Prirodoslovno-matematički fakultet u Splitu

Matematika: Računarski smjer

Raspodijeljene i nerelacijske baze podataka

**Đana Krstić**

OrientDB i primjer baze podatka

Dokumentacija projekta

Split, 6. 2023.

Sadržaj

1. **Uvod…………………………………………………………………………………..1**
2. **Razvojna tehnologija – OrientDB…………………………………………………..2**
   1. Osnovne informacije…………………………………………………………2
   2. Razlika graf modela i relacijske BP……………………………………….. 3
   3. Prikaz OrientDB-a…………………………………………………………..3
   4. Record, klase i cluster……………………………………………………….6
3. Izrada projekta i rad u OrientDB-u……………………………………………….8
   1. Ogledni opis baze……………………………………………………………..8
   2. Stvaranje i struktura klasa – vrhovi i rubovi……………………………….8
      1. Vrhovi………………………………………………………………8
      2. Rubovi……………………………………………………………..12
      3. Indexi i dodatne značajke………………………………………..15
   3. Unos podataka………………………………………………………………16
   4. Functions Managment + generiranje podataka…………………………..18
4. Upiti na bazi „intertests“ i grafovi………………………………………………...20
   1. Graph Editor………………………………………………………………..21
   2. Dohvaćanje podataka iz baze………………………………………………23
      1. Rad s vrhovima…………………………………………………………..24
      2. Rad s rubovima…………………………………………………..26
      3. Primjeri upita s grafovima………………………………………30
5. Zaključak…………………………………………………………………………..39
6. **Literatura…………………………………………………………………………..40**
7. Uvod

Tema projekta je izrada nerelacijske baze podataka tipa u open source bazi OrientDB. Govorit ćemo o samom OrientDB-u, njegovom načinu rada kroz postupak izrade baze i o njegovom vezi sa SQL-om. Dodatno, obradit ćemo bazu *interests* u koju možemo spremati podatke o korisnicima i njihovim interesima (knjige, filmovi, igre), te njihove odnose s drugim korisnicima.

1. Razvojna tehnologija – OrientDB
   1. Osnovne informacije

OrientDB je open source NoSQL baza podataka koja je napisana u Javi. Podržava različite tipove nerelacijskih baza podataka – baze temeljene na grafovima, parovima ključ/vrijednost i dokumentima tj. OrientDB je multi-model baza. Glavna karakteristika OrientDB-a je mogućnost grafičkog prikaza odnosa bazi neovisno o odabranom tipu.

OrientDB se najčešće koristi kod bankarstva, društvenih mreža, sustava preporuke, prometa, prilikom baratanja s Big Data itd. Prednost OrientDB-a u odnosu na druge baze podataka je taj da kombinira tip baze dokument i graf i na taj način ostvaruje veću pristupačnost podacima, bolji i brži prikaz.

OrientDB dopušta schema-less, schema-full i schema-mixed način rada tj. dopušta nam da unaprijed definiramo svojstva s kojim radimo, ali i ne moramo.

OrientDB radi s različitim programskim jezicima, a dopušta pisanje query-ja sa Gremlinom kao i sa SQL sinatksom. Gremlin je virtualna mašina namijenjena sa za baze podataka tipa graf. Gremlin je nerlacijskim bazama tipa graf isto što je i SQL relacijskim.

OrientDB dozvoljava nasljeđivanje i polimorfizam, ima svoj development studio i ima vlastite ugrađene funkcije.

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Rank OrientDB-a kao graf tipa

OrientDB se nalazi u top 10 najkorištenijih baza tipa s modelom graf. 2023. se nalazi na 15. mjestu kao model document.

|  |  |
| --- | --- |
| PROS | CONS |
| Multi-model | Relativno nepoznata |
| Besplatni clustering support | Treba „naučiti“ kako funkcionira |
| Brza – pogotovo s linkovima | Bulk-ovi mogu stvarati greške |

* 1. Razlika graf modela i relacijske BP

Glavna razlika graf nerelacijskih baza podataka koje koristi OrientDB i relacijskih baza podataka je ta da relacijske baze podataka za povezivanje različitih tablica koriste strani ključ i operaciju JOIN. Na taj način se utvrđuje njihov odnos.

Kod graf nerelacijskih baza podataka, tablice su predstavljene vrhovima (eng. vertex), a veze među njima se ostvaruju stvaranjem rubova (eng. edges). Na ovaj način, moguće je intuitivnije stvoriti baze podatke.

Zbog lakog načina stvaranja veza među podacima, graf BP su idealne za rad s hijerarhijskim podacima jer se s njima lakše određuju pojedine veze.

Još jedna razlika (prednost) graf tipa BP je taj što ne trebamo kreirati međutablicu kod veze „više-prema-više“ već rubove možemo stvarati proizvoljno.

* 1. Prikaz OrientDB

Slika na kojoj se prikazuje web-mjesto

Opis je automatski generiran

Slika - Open Studio Web Tool

Nakon instalacije OrientDB-a, pokrećemo server klikom na server.sh ili server.bat (shell script (for unix shell prompt) ili batch file (windows commands) .

OrientDB dopušta rad u konzoli, ali ima i Open Studio Web Tool za olakšani rad. Web tool-u pristupamo tako da u browser upišemo „localhost:2480“.

Slika na kojoj se prikazuje snimka zaslona, tekst, crno

Opis je automatski generiran

Slika - OrientDB server pokrenut u konzoli

Nova baza se stvara klikom na „NEW DB“ ili možemo odabrati jednu od postojećih baza podataka koje dolaze s instalacijom OrientDB-a (npr. demodb) ili koje smo već prije stvorili.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika – Primjer kreiranja nove baze

Nakon stvaranja ili otvaranja baze ulazimo u karticu **„BROWSE“** koja sadrži query prozor za unos naredbi – tu se koristi SQL API.

Slika na kojoj se prikazuje tekst, softver, snimka zaslona, Multimedijski softver

Opis je automatski generiran

Slika - Query prozor i različite opcije

Ostale dostupne kartice su: „SCHEMA“, „SECURITY“, „GRAPH“, „FUNCTIONS“ i „DB“.

**„SCHEMA“** sadrži korisničke klase (vrhove i rubovi) i sistemske klase.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika – Shema

Postoje 3 glavne skupine klasa: klase čvorova (**Vertex** Classes), klase rubova (**Edge** Classes) i generičke klase (Generic Classes). Više o klasama u nadolazećim sekcijama.

**„SECURITY“** dopušta kontrolu pristupa – dodavanje novih korisnika baze i određivanje njihove uloge pri radu sa samom bazom

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, broj

Opis je automatski generiran

Slika - Security Manager

U bazu možemo dodavati više korisnika, te im dati ulogu – admin, writer, reader. Možemo i specificirati restrikcije.

**„GRAPH“** nam dopušta vizualni prikaz odnosa klasa BP – objasnit ćemo detaljnije u nadolazećim sekcijama.

**„FUNCTIONS“** nam dopušta pisanje i rad sa funkcijama – objasnit ćemo detaljnije u nadolazećim sekcijama.

**„DB“** nam dopušta pregled same baze i clustera, te mogućnost export-a baze. Zapisana su imena svih clustera, te koliko je podataka pohranjeno u pojedini cluster.

Nakon što se baza exporta, sprema se kao .gz file (tj. gzip) – baza podatka se zapisuje u JSON formatu i može se ponovno importati u OrientDB s naredbom **IMPORT DATABASE** **<putanja>** .

* 1. Record, klase i cluster

OrientDB koristi *Record* kao pohranu. **Record** je najmanja jedinica koja se može spremiti u bazu podataka. Postoje 4 tipa Record-a : dokumenti (najfleksibilniji), blobs (binarni zapisi), vrhovi i rubovi.

**Klase** su koncept uzet iz objektno-orijentiranog programiranja i sastoje se od svojstava koja se koriste za opisivanje nekih koncepata. U relacijskim bazama podataka, klase su bile „tablice“. Kod nerelacijskih baza tipa graf, klase su čvorovi i rubovi i oni imaju svoja svojstva.

Klase podržavaju nasljeđivanje i sadrži clustere. **Cluster** je fizička komponenta OrientDB-a u koju se pohranjuju stvarni podaci. Clusteri bi se mogli usporediti s tablicama u relacijskim bazama podataka. Prilikom kreiranja klase, automatski se stvaraju i clusteri. Jedna klasa može imati i više clustera.

Postoje 2 vrste cluster-a: fizički i in-memory.

Fizički su uporni (eng. persistent) tj. funkcionalni su i u kriznim situacijama kao prilikom nestanka struje.

In-memory su nepostojani (eng. volatile). Vodi se po principu „velike promjene prate velike promjene, a male male“.

Clustere u OrientDB-u možemo stvoriti s upitom „CREATE CLUSTER“ nakon ćega specificiramo naziv, tip, lokaciju i druga svojstva. Clustere možemo stvoriti i prilikom stvaranja klasa (jer podatke vezane za klase moramo spremiti negdje).

Klasu naknadno dodajemo clusteru sa naredbom :

**ALTER CLASS <ime\_klase> ADDCLUSTER <ime\_clustera>.**

Record-e u cluster možemo rasporediti na više načina. Prilikom kreiranja clustera koristi se *default* svojstvo – cluster se bira koristeći id.

Možemo promijeniti način dodavanja recorda u klasu s naredbom:

**ALTER CLASS <ime\_klase> CLUSTERSELECTION <način\_dodavanja>.**

Osim default, način dodavanja može biti i *round.robin* (organizira konfigurirane clustere za klasu u niz i svakom novom pridoda sljedeći cluster u nizu), *balanced* (jednoliko raspoređeno) i *local*(bira master cluster). Moguće je u OrientDB-u stvoriti i svoj način dodavanja koristeći Java API.



Slika - Clusteri ruba Plays

Na (Slici 8 - gore) vidimo primjer clustera (dobijemo ulaskom u prozor „DB“). Prvi stupac predstavlja ID clustera koji se daje automatski. Drugi stupac sadrži imena clustera, a treći koliko je recorda pohranjeno u njemu.

Važno je još spomenuti i **@rid** – jedinstveni id pridružen svakom recordu. @rid ima svoju strukturu.

Npr. @rid=34:45 je id takav da broj prije „:“ označava id clustera u kojem se record nalazi, a broj poslije njegov redni broj unutar clustera.

Clusteri i klase su vrlo korisni kad trebamo grupirati neke veće količine podataka. Npr. ako imamo klasu izvještaji, možemo stvoriti klastere izvještaj\_2021 i izvještaj\_2022 koji će grupirati izvještaje ovisno o godini.

U OrientDB-u imamo i apstraktne klase. Apstraktne klase nemaju instanci i ne moraju imati clustere (za razliku kod običnih klasa koje moraju imati bar jedan).

1. Izrada projekta i rad u OrientDB-u

U ovom odjeljku ćemo stvoriti bazu *„interests“* i pomoću nje opisati glavne značajke OrientDB-a.

* 1. Ogledni opis baze

Baza podataka nosi naziv *„interests“* i može se gledati kao simplicirana društvena mreža koja se temelji se na interesima korisnika.

Korisnici i njihovi osnovni podaci se pohranjuju u bazu kao vrh *Users*, a u bazu se spremaju i svi flmovi/serije (*Movies*) koje je korisnik gledao, sve igre (*Games*) koje je igrao i sve knjige (*Books*) koje je čitao.

Moguće je i ostvariti vezu s drugim korisnicima tj. može se označiti s kojim drugim korisnicima je prijatelj, obitelj ili u vezi. Također postoji opcija blokirati druge korisnike.

Više o svojstvima pojedinih klasa u nastavku.

* 1. Stvaranje i struktura klasa – vrhovi i rubovi
     1. Vrhovi

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Vrhovi klase "interests"

U bazi *„interests“* vrhovi su klase: Users, Books, Games i Movies.

Na listi klasa se nalazi i zadana klasa V koja pretstavlja klasu „Vertex“ na kojoj gradimo sve vrhove tj. iz nje sve klase koje označimo kao vrh klase nasljeđuju svojstva.

Postoje različiti načini za stvoriti klasu u OrientDB-u. Možemo iskoristiti SQL API i upisati naredbu.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika - Kreiranje klase User s SQL upitom i njezino izvršenje

„*extends V*“ nam govori da je klasa Users proširenje klase Vertex.

Vrh možemo kreirati i u „SCHEMA“ klikom na „new Vertex“.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika - Prozor za kreiranje vrha u "SCHEMA"

Klasama pridružujemo i određena svojstva. Ponovno možemo birati želimo li to učiniti SQL naredbom ili u korisničkom sučelju.

Primjerice, naredba za stvoriti svojstvo *username* u klasi User (tipa string) bi bila:

**create property User.username string** .

U sučelju je dovoljno kliknuti na klasu i potom dodati „new property“.

Osim tipa svojstva, možemo odabrati i je li obavezno, read\_only, koja je minimalna i maksimalna vrijednost koju smije poprimiti, hoće li svojstvo biti case sensitive, te je li klasa vezana za neku drugu.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika - Primjer kreiranja svojstva

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, račun

Opis je automatski generiran

Slika - Svojstva klase Users

Obavezna svojstva klase Users su: *username*, *user\_id*, *membership* i *date\_of\_account\_creation*.

Uočimo da imamo podatke tipa „**LINKEDLIST**“. Podaci tipa „LINKEDLIST“ predstavljaju listu „linkova“ tj. veza s drugim klasama. Tako npr. svojstvo *friend\_with* sadrži listu podatka @rid koji predstavljaju jedan objekt tj. u ovom slučaju korisnika. Način na koji smo pristupili ovim podacima je sa klasama rubova koje ćemo definirati kasnije.

*user\_id* je postavljen na read\_only kao što ćemo napraviti i za id integere i u drugim klasama. *age* uzimamo između 1 i 200.

Slika na kojoj se prikazuje tekst, broj, snimka zaslona, račun

Opis je automatski generiran

Slika - Svojstva klase Books

Obavezna svojstva klase Books su *bookID* i *title*. *Average\_rating* mora biti double između 1 i 5.

Slika na kojoj se prikazuje tekst, broj, snimka zaslona, Font

Opis je automatski generiran

Slika - Svojstva klase Games

Klasi Games je obavezno svojsvo *name*, a *meta\_Score* mora biti između 1 i 100.

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Svojstva klase Movies

Klasa Movies, kao i prethodne klase ima obavezan *show\_id* i *title*. Svojstvo *rating* mora biti između 1 i 10.

Podsjetimo se da OrientDB dopušta schema-less, schema-full i schema-mixed.

Ovo znači da se svojstva uopće nisu trebala definirati (učinili radi primjera i stvaranja nekih granica). Prilikom dodavanja podatka, svojstva se mogu izostaviti ili dodati po želji. Shema-less pristup je svojstvo dokument BP.

* + 1. Rubovi

Promotrimo sada rubove baze *„interests“*.

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Rubovi baze "interests"

Baza sadrži rubove: Banned, Plays, Watches, Reads, isFamilywith, isFriendsWith i inRelationshipWith.

Klase rubova, baš kao i vrhova, možemo stvoriti sa SQL upitom ili u sučelju.

Iskoristimo i koncept apstraktne klase. Apstraktne klase su takve da se u njih ne mogu unositi vrijednosti nego one služe kao nadklase drugim klasama – nadklasa čija svojstva nasljeđujemo.

Za primjer načinimo apstraktnu klasu Relationship koja je potomak klase E (klase **Edge**).

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, račun

Opis je automatski generiran

Slika - Stvaranje apstraktne klase ruba Relationship

Klasu smo definirali tako da ona ide iz vrha User u vrh User i ona će predstavljati bilo kakav odnos među korisnicima.

Ovoj klasi dodajemo i svojstvo *how\_long* koje će označavati koliko je dugo trajao odnos u pitanju među korisnicima.

Ova svojstva su naslijedile i klase inRelationshipWith (koja ima i bool svojstvo *married* koje bilježi je li korisnik u braku), isFriendsWith i isFamilywith (koja još bilježi i *relation* tj. o kakvom je obiteljskom odnosu riječ).

Promotrimo ostale klase rubova.

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Svojstva ruba Banned

Rub Banned ima svojstvo *reason*, ali i svojstva *in* and *out* tipa „LINK“. *in* i *out* su svojstva zajednička svim klasama tipa rub i ona se stvore prilikom stvaranja klase. Ona nam govore iz kojeg tipa čvora idemo u koji.

Slika na kojoj se prikazuje tekst, broj, Font, snimka zaslona

Opis je automatski generiran

Slika - Svojstva ruba Plays

Slika na kojoj se prikazuje tekst, broj, snimka zaslona, Font

Opis je automatski generiran

Slika - Svojstva ruba Reads

Slika na kojoj se prikazuje tekst, broj, snimka zaslona, Font

Opis je automatski generiran

Slika - Svojstva ruba Watches

Svi rubovi imaju svojstvo *rating.* Rub Plays ima i svojstva *difficulty* i *hours\_played*, a rub Reads *times\_read* i *favorite\_character*. Ni jedno svojstvo nije obavezno.

Ovakvi rubovi se smatraju regularnima, ali postoje i tzv. Light Weight Edges. Za razliku od regularnih, oni se ostvaruju kao cijela klasa, već se sprema samo veza među vrhovima. Ovaj postupak je sličan pojmu stranog ključa kod relacijskih baza podataka i može biti koristan jer zauzima manje prostora nego regularni rub. Ali, za razliku od regularnog ruba, Light Wight Edge ne može imati svojstva, pa se ne koristi tako često.

* + 1. Indexi i dodatne značajke

Nad nekim svojstvima moramo postaviti dodatna ograničenja unosa koja ne možemo označiti kvačicom u sučelju.

Npr. želimo da nam sva svojstva oblika *id* budu jedinstvena. To možemo ostvariti dodavanjem indexa.



Slika - Indexi vrha Users

Vidimo da u klasi Users imamo stvorena 2 indexa Users.user i Users.user\_id.

Ova dva indexa ostvaruju jedinstvenost svojstava *user\_id* i *username*. Tako, ako npr. korisnik želi unijeti username koji već postoji javit će mu grešku.

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Opis je automatski generiran

Slika - Stvaranje indexa

Index možemo stvoriti ključnom riječi „CREATE INDEKS“ ili klikom na „Add new Index“ koji se nalazi zajedno sa svojstvima.

**Indeks** služi za brzo pretraživanje u BP, a možemo mu dati ime, na koje svojstvo se odnosi i koji je tip. Tip može biti unique, notunique, dictionary, fulltext, unique\_hash\_index…

Primijetimo da kada stvorimo index imamo kategoriju „Engine“ u kojoj piše „CELL\_BTREE“. Ovo se odnosi na algoritam indexiranja.

OrientDB podržava 5 algoritama indexiranja: SB Tree, Hash, Auto Sharding, Lucene Full Text i Lucene Spatial index.

Svaki od ovih algoritama je određen tipom indexa i nakon što jednom postavimo tip, više ga ne možemo mijenjati.

Tako npr. svojstvo UNIQUE ima generalni SB-Tree algoritam, baš kao i NOTUNIQUE, FULLTEXT i DICTIONARY.

Hash tipovi koriste Hash algoritam (neki hash tipovi koriste i sharding), a Lucene se većinom koriste za full text.

Također želimo da rubovi budu jedinstveni – opet pravimo indeks, ali označavamo polja in i out tj. želimo da se kombinacija pojavi samo jednom.



Slika - Index za jedinstvenost rubova

Osim jedinstvenosti, želimo da u neka svojstva možemo unijeti samo specifične vrijednosti.

Npr. u klasu Users želimo limitirati mogućnost unosa svojstva *membership* na samo „Regular“ i „Premium“.

To činimo SQL naredbom koja koristi regularne izraze:

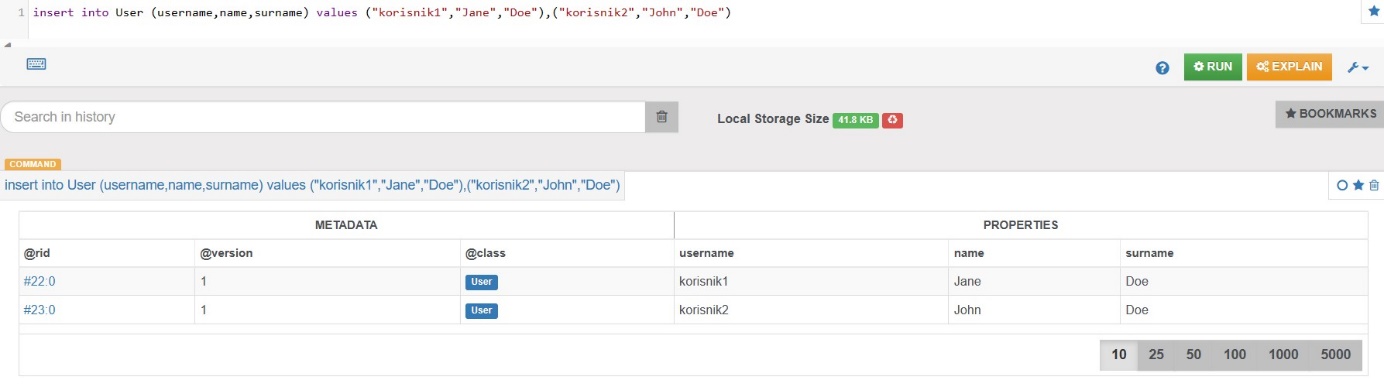
**ALTER PROPERTY Users.membership REGEXP "[Regular|Premium]"**

Na ovaj način smo još limitirali i svojstvo gender:

**ALTER PROPERTY Users.gender REGEXP "[M|F|Other]"**

* 1. Unos podataka

Podatke možemo unijeti na više načina.



Slika - Dodavanje korisnika koristeći SQL

Koristeći SQL naredbu možemo npr. dodati novog korisnika tako da koristimo „**INSERT INTO** Class“ i u zagrade nabrojimo svojstva koja želimo unijeti.

Pošto OrientDB podržava shema-less pristup, smijemo upisati što god želimo kao svojstvo.

Vrijednosti (eng. values) koje unosimo također stavljamo u zagrade i odvajamo zarezom.

Slika na kojoj se prikazuje tekst, broj, softver, snimka zaslona

Opis je automatski generiran

Slika - Add New Record

Možemo dodati novi record i ako uđemo u „SCHEMA“ i kliknemo „Add New Record“ kraj klase čiju vrijednost želimo unijeti.



Slika - Stvaranje ruba između korisnika i knjige

Rub stvaramo sa

**CREATE EDGE <name> from <out> to <in>**

tako da specificiramo od kojeg do kojeg vrha ćemo stvoriti vezu.

Ovaj pristup nije efikasan kada želimo unijeti veliku količinu podataka, pa nam tu koristi mogućnost unosa podataka kao .csv ili .json file-ova.

U *„interests“* smo importali klase Movies, Books i Games iz .csv datoteka.

Ovo smo učinili koristeći ETL (The Extractor Transformer and Loader).

**ETL** je modul OrientDB-a koji nam dopušta učitavanje podataka korištenjem ETL procesa:

1. Konfiguracija – stvaranje JSON dokumenta za konfiguraciju
2. Ekstraktor – izvlači podatke iz originalne datoteke
3. Transformer – modificira podatke i postavlja INPUT
4. Loader – sprema modificirane podatke u OrientDB

Slika na kojoj se prikazuje tekst, snimka zaslona, crta

Opis je automatski generiran

Slika - JSON konfiguracija

U JSON datoteku spremamo sve upute potrebne za izvođenje ETL procesa.

U „source“ specificiramo putanju na .csv datoteku.

Ekstraktor postavimo na „csv“ svojstvo (da smo odlučili importati .json, koristili bi json extractor). Postoji mogućnost i detaljnijeg opisa ekstrakcije - separatori, gledamo li po retku, stupcu, preskačemo li vrijednosti…

Transformeru govorimo u koju klasu spremamo podatke.

Na kraju, loader sprema podatke u OrientDB.

Zapisujemo putanju do naše baze podataka (sama će se stvoriti ako ne postoji – „dbAutoCreate“ postavili na true), navodimo tip baze (graf – iako nismo importali veze, pa je sličnije document tipu), korisničko ime i lozinku za pristup bazi. Na sličan način možemo stvarati svojstva, indexe, clustere…

Nakon stvaranja .json datoteke odlazimo u command prompt i premještamo se u folder koji sadrži oetl.bat (ili .sh). Zatim importamo podatke tako što pokrećemo oetl skriptu i pišemo ime .json konfiguracije s uputama za unos (npr. oetl.bat books1.json).

Na ovaj način smo stvorili klase Movies, Books i Games i nakon toga smo unijeli odgovarajuća svojstva.

Neka svojstva kao što je npr. svojstvo title u Books nisu dopuštala unos zbog neodgovarajućeg tipa podataka tj. tip u svojstvu nije bio jedinstven.

Problem se ispravi prebacivanjem svih vrijednosti svojstva u jedan tip naredbom:

**UPDATE Books SET title = title.asString() WHERE title.type() <> 'STRING'**

Sada možemo označiti da je tip podatka STRING.

* 1. Functions Managment + generiranje podataka

OrientBD dopušta pisanje funkcija koristeći SQL ili javascript (API) u sučelju Functions Managment.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Ikona na računalu

Opis je automatski generiran

Slika - Functions Managment

U textbox „Name“ upisujemo naziv funkcije, a potom biramo jezik. Označujemo želimo li da funkcija bude idempotentna (mijenja BP – odnosi se na HTTP GET (dohvaćanje iz baze), a inače je HTTP POST (unos u bazu) – postoje još i DELTE i PUT HTTP naredbe) i dodajemo parametre i njihove nazive ako nam trebaju. Funkcija se sprema i pokreće na EXECUTE.

Podatke za klasu Users, kao i rubove, smo kreirali nasumičnim generiranjem podataka korištenjem funkcija.

Primjerice, na (slici 30 – gore), imamo kod funkcije **ban()** koja za nasumične korisnike stvara rub Banned.

Na bazu podataka djelujemo tako da koristimo već ugrađenu ključnu riječ **db** koja predstavlja bazu i sa metodom **command()** upisujemo SQL upit koji želimo izvršiti.

Generirali smo slučajne brojeve koji predstavljaju *user\_id* za pojedinog koirsnika.

Funkcija **createUsr()** je služila za kreiranje jednog korisnika i vraćala je objekt oblika:

**obj**={

username: name + sur + rand(0,5000),

name: name,

surname: sur,

gender: gen,

membership: memb,

dat: dat

};

name i surname biramo nasumično iz lista imena za Male, Female, Neutral i Surname, te username stvaramo tako da spojimo ime i prezime i dodamo random broj između 0 i 5000. Dodajemo još gender, membership i dan stvaranja računa jer su obavezna svojstva.

Potom sa funkcijom **createUsers()** unosimo objekte u klasu Users.

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

Slika - Funkcija readBooks()

Funkcija **readBooks()** spaja nasumičnog korisnika s nasumičnom knjigom s rubom Reads ovisno o njenom *bookID*-ju. Ovisno o nasumičnom broju dodajemo i neka svojstva – npr. ovdje smo dodavali *rating* ovisno o kojoj iteraciji je riječ. Na sličan način funkcioniraju i funkcije **playGames()** i **watchMovies()**.

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Opis je automatski generiran

Slika - funkcija makeFriends()

Funkcija **makeFriends()** stvara rub isFriendsWith između nasumičnih korisnika.

Uočimo da smo stvarali vezu u oba smjera jer najčešće ovakvi odnosi idu u oba smjera, no nije nužno (npr. na društvenim mrežama se dvije osobe ne moraju uzajamno pratiti). Ovo u grafu tvori tzv. višestruku brid.

Ovdje smo dodavali svojstvo *how\_long* ovisno o tome je li nasumični broj djeljiv sa 5.

Slično smo napravili i funkcije **makeRealtions()** i **makeCouples()**.

Pokuša li ikoja funkcija unijeti iste podatke vraća se greška.

1. Upiti na bazi „interests“ i grafovi
   1. Graph Editor

Graf Editor nam omogućuje grafički prikaz odnosa vrhova i vizualno upravljanje svojstvima.

Slika na kojoj se prikazuje tekst, snimka zaslona, karta, dijagram

Opis je automatski generiran

Slika - Primjer izgleda grafa

Prikazujemo vrhove i rubove koristeći SQL upite. Svakim novim upitom dodajemo još vrhova/rubova na graf. U gornjem desnom kutu piše koliko je rubova i vrhova prikazano. Vrhove također možemo pomicati i mijenjati im veličinu.

Slika na kojoj se prikazuje tekst, softver, snimka zaslona, Multimedijski softver

Opis je automatski generiran

Slika - Vrh na grafu

Klik na vrh nam dopušta da vidimo svojstva tog vrha i upravljanje njima. Možemo mu vidjeti i sve ostale čvorove s kojim je povezan i kako (klikom na određeni tip brida možemo proširiti prikaz grafa. Zbrojimo li sve rubove vrha dobili bismo njegov **stupanj**.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, dizajn

Opis je automatski generiran

Slika - Postavke vrha na grafu

Klikom na tab settings možemo dodatno mijenjati vizualna svojstva grafa. Možemo odabrati boju vrha, naziv, veličinu, koristi li ikonu…

* 1. Dohvaćanje podataka iz baze

Slika na kojoj se prikazuje tekst, snimka zaslona, crta

Opis je automatski generiran

Slika - Broj vrhova

Slika na kojoj se prikazuje tekst, snimka zaslona, crta

Opis je automatski generiran

Slika - Broj bridova

U bazi *„interests“* ima 29972 vrhova i 16211 bridova.

Provedemo li isti upit na pojedinačnim klasama dobijemo sljedeće podatke.

U bazi se nalazi 799 korisnika, 8807 filmova/serija, 1569 knjiga i 18797 igara.

Ima 753 prijateljstva, 321 obiteljskih odnosa, 287 parova i 685 bannova.

Promotrimo prikaz podataka.

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

Slika - Podaci klase Books

Slika na kojoj se prikazuje tekst, broj, Font, crta

Opis je automatski generiran

Slika - Podaci klase isFriendsWith

Iz klase Books (tipa V) vidimo da svaki redak predstavlja novi podatak tj. novi vrh. Svaki stupac predstavlja svojstvo, a prikazano je povezan li je (i kako) s nekim drugim vrhom.

U klasi isFriendsWith (tipe E), svaki redak predstavlja novi rub.

Iako smo odabrali „select all“ opciju, OrientDB nam prikazuje samo prvih 10 podataka. Broj prikazanih podataka možemo povećati klikom na brojku u donjem desnom kutu.

* + 1. Rad s vrhovima

Uklonimo li rubove, čvorovi nam ostaju kao jedine klase. Raditi samo s čvorovima možemo smatrati ekvivalentnim radom u BP tipa dokument. Tako će, iz perspektive dokument BP, vrhovi zapravo biti različite kolekcije.

* Koliko filmova je snimljeno u kojoj državi? (npr. SAD)

**select count(\*) from Movies where country="United States"**

2818

* Koji filmovi imaju više od 1 direktora?

**select title,director.size() from Movies where director.size()>1**

Slika na kojoj se prikazuje tekst, račun, snimka zaslona, crta

Opis je automatski generiran

Slika - Filmovi koji imaju više od jednog direktora

* Filmovi u kojima je glumio određeni glumac? (npr. Ryan Reynolds)

**select title from Movies where („Ryan Reynolds“ in cast)**

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, crta

Opis je automatski generiran

Slika - Filmovi u kojima je glumio Ryan Reynolds

* Vrati imena svih ljubavnih komedija snimljenih nakon 2000.

**select title,description from Movies where (" Romantic Movies" in listed\_in) and ("Comedies" in listed\_in) and release\_year>2000 and type=="Movie"**

Slika na kojoj se prikazuje tekst, Font, broj, dokument

Opis je automatski generiran

Slika - Ljubavne komedije

* Koliko knjiga nije napisano na engleskom jeziku?

**select count(\*) from Books where language\_code!="eng" and language\_code!="en-US" and language\_code!="en-GB"**

86

* Prosječna ocjena čitatelja ovisno o publisher-u?

**select publisher,avg(average\_rating).asDecimal() from Books group by publisher**

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Prosječan rating knjige po publisher-u

* Prosječan meta\_score igrica?

**select avg(meta\_score) from Games**

70

* Koliko korisnika su žene s Premium Membershipom?

**select count(\*) from Users where gender=="F" and membership=="Premium"**

46

* + 1. Rad s rubovim
* Vrati igre koje igraju neki korisnici i koje su sa napravljene prije 2000. godine.

**select from Games where (release\_date.substring(release\_date.length()-4,release\_date.length())<2000 and in(Plays).size()!=0)**

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Igre koje igraju korisnici napravljene prije 2000.

S metodom **.substring()** smo kidali zadnja 4 znaka u stringu *release\_date* i ta 4 znaka su predstavljala brojeve.

**.size()** nam vraća duljinu liste.

**in(<ime\_ruba>)** nam vraća sve vrhove u koje se ulaze npr. za Plays bi to bile igre. **out(<ime\_ruba>)** vraća sve vrhove iz kojih rub izlazi.

* Koji film je gledalo najviše korisnika?

**select title,in(Watches).size() from Movies where in(Watches).size()==(select**

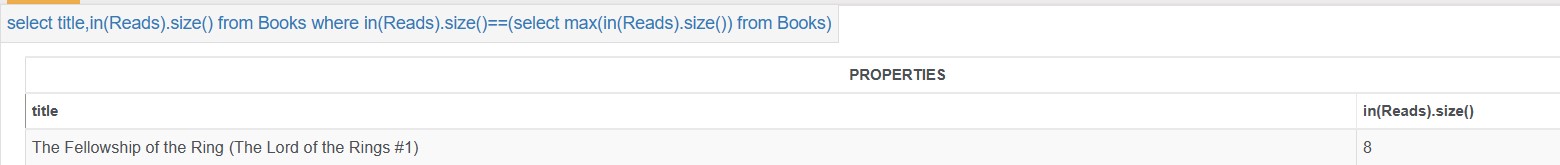
**max(in(Watches).size()) from Movies)**



Slika - Najgledaniji film

* Knjiga koju je čitalo najviše korisnika?

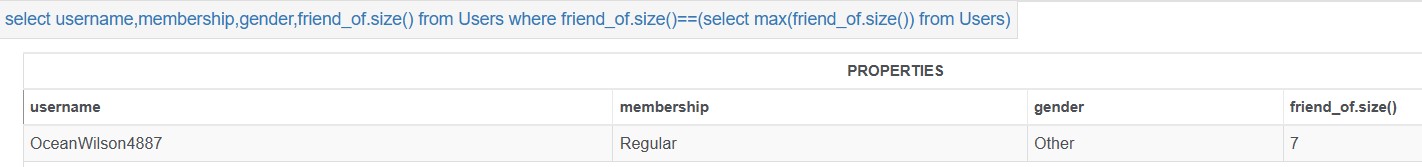
**select title,in(Reads).size() from Books where in(Reads).size()==(select max(in(Reads).size()) from Books)**



Slika - Najčitanija knjiga

* Koji korisnik ima najviše prijatelja?

**select username,membership,gender,friend\_of.size() from Users where friend\_of.size()==(select max(friend\_of.size()) from Users)**



Slika - Korisnik s najviše prijatelja

* Koliko korisnika nema dodanih prijatelja?

**select count(\*) from Users where friend\_of.size()==0**

733

* Koji korisnici su u braku i imaju dijete?

**SELECT @rid,username,outE("inRelationshipWith")[married==true],outE("isFamilywith")[relation=="Parent"]**

**FROM Users where outE("inRelationshipWith")[married==true].size()>0 and outE("isFamilywith")[relation=="Parent"].size()>0**

Slika na kojoj se prikazuje tekst, Font, crta, broj

Opis je automatski generiran

Slika - Korisnici koji su u braku i imaju djecu

Npr. [relation==“Parent“] nam označava da gledamo samo rubove gdje svojstvo *relation* ima vrijednost „Parent“.

* Korisnici koji su bannali najviše korisnika i koliko njih?

**select \* from (select out.@rid, out.username,count(out) as zbroj from Banned group by out.username) where zbroj==(select max(zbroj) from (select out.@rid,out.username,count(out) as zbroj from Banned group by out.username))**

Slika na kojoj se prikazuje tekst, crta, broj, Font

Opis je automatski generiran

Slika - Korisnici koji su najviše bannali

**GROUP BY** je SQL upit koji grupira rezultate ovisno o odabranim svojstvima. Npr. na (slici 49 – gore) smo grupirali po tome koliko puta su bannali tj. koliko su puta bili korisnici na out kraju ruba Banned i prebrojali ovisno u username-u.

Pošto su vrhovi klase, želimo li pristupiti svojstvu unutar te klase, stavljamo <klasa>.<svojstvo>.

Tako npr. u kodu gore smo koristili out.username kako bi pristupili username-u korisnika koji je out dio ruba Banned.

Ovako možemo nizati više svojstava ovisno o tope koji je tip.

* Korisnici s najviše bannova i koliko?

**select \* from (select in.@rid,out.username,count(in) as zbroj from Banned group by in.username) where zbroj==(select max(zbroj) from (select in.@rid,in.username,count(in) as zbroj from Banned group by in.username))**



Slika - Korisnik s najviše bannova

* Koliko korisnika je bilježilo samo igre i knjige, ali ne i filmove?

**SELECT count(\*)**

**FROM Users where outE("Reads").size()>0 and outE("Plays").size()>0 and outE("Watches").size()==0**

32

* Koji korisnik je gledao najviše filmova?

**select \* from (select out.@rid,out.username,count(out) as zbroj from Watches group by out.username) where zbroj==(select max(zbroj) from (select count(out) as zbroj from Watches group by out.username))**



Slika - Korisnik koji je gledao najviše filmova

* Igre koje su igrane manje od 5 sati i koje su loše ocijenjene (<5) od strane korisnika?

**select in.name,rating,hours\_played from Plays where rating<5 and hours\_played<5**

Slika na kojoj se prikazuje tekst, račun, crta, broj

Opis je automatski generiran

Slika - Lošije igre

* Svi filmovi/serije/knjige/igre koje su korisnici ocijenili s 10?

**select @rid,username,outE("Watches")[rating==10].in.title as movies ,outE("Plays")[rating==10].in.name as games ,outE("Reads")[rating==10].in.title as books from Users**

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Slika - Mediji ocjenjeni s 10

* + 1. Primjeri upita s grafovima

Radi preglednosti, u sljedećim upitima ćemo se limitirati na jednog korisnika.

Uzmimo korisnika sa @rid=#34:84 (username=“RoyalPip3261“)

Bitna naredba u ovoj sekciji je **TRAVERSE** koja vraća cross-relationships od klase koju promatramo. Primjerice:

**TRAVERSE out() FROM #34:84**

Slika na kojoj se prikazuje snimka zaslona, dijagram, krug, tekst

Opis je automatski generiran

Slika – TRAVERSE

Traverse je podržan na graf i document modelu BP.

Sa TRAVERSE možemo specificirati i dubinu upita (koliko je udaljena veza) i vrstu veze tj. ruba.

* Vrati sve prvorazredne prijatelje korisnika.

**SELECT FROM (TRAVERSE outE(), inV() FROM #34:84 MAXDEPTH 1) WHERE @class = 'isFriendsWith'**

Slika na kojoj se prikazuje krug, snimka zaslona, dijagram

Opis je automatski generiran

Slika - Prvorazredni prijatelji

Sa **MAXDEPTH** smo specificirali dubinu. Alternativni naziv za dubinu je bliskost.

* Vrati trećerazredne prijatelje.

**SELECT FROM (TRAVERSE outE(), inV() FROM #34:84 MAXDEPTH 2) WHERE @class = 'isFriendsWith'**

Slika na kojoj se prikazuje dijagram, karta, vizualizacija

Opis je automatski generiran

Slika - Trećerazredni prijatelji

Ovim upitom smo dobili i prijatelje prijatelja.

* Vrati sve veze korisnika (dubina 4).

**SELECT FROM (TRAVERSE outE(), inV() FROM #34:84 MAXDEPTH 4) WHERE @class = 'isFriendsWith' or @class="Banned" or @class="inRelationshipWith" or @class="isFamilywith"**

Slika na kojoj se prikazuje tekst, dijagram, snimka zaslona, crta

Opis je automatski generiran

Slika - Odnosi korisinika RoyalPip3261

* Knjige/filmovi/igre koje je korisnik konzumirao?

**SELECT FROM Watches WHERE** [**out.@rid==#34:84**](mailto:out.@rid==#34:84)**;**

**SELECT FROM Reads WHERE out.@rid==#34:84;**

**SELECT FROM Plays WHERE** [**out.@rid==#34:84**](mailto:out.@rid==#34:84)**;**

Slika na kojoj se prikazuje dijagram, snimka zaslona, dizajn

Opis je automatski generiran

Slika - Interesi korisnika RoyalPip3261

Graf sučelje funkcionira tako da se novim upitom samo nadodaju (prošli upit i (slika 58)).

Ovakav tip grafa se zove stablo. Specifičnije, ovakav tip grafa se zove zvijezda tj. imamo jedan centralni vrh i korisničke interese koji su listovi i ne idu u druge vrhove.

Postavlja se pitanje kako pristupiti svim putovima vezanim za korisnika.

Koristimo metodu **path()** koja vraća sve moguće putove.

* Vratimo sve moguće putove sa početkom u vrhu našeg korisnika.

**select $path from (traverse in() from #34:84)**

Slika na kojoj se prikazuje tekst, snimka zaslona, broj, Font

Opis je automatski generiran

Primijetimo da je *$path* jedan on rezultata naredbe TRAVERSE.

Često kod grafova bude bitan najkraći put između 2 čvora.

* Vrati nam najkraći put između korisnika #34:84 (RoyalPip3261) i „SterlingLogan383“.

**SELECT expand(path)**

**FROM (**

**SELECT shortestPath($from, $to) AS path**

**LET**

**$from = (SELECT FROM Users WHERE @rid==#34:84),**

**$to = (SELECT FROM Users WHERE username="SterlingLogan838")**

**UNWIND path**

**)**

Slika na kojoj se prikazuje tekst, snimka zaslona, dijagram, krug

Opis je automatski generiran

Slika - Najkraći put između RoyalPip3261 i SterlingLogan838

OrientDB ima već ugrađenu metodu **shortestPath()** koja prima 2 parametra : početni i završni vrh.

Potom sa **unwind path** koristimo rezultat – unwind razbije vraćenu strukturu i omogućuje SELECT da pristupi podacima.

**expand()** proširuje rezultat – vrati sva svojstva umjesto jednog koji bi bio rezultat upita.

Promotrimo put dobiven gornjim upitom.

Ova dva korisnika su povezana na sljedeći način:

Korisnik RoyalPip3261 je prijatelj s korisnikom GustavoEmber4826 koji je pročitao istu knjigu kao i korisnik SterlinPolly4837 koji je prijatelj s korisnikom SterlingLogan838.

* Koji je najkraći put između filma #25:2109 i knjige #20:7?

**SELECT expand(path)**

**FROM (**

**SELECT shortestPath($from, $to) AS path**

**LET**

**$from = (SELECT FROM Movies WHERE @rid==#25:2109),**

**$to = (SELECT FROM Books WHERE @rid==#20:7)**

**UNWIND path**

**)**

Slika na kojoj se prikazuje tekst, snimka zaslona, dijagram, krug

Opis je automatski generiran

Slika - Najkraći put između filma 25:2109 i knjige #20:7

Korisnik koji je pogledao zadani film je odigrao istu igru kao i korisnik koji je pročitao zadanu knjigu tj. veza među zadanim filmom i knjigom je ta da su ih gledali/čitali korisnici koji su igrali istu igru.

Na ovaj način bi grafovi bili korisni u sustavima preporuke. Na temelju činjenice da su oba korisnika igrala istu igru bi im preporučili pripadni film/knjigu koja se svidjela nekom drugom korisniku. Analogno za sve druge kombinacije interesa.

U OrientDB-u preporuke možemo napraviti koristeći **MATCH** – gleda rezultate koje odgovaraju onima koje smo naveli da želimo dobiti.

* Preporuči nove prijatelje korisniku #34:84 .

**MATCH**

**{class: Users, as: user, where: (@rid="#34:84")}-isFriendsWith-{as: friend},**

**{as: friend}-isFriendsWith-{as: friendOfFriend, where: ($matched.user not in $currentMatch.both('isFriendsWith') and $matched.user != $currentMatch)}**

**RETURN DISTINCT friendOfFriend.username**

Slika na kojoj se prikazuje tekst, crta, Font, snimka zaslona

Opis je automatski generiran

Slika – Preporuka prijatelja za korisnika RoyalPip3261

{uvjeti klase}-<rub>-{as: …}… radi na način da uzima klasu povezanu rubom za neku drugu klasu (prvu klasu spremamo s *„as:“* i koristimo dalje. Sa -<rub>- „ručno“ radimo put, nadovezivanjem rubova na vrhove i tako dobijemo rezultat koji želimo.

**$matched** nam vraća rezultat koji je odgovarao uvjetima, **$currentmatch** vraća trenutno poklapanje, a **.both(<rub>)** gleda istovremeno i in i out vrhove ruba. **DISTINCT** vraća jedinstvene rezultate.

Uočimo da smo u upitu gore izbacili ime korisnika i ljudi s kojima je već prijatelj.

Alternativno smo upit (starija verzija OrientDB-a) mogli napisati i kao :

**MATCH**

**{class: User, as: user, where: (@rid="#34:84")}**

**.both('isFriendsWith').both('isFriendsWith')**

**{as: friendOfFriend}**

**RETURN user, friendOfFriend**

* Vrati prijatelje koji su zajednički prijateljima korisnika #34:84?

**MATCH {class: Users, as: user, where: (@rid="#34:84")}-isFriendsWith-{as: friend},**

**{ as: user }-isFriendsWith-{ as: friend }**

**RETURN user.username, friend.username**

**Slika na kojoj se prikazuje tekst, snimka zaslona, Font, crta

Opis je automatski generiran**

Slika - Zajednički prijatelji od korisnika RoyalPip3261 i njegovih prijatelja

* Preporuči igre nekom korisniku zadanom korisniku.

**MATCH**

**{Class: Users, as: user, where: (@rid=#34:84)}**

**-isFriendsWith->{class: Users, as: friend},**

**{as: friend}-Plays->{Class: Games, as: games}**

**RETURN $pathelements**

Slika na kojoj se prikazuje snimka zaslona, krug, tekst, dijagram

Opis je automatski generiran

Slika - Preporuka igara

Zapišimo li „{iz klase1}-<rub>->{iz klase2}“ na ovaj način tj. sa strelicom, onda određujemo out, in klase ruba.

U upitu gore, preporuku smo vršili tako da smo korisniku preporučili igre koje su igrali njegovi prijatelji.

Na (slici 63) se vraćaju igre samo 1 prijatelja. Ne zato što samo jedan prijatelj korisnika igra igre nego zato što se vraća samo 1 od mogućih putova.

* Preporuči prijatelje ovisno o knjigama koje je korisnik čitao.

**MATCH**

**{Class: Users, as: user, where: (@rid=#35:75)}-Reads->**

**{class: Books, as: book},**

**{as: book}<-Reads-{Class: Users, as: reader}**

**RETURN reader,reader.outE("Reads").in.title**

Slika na kojoj se prikazuje tekst, pismo, papir, dokument

Opis je automatski generiran

Slika - Korisnici i zajedničke knjige njih i zadanog korisnika

Prethodna preporuka radi na način da se prvo gledaju knjige koje je korisnik čitao (spremaju se kao *books*) i onda se gleda koji korisnici su čitali te knjige.

* Preporuči knjige korisniku.

**MATCH**

**{Class: Users, as: user, where: (@rid=#35:75)}-Reads->{class: Books, as: book},**

**{as:book}.inE("Reads"){as: ReviewStars, where: (rating>5)}.inV(){as: rate},**

**{as:rate}<-Reads-{Class:Users,as:reader},**

**{as:reader}.outE("Reads"){as: ReviewStars, where: (rating>5)}.outV(){as: rate1},**

**{as:rate1, where:(@rid!=#35:75)}-Reads->{class:Books,as:bookrec},**

**{as:bookrec}.inE("Reads"){as: ReviewStars, where: (rating>5)}.inV(){as: rate2}**

**RETURN distinct(rate1.name)**

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, bijelo

Opis je automatski generiran

Slika - Knjige preporučene korisniku RoyalPip3261

Gornja preporuka funkcionira na način da među knjigama koje je korisnik čitao gleda one koje je pozitivno ocijenio (rating>5), te gleda korisnike koji su čitali te knjige. Među knjigama koje su čitali ti korisnici opet probire one koji su pozitivno ocijenjeni i vraća ih kao preporuku.

Primijetimo još da su svi bridovi u BP usmjereni tj. imaju ulaz i izlaz.

1. Zaključak

OrientDB je vrlo korisna BP jer dozvoljava primjenu više različitih modela. Document model nam omogućuju shema-less pohranu velikog broja podataka, do graf model omogućava intuitivno i brzo povezivanje različitih klasa podataka.

OrientDB ima i Open Studio Web Tool koji dodatno olakšava rad korisniku, ali je način rada OrientDB-a specifičan i potrebno ga je naučiti.

Sve podatke možemo dobiti grafički i u obliku tablice, te postoje mnoge ugrađene funkcije. OrientDB ima SQL i Java API-je, te omogućava pisanje vlastitih funkcija. Podržava unos .csv i .json oblika podataka, a export izvršava u .json obliku.

U zaključku, OrientDB je vrlo koristan način spremanja i rada s različitim podacima, te je napravljen na način da se praktični dio lako i brzo izvršava.

Popis literature

* [OrientDB tutorial](https://youtu.be/uMpb7aRzVM4)
* [Stack Overflow](https://stackoverflow.com/)
* [OrientDB manual (dokumentacija)](http://orientdb.com/docs/3.0.x/)