Drugi kolokvijum iz Upravljanja Informacijama
Grupa 1

Zadatak 1:

Pokrenuti Lucene alat (Lucene folder) pomoću Eclipse-a. Indeksirati dokumente koji se nalaze u “data” direktorijumu, i zatim izvrsiti pretrage koje vracaju:

a) Sve dokumente koji sadrže reč “president” ali ne i reč “nelson”
b) Sve dokumente koji imaju reči koje u sebi sadrže reč “preisdent” (ako dokument sadrži reč presidents, na primer, treba da bude prikazan)
c) Sve dokumente koji sadrže frazu “president of united states”, ali ako se nađu još 3 (ili manje) reči unutar fraze (npr. “President of the new united states), dokument takođe treba prikazati.
d) Promeniti IndexFiles.java klasu tako da se kreiraju 2 indeksa, jedan sa naslovom članka (prva linija svakog .txt fajla), a drugi sa sadržajem članka (ostatak teksta u fajlu). Zatim napraviti upit koji vraća sve dokumente koji sadrže reč “dead” i u naslovu i u sadržaju članka.

Same upite možete napisati u tekstualnom dokumentu resenja.txt, i dokument sacuvati u Lucene folderu

Zadatak 2:

Podatke za zadatke (kao i mesto gde treba da sačuvate vaše .py skripte) imate u folderu MapReduce

a) Lak zadatak:

U “data” fajlu se nalaze informacije o različitim hranama u vidu csv fajla (nayiv hrane u prvoj koloni i grupa kojoj ta hrana pripada u trećoj koloni). Sumirajte koliko stavki pripada svakoj od navedenih grupa hrana.

b) Srednji zadatak:

U “data” fajlu se nalaze informacije o različitim “ukusima” i grupi kojoj oni pripadaju. Zadatak je da na kraju za svaku grupu ispišete broj ukusa koji joj pripada, kao i ukus sa najdužim i najkraćim nazivom (ako ih ima više sa istim nazivom napišite bilo koji). Ako se u datoteci “data” nalaze neispravni podaci, drugim rečima ako jedna od dve kolone ne postoji ili ima NULL vrednost, takvi podaci treba da se preskoče.

c) Težak zadatak:

U “data” fajlu se nalaze informacije o transakcijama u CSV formatu. Prva kolona označava naziv osobe koja dobija novac, druga kolona naziv osobe koja plaća, i treća kolona sadrži količinu novca za tu transakciju. Zadatak map reduce algoritma je da sve transakcije između dve osobe sumirau jednu stavku. Za primer koji ste dobili, između ostalog, na izlazu treba da se pojavi informacija da osoba F treba da dobije od osobe A 200 dinara. Na izlazu ne sme da se pojave negativni brojeci (osoba A treba da dobije od osobe F
-200 dinara), i ako je na kraju sumiranja svih transakcija dug između dve osobe 0, ta transakcija ne treba da se pojavi na krajnjem ispisu.

Zadatak 3:

U folderu KafkaStreaming se nalaze folderi sa spremljenim producerima za lak, srednji i težak zadatak. U tim istim folderima treba da ostavite i vaše .py skripte koje će dalje obrađivati poruke. Pre nego što budete pokrenuli yadatak treba da pokrenete kafka_dummy.py skriptu i kreirate potrebne nizove za poruke (message queues)

a) **Lak zadatak:**
   Napraviti consumer-a koji će u realnom vremenu brojati:
   - Ukupnu količinu majica za svaku od veličina (XL, L, M, S)
   - Ukupnu količinu majica za svaki od brendova (Adidas, Nike, Polo, Uniqlo)

   Te agregirane podatke ne treba nigde da šalje samo ih ispišite na standardni output svaki put kada budu updateovani.

b) **Srednji zadatak:**
   Napraviti Faust filter koji će da u realnom vremenu da nadgleda sve porudžbine (poruke) i:
   - Ako je veličina majice XXL (koju prodavnica ne prodaje više), porudžbina treba da se preusmeri na consumer_non_fulfilled_orders servis
   - Ako je bilo koja druga veličina porudžbina ide do consumer_shipping_shirts servisa

c) **Težak zadatak:**
   Servis 'Zahtev za letove' periodično na *topic* sa nazivom "zahtevi_za_letove" šalje poruku za zauzimanje jedne od pista. Zahtev se šalje u vidu JSON-a sa sledećim poljima:

   {
   id_leta: string,
   broj_piste: integer
   }

   Može da zauzme let tek kada svi servisi koji nadgledaju pistu odobre zahtev. Servis koji šalje zahteve za zauzimanje piste očekuje u nekom trenutku odgovor da li je let prihvaćen ili odbijen u formatu:

   {
   id_leta: string,
   let_odobren:
   boolean
   }

   Svaki od 3 servisa za nadgledanje piste (kontrolni toranj, održavanje piste, i uprava aerodroma) primaju zahteve za let i nakon odredjene pauze (izmedju 1 i 5 sekundi) vraćaju svoj odgovor u formatu:

   {
   id_leta: string,
   let_odobren:
   boolean
   }

   na topic sa nazivom "odgovori_na_zahteve".

   Treba da napišete skriptu koja:

   1) akumulira odgovore koje salju svaki od 3 servisa za nadgledanje piste, i nakon toga pošalje konačni odgovor servisu za zahtev letova. Zahtev je odobren jedino ako ga sva tri servisa za nadgledanje piste odobre.

2) Sve dok traje odobravanje leta za pistu sa odredjenim rednim brojem (koji se može posmatrati i kao id piste), faust aplikacija treba automatski da odbije svaki zahtev koji traži pistu sa istim rednim brojem.

NAPOMENA: Servisi koje ste dobili (cetiri .py skripte) koriste 2 kafka topic-a ("zahtevi_za_letove" i "odgovori_na_zahteve"). Ako je potrebno za resenje zadatka, možete kreirati nove topic-e, ili brisati postojece.