## Napredni modeli i baze podataka – završni ispit 23.01.2017.

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

Baza podataka je namijenjena pohrani podataka o letovima zrakoplovne tvrtka. Putnici temeljem ostvarenih letova sakupljaju nagradne milje (***putnikLet.nagMilje***) koje naknadno mogu koristiti za različite pogodnosti.

**osoba**

| ***osobaId*** | ***prezime*** | ***ime*** | ***…*** |
|---|---|---|---|
| 100 | Han | Ana | |
| 101 | Kos | Filip | |
| … | | … | |

**zrLuka**

| ***zrLukaId*** | ***nazivZrLuka*** | ***…*** | ***geom*** |
|---|---|---|---|
| 100 | Pleso | … | <point> |
| 101 | Charles de Gaulle | … | <point> |
| 102 | Heathrow | … | <point> |
| … | … | … | … |

**let**

| ***letId*** | **zrLukaOdId** | **zrLukaDoId** | **datVrijPolazak** | **…** |
|---|---|---|---|---|
| 400 | 100 | 150 | 2016-12-25 10:00:00.00 | |
| 450 | 100 | 250 | 2017-01-05 08:30:00.00 | |
| … | … | … | … | … |

**putnikLet**

| ***letId*** | **osobaId** | **…** | **nagMilje** |
|---|---|---|---|
| 400 | 100 | … | 1500 |
| 450 | 100 | … | 500 |
| … | … | … | … |

**slika 1**

1. **(5 bodova)** Korištenjem funkcionalnosti PostgreSQL SUBP napišite SQL naredbu kojom će se ispisati prezime i ime osobe, godina leta i
   - u toj godini ukupan broj prikupljenih nagradnih milja (**nagMiljeG**),
   - broj prikupljenih nagradnih milja u toj i u prethodnoj godini u kojoj je osvojio nagradne milje (**nagMiljeGiPrethG)**.
   - ukupan broj prikupljenih nagradnih milja bez obzira na datum leta (***nagMiljeUk***):

| ***prezime*** | ***ime*** | ***godina*** | ***nagMiljeG*** | ***nagMiljeGPretG*** | ***nagMiljeUk*** |
|---|---|---|---|---|---|
| Han | Ana | 2015 | 1500 | 1500 | 3000 |
| Han | Ana | 2016 | 500 | 2000 | 3000 |
| Han | Ana | 2017 | 1000 | 1500 | 3000 |
| … | … | … | … | … | … |

   **Pomoć:** EXTRACT (YEAR FROM datVrijemePolazak) će izdvojiti datum iz TIMESTAMP atributa datVrijemePolazak u PostgreSQL-u.

2.

a) **(1 bod)** Objasnite koje se n-torke u kontekstu transakcijskog vremena smatraju trenutnim sistemskim n-torkama, a koje povijesnim sistemskim n-torkama.

b) **(5 bodova)** Vlasnik sustava želi redundantno pohraniti informaciju o ukupnom broju nagradnih milja za svakog putnika. Također, želi pratiti ukupne nagradne milje putnika obzirom na vrijeme valjanosti i obzirom na transakcijsko vrijeme. Korištenjem funkcionalnosti PostgreSQL SUBP napišite SQL naredbe čijim bi izvođenjem u bazi podataka bili definirani svi objekti potrebni za ostvarenje opisanog zahtjeva**.** Obavezno implementirajte ključ nove relacije/relacija.
   Obzirom na sheme predloženih objekata/relacija**,** i uz pretpostavku da *ne* sadrže nijedan zapis**,** skicirajte njihov sadržaj nakon događaja s rednim brojem 1, a potom novi sadržaj nakon događaja s rednim brojem 2.

| ***rbrDogađaj*** | ***Što se dogodilo u stvarnom svijetu*** | ***Kada se dogodilo u stvarnom svijetu*** | ***Evidentirano u bazu podataka*** |
|---|---|---|---|
| 1 | Ana Han je osvojila 1000 nagradnih milja | ***01.05.2015*** | ***02.05.2015 08:00:00.00*** |
| 2 | Ana Han je osvojila dodatnih 500 nagradnih milja | ***01.07.2016*** | ***02.07.2016 10:00:00.00*** |

3. **(5 bodova**) Napisati funkciju koja će za zadano polazište (zračnu luku) i dolet aviona odrediti najdalje moguće odredište pri čemu je dozvoljeno jedno punjenje rezervoara (tj. slijetanje u usputnu zračnu luku). Npr. za Pleso (Zagreb) i 6000 km, funkcija bi mogla vratiti John F. Kennedy (New York) (preko Heathrow u Londonu). Možete smatrati da avion može sletjeti u bilo koju zračnu luku.
   Rješenja pišite u formi pseudokoda i SQL naredbi, te komentara, pri čemu **navedite GIS funkcije koje planirate koristiti**.

4. **(5 bodova)** Neka je ulaz u M/R relacija koja nastaje prirodnim spajanjem relacija let i putnikLet. Napišite M/R upit koji će za svakog putnika vratiti postotak nagradnih milja ostvarenih ove godine u odnosu na nagradne milje svih vremena. Pretpostavite istu arhitekturu (ponašanje) M/R-a kao kod MongoDB sustava.

5. **(3 boda)** Navedite replikacijske modele kod NoSQL sustava i komentirajte njihove prednosti i mane.

6. **(3 boda)** Što je *combinable reducer* i koje su mu prednosti i ograničenja. Objasnite na primjeru.

7. **(5 bodova)** Korištenjem OWL i Turtle sintakse napišite naredbe za definiranje ontologije kojom će biti modeliran mali segment iz svijeta filmske industrije. Ontologijom je potrebno definirati sve konstrukte (TBox) potrebne za opisivanje podataka o nazivu filma, njegovom distributeru i glumcima. Pri uvođenju klasa u ontologiju možete preuzeti klase Person, Actor i distributor iz <http://dbpedia.org/ontology/> i klasu Movie iz <http://www.schema.org/>. Predložena ontologija mora omogućiti npr.. evidenciju sljedećih izjava o filmu „Passengers“:
   - naziv filma ***<http://dbpedia.org/resource/Passengers>*** je „Passengers“.
   - distribuiran je od strane tvrtke ***<http://dbpedia.org/resource/Columbia_Pictures>***.
   - u filmu glumi glumica ***<http://dbpedia.org/resource/Jennifer_Lawrence>***
   - glumica ***<http://dbpedia.org/resource/Jennifer_Lawrence>*** se zove „Jennifer Lawrence“.

   Za ontologiju koju ste predložili napišite Turtle izraze (ABox) kojima ćete opisati gornje izjave vezane uz film, glumicu i distributera filma Passengers.
   **Napomena:** Prefikse svih prostora imena koji su navedeni u službenom podsjetniku možete koristiti i bez prepisivanja definicija prefiksa.

8. **(3 boda)** Korištenjem SUTP pratimo transakcije između korisnika na burzi Bitstamp, jednoj od najpoznatijih svjetskih burzi za trgovanje kriptovalutom Bitcoin u zamjenu za američki dolar. Nakon svake obavljene transakcije na ulaz SUTP-a pristiže n-torka koja opisuje obavljenu transakciju između neka 2 korisnika burze, opisana sljedećom shemom: <*vremenska_oznaka, id_prodavaca, id_kupca, kolicina_bitcoina*>. U okviru SUTP pokrenut je kontinuirani upit koji na izlaz svake minute šalje ukupno stanje računa (količinu Bitcoina) za sve korisnike burze.
   Navedite raspoložive modele tokova podataka i objasnite koji model toka podataka i zašto je adekvatan za realizaciju opisanog scenarija.

**Rješenja:**

**1.**
```
SELECT osoba.prezime, osoba.ime, EXTRACT (YEAR FROM datVrijPolazak),
       SUM(nagMilje),
       SUM(SUM(nagMilje)) OVER (PARTITION BY osoba.osobaId
                                ORDER BY EXTRACT (YEAR FROM datVrijPolazak)
                                ROWS BETWEEN 1 PRECEDING AND CURRENT ROW),
       SUM(SUM(nagMilje)) OVER (PARTITION BY osoba.osobaId)
  FROM osoba
  JOIN putnikLet ON putnikLet.putnikId = osoba.osobaId
  JOIN let ON putnikLet.letId = let.letId
  GROUP BY osoba.osobaId, osoba.prezime, osoba.ime, EXTRACT (YEAR FROM datVrijPolazak)
```

**Krivo, ujedno i najčešće studentsko, rješenje:**
```
SELECT osoba.prezime, osoba.ime, EXTRACT (YEAR FROM datVrijPolazak),
       SUM(nagMilje) OVER (PARTITION BY osoba.osobaId,
                           EXTRACT (YEAR FROM datVrijPolazak),
       SUM(nagMilje) OVER (PARTITION BY osoba.osobaId
                           ORDER BY EXTRACT (YEAR FROM datVrijPolazak)
                           ROWS BETWEEN 1 PRECEDING AND CURRENT ROW),
       SUM(nagMilje) OVER (PARTITION BY osoba.osobaId)
  FROM osoba
  JOIN putnikLet ON putnikLet.putnikId = osoba.osobaId
  JOIN let ON putnikLet.letId = let.letId
  GROUP BY osoba.osobaId, osoba.prezime, osoba.ime, EXTRACT (YEAR FROM datVrijPolazak)
```
Ovaj upit završi sljedećom pogreškom:
```
ERROR:  column "putniklet.nagmilje" must appear in the GROUP BY clause or be used in an
aggregate function
LINE 2:        SUM(nagMilje) OVER (PARTITION BY osoba.osobaId, EXTRA...
```

Budući da upit sadrži GROUP BY dio koji ne sadrži atribut ***nagMilje***, (međurezultat se evaluira prije računanja vrijednosti za particije), neće se moći računati SUm(nagMilje) niti u jednom od gornja tri izraza jer međurezultat ne sadrži ***nagMilje***.
Loše je i rješenje u kojem bi se u gornjem upitu izostavio GROUP BY jer se tada na izlazu pojavi onoliko n-torki koliko ih je u ***putnikLet***, umjesto jedna po osobi i godini.

**2.**

a) n-torke čiji transakcijski period ima presjek s trenutkom "sada" se nazivaju trenutnim sistemskim n-torkama (current system rows), sve ostale se nazivaju povijesnim sistemskim n-torkama (historical system rows).

b) Potrebno je „stvoriti“ bitemporalnu relaciju, a za to su moguća 2 rješenja:

1. *Kreirati 2 nove relacije:*

   ***putnikNagMilje*** u kojoj se uvijek nalaze samo trenutne sistemske n-torke i
   ***putnikNagMiljeHistory*** u kojoj se nalaze povijesne sistemske n-torke.
   Obje relacije su jednake sheme osim što za ***putnikNagMiljeHistory*** nije važan (ne mora biti definiran) ključ.
   Relacije sadrže 2 range atributa:
   ***periodValjanosti*** DATERange za praćenje ukupnih nagradnih milja u kontekstu vremena valjanosti
   i
   ***periodTransakcijski*** TSTZange za praćenje ukupnih nagradnih milja u kontekstu transakcijskog vremena
   Tekst obojan zelenom bojom nije potrebno pisati – tu se nalazi zbog boljeg razumijevanja rješenja:
```
CREATE EXTENSION bTree_gist;
CREATE TABLE putnikNagMilje
(putnikId            INT REFERENCES osoba(osobaId),
 nagMiljeUk          INT NOT NULL,
 periodValjanosti    DATERANGE NOT NULL,
```

```
 periodTransakcijski TSTZRange NOT NULL DEFAULT tstzrange(current_timestamp, null),
 PRIMARY KEY (putnikId, periodValjanosti),
 CONSTRAINT pkPutnikNagMilje EXCLUDE USING gist (putnikId WITH =, periodValjanosti WITH &&)
--ili PRIMARY KEY (putnikId, periodTransakcijski),
--CONSTRAINT pkPutnikNagMilje EXCLUDE USING gist (putnikId WITH =, periodTransakcijski WITH
&&)
 );
CREATE TABLE putnikNagMiljeHistory (LIKE putnikNagMilje);
CREATE TRIGGER versioning_trigger
BEFORE INSERT OR UPDATE OR DELETE ON osoba
FOR EACH ROW EXECUTE PROCEDURE versioning(
  'periodTransakcijski', ' putnikNagMiljeHistory', true
);
```

2. Proširiti postojeću tablicu ***osoba*** s 3 atributa:

```
nagMiljeUk          INT,          - za ukupne nagradne milje osobe
periodValjanosti    DATERANGE     - za period valjanosti
periodTransakcijski TSTZRange     – za transakcijski period.
```

*Ključ osoba proširiti tako da pored sifOsoba uključuje i periodTransakcijski.*

Kreirati tablicu osobaHistory tablicu jednake sheme kao izmijenjena osoba ali ključ nije potrebno definirati.

Sadržaj novih tablica je dan za rješenje opisano pod 1.

Nakon 1. događaja:

***putnikNagMilje***

| *putnikId* | *periodValjanosti* | *nagMiljeUk* | *periodTransakcijski* |
|---|---|---|---|
| 100 | [01.05.2015,) | 1000 | *[2015-05-02 08:00:00.00, )* |

***putnikNagMiljeHistory***

| *putnikId* | *periodValjanosti* | *nagMiljeUk* | *periodTransakcijski* |
|---|---|---|---|

Nakon 2. događaja:

***putnikNagMilje***

| *putnikId* | *periodValjanosti* | *nagMiljeUk* | *periodTransakcijski* |
|---|---|---|---|
| 100 | [01.05.2015, 01.07.2016) | 1000 | *[2016-07-02 10:00:00.00,)* |
| 100 | [01.07.2016,) | 1500 | *[2016-07-02 10:00:00.00,)* |

***putnikNagMiljeHistory***

| *putnikId* | *periodValjanosti* | *nagMiljeUk* | *periodTransakcijski* |
|---|---|---|---|
| 100 | [01.05.2015,) | 1000 | *[2015-05-02 08:00:00.00,2016-07-02 10:00:00.00)* |

3.

```
-- s jednim upitom:
func getNajdalje(idPolaziste, dolet) {

          SELECT zrLukaId
              FROM zrLuka pol, zrLuka srednja, zrLuka odrediste
              WHERE pol.zrLukaId = idPolaziste
                AND st_distance(pol.geom, srednja.geom) <= dolet
                AND st_distance(srednja.geom, odrediste.geom) <= dolet
   -- mogu se dodati i uvjeti da se izbjegne spajanje luke same sa sobom, ali nije nužno
          ORDER BY st_distance (pol.geom, odrediste.geom) DESC
          LIMIIT 1
}
 -- s dva upita, u dva koraka:

func getNajdalje(idPolaziste, dolet) {
```

```
            SELECT zrLukaId
               INTO TEMP prviKrug
                FROM zrLuka
                WHERE st_distance(geom,
                          (SELECT geom FROM zrLuka WHERE zrLukaId = idPolaziste))< dolet

            SELECT distinct zrLuka.zrLukaId, zrLuka.geom
                   INTO TEMP drugiKrug
                 FROM zrLuka JOIN prviKrug
                WHERE st_distance(zrLuka.geom, prviKrug.geom)< dolet

            SELECT zrLukaId
             FROM drugiKrug
            ORDER BY st_distance (geom,
                          (SELECT geom FROM zrLuka WHERE zrLukaId = idPolaziste) ) DESC
            LIMIIT 1
        }
```

4.

```
        map (k, v) {

           emit(v.osobaId, {
                      ovaGod: ((year(v.datVrijPolazak) == year(today)) ? v.nagMilje : 0),
                      svihVremena: v.nagMilje
            });

        }
        reduce (k, vlist) {
           var agg = {ovaGod: 0, svihVremena: 0};
           foreach (v in vlist) {
              agg.ovaGod += v.ovaGod;
              agg.svihVremena += v.svihVremena;
           }
           return agg;
        }

        finalize (k, vlist) {
           return vlist.first().ovaGod / vlist.first().svihVremena; // * 100, može i ne mora;
        }
```

5. vidi predavanja
6. vidi predavanja
7.

Tekst obojan zelenom bojom nije potrebno pisati – tu se nalazi zbog kompletnosti rješenja:

```
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbr:    <http://dbpedia.org/resource/> .
@prefix dbo:    <http://dbpedia.org/ontology/> .
@prefix schema: <http://www.schema.org/> .
@base           <http://exampple.org/myontology.owl/> .

<http://exampple.org/myontology.owl/> rdf:type owl:Ontology ;
                                      owl:imports dbo: .
#    Classes
dbo:Person rdf:type owl:Class .
dbo:Actor rdf:type owl:Class ;
          rdfs:subClassOf dbo:Person .
dbo:distributor rdf:type owl:Class .
```

```
schema:Movie rdf:type owl:Class .
#    Object Properties
:isDistributedBy rdf:type owl:ObjectProperty ;
                 rdfs:domain schema:Movie ;
                 rdfs:range dbo:distributor .

:isActorIn       rdf:type owl:ObjectProperty ;
                 rdfs:domain dbo:Actor ;
                 rdfs:range schema:Movie .

:actorName rdf:type owl:DatatypeProperty ;
            rdfs:domain dbo:Actor ;
            rdfs:range xsd:string .

:movieName  rdf:type owl:DatatypeProprty ;
            rdfs:domain dbo:Movie ;
            rdfs:range xsd:string .
#    Individuals
dbr:Columbia_Pictures rdf:type dbo:distributor .
dbr:Passengers rdf:type schema:Movie ;
               :movieName "Passengers"^^xsd:string;
               :isDistributedBy dbr:Columbia_Pictures.
dbr:Jennifer_Lawrence rdf:type dbo:Actor ;
                :actorName "Jennifer Lawrence"^^xsd:string;
                :isActorIn dbr:Passenger.
```

8.

Opisani scenarij je primjer *Turnstile modela* toka podataka jer različiti elementi ulaznog toka mogu povećati ili smanjiti ukupnu vrijednost za istog korisnika (prema formalnoj definiciji, ulazni elementi $a_i$ su ažuriranja elemenata A[j])