U zadacima 1-3 pretpostavlja se korištenje PostgreSQL SUBP-a i baze podataka sa slike 1.

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 odabrati 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.

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 pretpostaviti da vam se nalazi u varijabli `znam`.

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/…).

4. (5 bodova)
Napisati MapReduce upit koji će vratiti listu smještaja u Parizu s njihovim prosječnim ocjenama (`AccId, AvgRating`).
Ulazni podatci 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.

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.

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.

Rješenja:

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-torke u potpunosti sadržan u periodu jedne referencirane n-torke 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-torke ili u kombiniranom periodu dvije ili više uzastopnih referenciranih n-torki

2.

```
LET znam = POINT(x, y); // zadana je
// Moze se naravno i drugacije rijesiti, 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
```

3.

1. algoritmi temeljeni na udaljenosti znakovnih nizova

- mjera udaljenosti između znakovnih nizova s1 i s2 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 jednako ili slično izgovaraju ali drugačije zapisuju
- Algoritam riječ predstavlja znakovnim nizom koji prezentira izgovor (zvučanje) riječi
- PostgreSQL: Soundex i Metaphone

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

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;
}
```

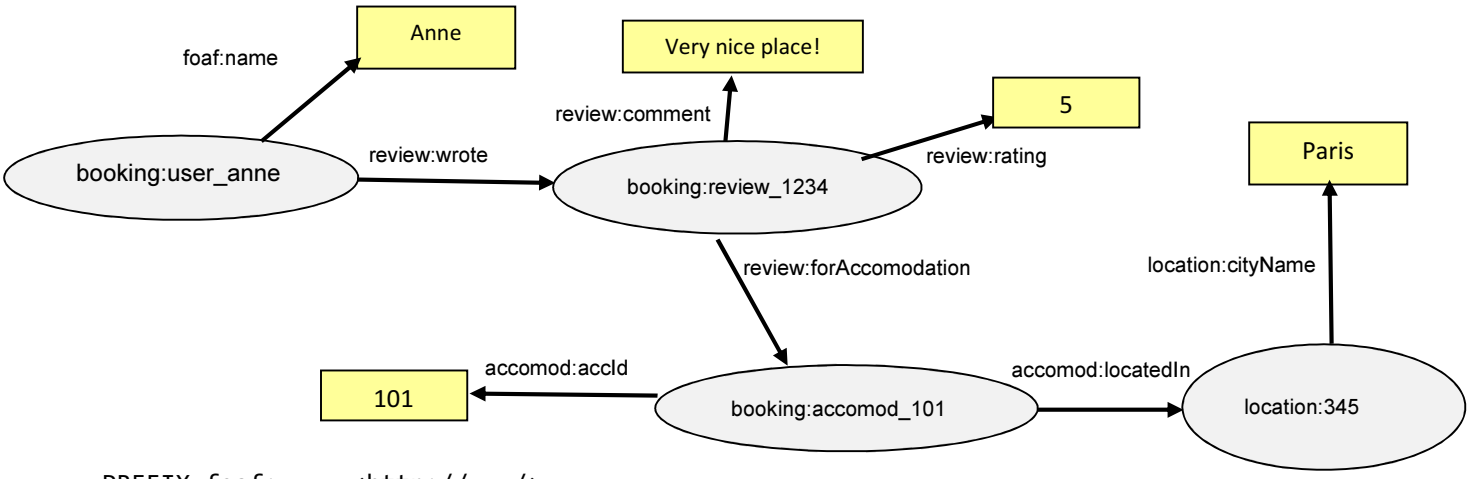
5. (5 bodova)

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.

Primjer u pred.

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")
 }
```