTermin				
sifPut	rbrDio	sifLokPoc	sifLokZav	geom	sifPut	datPolazak	cijenaHRK	ukMjesta	prodano Mjesta
20	2	12	11	<polyline>	20	01.05.2016	1200	50	50
20	3	11	13	<polyline>	20	01.07.2016	1400	80	72
21	1	8	13	<polyline>	21	15.06.2016	400	50	46
21	4	11	15	<polyline>	22	04.06.2016	3200	50	50
21	6	16	13	<polyline>
...

Slika 1.

1. **(4 boda)** Po godinama u kojima su putovanja realizirana, ispisati godinu i naziv putovanja te ukupan i kumulativni broj kilometara koje autobusi pređu prevozeći putnike na tom putovanju.
Npr. kumulativni broj kilometara za putovanje naziva 'Ljepote Jadrana' u 2016. godini pored broja kilometara pređenih 2016. uključuje i kilometre pređene (za to isto putovanje) u 2015. te kilometre u svim godinama prije 2015.

godina	nazivPut	kmGodina	kumulKm
...
2015	Ljepote Jadrana	5000	50000
2016	Ljepote Jadrana	6000	56000
...

Pomoć: godinu iz datuma možete izdvojiti pomoću funkcije EXTRACT
(npr. EXTRACT (YEAR FROM imeAtributa)).

RJEŠENJE

1. **(4 boda)**

```

SELECT EXTRACT (YEAR FROM datPolazak), putovanje.nazivPut,
       SUM(ST_length(putLok.geom)),
       SUM(SUM(ST_length(putLok.geom))) OVER (PARTITION BY putovanje.sifPut
                                                 ORDER BY EXTRACT (YEAR from datPolazak)
                                                 ROWS BETWEEN UNBOUNDED PRECEDING
                                                 AND CURRENT ROW)
--prethodna 2 retka se mogu izostaviti jer je to i defaultna definicija granica okvira
FROM putovanje, putLok, putTermin
WHERE putovanje.sifPut = putLok.sifPut
      AND putovanje.sifPut = putTermin.sifPut
GROUP BY EXTRACT (YEAR from datPolazak), putovanje.nazivPut, putovanje.sifPut
    
```

Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o putovanjima autobusom u ponudi turističke agencije. U relaciji ***putLok*** su opisane dionice konkretnog putovanja. Atribut ***putLok.geom*** predstavlja dionicu ceste između početne i završne lokacije te dionice. Atribut ***putTermin.cijenaHRK*** predstavlja cijenu putovanja za jednu osobu. Ključevi relacija su podrtani.

lokacija				putovanje			drzava		
<i>sifLok</i>	<i>nazivLok</i>	<i>oznDrz</i>	<i>geom</i>	<i>sifPut</i>	<i>nazivPut</i>	...	<i>oznDrz</i>	<i>nazivDrz</i>	<i>geom</i>
11	Split	HR	<point>	20	Ljepote Jadrana		HR	Hrvatska	<polygon>
12	Plitvice	HR	<point>	21	Vikend u Dubrovniku		AT	Austrija	<polygon>
13	Dubrovnik	HR	<point>	23	Austrijska avantura		PL	Poљska	<polygon>
...	24	Finske bijele noći

putLok					putTermin				
<i>sifPut</i>	<i>rbrDio</i>	<i>sifLokPoc</i>	<i>sifLokZav</i>	<i>geom</i>	<i>sifPut</i>	<i>datPolazak</i>	<i>cijenaHRK</i>	<i>ukMjesta</i>	<i>prodanoMjesta</i>
20	2	12	11	<polyline>	20	01.05.2016	1200	50	50
20	3	11	13	<polyline>	20	01.07.2016	1400	80	72
21	1	8	13	<polyline>	21	15.06.2016	400	50	46
21	4	11	15	<polyline>	22	04.06.2016	3200	50	50
21	6	16	13	<polyline>
...

Slika 1.

2. (4 boda) Za sva putovanja ispisati podatke o ukupnoj zaradi po mjesecima u sljedećem obliku:

<i>putovanje</i>	<i>siječanj</i>	<i>veljaca</i>	<i>ozujak</i>	<i>aprīl</i>					<i>prosinac</i>
Jug Dalmacije	16864.75		25634.00	123265.50
Ljepote Jadrana

Pomoć: Upotrijebiti funkciju ***crosstab***. Voditi računa da se podaci moraju ispravno ispisati i u slučaju da postoje mjeseci u kojima nije bilo polazaka za određeno putovanje.

RJEŠENJE

2. (4 boda)

```

CREATE EXTENSION tablefunc;
CREATE TEMP TABLE mjesec
(rbrMjesec int);
INSERT INTO mjesec VALUES ( 1 );
INSERT INTO mjesec VALUES ( 2 );
...
INSERT INTO mjesec VALUES ( 12 );

SELECT *
  FROM crosstab ('SELECT putovanje.nazivPut AS naziv
                  , CAST(EXTRACT(MONTH FROM datPolazak) AS int) AS mjesec
                  , CAST(SUM(putTermin.cijenaHrk*prodanoMjesta)
                        AS decimal (10,2)) AS zarada
                FROM putovanje, putTermin
               WHERE putovanje.sifPut = putTermin.sifPut
             GROUP BY putovanje.nazivPut, EXTRACT(MONTH FROM datPolazak)
            ORDER BY putovanje.nazivPut, EXTRACT(MONTH FROM datPolazak)'
                  , 'SELECT rbrMjesec FROM mjesec ORDER BY rbrMjesec')
 AS pivotTable (nazivPutovanje CHAR(50)
              , siječanj decimal (10,2), veljača decimal (10,2)
              , ožujak decimal (10,2), travanj DECIMAL(10,2)
              , svibanj DECIMAL(10,2), lipanj DECIMAL(10,2)
              , srpanj DECIMAL(10,2), kolovoz DECIMAL(10,2)
              , rujan DECIMAL(10,2), listopad DECIMAL(10,2)
              , studeni DECIMAL(10,2), prosinac DECIMAL(10,2))

```

ORDBMS

proizvod		prodaja						
sifProizv	nazivProizv	...	datum	sifProd	sifProizv	kolicina	iznos	...
15	Clarins maskara		15/01/2015	1	15	1	200	
25	Chanel smokey eyes sjenilo		15/01/2015	1	15	2	400	
35	Shiseido BB krema		16/01/2015	1	35	2	300	
...	...		16/01/2015	1	35	2	300	
			15/01/2015	2	35	2	300	
			15/01/2015	2	15	1	200	
			16/01/2015	2	25	2	200	
			16/01/2015	2	25	3	300	
		

2. U zadatku se pretpostavlja korištenje PostgreSQL SUBP-a u kojem postoji intarray proširenje.

- (2 boda) Napišite SQL naredbu za kreiranje tablice **prodavaonicaOR** i svih ostalih objekata korištenih u definiciji te tablice. Tablica, osim šifre (primarni ključ) i naziva prodavaonice, sadrži i atributе:
 - *poslovodja* - atribut tipa *poslovodjaT* čiji su atributi (elementi): *sifOsoba*, *datumOd*
 - *zaposlenici* – šifre zaposlenika u prodavaonici. Zaposlenici su poredani prema trenutku zapošljavanja u prodavaonici (kao prvi je naveden zaposlenik koji je prvi zaposlen).
- (1 bod) Napišite upit kojim će se ispisati šifra i naziv prodavaonica s više od jednog zaposlenika, u kojima je poslovođa zadnja zaposlena osoba u toj prodavaonici.
- (2 boda) Napišite upit kojim će se provjeriti postoje li osobe koje su zaposlene u više prodavaonica. Za takve osobe potrebno je ispisati šifru osobe i polje koje sadrži šifre prodavaonica u kojima je osoba zaposlena. Zadatak riješiti bez korištenja podupita.

RJEŠENJE

2.a) CREATE TYPE poslovodjaT AS (sifOsoba INT, datumOd DATE);
CREATE TABLE prodavaonicaOR (sifProd INTEGER PRIMARY KEY,
 nazProd VARCHAR(60),
 poslovodja poslovodjaT,
 zaposlenici INTEGER[]);

b) SELECT sifProd, nazProd FROM prodavaonica
WHERE zaposlenici[icount(zaposlenici)] = (poslovodja).sifOsoba
AND icount(zaposlenici) > 1;

c) SELECT UNNEST(zaposlenici) AS sifOsoba, array_agg(sifProd)
 FROM prodavaonicaOR
 GROUP BY sifOsoba
 HAVING COUNT(*) > 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 objavljaju prema izbornim jedinicama i kandidacijskim listama (**electList**). Kandidacijske liste u nekim slučajevima objedinjavaju više političkih stranaka.

electUnit			city		electPlace				
unitId	unitName	UnitShort Name	cityId	cityName	population	placeId	placeName	unitId	cityId
1	I. izborna jedinica	I	1	Zadar	71471	1001	OŠ Tina Ujevića	7	10
2	II. izborna jedinica	II	2	Split	167121	1002	Mjesna zajednica Kruse	2	10
...

electList		party		
listId	listName	partyId	partyName	shortName
1	SDP + HNS + HSU + SR+ HSS + ZS	88	Socijaldemokratska partija Hrvatske	SDP
2	HDZ + HSS + HSP AS + BUZ +...	245	Most nezavisnih lista	MOST
...

person					vote			
personId	fName	lName	partyId	...	votId	placeId	personId	electDate
100	Zoran	Milanović	88	...	100001	1001	104	7.11.2015
104	Drago	Prgomet	245	...	100002	1001	101	7.11.2015
...

slika 1

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

electResultOR				listMembers
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.

RJEŠENJE

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)
)

```

3. (4 boda) Prepostavite 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*.

RJEŠENJE

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
```

5. (6 bodova) U zadatku se pretpostavlja korištenje SQL standarda.

U bazi podataka pohranjuju se podaci o projektima u nekom poduzeću i osobama koje rade na projektima. Prikazanim naredbama kreirani su tipovi *projektT* i *osobaT*. Kreirane su i tipizirane tablice *projektOR* i *osobaOR* temeljene na tim tipovima. Imena stupaca koji sadrže jedinstvene identifikatore objekata u tipiziranim tablicama su *OIDprojekt* odnosno *OIDosoba*. U atributu *osobaT.projekti* evidentirani su projekti na kojima osoba trenutno radi.

```
CREATE TYPE projektT AS (
    sifProjekt INTEGER,
    nazProjekt CHAR(50))
INSTANTIABLE NOT FINAL
REF IS SYSTEM GENERATED;
```

```
CREATE TYPE osobaT AS (
    sifOsoba    INTEGER,
    ime         CHAR(50),
    prezime     CHAR(75),
    projekti   REF(projektT) MULTISET)
INSTANTIABLE NOT FINAL
REF IS SYSTEM GENERATED;
```

U tablici *osobaOR* evidentiraju se **sve** osobe koje rade na projektima. Međutim, osobe mogu, ali i ne moraju biti zaposlenici poduzeća. Za osobe koje nisu zaposlenici poduzeća potrebno je dodatno, u objektno relacijskoj tablici *honoraracOR*, evidentirati poduzeće u kojem je ta osoba trenutno zaposlena. Informacija o poduzeću u kojоj je osoba zaposlena pohranjena je u atributu *poduzece* koji je tipa ROW, a sastoji se od dva elementa: *nazPoduzece* (znakovni niz maksimalne duljine 50 znakova) i *datZaposlenja* (datum).

- Napisati niz SQL naredbi kojima će se kreirati tablica *honoraracOR* te svi za to potrebni objekti objektno-relacijske baze podataka, pri čemu je potrebno primijeniti nasljeđivanje.
- Napisati SQL naredbu/naredbe kojom će se upisati podaci o osobi koja radi na projektu sa šifrom 1, nije zaposlenik poduzeća, a od 1.3.2014. je zaposlena u poduzeću naziva "INA". Projekt sa šifrom 1 je već evidentiran u tablici *projektOR*. Ostali podaci o osobi (šifra, ime i prezime) su proizvoljni.
- Napisati SQL naredbu kojom će se dohvatiti ime i prezime osoba koje trenutno rade na barem 3 projekta, a jedan od tih projekata je projekt sa šifrom 1. Zadatak riješiti bez korištenja podupita.

RJEŠENJE

a)

```
CREATE TYPE honoraracT UNDER osobaT AS (
    poduzece ROW (nazivPoduzece CHAR(100), datumZaposlenje DATE)
    INSTANTIABLE NOT FINAL;
CREATE TABLE honoraracOR OF honoraracT UNDER osobaOR;
```

b)

```
INSERT INTO honoraracOR VALUES (101, 'Perica', 'Perić',
                                MULTISET (SELECT p.OIDprojekt FROM p.projektOR
                                           WHERE p.sifProjekt = 1)
                                ROW ('INA', '1.3.2014');
```

c)

```
SELECT o.ime, o.prezime
  FROM osobaOR o, UNNEST (o.projekti) AS p(OIDprojekt)
 WHERE CARDINALITY (o.projekti) >= 3
   AND p.OIDprojekt -> sifProjekt = 1;
```

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 podrtani.

osoba			nekretnina				
sifOsoba	imeOsoba	prezOsoba	sifNekretnina	povrsina	vlasnici	adresa (pbr, ulica, broj)	geom
1	Slack	David	1	45.6	(13, 22, 5)	(10000, "Ilica", 12)	Polygon (...)
2	Segel	Amanda	2	380.4	{555}	(10000, "Vinogradска", 14)	Polygon (...)
3	Smith	Bob	3	33.5	{22}	(10000, "Ilica", 12)	Polygon (...)
4	Cardell	Ema	4	1087.3	{22, 13}	(10000, "Grada Vukovara", 12)	Polygon (...)
5	Tillman	Joan	5	234.5	{1}	(10000, "Grada Vukovara", 345)	Polygon (...)
			6	22.1	{5}	(44000, "Vinogradска", 12)	Polygon (...)
				

parcela		
brParcela	vlasnici	geom
101	{1}	Polygon (...)
102/1	{5}	Polygon (...)
102/2	{22, 13}	Polygon (...)

1. **(4 bodova)** 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. Prepostaviti 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.

RJEŠENJE

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)** 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 rješiti bez korištenja podupita.

RJEŠENJE

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
```



Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o studijskim boravcima studenata na inozemnim visokim učilištima u okviru Erasmus+ programa. Relacija **visokoUciliste** sadrži sva sveučilišta i njima podređene visokoobrazovne institucije (fakultete, visoke škole,...) koje sudjeluju u programu. Sveučilišta nemaju nadređeno visoko učilište. Za studijske boravke se bilježi broj mjeseci koje je student boravio na inozemnoj instituciji (**erasmusBoravak.brMjes**).

U relaciji **ugovorAkGodina** su pohranjeni podaci o stranim visokim učilištima s kojima visoka učilišta imaju sklopljene ugovore o razmjeni studenata u određenoj akademskoj godini. Iznos mjesечne potpore/stipendije (relacija **potpora**) koja se studentu dodjeljuje, ovisi o državi u kojoj student boravi za vrijeme razmjene i vremenom se mijenja. Ključevi relacija su podrtani.

Slika 1.

drzava			visokoUciliste		
Ozn Drzava	nazivDrzava	geom	sifVU	nazivVU	Ozn Drzava
HR	Republika Hrvatska	<polygon>	10	Sveučilište u Zagrebu	HR
SE	Kraljevina Švedska	<polygon>	15	Sveučilište u Splitu	HR
PL	Poljska	<polygon>	53	Fakultet elektrotehnike i računarstva	HR
...	...		54	Ekonomski fakultet	HR
			112	Kraljevsko tehničko sveučilište	SE
		

erasmusBoravak					potpora		ugovorAkGodina				
JMBAG	AK Godina	sifVU Odl	sifVU Dol	brMjes	ak Godina	oznDrz Odl	oznDrz Dol	mjIzn Eur	ak Godina	sifVU Odl	sifVUDol
...	2015	53	112	5	2015	HR	SE	400	2015	53	[112, 345, ...]
...	2015	54	117	10	2016	HR	SE	460	2015	54	[113, 114, ...]
...	2015	SI	SE	500
							

1. **(4 boda)** Za visoka učilišta čiji studenti su sudjelovali u Erasmus+ programu, po akademskim godinama u kojima su boravci realizirani, ispisati ukupan i kumulativan broj studenata te ukupan i kumulativan iznos dodijeljenih potpora/stipendija.

Npr. kumulativni broj studenata za FER u 2015./2016. akademskoj godini pored broja studenata u 2015./2016. uključuje i broj studenata za godinu 2014./2015. te brojceve studenata u svim godinama prije 2015./2016. Slično vrijedi za kumulativan iznos.

akGodina	nazivVU	brojStud	kumulBrojStud	uklznosEur	kumulUklznosEur
...
2015	Fakultet elektrotehnike i računarstva	50	200	10000	50000
2016	Fakultet elektrotehnike i računarstva	60	260	12000	62000
...

RJEŠENJE

1. **(4 boda)**

```

SELECT erasmusBoravak.akGodina, VUOdlazno.nazivVU
    , COUNT(erasmusBoravak.JMBAG)
    , SUM(COUNT(erasmusBoravak.JMBAG)) OVER (PARTITION BY VUOdlazno.nazivVU
                                                ORDER BY erasmusBoravak.akGodina
                                                RANGE BETWEEN UNBOUNDED PRECEDING
                                                AND CURRENT ROW)

//prethodna 2 retka se mogu izostaviti jer je to i defaultna definicija granica okvira
    , SUM(brMjes*mjIznosEur)
    , SUM(SUM(brMjes*mjIznosEur)) OVER (PARTITION BY VUOdlazno.nazivVU
                                                ORDER BY erasmusBoravak.akGodina
                                                RANGE BETWEEN UNBOUNDED PRECEDING
                                                AND CURRENT ROW)

//prethodna 2 retka se mogu izostaviti
    FROM erasmusBoravak
        , visokoUciliiste VUOdlazno
        , visokoUciliiste VUDolazno
        , potpora
WHERE erasmusBoravak.sifVUOdl = VUOdlazno.sifVU
    AND erasmusBoravak.sifVUDol = VUDolazno.sifVU
    AND VUOdlazno.oznDrzava = potpora.oznDrzavaOdl
    AND VUDolazno.oznDrzava = potpora.oznDrzavaDol
    AND erasmusBoravak.akGodina = potpora.akGodina
GROUP BY erasmusBoravak.akGodina, VUOdlazno.nazivVU

```

2. **(3 boda)** Ispisati nazive svih stranih visokih učilišta (bez ponavljanja) s kojima, u akademskoj godini 2015./2016., bilo koje hrvatsko visoko učilište ima sklopljen ugovor o razmjeni studenata.

RJEŠENJE

2. **(3 boda)**

```
SELECT DISTINCT stranoVU.nazivVU
FROM ugovorAkGodina,
     UNNEST(sifVuDol) AS sifStranoVU,
     visokoUciline hrVU,
     visokoUciline stranoVU
WHERE sifStranoVU = stranoVU.sifVU
AND ugovorAkGodina.akGodina = 2015
AND ugovorAkGodina.sifVuOdl = hrVU.sifVU
AND hrVU.oznDrzava = 'HR' -- ili spojiti još s državom, ali to nije važno
```

ili

```
SELECT DISTINCT stranoVU.nazivVU
FROM ugovorAkGodina,
     visokoUciline hrVU,
     visokoUciline stranoVU
WHERE ugovorAkGodina.akGodina = 2015
AND ugovorAkGodina.sifVuOdl = hrVU.sifVU
AND hrVU.oznDrzava = 'HR'
AND sifVUDol @> ARRAY[stranoVU.sifvu]
```

Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o putovanjima autobusom u ponudi turističke agencije. U relaciji **putLok** su opisane dionice konkretnog putovanja. Atribut **putLok.geom** predstavlja dionicu ceste između početne i završne lokacije te dionice. Atribut **putTermin.cijenaHRK** predstavlja cijenu putovanja za jednu osobu. Ključevi relacija su podcrteni.

lokacija				putovanje			drzava		
<u>sifLok</u>	nazivLok	oznDrz	geom	<u>sifPut</u>	<u>nazivPut</u>	...	<u>oznDrz</u>	nazivDrz	geom
11	Split	HR	<point>	20	Ljepote Jadrana		HR	Hrvatska	<polygon>
12	Plitvice	HR	<point>	21	Vikend u Dubrovniku		AT	Austrija	<polygon>
13	Dubrovnik	HR	<point>	23	Austrijska avantura		PL	Poљska	<polygon>
...	24	Finske bijele noći

putLok					putTermin				
<u>sifPut</u>	rbrDio	<u>sifLokPoc</u>	<u>sifLokZav</u>	geom	<u>sifPut</u>	<u>datPolazak</u>	cijenaHRK	ukMjesta	prodano Mjesta
20	2	12	11	<polyline>	20	01.05.2016	1200	50	50
20	3	11	13	<polyline>	20	01.07.2016	1400	80	72
21	1	8	13	<polyline>	21	15.06.2016	400	50	46
21	4	11	15	<polyline>	22	04.06.2016	3200	50	50
21	6	16	13	<polyline>
...

Slika 1

3. (4 boda) Korištenjem PostgreSQL sintakse potrebno je napisati SQL naredbe za kreiranje relacije putovanjeOR čija je **shema i sadržaj** skiciran donjom tablicom a definiran shemom i sadržajem tablica sa slike 1.

<u>sifPut</u> smallint	<u>nazivPut</u> ncharacter(250)	<i>lokacije</i> <i>lokacija[]</i>
20	Ljepote Jadrana	"(" (8, \"Zagreb\", \"HR\",), (9, \"Plitvice\", \"HR\",), (10, \"Zadar\", \"HR\",), ...)"
21	Vikend u Dubrovniku	...

U polju lokacije se pojavljuju sve lokacije koje putnici obilaze na određenom putovanju. Voditi računa da se lokacije ne ponavljaju.

RJEŠENJE

3. (4 boda)

Budući da kreiranjem tablice u bazi podataka nastaje i istoimeni tip, tip **lokacija** možemo iskoristiti kao tip kojeg će biti elementi polja **putovanjeOR.lokacije**. Nije potrebno definirati ekstra tip.

```

CREATE TABLE putovanjeOR
(
    sifPut      smallint CONSTRAINT pkPutovanjeOR PRIMARY KEY,
    nazivPut    NCHAR(250),
    lokacije    lokacija[]
);

INSERT INTO putovanjeOR
SELECT putovanje.*,
       (SELECT array_agg(DISTINCT lokacija.*)
        FROM putLok, lokacija
        WHERE putovanje.sifPut = putLok.sifPut
          AND (lokacija.sifLok = putLok.sifLokPoc OR
               lokacija.sifLok = putLok.sifLokZav))
FROM putovanje
DISTINCT treba da se izbjegne ponavljanje lokacija.

CAST nije potreban u array_agg(CAST( ROW(DISTINCT lokacija.*) AS lokacija))
jer je lokacija.* istog tipa kao i element polja putovanjeOR.lokacije

```

Vremenske baze

Baza podataka prikazana na **slici 1** služi za evidenciju podataka o rezultatima referendumu koji se održavaju u Ujedinjenom Kraljevstvu Velike Britanije i Sjeverne Irske. Rezultati referendumu se bilježe u relaciji **result**. Osobe koje mogu pristupiti određenom referendumu (zajedno s detaljima kada što su predviđeno glasačko mjesto) evidentirane su u relaciji **personRef**. Teritorijalna podjela države, te glasačka mjesta u teritorijalnim jedinicama evidentirani su u relacijama **terUnit** i **poolPlace**. Ključevi relacija su podcrtni.

terUnit				poolPlace			
terUnitId	terUnitName	geom	supTerUnitId	poolPlaceId	poolPlaceName	terUnitId	terUnitName
53	England	<polygon>	NULL	184	Islington Town Hall	856	London
32	Scotland	<polygon>	NULL	132	Olive Morris House	437	Edinburgh
5478	East England	<polygon>	53	11	York House	976	West London
9056	London	<polygon>	53	East London
437	City of Edinburg	<polygon>	32
856	West London	<polygon>	9056
876	East London	<polygon>	9056
...

result				
refDate	poolPlaceId	ordinal	voteFor	valid
18.09.2014	132	1	1	1
23.06.2016	11	2	0	1
23.06.2016	132	105	1	1
...

person			
personId	FName	LName	...
...

Slika 1.

2. (4 boda) Zbog promjene teritorijalnog ustroja, moguća je promjena pripadnosti glasačkog mesta teritorijalnoj jedinici (**poolPlace.terUnitId**). Postojeći segment sa slike 1 potrebno je izmijeniti (proširiti) tako da omogući praćenje promjene atributa **poolPlace.terUnitId** u kontekstu vremena valjanosti. Modelom osigurati pripadnost glasačkog mesta samo jednoj teritorijalnoj jedinici u konkretnom periodu. Također, modelom osigurati da se zapisi u relaciji **personRef** referenciraju na ispravnu n-torku iz **poolPlace** (aktualnu u trenutku odvijanja referendumu).

Korištenjem funkcionalnosti PostgreSQL SUBP napisati SQL naredbe za provođenje predloženih izmjena. Definirati temporalna integritetska ograničenja (primarne i strane ključeve) koja je moguće definirati u okviru PostgreSQL SUBP.

Za integritetska ograničenja koja nije moguće implementirati u PostgreSQL-u objasnite kakvu implementaciju predviđa SQL standard.

RJEŠENJE

2. (4 boda)

Shemu **poolPlace** treba proširiti atributom **poolPlacePeriod** tipa DATERANGE, ažurirati vrijednost tog atributa, uništiti postojeći PRIMARY KEY i kreirati temporalni primarni ključ.

Shemu **personRef** treba proširiti atributom **polPlacePeriod** tipa DATERANGE, ažurirati mu vrijednost i kreirati temporalni strani ključ. PostgreSQL ga ne podržava.

```
CREATE EXTENSION bTree_gist; //studenti ne moraju napisati, ali inače treba
ALTER TABLE poolPlace ADD poolPlacePeriod DATERANGE;
ALTER TABLE poolPlace ADD PRIMARY KEY (poolPlaceId, poolPlacePeriod);
ALTER TABLE poolPlace ADD CONSTRAINT temporalPKPoolPlace EXCLUDE USING gist
(poolPlaceId WITH =, poolPlacePeriod WITH &&)
```

```
ALTER TABLE personRef ADD polPlacePeriod DATERANGE;
PostgreSQL ne podržava temporalni referencijski integritet.
```

SQL standard za temporalni referencijski integritet predviđa da je period referencirajuće n-torce u potpunosti sadržan u periodu jedne referencirane n-torce ili u kombiniranom periodu dvije ili više uzastopnih referenciranih n-torki.

Primijenjeno na gornje dvije relacije to bi značilo sljedeće:

Za isti **poolPlaceId** (**poolPlace.poolPlaceId = personRef.poolPlaceId**)
personRef.polPlacePeriod treba biti u potpunosti sadržan u **poolPlace.poolPlacePeriod**
-u jedne n-torce ili u kombiniranom periodu dvije ili više uzastopnih referenciranih n-torki

2. **(5 bodova)** Objasnite razliku između stanja i događaja u kontekstu vremenskih baza podataka. Koristeći samo SQL (bez dodatne podrške za upravljanje vremenom), napišite primjer naredbe za stvaranje relacije stanja i relacije događaja.

RJEŠENJE

2

- **Stanja** opisuju činjenice vezane uz neki objekt u bazi podataka koje su istinite u nekom vremenskom intervalu ili periodu. Te se činjenice ne smatraju točnima izvan pridruženog perioda.
- **Događaji** opisuju činjenice vezane uz neki objekt u bazi podataka koje su se dogodile u određenom trenutku (*chrononu*) i nemaju trajanje.

Relacija stanja:

```
CREATE TABLE zaposlenik (
    id      INTEGER,
    ime     CHAR(20),
    prezime CHAR(20),
    placa   NUMBER,
    vrijediOd TIMESTAMP,
    vrijediDo TIMESTAMP
);
```

Relacija događaja

```
CREATE TABLE ispit (
    sifStudent    INTEGER,
    sifNastavnik  INTEGER,
    sifPredmet    INTEGER,
    ocjena,
    datum        DATE
);
```

stanje			
placa			
idZaposlenika	iznos	vrijediOd	vrijediDo
100	5000	1.1.2010.	NULL
101	8000	6.12.2001.	1.1.2005
101	10000	2.1.2005.	NULL
102	6000	1.7.2009.	1.7.2011.
105	6000	20.5.1998.	19.5.2008.
105	7000	20.5.2008.	NULL

događaj		
nagrada		
idZaposlenika	iznos	datum
100	5000	10.6.2011.
105	5000	24.12.2001.
101	10000	1.7.2011.
105	5000	20.5.2008.

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

RJEŠENJE

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



electUnit			city		electPlace				
unitId	unitName	UnitShortName	cityId	cityName	population	placeId	placeName	unitId	cityId
1	I. izborna jedinica	I	1	Zadar	71471	1001	OŠ Tina Ujevića	7	10
2	II. izborna jedinica	II	2	Split	167121	1002	Mjesna zajednica Kruge	2	10
...

electList		party		
listId	listName	partyId	partyName	shortName
1	SDP + HNS + HSU + SR+ HSS + ZS	88	Socijaldemokratska partija Hrvatske	SDP
2	HDZ + HSS + HSP AS + BUZ +...	245	Most nezavisnih lista	MOST
...

person			
personId	fName	lName	partyId
100	Zoran	Milanović	88
104	Drago	Prgomet	245
...

vote			
votId	placeId	personId	electDate
100001	1001	104	7.11.2015
100002	1001	101	7.11.2015
...

slika 1

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 ujek na istim kandidacijskim listam odnosno, ona nije relacija vremena valjanosti. Opišite što biste promjenili 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-torce u relaciju.
- Napišite SQL naredbe za promjenu sheme relacije **party**. Trenutno ograničenje ključa je imenovano s **pkPartyMember**.

RJEŠENJE

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-torce 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-torce: 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-torce 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. (3 boda) Objasnite mehanizam **kvorum** kod ostvarivanja konzistencije. Navedite primjere i za čitanje i za pisanje gdje se objašnjava kako se (ne)postiže konzistencija.

RJEŠENJE

6. (3 boda)

Kvorum je mehanizam kojim je moguće ostvariti konzistenciju u distribuiranim sustavima. Neka imamo sustav s 1000 čvorova (nije bitno), a faktor replikacije $N = 3$, što znači da je isti podatak zapisan na 3 mesta od tih 1000 čvorova.

Ako želimo osigurati konzistenciju kod pisanja, nužno je da novi zapis upišemo na barem:

$$W > N/2$$

mesta (W je broj pisanja) zato jer onda kasnije, kod čitanja, možemo pročitati iste te, što čini većinu i pobijeđuje kod odlučivanja što je ispravna verzija. Npr. Neka je za studenta pisalo da ima 20 godina na 3 mesta – kada upišemo da sad ima 21 godinu na barem 2 mesta, sigurni smo da će naknadno kod glasanja biti barem dva zapisa s 21, što će odnijeti prevagu nad verzijom gdje se tvrdi da student ima 20 godina.

Konzistentno pisanje ne znači da kasnije ostvarujemo konzistentno čitanje!

Naime, moram pročitati ta dva zapisa da bismo bili sigurni da smo ostvarili konzistenciju. Ako pročitamo npr. samo jedan, možemo u našem primjeru krivo zaključiti da student ima 20 godina (ako pogodimo baš taj čvor u kojem piše 20).

Konzistencija kod čitanja je tako povezana s W , te je za ostvariti konzistenciju dovoljno:

$$R + W > N$$

U našem primjeru, ako je $W = 2$ i $N = 3$, slijedi da $R > 1$.

Sve te parametre (R, W) možemo definirati na razini sjednice, ali i na razini requesta!

(nije nužno reći na ispitu): valja primijetiti i što se događa s ovom formulom kada imamo nekonzistentno pisanje, npr. $W = 1$ u tom slučaju je $R > 2$, odnosno moramo pročitati sve čvorove da bi imali „konzistentno“ čitanje – ali sada konzistentno samo u smislu da detektiramo konflikt.

Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o studijskim boravcima studenata na inozemnim visokim učilištima u okviru Erasmus+ programa. Relacija **visokoUciliste** sadrži sva sveučilišta i njima podređene visokoobrazovne institucije (fakultete, visoke škole,...) koje sudjeluju u programu.

Sveučilišta nemaju nadređeno visoko učilište. Za studijske boravke se bilježi broj mjeseci koje je student boravio na inozemnoj instituciji (**erasmusBoravak.brMies**).

U relaciji **ugovorAkGodina** su pohranjeni podaci o stranim visokim učilištima s kojima visoka učilišta imaju sklopljene ugovore o razmjeni studenata u određenoj akademskoj godini. Iznos mjesечne potpore/stipendije (relacija **potpora**) koja se studentu dodjeljuje, ovisi o državi u kojoj student boravi za vrijeme razmjene i vremenom se mijenja. Ključevi relacija su podrtani.

Slika 1.

drzava			visokoUciliste		
Ozn Drzava	nazivDrzava	geom	sifVU	nazivVU	Ozn Drzava
HR	Republika Hrvatska	<polygon>	10	Sveučilište u Zagrebu	HR
SE	Kraljevina Švedska	<polygon>	15	Sveučilište u Splitu	HR
PL	Poljska	<polygon>	53	Fakultet elektrotehnike i računarstva	HR
...	...		54	Ekonomski fakultet	HR
			112	Kraljevsko tehničko sveučilište	SE
		

erasmusBoravak					potpora		ugovorAkGodina				
JMBAG	Ak Godina	sifVU Odl	sifVU Dol	brMjes	ak Godina	oznDrz Odl	oznDrz Dol	mjlzn Eur	ak Godina	sifVU Odl	sifVUDol
...	2015	53	112	5	2015	HR	SE	400	2015	53	[112, 345, ...]
...	2015	54	117	10	2016	HR	SE	460	2015	54	[113, 114, ...]
...	2015	SI	SE	500

3. **(4 boda)** Vlasnik poslovnog sustava koji koristi segment baze podataka sa slike 1 ne želi evidentirati iznose finansijskih potpora koje se isplaćuju studentima svake akademske godine nego samo kad se promijene. Smatrati da se iznosi potpora ne mijenjaju unutar akademske godine.

Potrebitno je predložiti izmjene modela sa slike 1 tako da omogući praćenje promjene atributa **potpora.mjlnosEur** u kontekstu vremena valjanosti. Modelom osigurati jedinstvenost iznosa potpore za određenu odlaznu i dolaznu državu u konkretnom periodu valjanosti.

Korištenjem funkcionalnosti PostgreSQL SUBP, napisati SQL naredbe za provođenje predloženih izmjena. Definirati temporalna integritetska ograničenja koja je moguće definirati u PostgreSQL SUBP. Skicirajte sadržaj relacije **potpora** nakon predloženih izmjena uz pretpostavku da u njoj postoe samo n-torce prikazane na slici 1. Pretpostavite da iznosi potpora važeći u 2016./2017. vrijede i u 2017./2018., ali ne i nakon. Smatrati da akademska godina počinje 1.10. a završava 30.9. (sljedeće kalendarske godine).

Napišite jednu SQL naredbu čije bi izvođenje završilo pogreškom zbog narušenja ograničenja temporalnog primarnog ključa. Na primjeru predložene SQL naredbe objasnite na koji način PostgreSQL SUBP (logički) provjerava integritet temporalnog primarnog ključa.

RJEŠENJE

3. (4 boda)

Shemu **potpora** treba izmijeniti – atribut akGodina SMALLINT treba zamijeniti atributom akGodinaPeriod tipa DATERANGE (prvo dodati novi atribut, ažurirati mu vrijednost u skladu s vrijednošću akGodina, potom ukloniti akGodina), uništiti postojeći PRIMARY KEY i kreirati temporalni primarni ključ.

```
CREATE EXTENSION bTree_gist; //studenti ne moraju napisati, ali inače treba
ALTER TABLE potpora DROP CONSTRAINT pkPotpora;
```

```
ALTER TABLE potpora ADD akGodinaPeriod DATERANGE;
Ažurirati vrijednosti ... UPDATE naredbe
```

```
ALTER TABLE potpora ADD CONSTRAINT pkPotpora
    PRIMARY KEY (akGodinaPeriod, oznDrzavaOdl, oznDrzavaDol); //ovo još
    uvijek ne osigurava temporalni PK, samo osigurava jednistvenost ove tri atributa
ALTER TABLE potpora ADD CONSTRAINT temporalPKPotpora EXCLUDE USING gist
    (oznDrzavaOdl WITH =, oznDrzavaDol WITH =, akGodinaPeriod WITH &&)
```

Sadržaj izmijenjene relacije:

<u>akGodinaPeriod</u>	<u>oznDrzavaOdl</u>	<u>oznDrzavaDol</u>	<u>mjlnosEur</u>
[1.10.2015, 1.10.2016)	HR	SE	400.00
[1.10.2016, 1.10.2018)	HR	SE	460.00
[1.10.2015, 1.10.2016)	SI	SE	500.00

Zbog u Postgresu podrazumijevane open-closed notacije za DATERANGE tip podatka ispravno je napisati periode [1.10.2015, 1.10.2016), a ne [1.10.2015, 30.9.2016].

Netrivialna naredba koja bi narušila constraint temporalPKPotpora:

```
INSERT INTO potpora VALUES ('[1.12.2015, 1.3.2016)', 'HR', 'SE', 420.00);
```

PostgreSQL pri provjeri constraint temporalPKPotpora provjerava postoji li u potporu n-torka sa istim oznDrzavaOdl i oznDrzavaDol (zbog oznDrzavaOdl WITH =, oznDrzavaDol WITH =) i

periodom koji se preklapa s periodom n-torke koju želimo insertirati (akGodinaPeriod WITH &&). Budući da takav period postoji (n-torka broj 1) sustav neće dozvoliti unos.

accomodUnit				
accomodUnitId	unitName	noOfPersons	Geom	...
1	Apartman Lavanda	4	<point>	
2	Apartman Oliva	2	<point>	
...	

metro			
lineId	lineName	Geom	...
1	From here to there	<polyline>	
2	And back again	<polyline>	
...	

user				
userId	FName	Lname	email	...
1	Ana	Car	...	
2	Ivo	Beg	...	
...	

slika 1

1. (5 bodova)

Postojeći segment baze podataka sa slike 1 potrebno je proširiti segmentom za pohranu podataka o cjeniku i rezervaciji smještajnih jedinica. Cjenikom se definira cijena smještaja u smještajnoj jedinici u određenom vremenskom periodu određenom datumom početka i završetka (npr. za smještajnu jedinicu s `accomodUnitId=1` cijena u periodu 01.05.2016-15.06.2016 iznosi 70€, a u periodu 15.06.2016-15.07.2016 iznosi 80€ po danu). Modelom osigurati jedinstvenost cijene za smještajnu jedinicu u konkretnom periodu.

Rezervacijom registrirani korisnik (**user**) rezervira smještaj u smještajnoj jedinici za određeni period boravka (određen datumom početka i kraja boravka). Za svaku rezervaciju se evidentira ukupna cijena određena temeljem cjenika za period. Modelom osigurati nemogućnost višestruke rezervacije iste smještajne jedinice u istom vremenskom periodu.

Korištenjem funkcionalnosti PostgreSQL SUBP napisati SQL naredbe za kreiranje potrebnih relacija. Tipove podataka odabratи proizvoljno, ali smisleno. Definirati temporalna integritetska ograničenja (primarne i strane ključeve) koja je moguće definirati u okviru PostgreSQL SUBP. Za integritetska ograničenja koja **nije** moguće implementirati u PostgreSQL-u objasnite kakvu implementaciju predviđa SQL standard.

RJEŠENJE

```

1.
CREATE TABLE priceList
(accomdUnitId      INTEGER REFERENCES accomodUnit (accomdUnitId),
priceListPeriod   DATERANGE,
price             DECIMAL (10,2) NOT NULL
PRIMARY KEY (accomdUnitId, priceListPeriod) CONSTRAINT PKPriceList,
CONSTRAINT temporalPKPriceList EXCLUDE USING gist
  (accomdUnitId WITH =, priceListPeriod WITH &&)
);

CREATE TABLE booking
(accomdUnitId      INTEGER,
bookingPeriod     DATERANGE,
userId            INTEGER REFERENCES user (userId),
totalPrice        DECIMAL (10,2) NOT NULL
PRIMARY KEY (accomdUnitId, bookingPeriod) CONSTRAINT PKBooking,
CONSTRAINT temporalPKBooking EXCLUDE USING gist
  (accomdUnitId WITH =, bookingPeriod WITH &&)
);

```

Modelom bi trebalo definirati temporalni strani ključ kojim se `booking` referencira na `priceList`

```

  booking (accomdUnitId, bookingPeriod) REFERENCES
priceList (accomdUnitId, priceListPeriod)

```

PostgreSQL ne podržava temporalni referencijski integritet.

SQL standard za temporalni referencijski integritet predviđa da je period referencirajuće n-torce u potpunosti sadržan u periodu jedne referencirane n-torce ili u kombiniranom periodu dvije ili više uzastopnih referenciranih n-torki.

Primijenjeno na gornje dvije relacije to bi značilo sljedeće:

```

Za isti accomdUnitId (booking.accomdUnitId = priceList.accomdUnitId)
  booking.bookingPeriod treba biti u potpunosti sadržan u priceList.priceListPeriod-uu
  jedne n-torce ili u kombiniranom periodu dvije ili više uzastopnih referenciranih n-torki

```

GIS

3. (4 boda) Definirajte običan i dimenzijski prošireni model devet presjeka. Objasnite čemu služe i zašto je prošireni „bolji“ od običnog. Napišite primjer matrice za oba modela.

RJEŠENJE

Model 9 presjeka (9IM)

- Binarna topološka relacija \mathbf{R} između dva prostorna objekta \mathbf{A} i \mathbf{B} opisuje se usporedbom unutrašnjosti (A^0), granice (∂A) i vanjštine (A^-) dvaju objekata.
- Tih 6 komponenata moguće je kombinirati tako da oblikuju 9 temeljnih vrijednosti (presjeka) za opis topoloških relacija
- Svaki presjek može poprimiti vrijednosti \emptyset i $\neg\emptyset$
- Uređeni skup 9 presjeka može se prikazati matricom:
-

$$R(A, B) = \begin{pmatrix} A^0 \cap B^0 & A^0 \cap \partial B & A^0 \cap B^- \\ \partial A \cap B^0 & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^0 & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

Dimenzijski prošireni model 9 presjeka (DE-9IM)

- Pored operatora unutrašnjosti (0), granice (∂) i vanjštine ($^-$), uvodi se i operator dimenzije
 - ako je $S = \emptyset$
 - 0 ako S sadrži barem točku, ali ne i linije i površine
 - 1 ako S sadrži barem liniju, ali ne površinu
 - 2 ako S sadrži barem površinu

Svaki element matrice proširuje se dimenzijom

- Novu matricu moguće je zapisati ovako:

$$DE9I = \begin{pmatrix} \dim(\partial\lambda_1 \cap \partial\lambda_2) & \dim(\partial\lambda_1 \cap \lambda_2^0) & \dim(\partial\lambda_1 \cap \lambda_2^-) \\ \dim(\lambda_1^0 \cap \partial\lambda_2) & \dim(\lambda_1^0 \cap \lambda_2^0) & \dim(\lambda_1^0 \cap \lambda_2^-) \\ \dim(\lambda_1^- \cap \partial\lambda_2) & \dim(\lambda_1^- \cap \lambda_2^0) & \dim(\lambda_1^- \cap \lambda_2^-) \end{pmatrix}$$

Dimenzijski prošireni model i matrica mogu se okarakterizirati kao „bolji“ od običnog modela i matrice jer su u stanju razlikovati veći broj različitih odnosa između dva prostorna objekta.

Baza podataka prikazana na slici 1 služi za evidenciju podataka o rezultatima referendumu koji se održavaju u Ujedinjenom Kraljevstvu Velike Britanije i Sjeverne Irske. Rezultati referendumu se bilježe u relaciji **result**. Osobe koje mogu pristupiti određenom referendumu (zajedno s detaljima kao što su predviđeno glasačko mjesto) evidentirane su u relaciji **personRef**. Teritorijalna podjela države, te glasačka mjesta u teritorijalnim jedinicama evidentirani su u relacijama **terUnit** i **poolPlace**. Ključevi relacija su podrtcani.

terUnit					poolPlace				
terUnitId	terUnitName	...	geom	supTerUnitId	poolPlaceId	poolPlaceName	...	terUnitId	
53	England		< polygon >	NULL	184	Islington Town Hall		856	
32	Scotland		< polygon >	NULL	132	Olive Morris House		437	
5478	East England		< polygon >	53	11	York House		976	
9056	London		< polygon >	53	
437	City of Edinburg		< polygon >	32					
856	West London		< polygon >	9056					
876	East London		< polygon >	9056					
...					

personRef				result				
refDate	personId	poolPlaceId	voted	refDate	poolPlaceId	ordinal	voteFor	valid
18.09.2014	234567	132	1	18.09.2014	132	1	1	1
23.06.2016	38954	184	0	23.06.2016	11	2	0	1
23.06.2016	6903423	11	1	23.06.2016	132	105	1	1
23.06.2016	5689045	184	1
...					

person			
personId	FName	LName	...
...

Slika 1.

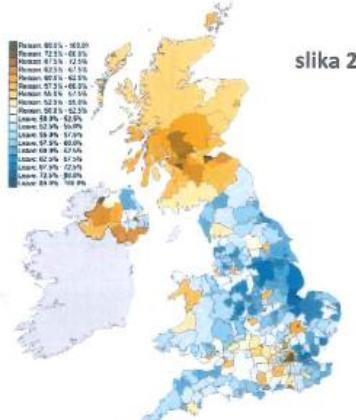
4. Prepostavimo da je vaš zadatak analizirati i vizualizirati rezultate izbora.

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.

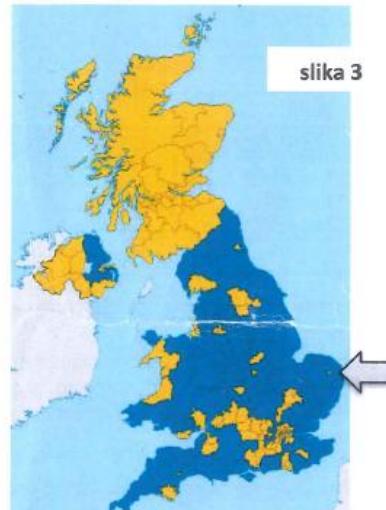
Potrebno je:

- (a) (3 boda) Napraviti vizualizaciju rezultata izbora po teritorijalnim jedinicama na dnu hijerarhije (listovi), poput ove preuzete s wikipedije:
Opišite kako biste napravili takvu vizualizaciju.
Kod dohvata podataka je potrebno navesti odgovarajuće SQL naredbe.
- (b) (4 boda) Potrebno je pronaći „relativno najmanju u odnosu na susjede“ teritorijalnu jedinicu koja je:
 - a. glasala za ostanak, i
 - b. koja je okružena teritorijalnim jedinicama koje su glasale za izlazak.
 Ili, preciznije rečeno, koja nema niti jednog susjeda koji je glasao za ostanak.
Pod „relativno najmanju u odnosu na susjede“ se misli da je potrebno razmatrati omjer površine te jedinice i sume površine njenih susjeda.
Opet, potrebno je razmatrati samo teritorijalne jedinice koje su najniže u hijerarhiji (listovi). Na slici 3 je strelicom označena jedinica koju je potrebno programski pronaći. Napišite pseudokod kojim biste pronašli takvu teritorijalnu jedinicu.

Pritom koristite SQL za opis dohvata podataka, pri čemu navedite i GIS funkcije koje koristite.



slika 2



slika 3

RJEŠENJE

GIS

(a)

Napraviti vizualizaciju rezultata izbora po najmanjim teritorijalnim jedinicama (najnižima u hijerarhiji), poput ove preuzete s wikipedije:
Opišite kako biste napravili takvu vizualizaciju.

Potrebno je opisati potencijalne izmjene na bazi u pogledu dodatnih podataka, te SQL naredbe kojima se pripremaju podaci potrebni za vizualizaciju.

```
// priznaje se i ako grupiraju po geom, inače trebalo bi napraviti podupit u
select
// listi ili sl.
SELECT terUnitId, geom, 100 * sum(voteFor) / COUNT(*) as remain
  FROM terUnit
    JOIN poolPlace ON terUnit.terUnitId = poolPlace.terUnitId
      JOIN result ON poolPlace.poolPlaceId = result.poolPlaceId
 WHERE valid = 1
   AND NOT EXISTS (SELECT * fromterUnitchild WHERE superUnitId =
terUnit.terUnitId)
 GROUP BY terUnitId, geom
```

Ovaj upit je ulaz u alat za vizualizaciju kao QGIS, gdje onda treba jednostavno dodijeli boja na temelju one mjere „remain“.

(b)

```
CREATE VIEW glasaleZa AS
SELECT terUnitId, geom
  FROM terUnit
    JOIN poolPlace ON terUnit.terUnitId = poolPlace.terUnitId
      JOIN result ON poolPlace.poolPlaceId = result.poolPlaceId
 WHERE valid = 1
   AND NOT EXISTS (SELECT * fromterUnitchild WHERE superUnitId =
terUnit.terUnitId)
 GROUP BY terUnitId, geom
 HAVING SUM(voteFor) > SUM(1 - voteFor);
MIN = {id: null, relArea: Float.MAX};

foreach (unitingglasaleZa) {
  LET cnt= SELECT COUNT(*)
    FROM glasaleZa
      WHERE st_touches(unit.geom, glasaleZa.geom)
        AND unit.terUnitId<>glasaleZa.terUnitId
  IF (cnt == 0) { // Nema susjeda istomišljenika
    currRelArea = SELECT st_area(unit.geom)/SUM(st_area(geom))
      FROM terUnit
        WHERE st_touches(unit.geom, terUnit.geom)
          AND NOT EXISTS (SELECT * fromterUnitchild
            WHERE superUnitId = terUnit.terUnitId);
    // ne treba gledati glasove, svi susjedi su protiv
    IF (MIN.relArea<currRelArea) {
      MIN.relArea = currRelArea;

      MIN.id = unit.terUnitId;
    }
  }
}
PRINT MIN
```

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 podrtani.

osoba			nekretnina				
sifOsoba	imeOsoba	prezOsoba	sifNekretnina	povrsina	vlasnici	adresa (pbr, ulica, broj)	geom
1	Slack	David	1	45.6	(13, 22, 5)	(10000, "Ilica", 12)	Polygon (...)
2	Segel	Amanda	2	380.4	{555}	(10000, "Vinogradска", 14)	Polygon (...)
3	Smith	Bob	3	33.5	{22}	(10000, "Ilica", 12)	Polygon (...)
4	Cardell	Ema	4	1087.3	{22, 13}	(10000, "Grada Vukovara", 12)	Polygon (...)
5	Tillman	Joan	5	234.5	{1}	(10000, "Grada Vukovara", 345)	Polygon (...)
			6	22.1	{5}	(44000, "Vinogradска", 12)	Polygon (...)
				

parcela		
brParcela	vlasnici	geom
101	{1}	Polygon (...)
102/1	{5}	Polygon (...)
102/2	{22, 13}	Polygon (...)

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)

RJEŠENJE

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

RJEŠENJE

4. (3 boda)

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

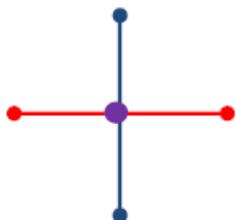
Presijecanje (*crosses*):

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

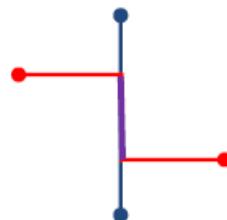
Preklapanje (*overlaps*):

$$\langle \lambda_1, \text{overlap}, \lambda_2 \rangle \Leftrightarrow (\dim(\lambda_1^0) = \dim(\lambda_2^0) = \dim(\lambda_1^0 \cap \lambda_2^0)) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_1) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_2)$$

Presijecanje:



Preklapanje:



electUnit			city		electPlace				
unitId	unitName	UnitShortName	cityId	cityName	population	placeId	placeName	unitId	cityId
1	I. izborna jedinica	I	1	Zadar	71471	1001	OŠ Tina Ujevića	7	10
2	II. izborna jedinica	II	2	Split	167121	1002	Mjesna zajednica Kruge	2	10
...

electList		party		
listId	listName	partyId	partyName	shortName
1	SDP + HNS + HSU + SR+ HSS + ZS	88	Socijaldemokratska partija Hrvatske	SDP
2	HDZ + HSS + HSP AS + BUŽ +...	245	Most nezavisnih lista	MOST
...

person				
personId	fName	lName	partyId	...
100	Zoran	Milanović	88	...
104	Drago	Prgomet	245	...
...

vote			
votId	placeId	personId	electDate
100001	1001	104	7.11.2015
100002	1001	101	7.11.2015
...

slika 1

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:

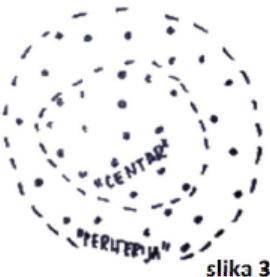
- (a) **(3 boda)** Za jednu stranku (bilo koju), prikazati **uspjeh te stranke u nijansama** neke boje **po izbornim jedinicama** (npr. slika 2). Uzmite u obzir da nemaju sve izborne jedinice isti broj glasača te stoga odaberite prikladnu mjeru uspješnosti.

Potrebno je opisati potencijalne izmjene na bazi u pogledu dodatnih podataka, te SQL naredbe kojima se pripremaju podaci potrebni za vizualizaciju.



slika 2

- (b) **(4 boda)** Za grad Zagreb je potrebno vizualizirati izborne rezultate s obzirom na udaljenost **glasačkih mesta** 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

RJEŠENJE

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 brojem glasača. Ne koristiti populaciju – nije dobro (npr. što ako 10% ljudi izđe na izbole 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 particoniranje, 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 radiusom, 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);
```

3. Vaš zadatak je napraviti turističku gastronomsku auto-kartu hrvatske.

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 i da možete raditi izmjene na bazi ako je potrebno.

(a) **(4 boda)** Potrebno je napraviti vizualizaciju auto-karte tako da se **motorne ceste i autoseste** po dionicama prikažu različitim bojama s obzirom na kvalitetu i brojnost ponude restorana koji pripadaju dionici. Možete smatrati da restoran pripada **najbližoj** motornoj cesti ili autosesti, a da kvalitetu restorana određuje prosječna ocjena.

Na primjer:

- dionica od 10 km koja ima jedan restoran ocjene 4 bi trebala biti bolje ocijenjena od dionice od 15 km kojoj također pripada jedan restoran ocjene 4.
- Isto tako, dionica od 10 km kojoj pripadaju dva restorana ocjene 4 bi trebala biti bolje ocijenjena od dionice od 10 km kojoj pripadaju dva restorana ocjene 2!

(b) **(4 boda)** Kako biste odredili područje s najvećom ponudom restorana po:

- „glavi stanovnika“
- m^2

Područje definirajmo kao veće mjesto (npr. $N > 30k$) s pripadajućim manjim mjestima.

Potrebno je opisati potencijalne izmjene na bazi u pogledu dodatnih podataka, te SQL naredbe kojima se pripremaju podaci potrebni za vizualizaciju.

RJEŠENJE

(a)

Potrebno je restoranu pridijeliti id dionice kojoj pripada na temelju udaljenosti koju računamo s `st_distance()`, npr.:

```
ALTER TABLE restaurant ADD idRoad int ;
UPDATE restaurant
SET idRoad = (SELECT idRoad
               FROM Road r
              WHERE st_distance(restaurant.geom, r.geom) =
                (
                  SELECT min(st_distance(resautarant.geom, allRoads.geom))
                     FROM road allRoads
                     LIMIT 1
                )
            )
```

Može i preko LIMIT:

```
UPDATE restaurant
SET idRoad = (SELECT idRoad
               FROM Road r
              ORDER BY st_distance(restaurant.geom, r.geom) ASC
                 LIMIT 1
            )
```

Zatim:

```
ALTER TABLE road ADD restDensity decimal(5,2);
```

```
UPDATE road
SET restDensity = 1./st_length(geom) * (SELECT AVG(avgGrade)
                                         FROM restaurant
                                         where idRoad = road.idRoad)
```

I sve slične mjere bi ja uvažio, koje zadovoljavaju zahtjeve.

Npr.: `max(grade)` + ovo gore

(b)

Odabro bih mjesta $>30k$, te napravio Voronoi dijagram kao na projektu.

(uglavnom se priznaju i rješenja s `st_distance`, pa `convexhull`, iako to nije skroz točno).

Potom bi to pohranio u bazu kao geometriju, jedan zapis po jednom $>30k$ mjestu u tablicu `podrucja`.

Zatim bi u:

(b1) dijelu uzeo:
(COUNT(*) FROM restaurant where `st_contains(područje.geom, rest.geom)`)
/(SUM(population) FROM place where `st_contains(područje.geom, place.geom)`)
a u (b2):
(COUNT(*) FROM restaurant where `st_contains (područje.geom, rest.geom)`)
/(SUM(`st_area(područje.geom)`)

Slika 1.

drzava			visokoUciliste		
Ozn Drzava	nazivDrzava	geom	sifVU	nazivVU	Ozn Drzava
HR	Republika Hrvatska	<polygon>	10	Sveučilište u Zagrebu	HR
SE	Kraljevina Švedska	<polygon>	15	Sveučilište u Splitu	HR
PL	Poljska	<polygon>	53	Fakultet elektrotehnike i računarstva	HR
...	...		54	Ekonomski fakultet	HR
			112	Kraljevsko tehničko sveučilište	SE
		

erasmusBoravak			potpora		ugovorAkGodina						
JMBAG	Ak Godina	sifVU Odl	sifVU Dol	brMjes	ak Godina	oznDrz Odl	oznDrz Dol	mjln Eur	ak Godina	sifVU Odl	sifVUDol
...	2015	53	112	5	2015	HR	SE	400	2015	53	[112, 345, ...]
...	2015	54	117	10	2016	HR	SE	460	2015	54	[113, 114, ...]
...	2015	SI	SE	500

4. Pretpostavimo da je vaš zadatak analizirati i vizualizirati podatke o Erasmus+ programu. 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**.

Potrebno je:

- (a) **(3 boda)** Označiti različitim bojama države s obzirom na razmjenu s ostalim državama, svih vremena. Kao mjeru razmjene uzeti broj razmijenjenih studenata (smjer razmjene nije bitan). Na primjer, ako je Hrvatska u cijeloj svojoj povijesti razmijenila s Poljskom 100 studenata, Švedskom 200, te Portugalom 30, suradnja Hrvatske je 330. Primijetite da se, npr., i Poljskoj broji tih istih 100 studenata, odnosno da se razmijenjeni studenti broje i jednoj i drugoj državi.
Opisite kako biste napravili takvu vizualizaciju.
Kod dohvata podataka je potrebno navesti odgovarajuće SQL naredbe.
- (b) **(4 boda)** Za svako sveučilište u Erasmus+ programu odrediti njemu najbliže sveučilište iz druge države
Pretpostavite da za svako sveučilište postoji samo jedno najbliže sveučilište.
Za ona sveučilišta:
 - koja su u simetričnom odnosu (međusobno su jedna drugom najbliža)
 - i čije države graničena karti je potrebno označiti poligon koji obuhvaća sva njihova visoka učilišta.
Napišite pseudokod kojim biste to ostvarili, pritom koristite SQL za opis dohvata podataka, pri čemu navedite i GIS funkcije koje koristite.

RJEŠENJE

4.

(a)

```
// priznaje se i grupiranje po geom, inače trebalo bi napraviti podupit u select
// listi ili sl.
CREATE TEMP TABLE suradnja AS
SELECT drzava.OznDrzava, nazDrzava, geom, COUNT(*) as promet
FROM visokoUciliste
JOIN drzava
    ON visokoUciliste.OznDrzava = drzava.OznDrzava
JOIN erasmusBoravak
    ON (sifVuOdl = sifVu OR sifVuDol = sifVu)
GROUP BY drzava.OznDrzava, nazDrzava, geom
```

Ovaj upit je ulaz u alat za vizualizaciju kao QGIS, gdje onda treba jednostavno dodijeli boja na temelju mjere „promet“.

(b)

```
CREATE VIEW sveucilista AS
SELECT sv1.sifVu, sv1.oznDrzava,
       sv2.sifVu as sifVu2, sv2.oznDrzava as oznDrzava2,
       st_distance(sv1.geom, sv2.geom) as distance
FROM visokoUciliste sv1
JOIN visokoUciliste sv2 ON sv1.oznDrzava <> sv2.oznDrzava -- suvišno
WHERE sv1.sifNadVu IS NULL
      AND sv2.sifNadVu IS NULL
      AND st_distance(sv1.geom, sv2.geom) =
          (SELECT MIN(st_distance(sv1.geom, sv2.geom))
           FROM visokoUciliste sv3
           WHERE sv1.oznDrzava <> sv3.oznDrzava
             AND sv3.sifNadVu IS NULL
          )
CREATE VIEW sim_parovi AS
SELECT s1.sifVu, s1.sifVu2

      FROM sveucilista s1
      JOIN sveucilista s2 ON s1.sifVu2 = s2.sifVu
                           AND s2.sifVu2 = s1.sifVu
      JOIN drzava d1 ON s1.oznDrzava = d1.oznDrzava
      JOIN drzava d2 ON s2.oznDrzava = d2.oznDrzava
      WHERE st_touches(d1.geom, d2.geom);

foreach (par in sim_parovi) {
    LET geom = SELECT ST_ConvexHull(ST_Collect(visokoUciliste.geom))
                  FROM visokoUciliste
                  WHERE sifNadVu in (par.sifVu, par.sifVu2)
    // draw geom
}
```



Can they solve it? Can they?

Segment baze podataka prikazan na **slici 1** služi za evidenciju podataka o putovanjima autobusom u ponudi turističke agencije. U relaciji ***putLok*** su opisane dionice konkretnog putovanja. Atribut ***putLok.geom*** predstavlja dionicu ceste između početne i završne lokacije te dionice. Atribut ***putTermin.cijenaHRK*** predstavlja cijenu putovanja za jednu osobu. Ključevi relacija su podcrteni.

lokacija				putovanje			drzava		
<i>sifLok</i>	<i>nazivLok</i>	<i>oznDrz</i>	<i>geom</i>	<i>sifPut</i>	<i>nazivPut</i>	...	<i>oznDrz</i>	<i>nazivDrz</i>	<i>geom</i>
11	Split	HR	<point>	20	Ljepote Jadrana		HR	Hrvatska	<polygon>
12	Plitvice	HR	<point>	21	Vikend u Dubrovniku		AT	Austrija	<polygon>
13	Dubrovnik	HR	<point>	23	Austrijska avantura		PL	Poljska	<polygon>
...	24	Finske bijele noći

putLok					putTermin				
<i>sifPut</i>	<i>rbrDio</i>	<i>sifLokPoc</i>	<i>sifLokZav</i>	<i>geom</i>	<i>sifPut</i>	<i>datPolazak</i>	<i>cijenaHRK</i>	<i>ukMjesta</i>	<i>prodano Mjesta</i>
20	2	12	11	<polyline>	20	01.05.2016	1200	50	50
20	3	11	13	<polyline>	20	01.07.2016	1400	80	72
21	1	8	13	<polyline>	21	15.06.2016	400	50	46
21	4	11	15	<polyline>	22	04.06.2016	3200	50	50
21	6	16	13	<polyline>
...

- 4. (5 bodova)** Napišite jedan ili više upita kojim ćete dobiti popis međunarodnih putovanja koja kreću iz Hrvatske, pri čemu za svako putovanje treba ispisati ukupno duljinu, te duljinu putovanja po cestama izvan Hrvatske.

RJEŠENJE

4. (5 bodova)

```
// može i bez pomoćnog VIEW-a sve u jednom upitu:
CREATE VIEW putMedjSPolaskomIzHR AS
SELECT DISTINCT putovanje.sifPut,
               putovanje.nazivPut
FROM putovanje
JOIN putLok ON putovanje.sifPut= putLok.sifPut
           AND rbrDio = 1
JOIN lokacija ON putLok.sifLokPoc = lokacija.sifLok
WHERE lokacija.oznDrzava = 'HR'
AND EXISTS (SELECT *
             FROM putLok
             JOIN lokacija ON putLok.sifLokPoc = lokacija.sifLok
             WHERE sifPut = putovanje.sifPut
             AND oznDrz <> 'HR')

SELECT putMedjSPolaskomIzHR.*,
       SUM(st_length(putLok.geom)) as ukPut,
       SUM(st_length(st_intersection(putLok.geom,
                                     (SELECT st_union(geom) FROM drzava WHERE oznDrz <> 'HR'))--sve ostale
          )))
FROM putMedjSPolaskomIzHR
JOIN putLok ON putMedjSPolaskomIzHR.sifPut = putLok.sifPut
```

--treba paziti, neke dionice mogu biti skroz vanjske, ili između dvije strane države, --npr. Putovanje iz Hr u Mađ u Austr u Slo u Hr.

Može još elegantnije ako se napravi presjek s Hrvatskom, pa izračuna ukupno minus ceste po Hr:

```
putMedjSPolaskomIzHR.*,
SUM(st_length(putLok.geom)) as ukPut,
SUM(st_length(putLok.geom)) -
SUM(st_length(st_intersection(putLok.geom,
                             (SELECT st_union(geom) FROM drzava WHERE oznDrz = 'HR')
          )))
```

accomodUnit				
accomodUnitId	unitName	noOfPersons	Geom	...
1	Apartman Lavanda	4	<point>	
2	Apartman Oliva	2	<point>	
...	

metro				
lineId	lineName	Geom	...	
1	From here to there	<polyline>		
2	And back again	<polyline>		
...		

user				
userId	FName	Lname	email	...
1	Ana	Car	...	
2	Ivo	Beg	...	
...		

slika 1

2. (5 bodova)

Pretpostavimo da želite rezervirati smještaj u nekom gradu, pri čemu ste posebno zainteresirani za određenu **znamenitost** (npr. Louvre muzej).

Želimo odrediti koliko ima smještaja (tablica accomodUnit) u dvije zone:

- **Zona A** – smještaj udaljen **manje od 10 km** od znamenitosti
- **Zona B** – smještaj udaljen **više od 10 km, a manje od 20km** od znamenitosti

Pritom, zanimaju nas samo oni smještaji koji su u dosegu metroa. Smatramo da je smještaj **u dosegu metroa** ako je udaljen **manje od 500m od neke metro linije** (tablica metro).

Napišite pseudokod koji sadrži odgovarajuće SQL naredbe pomoću kojeg bi doznali **broj smještaja, prosječnu udaljenost i minimalnu udaljenost po zonama A i B za smještaje za barem dvije osobe koji su u dosegu metroa**. Znamenitost je predstavljena točkom i možete prepostaviti da vam se nalazi u varijabli **znam**.

RJEŠENJE

2.

```

LET znam = POINT(x, y); // zadana je
// Može se naravno i drugacije rjesiti, npr. pomocu st_distance.
LET zonaA = st_buffer (znam, 10000);
LET zonaB = st_difference(st_buffer (znam, 20000), zonaA);
LET metro = SELECT st_union(st_buffer(geom, 500)) FROM metro;
// Ja cu napisati jedan upit, ali moglo se i dva, posebno za zonu A i B:
SELECT CASE WHEN st_contains(zonaA, geom) THEN 'Zona A' ELSE 'Zona B' END as zona
        COUNT(*),
        AVG (st_distance(znam, geom)) as prosj,
        MIN (st_distance(znam, geom)) as minm
FROM accomodUnit
WHERE noOfPersons >= 2
    AND st_contains(metro, geom)
    AND (st_contains(zonaA, geom) OR st_contains(zonaB, geom))
GROUP BY zona

```



Svašta nešto

3. **(6 bodova)** Objasniti distribucijske modele u NoSQL sustavima i navesti kako utječu na čitanje odnosno pisanje podataka. Navedite jedan primjer.

RJEŠENJE

Pogledati predavanja, to je teoretsko pitanje, slajdovi 7-10 u „NoSQL 2/3“

5. **(3 boda)** Što je Bloomov filter? Koja su njegova svojstva? Navedite primjer s barem tri člana skupa.

RJEŠENJE

5. **(5 bodova)**

Što je Bloomov filter i čemu služi? Koja su njegova svojstva. Navedite parametre koje uzimamo u obzir kad projektiramo Bloomov filter i kako one utječu na njegovu točnost. Navedite primjer.

BF je probabilistička podatkovna struktura koja omogućuje kompaktno pohranjivanje informacije o članstvu u skupu na račun točnosti. BF može dati lažne pozitivne odgovore (da element jest član skupa), ali ne i lažne negativne. Razmatramo m - broj bitova BF-a, k - broj hash funkcija i n – očekivani broj elemenata u skupu. Veći broj bitova povećava točnost BF-a, veći n naravno smanjuje, a k ima neki optimum za m i n.

4. **(6 bodova)** Objasnite što je to vektorski sat i navedite pravila koja se koriste kod vektorskih satova. Objasnite na primjeru. Primjer mora uključivati barem jedan konflikt.

RJEŠENJE

4. Vidjeti predavanja:



3. (4 boda) Komentirajte potporu za transakcije u NoSQL sustavima.
Kakva je povezanost modela podataka s (ne)mogućnošću obavljanja transakcija u NoSQL sustavima?
Objasnite pomoću primjera.

RJEŠENJE

Map Reduce

5. (4 boda) Napisati MapReduce upit koji će prebrojiti rezultate izbora na temelju tablice **results**. Rezultat upita treba biti broj glasova za ostanak, izlazak i broj nevažećih glasova za sve identifikatore glasačkog mjesto, tako da na temelju rezultata, npr. možemo popuniti sljedeću tablicu:

poolPlaceId	voteStay	voteLeave	invalidVotes
1	12345	54321	123
2	2131	1233	31
3	34234	34432	34
...

Prepostaviti istu M/R arhitekturu (s obzirom na *(combinable) reducer* i sl.) kao MongoDB.

result				
refDate	poolPlaceId	ordinal	voteFor	valid
18.09.2014	132	1	1	1
23.06.2016	11	2	0	1
23.06.2016	132	105	1	1
...

RJEŠENJE

M/R rješenje:

```
map (k, v) {
    // nije nužno množiti s valid,
    emit(v.poolPlaceId, {
        stay: (v.voteFor == 1) * v.valid,
        leave: (v.voteFor == 0) * v.valid,
        invalid: (1 - v.valid)
    });
}
reduce (k, vlist) {
    var agg = { stay: 0, leave: 0, invalid: 0 };
    foreach (v in vlist) {
        agg.stay += v.stay;
        agg.leave += v.leave;
        agg.invalid += v.invalid;
    }
    returnagg;
}

// finalize nije potreban
```

4. **(10 bodova)** Ulagni skup podataka je (pojednostavljena) log datoteka nekog web servera. Primjer sadržaja, uz objašnjenje značenja atributa u zapisu, je prikazan na slici:

IP adresa	Vrijeme	Zahtjev	Response Code	Bytes	Client
161.53.200.56	[09/Dec/2013:11:38:15 +0100]	"GET /dummy/reports.css HTTP/1.1"	304	-	"Internet Explorer 6.0"
161.53.150.29	[09/Dec/2013:11:59:58 +0100]	"HEAD /robots.txt HTTP/1.0"	404	-	"-"
161.53.100.37	[09/Dec/2013:14:39:20 +0100]	"GET /fer/foo HTTP/1.1"	401	1373	"Mozilla Firefox v25"
161.53.100.37	[09/Dec/2013:14:39:24 +0100]	"GET /favicon.ico HTTP/1.1"	200	7274	"Mozilla Firefox v25"
161.53.100.37	[09/Dec/2013:14:40:14 +0100]	"GET /fer/ HTTP/1.1"	200	2773	"Mozilla Firefox v25"
161.53.100.37	[09/Dec/2013:14:40:14 +0100]	"GET /icons/blank.gif HTTP/1.1"	200	148	"Mozilla Firefox v25"
161.53.100.37	[09/Dec/2013:14:40:14 +0100]	"GET /icons/back.gif HTTP/1.1"	200	216	"Mozilla Firefox v25"
161.53.100.37	[09/Dec/2013:14:40:14 +0100]	"GET /icons/compressed.gif HTTP/1.1"	200	1038	"Mozilla Firefox v25"
161.53.100.19	[09/Dec/2013:14:41:30 +0100]	"GET /fer/ HTTP/1.1"	401	1367	"Chrome v31.0.1650.63"
...					

Potrebno je napisati MapReduce algoritam kojim će se dobiti ukupan broj zahtjeva i isporučenih bajtova za uspješne zahtjeve (HTTP Response Code je 200) po različitim klijentima (Internet preglednicima) u 2014. godini.

Objasnite (nacrtajte za npr. tri čvora) kako se obavlja vaš MR algoritam.

RJEŠENJE

4.

Uvažava se map koji prima jedan redak ili n redaka:

```
map (k, v)
for line = v.readLine()
    words = line.splitWords();
    if (w[3] == "200" && year(w[1]) == 2014)
        emit (w[5], {cnt: 1, bytes: w[4]})
```

Dobra su i rješenja koja u map funkciji agregiraju (najčešće koristeći dictionary/hashmap structure) podatke na razini preglednika, odnosno emitiraju samo jednu informaciju po pregledniku.

```
reduce (k, v)
sum = {cnt: 0, bytes: 0}
foreach req in v
    sum.cnt += req.cnt
    sum.bytes += req.bytes
return sum
```

6. (8 bodova) Ulazna datoteka sadrži podatke o mjerjenjima temperature i tlaka u 12:00 sati u svim mjestima u Hrvatskoj (za veći niz godina).

Napišite **M/R algoritam** koji će za sva mjesta vratiti prosječne mjesecne vrijednosti temperature i tlaka **po mjesecima 2014.** godine.

Nacrtajte čvorove, tijek podataka i **izlaze** iz map i reduce faza, te konačan rezultat za ove testne ulazne podatke:

```
(Zg, 31.12.2013, 1005, -3)
(Zg, 01.01.2014, 1010, -2)
(Zg, 02.01.2014, 1000, -4)
(Pu, 31.12.2013, 1000, +7)
(Pu, 01.01.2014, 1002, +5)
(Pu, 02.01.2014, 1004, +6)
```

i pritom pretpostavite da se izračunavanje obavlja na **tri čvora**. Prepostaviti općenitu paradigmu s više reducera (tj. u ovom primjeru - tri reducera).

Kod „implementacije“ možete koristiti bilo koji programski jezik ili pseudo kod. Pritom, ako koristite pseudo kod, ne smijete prelaziti granice mogućnosti modernih programskih jezika, odnosno smijete koristiti različite strukture podataka (npr. hashmap) i funkcije (npr. sort, splitWords), ali ne smijete koristiti izmišljene napredne funkcije primjenjive upravo na ovaj problem (npr. rjesiZadatak3()).

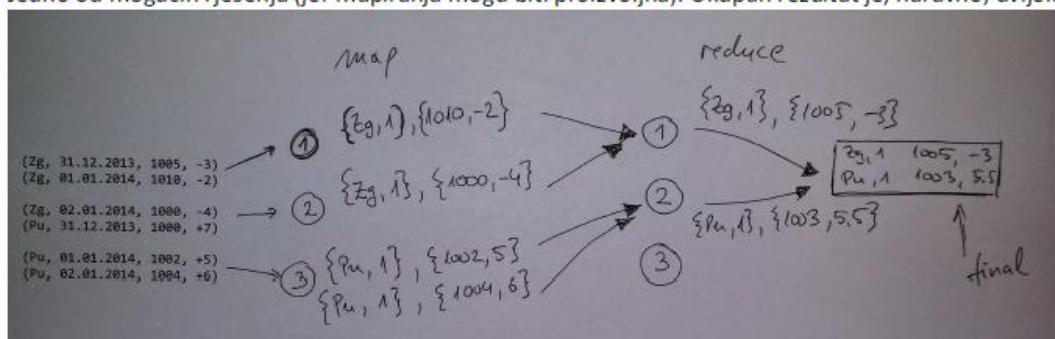
RJEŠENJE

6.

```
map (k, v) {
    foreach (item in v) {
        if (year(v[1]) == 2014) {
            key = {
                grad: v[0],
                mjesec: month(v[0])
            }
            val = {
                tlak: v[2],
                temp: v[3]
            }
            emit (key, val);
        }
    }
}

reduce (k, v) {
    foreach (pair in v) {
        sumP += pair.tlak;
        sumT += pair.temp;
    }
    val = {
        avgTlak: sumP/v.length,
        avgTemp: sumT/v.length,
    }
    return (k, val)
}
```

Jedno od mogućih rješenja (jer mapiranja mogu biti proizvoljna). Ukupan rezultat je, naravno, uvijek isti:



5. (10 bodova) Za svaki par članova društvene mreže potrebno je izračunati koliko imaju zajedničkih prijatelja, tako da je korisnika moguće obavijestiti npr.: „Vi i Pero imate 17 zajedničkih prijatelja“. Neka su podaci za svakog člana pohranjeni kao (član, lista_prijatelja) što ujedno predstavlja ulazne podatke u Map/Reduce algoritam kojim je potrebno rješiti ovaj zadatak.
- Napišite **M/R algoritam** kojim se za svaki par članova izračunava broj zajedničkih prijatelja. Napišite **izlaze** iz map i reduce faza, te **konačan rezultat** za ove testne ulazne podatke (članovi mreže su predstavljeni varijablama A, B, C i D):

- (A, BCD)
- (B, AD)
- (C, A)
- (D, AB)

i pritom prepostavite da se izračunavanje obavlja na **tri čvora**.

Kod „implementacije“ možete koristiti bilo koji programski jezik ili pseudo kod. Pritom, ako koristite pseudo kod, ne smijete prelaziti granice mogućnosti modernih programskih jezika, odnosno smijete koristiti različite strukture podataka (npr. hashmap) i funkcije (npr. sort, splitWords), ali ne smijete koristiti izmišljene napredne funkcije primjenjive upravo na ovaj problem (npr. rjesiZadatak3()).

RJEŠENJE

5. (10 bodova)

Prvo komentirajmo najčešća **pogrešna** rješenja:

Mnogi su studenti u map funkciji samo proslijedivali podatke dalje u reduce, ili još gore – razlomili ih na manje dijelove bez ikakve dodatne obrade, čime su samo povećali mrežni promet (npr. za A->BCD emitiraju A->B, A->C, A->D).

Dva tipična nastavka na ovo su:

- Studenti su prepostavili da postoji samo jedan reducer i sve, više ili manje „uspješno“, napravili u reducera. Ovdje je važno podsjetiti da je M/R namijenjen **paralelnom** obavljanju, a ako samo raspršimo podatke kako bi ih kasnije skupili i sve obavili u jednom jedinom čvoru onda smo **potpuno promašili smisao M/R-a** i zapravo sve obavili **na neučinkovitiji način**.
- Postoji više reducera, ali budući da zapisi nisu dobro raspršeni, ne zna se koji će gdje završiti i ne može se točno odrediti presjek.

Ključno u rješavanju ovog zadatka je bilo dosjetiti se da map funkcija raspršiti (emitira) podatke po **abecedno poredanim ključevima parova**.

Npr. za A->BCD, treba emitirati AB->CD, AC->BD i AD->BC.

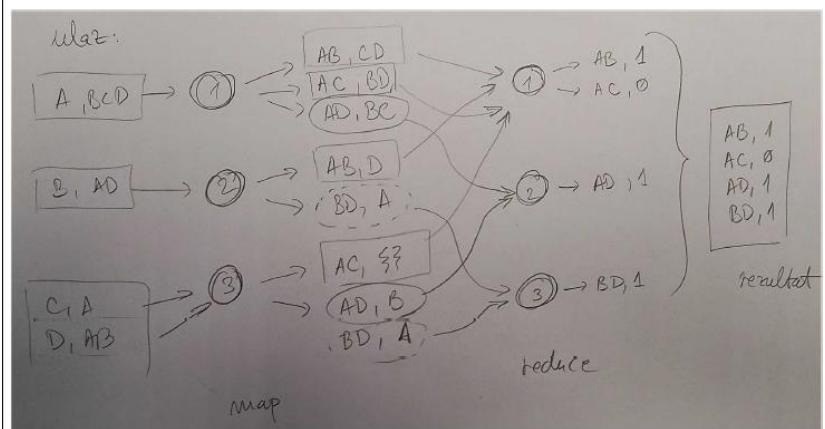
Na nekom drugom, ili istom čvoru, B će emitirati svoje parove i, ako je npr. A među njegovim prijateljima, svakako će emitirati **AB->ostali_B_prijatelji**.

M/R nam nakon toga **garantira** da će svi AB->^{*} završiti **u istom reducer čvoru**, koji onda samo treba napraviti presjek i odrediti njegovu kardinalnost.

```
map (k, v) {
    foreach (item in v) {
        list = k, item;
        list.sort();
        newkey = list.concat();
        newlist = v.minus(item);
        emit (newkey, newlist);
    }
}

reduce (k, v) {
    // v lista koja uvijek sadrži dvije liste
    mutual = v[0].intersect(v[1]);
    return (k, mutual.size());
}
```

Jedno od mogućih rješenja (jer mapiranja mogu biti proizvoljna). Ukupan rezultat je, naravno, uvijek isti:



6. (5 bodova) Napisati MapReduce upit koji će na temelju upita koji spaja tablice erasmusBoravak i visokoUciliste vratiti ukupan broj godina i mjeseci ostvarene suradnje između svakog para sveučilišta tako da npr. možemo popuniti sljedeću tablicu:

sveučiliše 1	sveučiliše 2	godina	mjeseci
Kraljevsko tehničko sveučilište	Sveučilište u Zagrebu	13	9
Kraljevsko tehničko sveučilište	Sveučilište u Splitu	15	0
...

Svaki par ispisati samo jednom.

Prepostaviti istu M/R arhitekturu (s obzirom na (*combinable*) reducer i sl.) kao MongoDB.

erasmusBoravak

JMBAG	Ak Godina	sifVU Odl	sifVU Dol	brMjes
...	2015	53	112	5
...	2015	54	117	10
...

RJEŠENJE

```

6.
map (k, v) {
    // poredamo ih, abecedno ili po šifri, svejedno, bitno je da je konzistentno
    // ideja je da se i (33, 35) i (35, 33) emitiraju kao jedan ključ, npr. (33, 35)
    Sv1 = MIN(k.nazivSveuc1, k.nazivSveuc2)
    Sv2 = MAX(k.nazivSveuc1, k.nazivSveuc2)
    emit({Sv1, Sv2}, k.brMjes);
}
reduce (k, vlist) {
    // ovdje se ne smije pretvarati u godine, jer je CR
    // zapravo, dalo bi se, ali nije elegantno
    var sumMj = 0;
    foreach (v in vlist) {
        sumMj += v;
    }
    return sumMj;
}
finalize (k, v) {
    return { k.Sv1, k.Sv2, v / 12, v % 12 };
}

```

5. (5 bodova) Napisati MapReduce upit koji će na temelju tablice putTermin za svako putovanje (šifru) ispisati kolika je promjena prihoda u postotcima u 2016. godini u odnosu na 2015. godinu, npr.

sifPut	Promjena
13	-10%
15	20%
...	...

Prepostaviti istu M/R arhitekturu (s obzirom na (*combinable*) reducer i sl.) kao u MongoDB.

putTermin

sifPut	datPolazak	cijenaHRK	ukMjesta	prodanoMjesta
20	01.05.2016	1200	50	50
20	01.07.2016	1400	80	72
21	15.06.2016	400	50	46
22	04.06.2016	3200	50	50
...

RJEŠENJE

5. (5 bodova)

```
map () {
    god = YEAR(this.datPolazak)
    emit(this.sifPut, {
        g15: god == 2015 ? (this.cijenaHRK*this.prodanoMjesta) : 0,
        g16: god == 2016 ? (this.cijenaHRK*this.prodanoMjesta) : 0
    });
}
reduce (k, vlist) {
    var uk = {g15: 0, g16: 0};
    foreach (v in vlist) {
        uk.g15 += v.g15;
        uk.g16 += v.g16;
    }
    return uk;
}
finalize (k, v) {
    return { k, (v.g16/v.g15) * 100 - 100 };
```

4. (5 bodova)

Napisati MapReduce upit koji će vratiti listu smještaja u Parizu s njihovim prosječnim ocjenama (AccId, AvgRating).

Ulagani podaci su korisničke ocjene smještaja u JSON obliku:

```
{User: "Anne", City: "Paris", AccId:101, Review: "Very nice place!", Rating: 5}
{User: "Peter", City: "London", AccId:111, Review: "Flea bag", Rating: 2}
{User: "Đuro", City: "Paris", AccId:101, Review: "Super-super!", Rating: 5}
{User: "Zoe", City: "Paris", AccId:102, Review: "Seen better", Rating: 3}
...

```

Pretpostaviti istu M/R arhitekturu (s obzirom na *(combinable) reducer* i sl.) kao MongoDB.

RJEŠENJE

4.

```
map (k, v) {
    if (v.City == 'Paris') {
        emit(v.AccId, {sum: v.Rating, count: 1});
    }
}
// moglo se i stalno konkatenirati u listu, ali to nije baš sjajno rješenje
// - posao se prebacuje na finalize
reduce (k, vlist) {
    var agg = {sum: 0, count: 0};
    foreach (v in vlist) {
        agg.sum += v.sum;
        agg.count += v.count;
    }
    return agg;
}

finalize (k, vlist) {
    return vlist.first().sum / vlist.first().count;
}
```

Semantički web

7. (5 bodova) Napravite RDF model podataka (nacrtajte graf) za nekoliko osoba koje:

- se mogu međusobno poznavati
- neke od njih mogu imati kućne ljubimce različitih vrsta (pas, mačka, ...)

Na temelju vlastitog modela napišite upit koji će za svaku osobu vratiti ime, prezime i broj poznanika koji imaju psa.

RJEŠENJE

```
PREFIX foaf: <http://.../>
PREFIX pets: http://.../
...
SELECT DISTINCT ?name (COUNT(?friend) AS ?num)
WHERE {
  ?user foaf:firstName ?name .
  ?user foaf:knows ?friend .
  ?friend pets:hasPet ?pet .
  ?pet rdf:type pets:Dog .
}
GROUP BY ?name
```

6. (3 boda) Objasnite „Open World“ i „Unique Name“ prepostavke. Vrijede li one u OWL-u?

RJEŠENJE

7. (3 boda) Što je konzistentno raspršenje? Navedite primjer.

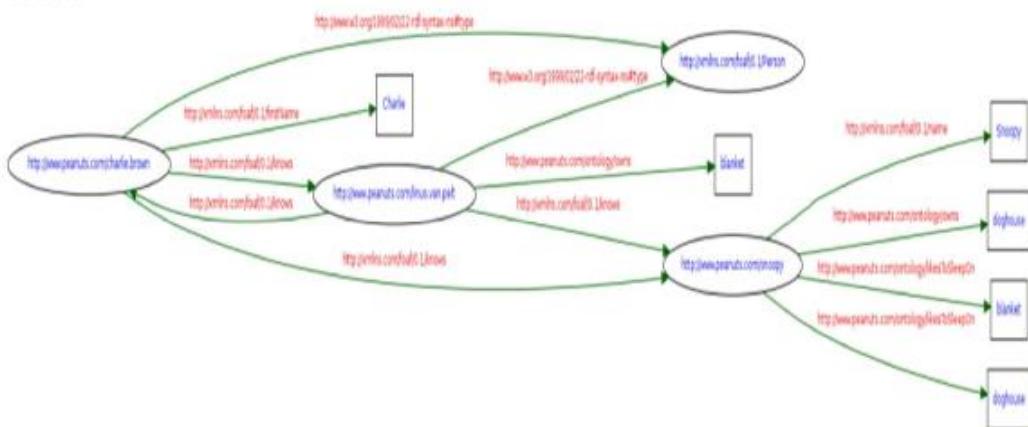
RJEŠENJE

5. (6 bodova) Na temelju zadanih podataka u N3/Turtle formatu, nacrtajte RDF graf:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@prefix pean: <http://www.peanuts.com/ontology/> .  
  
<http://www.peanuts.com/charlie.brown>  
    a                      foaf:Person ;  
    foaf:firstName "Charlie" ;  
    foaf:knows            <http://www.peanuts.com/linus.van.pelt> ;  
    foaf:knows            <http://www.peanuts.com/snoopy> .  
<http://www.peanuts.com/linus.van.pelt>  
    rdf:type              foaf:Person ;  
    foaf:knows            <http://www.peanuts.com/charlie.brown> ;  
    foaf:knows            <http://www.peanuts.com/snoopy> ;  
    pean:owns             "blanket" .  
<http://www.peanuts.com/snoopy>  
    foaf:name              "Snoopy" ;  
    pean:owns              "doghouse" ;  
    pean:likesToSleepOn    "blanket" ;  
    pean:likesToSleepOn    "doghouse" .
```

RJEŠENJE

5. zadatak



6. **(4 boda)** Nadopunite sljedeće SPARQL upite nad semantičkim izvorom DBpedia:

Prepostavite upotrebu sljedećih prefiksa:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
```

(a) (2 boda) Koliko regija u Njemačkoj ima više od 5 milijuna stanovnika?

```
WHERE {
    ?regija a dbpedia-owl:Region .
    ?regija dbpedia-owl:country <http://dbpedia.org/resource/Germany> .
    ?regija dbpprop:population ?brojSt
}
} -----
```

(b) (2 boda) Da li u Turskoj postoji naseljeno mjesto čiji engleski naziv je "Batman"?

```
WHERE {
    ?mjesto a dbpedia-owl:PopulatedPlace .
    ?mjesto dbpedia-owl:country <http://dbpedia.org/resource/Turkey> .
    ?mjesto foaf:name _____
}
} -----
```

RJEŠENJE

a)

```
SELECT COUNT(?regija)
WHERE {
    ?regija a dbpedia-owl:Region .
    ?regija dbpedia-owl:country <http://dbpedia.org/resource/Germany> .
    ?regija dbpprop:population ?brojSt
    FILTER(?brojSt > 5000000)
}
```

b)

```
ASK
WHERE {
    ?mjesto a dbpedia-owl:PopulatedPlace .
    ?mjesto dbpedia-owl:country <http://dbpedia.org/resource/Turkey> .
    ?mjesto foaf:name "Batman"@en
}
```

7. Nadopunite sljedeće SPARQL upite nad semantičkim izvorom DBpedia:

Prepostavite upotrebu sljedećih prefiksa

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
```

- a) **(1 bod)** Izračunajte ukupnu zaradu svih filmova u kojima je glumio George Clooney. Uzmite u obzir samo one filmove koji su pojedinačno zaradili preko 100 milijuna dolara.

```
WHERE {
  ?film a dbpedia-owl:Film .
  ?film dbpedia-owl:starring <http://dbpedia.org/resource/George_Clooney> .
  ?film dbpedia-owl:gross ?zarada .

}
```

- b) **(2 boda)** Napišite upit koji će dati odgovor na pitanje:

Da li su George Clooney (http://dbpedia.org/resource/George_Clooney) i Brad Pitt (http://dbpedia.org/resource/Brad_Pitt) glumili (dbpedia-owl:starring) u još nekom filmu osim u onima čiji naziv počinje sa „Ocean“? Rezultat upita mora biti true ili false.

```
WHERE {
  ?film a dbpedia-owl:Film .
  ?film dbpprop:name ?naziv .

}


```

RJEŠENJE

7. a)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT SUM(xsd:integer(?zarada))
WHERE {
  ?film a dbpedia-owl:Film .
  ?film dbpedia-owl:starring <http://dbpedia.org/resource/George_Clooney> .
  ?film dbpedia-owl:gross ?zarada
  FILTER (xsd:integer(?zarada) > 100000000)
}
```

Cast operator (xsd:integer) je nužan za funkcioniranje upita na Dbpedia krajnjoj točki (potrebno je pretvoriti tip podataka US\$ u cijeli broj), no u ispitu se priznaje i rješenje bez cast operatora.

b)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
ASK
WHERE {
  ?film a dbpedia-owl:Film .
  ?film dbpedia-owl:starring <http://dbpedia.org/resource/George_Clooney> .
  ?film dbpedia-owl:starring <http://dbpedia.org/resource/Brad_Pitt> .
  ?film dbpprop:name ?naziv .
  FILTER (!REGEX(?naziv, "Ocean"))
}
```

7. (4 boda) Kako bi mrežu cesta i gradova modelirali koristeći grafovski model podataka? Na vašem modelu riješite prvi zadatak, tj. napišite Cypher upit.

Mjesta su čvorovi sa svojstvima naziv, population i sl. Ceste se modeliraju kao relationships između mjesta, i mogu biti tipa Highway, motorway, ...
Cypher otplikle:

```
MATCH (zg: Place{name:"Zg"})-[r:Highway*]-(odrediste:Place{population>50000})
RETURN odrediste, SUM(r.length)
```

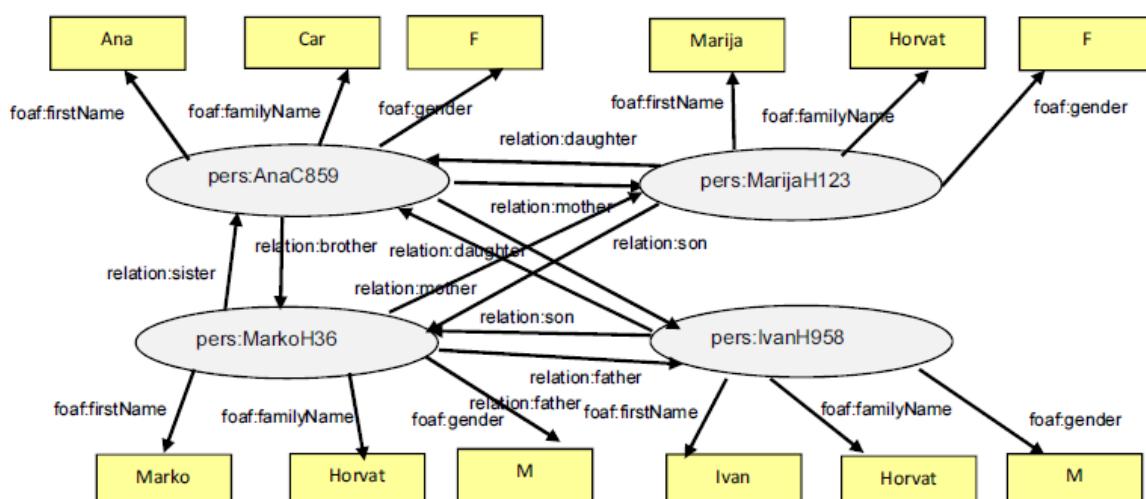
7. (5 bodova) Napravite RDF model podataka (nacrtajte graf) za jednostavno obiteljsko stablo kojim će opisati odnose majka/otac, brat/sestra i kći/sin za neku izmišljenu obitelj.

Minimalan skup svojstava koje modelom treba moći opisati za osobu su ime, prezime i spol. U modelu možete koristiti proizvoljne (postojeće ili zamišljene) rječnike. Vodite računa o tome da svaku osobu morate moći jedinstveno identificirati.

Na temelju vlastitog modela napišite SPARQL upit koji će vratiti prezimena i imena osoba čije bake imaju jednako ime. Primjer: Marija Car i Marija Lončar su bake Ane Horvat pa se Ana Horvat treba nalaziti u rezultatu upita.

RJEŠENJE

7. (5 bodova)



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/
PREFIX pers: < http://www.persons.com/allPersons/>
PREFIX relation: <http://.../>
```

```
SELECT ?familyName ?firstName
WHERE {
  ?person relation:father ?father .
  ?person relation:mother ?mother .
  ?father relation:mother ?fatherMother .
  ?mother relation:mother ?motherMother .
  ?fatherMother foaf:firstName ?fatherMotherFName .
  ?motherMother foaf:firstName ?motherMotherFName .
  ?person foaf:firstName ?firstName .
  ?person foaf:familyName ?familyName
  FILTER(?fatherMotherFName = ?motherMotherFName)
}
```

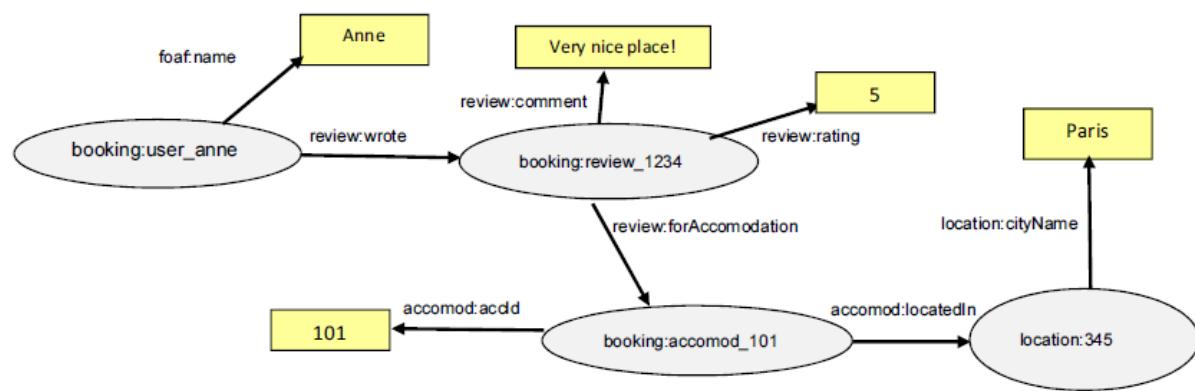
6. (5 bodova)

Napravite RDF model podataka iz 4. zadatka. Nad svojim modelom napišite SPARQL upit koji vraća sve osobe (samo jednom) koje su ocijenile neki smještaj u Parizu s ocjenom 5.

```
{User: "Anne", City: "Paris", AccId:101, Review: "Very nice place!", Rating: 5}
{User: "Peter", City: "London", AccId:111, Review: "Flea bag", Rating: 2}
{User: "Đuro", City: "Paris", AccId:101, Review: "Super-super!", Rating: 5}
{User: "Zoe", City: "Paris", AccId:102, Review: "Seen better", Rating: 3}
"
```

RJEŠENJE

6. (5 bodova)



```
PREFIX foaf: <http://.../>
PREFIX booking: <http://.../>
PREFIX review: <http://.../>
PREFIX accomod: <http://.../>
PREFIX location: <http://.../>
SELECT DISTINCT ?userName
WHERE {
    ?user foaf:name ?userName .
    ?user review:wrote ?reviewUserAccom .
    ?reviewUserAccom review:forAccomodation ?accomodation .
    ?reviewUserAccom review:rating ?revRating.
    ?accomodation accomod:locatedIn ?location .
    ?location location:cityName ?cityName .
    FILTER(?cityName = "Paris")
    FILTER(?revRating = "5")
}
```

DODATNI ZADACI

JESENSKI ROK 2013-14 |ZADATAK 6|

ZI 2014-15 |ZADATAK 7|



Sretno!