Grafovska NoSQL baza podataka kao izvještajni sustav

Matijašević, Mia

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:733445

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-10-13**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Mia Matijašević

GRAFOVSKA NOSQL BAZA PODATAKA KAO IZVJEŠTAJNI SUSTAV

Diplomski rad

Voditelj rada:
dr. sc. Ognjen Orel

Zagreb, rujan 2021.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

"Prijatelju moj, kad sam tražio od tebe da mi služiš, i ti si prihvatio, rekao sam ti da guraš onaj kamen svom svojom snagom, što si i uradio. Nijednom nisam spomenuo da sam očekivao da ćeš ga pomaknuti. Tvoj zadatak je bio da guraš. I sada, dolaziš k meni, iscrpljen i istrošene snage, misleći da nisi uspio. No, je li zaista tako? Pogledaj se. Tvoje ruke su snažne i mišićave, leđa nabijena i preplanula, koža na dlanovima je očvrsnula od stalnog pritiska, a noge su ti postale krupne i čvrste. I pored otpora ti si mnogo porastao, i tvoje mogućnosti su sada daleko veće nego ranije. Ipak, nisi pomaknuo kamen. Ali tvoj poziv bio je da budeš pokoran i guraš. Da vježbaš svoju vjeru i povjerenje u moju mudrost. To si uspio. Ja ću sada, prijatelju moj, pomaknuti kamen."

Hvala mojoj OBITELJI, ona mi je dala krila da letim.
Hvala mojim prijateljima, oni su me pratili u letu.
Hvala svim onima koji su mi davali vjetar u leđa.
I hvala mom mentoru koji mi je pomogao da sigurno sletim.

Sadržaj

Sadržaj **iv**

Uvod **1**

1 Osnovni pojmovi **2**
1.1 Osnovni pojmovi iz teorije grafova 2
1.2 Osnovni pojmovi vezani uz relacijske baze podataka 2

2 NoSQL **4**
2.1 Razlozi za nastajanje i karakteristike NoSQL baza podataka 4
2.2 Kategorije NoSQL baza podataka 7

3 Grafovske baze podataka **10**
3.1 Općenito o grafovskim bazama podataka 10
3.1.1 Karakteristike 11
3.1.2 Pohrana i obrada podataka 12
3.1.3 Prednosti u odnosu na relacijske baze podataka 13
3.1.4 Modeli grafovske baze podataka 16
3.2 Primjeri grafovskih baza podataka 18

4 Migracija podataka **20**
4.1 Generička migracija podataka 20
4.2 Specifična migracija podataka 21

5 Vizualizacije grafova **24**
5.1 Alati za vizualizaciju 24
5.1.1 Kriteriji za odabir alata za vizualizaciju 26
5.1.2 Vis.js 28
5.1.3 Neovis.js 29
5.1.4 Sigma.js 30

5.1.5 Gephi 31
5.1.6 Osvrt na izbor alata za vizualizaciju 33

6 Implementacija izvještajnog sustava na Neo4j + Neovis.js platformi 34
6.1 Tema izvještajnog sustava i ideja vizualizacije 34
6.2 Izvor podataka 35
6.3 Migracija podataka 36
6.3.1 Izvoz iz relacijske baze podataka 37
6.3.2 Uvoz u grafovsku bazu podataka 39
6.4 Vizualizacija podataka s Neovis.js bibliotekom 41
6.5 Ograničenja Neo4j + Neovis.js platforme 46

7 Implementacija izvještajnog sustava za veće količine podataka 47
7.1 Migracija podataka 48
7.2 Vizualizacija podataka sa Gephi Toolkit bibliotekom 50
7.2.1 Stvaranje radnog prostora 51
7.2.2 Uvoz podataka 51
7.2.3 Izgled vizualizacije 53
7.3 Sigma Exporter 55
7.4 Završni izgled 57

8 Zaključak 61

Bibliografija 63

Uvod

Većinu informacija čovjek opaža vizualno. Imamo sposobnost prepoznavanja i grupiranja objekata s obzirom na njihovu veličinu, oblik ili boju, a u našoj je prirodi koristiti i lakše razumjeti vizuale. Informacije su predstavljene podacima i kao takve pohranjuju se unutar baza podatka. Način na koji korisnik može vizualizirati i istraživati domenu podataka ostvaruje se putem izvještajnih sustava.

Brojne tvrtke i poslovni sustavi već dugi niz godina svoje podatke smještaju unutar relacijskih baza podataka. Zbog njihove pouzdanosti i funkcionalnosti i danas su čest odabir za pohranu. Popularizacijom internetskih usluga raste i količina podataka koju informacijski sustavi trebaju pohraniti i obraditi. Za to vrijeme relacijske baze podataka ulažu snage u pronalazak rješenja za nove poteškoće, no paralelno se pokreću ideje potpuno drugačijeg pristupa pohrani i obradi podataka. Realizacija novog pristupa rezultira novim bazama podataka pod zajedničkim imenom NoSQL. NoSQL baze podataka dijele se u nekoliko kategorija, a među njima posebno se ističu grafovske baze podataka. Korištenjem modela grafa, grafovske baze podataka predstavljaju dobro rješenje za domene složenije povezanih podataka. Zbog toga su brojne domene, inicijalno pohranjene u relacijskim bazama podataka, prikladne grafovskim bazama podataka, pa nastaju razni načini migracije podataka. Porastom korištenja web aplikacija općenito, nastaje i sve više izvještajnih sustava na webu. Interakcijom s podacima poboljšava se korisničko iskustvo, a načini implementacije izvještajnih sustava i danas su u porastu. Povrh svega, razvijeni su razni vizualizacijski alati koji pružaju jednostavno i dobro korisničko iskustvo nad analizom i pretraživanjem podataka. S obzirom da model grafa intuitivno stvara vizualnu predodžbu, od interesa je proučiti i alate kojima bi grafovsku bazu podataka mogli koristiti kao izvještajni sustav.

Ovaj diplomski rad sastoji se od teorijskog i praktičnog dijela. Teorijski dio započinje kratkim pregledom NoSQL baza podataka, a potom se orijentira na detaljniji prikaz grafovskih baza podataka. Opisani su načini za migraciju podataka iz relacijskih u grafovske baze podataka te naposljetku opisani neki od alata za njihovu vizualizaciju. Praktični dio sastoji se od opisa implementacije dviju web aplikacija kojima se opisana teorija prikazuje primjerom.

Poglavlje 1

Osnovni pojmovi

1.1 Osnovni pojmovi iz teorije grafova

S obzirom da se ovaj rad temelji na pregledu grafovskih baza podataka i vizualizacijom podataka u obliku grafa, radi konzistentnosti i cjelovitosti u ovom poglavlju iznosimo definicije usmjerenog i neusmjerenog grafa. Definicije su preuzete iz [35].

Definicija 1.1.1. *(Neusmjereni)* ***Graf*** *je uređeni par $G = (V, E)$, gdje je $V = V(G)$ neprazan skup čije elemente nazivamo* ***vrhovima****, a $E = E(G)$ je familija dvočlanih podskupova od V koje nazivamo* ***bridovima****.*

Definicija 1.1.2. ***Usmjereni graf*** *je uređeni par $G = (V, E)$, gdje je $V = V(G)$ neprazan skup čije elemente nazivamo* ***vrhovima****, a $E = E(G)$ je skup uređenih parova elemenata iz V, čije elemente zovemo* ***usmjerenim bridovima****.*

1.2 Osnovni pojmovi vezani uz relacijske baze podataka

Jedan od ciljeva ovog rada je dati pregled načina za migraciju podataka iz relacijske u grafovsku bazu podataka, a sam praktični dio kao izvor podataka koristi relacijsku bazu podataka. Radi konzistentnosti i cjelovitosti u ovom poglavlju iznosimo pregled osnovnih pojmova vezanih uz relacijske baze podataka. Definicije su preuzete iz [36] i [34].

Definicija 1.2.1. ***Baza podataka*** *(engl. database) je skup međusobno povezanih podataka, pohranjenih u vanjskoj memoriji računala.*

Definicija 1.2.2. ***Sustav za upravljanje bazama podataka*** *(SUBP, engl. Database Management System – DBMS) je računalni program koji omogućava rad s bazama podataka – definiranje, upravljanje, zauzimanje fizičkog prostora za smještaj podataka, zauzimanje*

računalnih resursa potrebnih za rad, omogućavanje pristupa korisnicima, posluživanje korisnika, zaštita od gubitka podataka, sigurnosne mjere zaštite baza podataka te obavljanje internih zadaća.

Definicija 1.2.3. ***Relacijska baza podataka*** *je baza podataka temeljena na relacijskom modelu podataka (engl. relational data model).* ***Relacijski model*** *zasnovan je na matematičkom pojmu relacije. I podaci i veze među podacima prikazuju se tablicama koje se sastoje od redaka i stupaca. Za opis strukture podataka u relacijskom modelu koriste se elementi: entitet, atribut, domena, relacija, n-torka.*

Definicija 1.2.4. ***Entitet*** *(engl. entity) je nešto što ima suštinu ili bit i posjeduje značajke po kojima se može razlikovati od svoje okoline.*

Definicija 1.2.5. ***Atribut*** *(engl. attribute) je karakteristika entiteta i definira se nad domenom. Atribut se također naziva i stupcem.*

Definicija 1.2.6. ***Domena*** *(engl. domain) je skup dopuštenih vrijednosti atributa i definira značenje podataka.*

Definicija 1.2.7. ***Relacija*** *(engl. relation) predstavlja konačan skup n-torki. Relacija se također naziva i tablicom.*

Definicija 1.2.8. ***N-torka*** *(engl. tuple) sadrži vrijednosti atributa koje odgovaraju atributima iz relacije. N-torka se također naziva i retkom.*

Definicija 1.2.9. ***Vrijednost atributa*** *je vrijednost konkretnog atributa u konkretnoj n-torki, pri čemu ta vrijednost mora pripadati domeni atributa.*

Definicija 1.2.10. ***Sustav za upravljanje relacijskim bazama podataka*** *(engl. Relational Database Management System – RDBMS) je sustav za upravljanje bazama podataka temeljenima na relacijskom modelu podataka. Omogućava izvođenje naredbi SQL jezika nad bazama podataka koje sadrži.*

Definicija 1.2.11. ***Ključ K*** *relacije R je podskup atributa od R koji ima sljedeća "vremenski neovisna" svojstva:*

1. *Vrijednosti atributa iz K jednoznačno određuju n-torku u R. Dakle ne mogu u R postojati dvije n-torke s istim vrijednostima atributa iz K.*

2. *Ako izbacimo iz K bilo koji atribut, tada se narušava svojstvo 1.*

Definicija 1.2.12. *Ključ jedne relacije koji je prepisan u drugu relaciju zove se* ***strani ključ*** *(u toj drugoj relaciji).*

Poglavlje 2

NoSQL

Velike količine do sada prikupljenih podataka smještene su u relacijskim bazama podataka. Relacijske baze podataka široko su poznate i korištene u brojnim tvrtkama i poslovnim sustavima već dugi niz godina. Njihova funkcionalnost i pouzdanost, uz nadasve dobru dokumentaciju, drži ih i dalje visoko na ljestvici odabira. Iako se razvijaju i dalje te podilaze raznim novim zahtjevima i rješenjima za nailazeće probleme, susreću se s brojnim poteškoćama često vezanim uz promjenjivu strukturu podataka te efikasnu pohranu i obradu velikih količina podataka. Sukladno tome, došlo je do promišljanja o potpuno drugačijem pristupu podacima i njihovoj pohrani. Tako je 2009. godine započeo pokret pod nazivom NoSQL (engl. *Not Only SQL*). Organiziranim sastankom na temu distribuiranih, nerelacijskih baza podataka otvorenog koda, pod vodstvom Johana Oscarssona i Erica Evansa, započela je popularizacija aspekata NoSQL pokreta te razvoj takozvanih NoSQL baza podataka.

2.1 Razlozi za nastajanje i karakteristike NoSQL baza podataka

Porast broja korisnika internetskih usluga preko raznih organizacija i modernih web aplikacija tijekom godina dovelo je do porasta količine podataka koje je potrebno obrađivati. Potreba za stalnom dostupnošću sustava, fleksibilnost sheme baze podataka, a uz to i niska cijena održavanja sustava, glavni su pokretači nastanka NoSQL baza podataka.

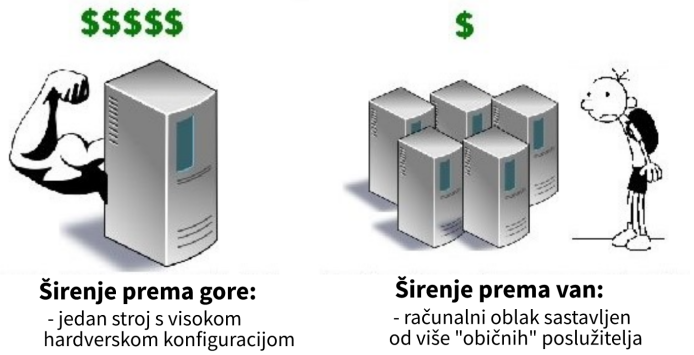
U širem smislu, NoSQL baze podataka možemo definirati kao klasu sustava za upravljanje bazama podataka za koje uglavnom vrijedi da: nisu relacijski, distribuirani su, horizontalno skalabilni i otvorenog koda [20]. To bi značilo sljedeće:

- **Nerelacijski sustavi**

  NoSQL baze podataka ne slijede relacijski model, već se podaci najčešće pohranjuju kao agregati, što znači da su svi logički povezani podaci spremljeni unutar jednog zapisa. Podaci jednog agregata u relacijskim bazama podataka mogu biti razmješteni u stotine tablica.

- **Horizontalna skalabilnost**

  NoSQL baze podataka imaju sposobnost iskorištavanja dodatnih računalnih resursa (u slučaju potrebe za njima) poput povećanja broja poslužitelja, dodatne memorije ili diskovnog prostora, pa kažemo da se mogu širiti prema van (engl. *scale out*), odnosno da su horizontalno skalabilne. Za relacijske baze podataka kaže se da su vertiklano skalabilne, odnosno da imaju sposobnost širenja prema gore (engl. *scale up*) jer se u slučaju potrebe za dodatnom memorijom, jačim procesorom i slično, ulaže u vrlo jaki poslužitelj.



Slika 2.1: Slikoviti prikaz karakteristika vertikalne i horizontalne skalabilnosti

- **Distribuiranost**

  Prikupljanjem i obradom sve veće količine podataka potrebno je ulagati u hardver kako bi se održale performanse sustava. Relacijske baze podataka uglavnom rade na specijaliziranim računalima jer se teško distribuiraju na više njih, stoga je potrebno ulagati u jedan vrlo jaki poslužitelj.
  S druge strane, NoSQL baze podataka često su dizajnirane za rad u računalnom oblaku, odnosno mogu biti podijeljene na više lokacija (poslužitelja), pri čemu su zajednički podaci najčešće kopirani.

- **Otvoreni kod**

  NoSQL baze podataka svoj kod drže javno otvorenim i dostupnim za proširenja.

Bitne značajke relacijskih baza podataka su podržavanje transakcija i njihovih ACID svojstava (od engl. *Atomicity, Consistency, Isolation, Durability* - nedjeljivost, konzistencija, izolacija, trajnost). Ovim se svojstvima obećava konzistentnost podataka u svakom trenutku, čak i na račun dostupnosti sustava, no u mnogim slučajevima to nije željena restrikcija.

Pojavom NOSQL baza podataka te naglaskom na stalnoj dostupnosti sustava, konzistentnost podataka ne predstavlja više nužno svojstvo baze podataka. Također, slijedeći mogućnost i prednost distribucije sustava, postaje bitna otpornost na particiju mreže. Odluka o tome koliko neki sustav treba biti dostupan, a koliko konzistentan, stvar je poslovne potrebe, no ono što postaje sigurno je da za moderne distribuirane sustave, u slučaju mrežne nedostupnosti, sustav i dalje mora raditi. Upravo navedeno, posljedica je CAP teorema (od engl. *Consistency, Availability, Partition tolerance*) koji kaže da u distribuiranom sustavu baza podataka je moguće postići samo dvoje od navedenog: konzistentnost, dostupnost i otpornost na particiju mreže [32]. A kako otpornost na particiju zapravo i nije opcionalna, ostaje izbor između dostupnosti i konzistentnosti sustava.

Tako većina NoSQL baza podataka podržava takozvana BASE svojstva (od engl. *Basically Available, Soft-state, Eventualy consistent*) koja predstavljaju sljedeće tri stavke:

1. **Načelna dostupnost:**
   Sustav je gotovo uvijek dostupan i svaki zahtjev dobiva odgovor.

2. **Labavo stanje:**
   Moguće je mijenjanje sustava i bez korisničkog zahtjeva (vezano uz naknadnu konzistentnost).

3. **Naknadna konzistentnost:**
   Sustav povremeno može biti u nekonzistentnom stanju, no s vremenom, ako u međuvremenu nema korisničkih zahtjeva, postaje konzistentan.

Uz sve navedeno, bitno je spomenuti fleksibilnost sheme podataka kao važnu karakteristiku NoSQL baza podataka. Shema podataka predstavlja detaljan opis smještaja svih podataka iz domene unutar baze podataka. U relacijskim bazama podataka to znači unaprijed definirati entitete i atribute, shvatiti domene atributa i njihove međusobne funkcijske veze, rasporediti atribute u tablice te normalizirati bazu podataka [20]. Upravo iz tog razloga, relacijske baze podataka nisu podložne promjenama sheme podataka pa nisu pogodne ni za aplikacije koje koriste agilne metode razvoja, čiji razvoj ide brzo i koje trebaju biti fleksibilne i otvorene prema raznim promjenama koje zahtijevaju korisnici.

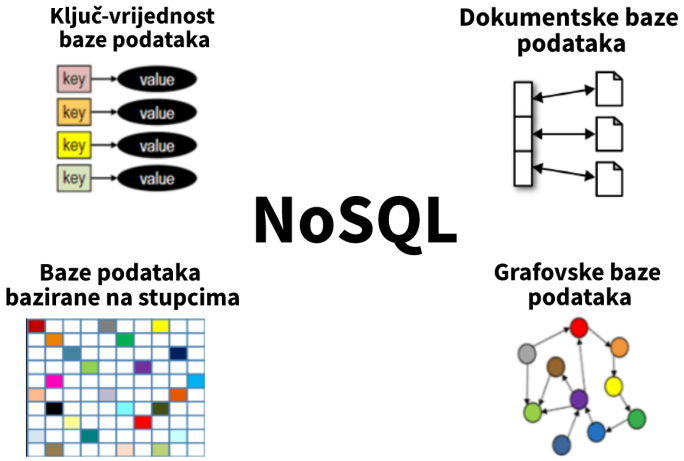
U implementaciji nekog softvera, čak i sitna izmjena na bazi, primjerice izmjena jednog atributa, treba se pomiriti sa svim aplikacijama koje ga koriste, s ili bez obustave rada sustava, što je nerijetko u većim sustavima skup i zahtjevan posao.

Kaže se da su NoSQL baze podataka *schemaless* jer je shemu baze podataka moguće promijeniti u bilo kojem trenutku. Tako je, primjerice, moguće promijeniti tip nekog atributa bez utjecaja na ostale podatke, dodavati proizvoljan broj novih atributa nekog entiteta, kao i postavljati tipove podataka koji prije nisu bili korišteni. Ovim se ostvaruje lakša i brža interakcija između baze podataka i aplikacija koje ju koriste. Istovremeno, postiže se otvorenost prema promjenjivom razvoju softvera iako se aplikacije i dalje trebaju držati u skladu jedna s drugom.

2.2 Kategorije NoSQL baza podataka

Ovisno o modelu podataka kojeg koriste, NoSQL baze podataka uglavnom se dijele u četiri kategorije: ključ-vrijednost (engl. *key-value*) baze podataka, dokumentske (engl. *document*) baze podataka, baze podataka bazirane na stupcima (engl. *column-family*) i grafovske baze podataka (engl. *graph databases*).

Model podataka zapravo se odnosi na logički model, odnosno način na koji su predstavljeni podaci i on je uglavnom različit od fizičkog modela, odnosno smještanja podataka na računalo.



Slika 2.2: Kategorije NoSQL baza podataka

Svaka od navedenih kategorija ima svoje specifičnosti te svaka manje ili više može odgovarati prirodi domene neke baze podataka i određenim poslovnim potrebama. S ciljem približavanja općeg koncepta pojedine kategorije, slijedi njihov kratki pregled.

Ključ-vrijednost baze podataka

Podaci ovih baza podataka reprezentirani su u obliku parova (ključ, vrijednost). Ovaj jednostavan model podataka sličan je rječniku, a dohvat, brisanje i unos podataka temelji na jedinstvenom identifikatoru - ključu. Vrijednost pridružena nekom ključu u pravilu može biti bilo kojeg tipa podatka, pa se tako može pospremiti tekst, slika, video i slično. Važno je napomenuti da vrijednost potpuno ovisi o ključu i pretraga po vrijednosti nije moguća, već se sve odvija putem ključa.

Ove baze podataka ističu se svojom jednostavnošću i brzinom, a posebno su učinkovite za učestale dohvate i unos podataka. Distribuirane su i poznate po dobroj i jeftinoj horizontalnoj skalabilnosti.

Dokumentske baze podataka

Slično kao ključ-vrijednost baze podataka, dokumentske baze podataka koriste model podataka koji se sastoji od parova ključ-dokument. Ključevi su također jedinstveni identifikatori kojima se pridružuju dokumenti raznih formata poput JSON, XML, BSON i drugih polustrukturiranih formata. Za razliku od ključ-vrijednost baza podataka, dokumentske baze podataka pretragu mogu vršiti ne samo po ključevima, već i po dokumentima. Moguće je imati i ugnježđene dokumente pa tako jedan dokument može unutar sebe sadržavati drugi.

Skup logički povezanih podataka smještaju se unutar istog dokumenta, a više takvih dokumenata može sačinjavati jednu *kolekciju*. Kolekcije u dokumentskim bazama podataka se konceptualno mogu poistovjetiti sa tablicama u relacijskim bazama podataka, pri čemu bi jedan redak u tablici odgovarao jednom dokumentu u kolekciji. Dokumenti unutar iste kolekcije mogu biti različitih formata, a s obzirom na pripadnu strukturu moguće ih je i indeksirati.

Horizontalna skalabilnost i fleksibilnost strukture dokumenata čini dokumentske baze podataka dobrim izborom za razne domene, a posebno se uklapaju u aplikacije koje koriste objektno orijentirane jezike jer se dokumenti mogu lako konvertirati u objekte, i obratno.

Baze podataka bazirane na stupcima

Ove se baze podataka temelje na smještanju podataka u obliku ključ-mapa. Naime, i ovdje postoje ključevi kao jedinstveni identifikatori, kao što je to slučaj u ključ-vrijednost i dokumentskim bazama podataka, no na ključ se ovdje vezuje mapa sačinjena od skupina-stupaca, vezanih podataka.

Ako bi se ova baza promatrala u terminima relacijske baze podataka, tada bi jedan redak baza podataka baziranih na stupcima predstavljao jedan entitet, a stupci atribute entiteta. Logički povezani stupci zajedno tvore obitelj stupaca, a svaki stupac također ima svoj ključ

i njemu pripadnu vrijednost. Vrijednosti mogu biti primitivni i složeni tipovi podataka poput brojeva, teksta, slika, JSON formata i drugih.

Baze podataka bazirane na stupcima također imaju mogućnost distribucije i horizontalne skalabilnosti, no u ovom je slučaju potrebno preciznije isplanirati izgradnju baze i njenu podjelu po poslužiteljima kako bi se ostvarilo donekle jednako opterećenje.

Grafovske baze podataka

Grafovske baze podataka ponešto su drugačije od ostalih kategorija NoSQL baza podataka, a najviše zbog toga što ih je teško distribuirati i što se podaci ne pohranjuju kao agregati, već koriste puno manju granulaciju čak i od relacijskih baza podataka. Naime, model podataka je graf i podaci su najčešće predstavljeni kao vrhovi koji mogu imati oznake (labele) i svojstva te kao bridovi koji predstavljaju veze. Bridovi također mogu imati svojstva, ali i smjer. Ono za što su ove baze podataka posebno efikasne, jesu domene u kojima su podaci jako povezani.

Svaka od navedenih kategorija zajedno sa svojim specifičnostima pokriva određeni raspon potreba. Promatrajući sve zajedno kao cjelinu, može se zaključiti da NoSQL baze podataka pružaju mogućnosti i rješenja za velik spektar zahtjeva, što od strane samih domena baza podataka, što od strane organizacija koje te baze podataka koriste u svojim aplikacijama, odnosno proizvodima.

Pored korištenja NoSQL baza podataka u aplikacijama baziranim na obradi i prikazu podataka vezanih uz određeni poslovni sustav, važno je naglasiti široku primjenu ovih baza podataka medu analitičarima i podatkovnim inženjerima koji provode dublje analize nad prirodom samih podataka. Isto tako i vizualizacija podataka postaje iznimno zanimljiva u olakšavanju njihove analize i obrade, a također rastući je trend kako među analitičarima, tako i među raznim organizacijama i njihovim proizvodima namijenjenim široj masi korisnika. No, više o vizualizacijama NoSQL baza podataka slijedi u narednim poglavljima, a u sljedećem ćemo se poglavlju posebno orijentirati na grafovske baze podataka koje su nam u tom smislu najzanimljivije.

Poglavlje 3

Grafovske baze podataka

Iako se modelom podataka, arhitekturom i karakteristikama same baze podataka poprilično razlikuju od ostalih kategorija, grafovske baze podataka također su svojevrsna kategorija unutar NoSQL baza podataka. Različiti modeli grafova, načini pohrane i obrade grafa, fleksibilnost i agilnost, samo su neke od svojstava ovih baza podataka čiji pregled slijedi u idućim poglavljima.

3.1 Općenito o grafovskim bazama podataka

Proteklih godina pojavili su se brojni projekti i proizvodi za upravljanje, obradu i analizu grafova. Kako prikazati osnovne koncepte i razlike pojedinih tehnologija, može predstavljati izazov. No, ipak neke se definicije, podjele i karakteristike mogu zasigurno odrediti.

Grafovsku bazu podataka možemo definirati kao online sustav za upravljanje bazama podataka čije osnovne CRUD (engl. *Create, Read, Update, Delete* - kreiranje, čitanje, ažuriranje, brisanje) operacije izlažu graf kao model podataka [39]. Model grafa generalno izlaže podatke domene u obliku skupa vrhova i bridova koji ih povezuju. Oni podaci koji predstavljaju entitete odnosno objekte, u grafu postaju vrhovima, dok su veze među njima opisane bridovima. Također svaki vrh posjeduje vlastiti jedinstveni identifikator.

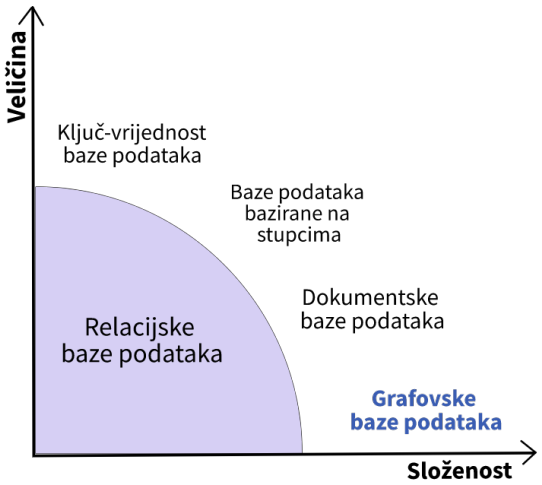
Prema [39], pod grafovske baze podataka se smatraju one tehnologije koje se koriste primarno za transakcijski integritet podataka prikazanih strukturom grafa i kojima se najčešće pristupa direktno iz aplikacija u stvarnom vremenu. One su ekvivalentne OLTP (engl. *Online Transaction Processing Systems*) sustavima u relacijskom svijetu. S druge strane, tehnologije koje se koriste za izvanmrežnu analizu podataka prikazanih strukturom grafa, najčešće izvođenu u skupinama koraka, [39] naziva strojevima za obradu grafa koji se donekle mogu poistovjetiti sa OLAP (engl. *Online Analitical Processing*) sustavima.

Grafovske baze podataka primjenu pronalaze u raznim područjima poput društvenih i računalnih mreža, prirodnih znanosti (biologija, biotehnologija, kemija...), računalnih sigurnosti (detekcije prijevara), preporuka u stvarnom vremenu i brojnim drugim kojima priroda domene nalaže bitnost odnosa među podacima.

3.1.1 Karakteristike

Za razliku od ostalih kategorija NoSQL baza podataka, specifično je da grafovske baze podataka podržavaju transakcije i njihova ACID svojstva, to jest, izgrađene su i optimizirane za očuvanje transakcijskog integriteta i postizanje što veće dostupnosti. Iz toga proizlazi činjenica da je sustav uvijek konzistentan (osim u slučajevima kada se distribuira na više polsužitelja), a agilnost i fleksibilnost sheme karakteristična su svojstva kao i za ostale kategorije opisana u poglavlju 2.1.

Glavni razlog nastajanja i najveća prednost u odnosu na ostale jest efikasna pohrana i obrada međusobno povezanih podataka, s naglaskom na pohrani informacija o vezama među podacima (entitetima, objektima). Kao što je rečeno, NoSQL baze podataka nastaju i iz potrebe za efikasnom pohranom i obradom što većih količina podataka, no glavna okosnica grafovskih baza podataka je efikasnost za probleme složene domene u smislu podataka međusobno povezanih složenim vezama. Na sljedećoj slici slikovito je prikazan motiv za nastajanje grafovskih baza podataka u odnosu na ostale.



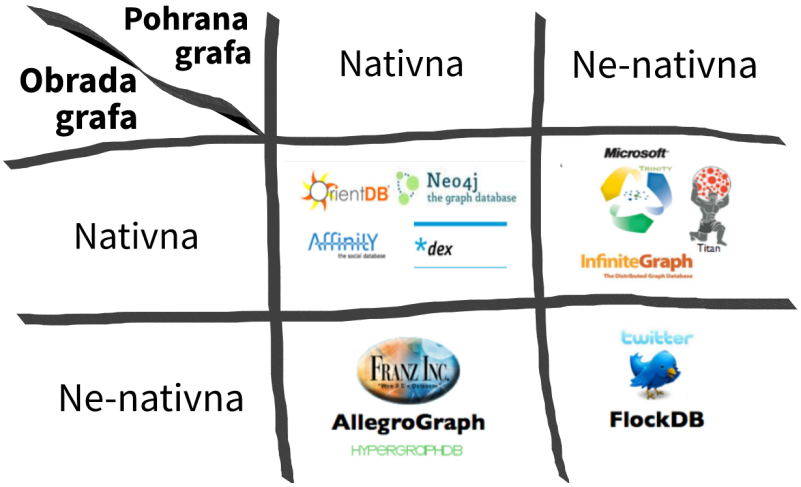
Slika 3.1: Dijagram baza podataka u ovisnosti o motivu nastanka

No sada, povezanost grafa postaje razlog slaboj horizontalnoj skalabilnosti i distribuiranosti. Primjerice, podijeliti graf na nekoj logičkoj razini na više poslužitelja, specifičan je zadatak svake domene ponaosob. Jedino što može doći u obzir je duplicirati podatke na više poslužitelja kako bi se ubrzale operacije čitanja, ali u tom slučaju izvršavanje operacija pisanja, brisanja i ažuriranja postaju zahtjevne za implementaciju.

3.1.2 Pohrana i obrada podataka

Stvorene od strana različitih proizvođača, grafovske baze podataka se često razlikuju u načinu pohrane i načinu obrade podataka. Neke koriste način pohrane koji je specijalno dizajniran i optimiziran za pohranu grafova, tzv. *nativna (autohtona) pohrana grafa* (engl. *Native graph storage*) [39], dok druge zapravo pohranjuju graf unutar relacijskih baza podataka, objektno-orijentiranih baza podataka ili nekih drugih skladišta podataka.

Ako promatramo način obrade grafa, postoje one grafovske baze podataka koje nemaju potrebe za korištenjem indeksa jer povezani vrhovi fizički već ”pokazuju” jedan na drugog. Taj se način naziva *nativna obrada grafa* (engl. *Native graph processing*) [39] i njime se pospješuju performase same baze podataka jer pogoduju bržem izvršavanju upita. No, postoje i one grafovske baze podataka koje nemaju to svojstvo i koje moraju koristiti indekse.



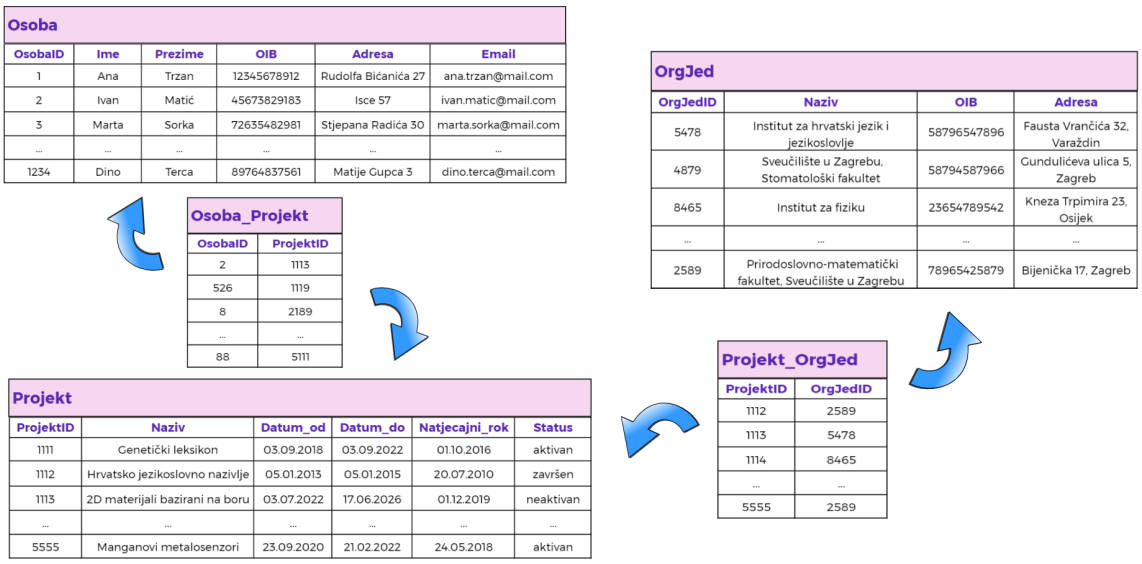
Slika 3.2: Prikaz grafovskih baza podataka u ovisnosti o načinu pohrane i obrade podataka

3.1.3 Prednosti u odnosu na relacijske baze podataka

Kako bi vjerodostojno prikazali samu svrhu i prednosti grafovskih baza podataka, najprije ćemo dati primjer modeliranja povezane domene relacijskom bazom podataka, a potom ćemo predočiti uočene poteškoće.

Na slici 3.3 dan je primjer relacijskog modela podataka koji se sastoji od 5 relacija:

- Osoba - sadrži podatke o pojedinoj osobi
- Projekt - sadrži podatke o pojedinom projektu
- OrgJed - sadrži podatke o pojedinoj organizacijskoj jedinici
- Osoba_Projekt - vezna tablica [1] kojom se prikazuje veza između osobe i projekta; predstavlja informacije o tome koja osoba je radila na kojem projektu
- Projekt_OrgJed - vezna tablica koja prikazuje vezu između projekta i organizacijske jedinice u sklopu koje je projekt rađen



Slika 3.3: Primjer relacijskog modela podataka

Odnosi među podacima u relacijskim bazama podataka se opisuju stranim ključevima direktnim navođenjem u sklopu neke relacije ili čak zasebnim relacijama koje se sastoje

[1] Vezne tablice predstavljaju veze "mnogo-naprama-mnogo"

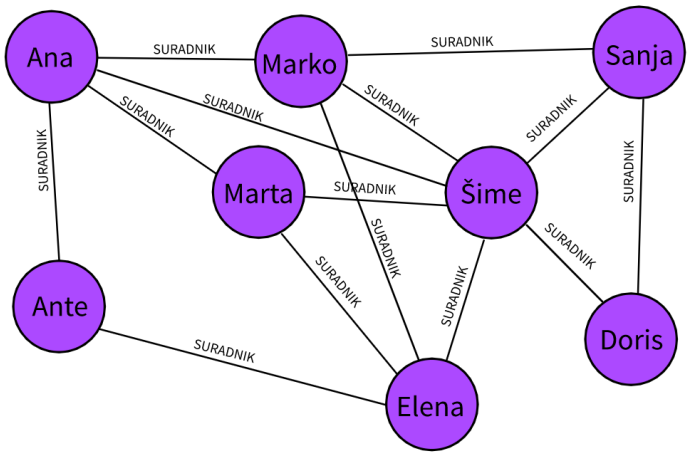
isključivo od stranih ključeva (kao vezne tablice na slici 3.3), sve kako bi se zabilježile veze među podacima.

Tada, ako bismo, primjerice, htjeli dohvatiti sve projekte u kojima je neka osoba sudjelovala, potrebno je izvršiti operaciju pridruživanja (engl. *join*) koja posredstvom vezne tablice Osoba_Projekt dohvaća podatke o projektima za točno određenu osobu. Već ako želimo dohvatiti i organizacijske jedinice u sklopu kojih je određena osoba sudjelovala u projektima, dodatno se komplicira i posljedično usporava izvođenje operacije pridruživanja jer u posredstvu se sada nalaze već dvije vezne tablice koje određuju odnose. Još složeniji i skuplji postaju recipročni upiti poput: dohvata svih osoba koje su pisale projekte u sklopu određene organizacijske jedinice.

Slijedeći navedeno, može se zaključiti kako se u relacijskim modelima podataka odnosi među podacima otkrivaju pretraživanjem tablica pa s povećanjem količine podataka i/ili povećanjem povezanosti među njima, raste i vrijeme potrebno za izvršavanje operacija pridruživanja. Svi gore navedeni primjeri dohvata podataka, temeljni su upiti koje zahtjeva bilo koji izvještajni sustav, iz čega proizlazi činjenica da za ovakve domene, relacijske baze podataka ne nude efikasno rješenje.

Kako to rješavaju grafovske baze podataka, poprilično je intuitivno jer se konkretne i apstraktne veze među podacima prikazuju bridovima grafa. Izvođenje svih operacija odnosi se na dio grafa zahvaćen samim upitom, dakle brzinom proporcionalnom veličini tog podgrafa, a ne na cijeli graf, što poprilično ubrzava njihovo izvođenje.

Pogledajmo još jedan primjer, temom nastavnom na prethodni, sada modeliran grafom (slika 3.4).



Slika 3.4: Znanstvene suradnje - model grafa

Prikazati sve znanstvenike (osobe) koje surađuju sa znanstvenicom Anom (spajanje relacije Osoba, sa slike 3.3, same sa sobom - prva razina rekurzije), efikasno bi se postiglo i relacijskim bazama podataka posredstvom eventualno jedne vezne tablice koja prikazuje odnose odnosno suradnje između znanstvenika. Važno je napomenuti kako u tom slučaju, pripadna vezna tablica opisuje samo jednosmjernu vezu, primjerice informaciju da je Anin suradnik Ante, ali ne i informaciju da je Ana Antin suradnik. Ako želimo ići dublje po mreži i prikazati sve osobe koje surađuju sa Aninim suradnicima (druga razina rekurzije), tada već slabe performase relacijskih baza podataka jer je potrebno više puta ući u rekurziju (pretražiti istu tablicu Osoba) kako bi se operacijom pridruživanja dobilo što se traži. Alternativno, moguće je "dopuniti" veznu tablicu suradnji na način da prikazuje dvostrane veze, primjerice za opisati suradnju između Ane i Ante, sada bi postojala 2 retka u veznoj tablici.

S povećanjem razina rekurzije, razlika u brzini dohvata grafovske u odnosu na relacijsku bazu podataka postaje neusporediva. Autori Watt, Vukotić, Abedrabbo i Fox u svojoj su knjizi *Neo4j in action* [41] prikazali rezultate izvođenja upita nad primjerom društvene mreže (osobe i njihova prijateljstva) u relacijskoj odnosno grafovskoj bazi podataka (Neo4j) kako bi se prikazale njihove performanse ovisno o dubini tj. razini rekurzije - slika 3.5. U eksperimentu, društvena mreža sastojala se od 1 000 000 osoba od koje svaka u prosjeku ima oko 50 prijatelja. S obzirom na srodnost primjera i problematike domene, rezultati se mogu jednako promatrati i u slučaju znanstvenih suradnji.

Dubina	RBP (sekunde)	GBP - Neo4j (sekunde)	Broj vraćenih rezultata
2	0.016	0.01	~2500
3	30.267	0.168	~110 000
4	1543.505	1.359	~600 000
5	Beskonačno	2.132	~800 000

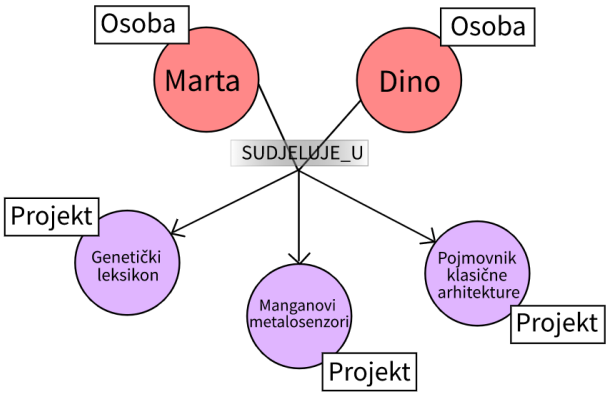
Slika 3.5: Usporedba performansi relacijske i grafovske baze podataka nad primjerom društvenih mreža

3.1.4 Modeli grafovske baze podataka

Osim načina pohrane i obrade podataka, grafovske se baze podataka mogu razlikovati i u podržavanju modela grafova. Postoji nekoliko različitih modela grafova, a neki od njih su: model grafa sa svojstvima, hipergraf i RDF model. Slijedi njihov kratki pregled.

Hipergraf

Hipergraf je generalizirani model grafa u kojem bridovi, takozvani *hiper-bridovi*, mogu povezivati proizvoljan broj vrhova [39]. Takav pristup dobar je u slučajevima kada domena sadrži mnogo veza s funkcionalnošću "mnogo-naprama-mnogo" (engl. *many-to-many*). Generalizacijom bridova postižu se informacijama bogati modeli, no u praksi se time često ispuštaju neki detalji. Hipergrafovi su izomorfni sa modelima grafova sa svojstvima, u smislu da se jedan može reprezentirati preko drugog, uz povećanje odnosno smanjenje broja bridova među njima, ovisno koji model se želi dobiti.

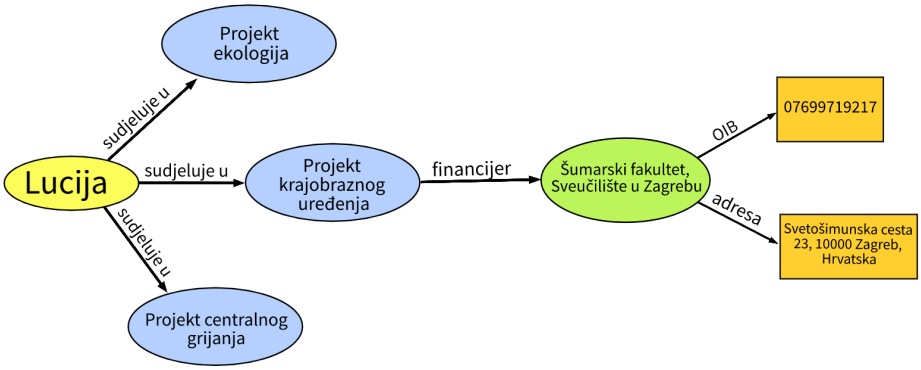


Slika 3.6: Slikoviti prikaz koncepta hipergrafa na primjeru sudjelovanja na projektima

RDF model

RDF (engl. *Resource Description Framework*) je model koji omogućava opisivanje resursa, najčešće informacija na Webu, trojkama subjekt-predikat-objekt. Subjekt predstavlja resurs, odnosno entitet koji se opisuje, predikat označava svojstvo subjekta, a objekt vrijednost tog svojstva. Promatrajući ih zasebno, trojke semantički ne znače mnogo i predstavljaju samo činjenice, no u mnoštvu pružaju bogat skup podataka kojim se zaključuju veze među informacijama i postiže znanje. RDF je zapravo običan tekst koji može biti zapisan u različitim formatima: trojna notacija (N3), Turtle, RDF/XML, a može se prikazati

i vizualno u obliku grafa kao na slici 3.7. Za upite nad ovim modelom najčešće se koristi SPARQL semantički upitni jezik koji je sintaksom vrlo sličan SQL-u.

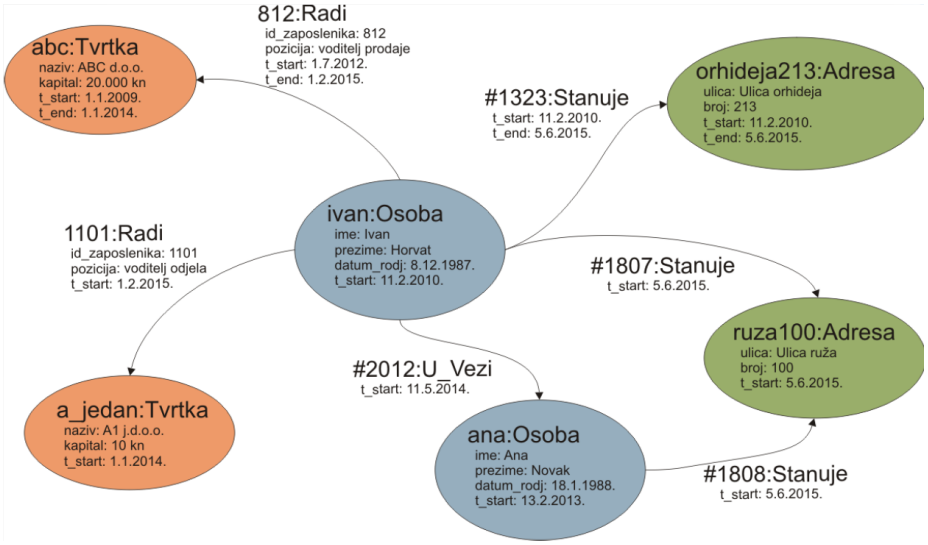


Slika 3.7: Primjer RDF modela

Model grafa sa svojstvima

Model grafa sa svojstvima temelji se na usmjerenom ili neusmjerenom grafu čiji vrhovi i bridovi imaju oznake (labele), a mogu imati i svojstva. Karakteristike su sljedeće:

- Svojstva vrhova odnosno bridova su oblika naziv-vrijednost pri čemu nazivi i vrijednosti mogu biti različiti tipovi podataka ovisno o implementaciji. Broj svojstava pojedinog vrha odnosno brida je proizvoljan.
- Ako oznake promatramo kao tipove vrhova, tj. bridova, tada one implicitno služe za njihovo grupiranje odnosno klasificiranje u svrhu lakšeg pretraživanja i obrade baze podataka. Vrhovi, za razliku od bridova, mogu imati više oznaka.
- Bridovi u suštini grade strukturu grafa, a opisani svojim smjerom, oznakom, svojstvima, početnim i završnim vrhom, dodatno semantički obogaćuju vrhove i cijeli graf. Za razliku od hipergrafova, jedan brid može povezivati isključivo dva vrha. Dodavanje svojstava bridovima korisno je za pohranu dodatnih meta podataka kojima se olakšava provođenje algoritama nad samim grafom, pojednostavljuju analize i obrade podataka.
- U skladu sa fleksibilnošću sheme grafovskih baza podataka, vrhovi odnosno bridovi koji su istog tipa (to jest imaju iste oznake), ne moraju nužno imati ista svojstva.



Slika 3.8: Primjer modela grafa sa svojstvima [20]

Ovaj je model najpopularniji u novijim NoSQL bazama podataka, a zbog svoje intuitivnosti u posljednje se vrijeme najviše i koristi.

Za potrebe ovog diplomskog rada model grafa sa svojstvima je najprikladniji jer nudi najveću fleksibilnost s obzirom na nepredvidivu domenu, stoga ćemo od sada pod grafovskom bazom podataka podrazumijevati onu koja koristi ovaj model.

3.2 Primjeri grafovskih baza podataka

S povećanjem popularnosti grafovskih baza podataka raste i broj sustava za njihovo upravljanje. Razlikujući se u načinima pohrane i obrade podataka, korištenim upitnim jezicima i modelima podataka, nude različite prednosti te nastoje održati konkurentnost na tržištu.

Neki od današnjih poznatih sustava za upravljanje grafovskim bazama podataka su: Neo4j, OrientDB, ArangoDB, DGraph, FlockDB, Cassandra, Memgraph, Titan i brojni drugi. Slijedi kratki pregled nekih od njih.

OrientDB

OrientDB je NoSQL sustav za upravljanje bazama podataka, otvorenog koda i pisan u Java programskom jeziku. Prva je baza podataka koja podržava više modela: dokument, ključ-vrijednost, objektno orijentirani i graf model podataka. No, sa svakim se modelom

upravlja kao sa grafovskim bazama podataka s izravnim vezama među zapisima. Kada je riječ o modelu grafa, koristi model grafa sa svojstvima te nativnu pohranu i obradu grafa. Podržava brojne programske jezike poput: .Net, C, C#, C++, Java, JavaScript, Node.js, Python, PHP i drugi, a pohranjivati može i do 120.000 zapisa u sekundi. Dostupan je u dva izdanja - besplatnom (engl. *Community*) i komercijalnom (engl. *Enterprise*) te podržava upitni jezik Gremlin ([12]).

Neo4j

Neo4j jedan je od vodećih sustava za upravljanje grafovskim bazama podataka. Otvorenog je koda i pisan u programskim jezicima Java i Scala. Razvoj Neo4j-a započet je 2003. godine, a u produkciji je od 2007. godine te od tada postaje sve popularniji. Dostupan je u besplatnom i komercijalnom izdanju. Kao i OrientDB, Neo4j također je primjer grafovske baze podataka koja koristi nativnu pohranu i obradu podataka te model grafa sa svojstvima. Neo4j tvrtka osnivač je i upitnog jezika Cypher koji se koristi za obradu i analizu podataka grafa i koji je postao vodeći podržavani jezik od strane raznih sustava za upravljanje grafovskim bazama podataka. Više o upitnom jeziku Cypher može se pronaći na [4].

Memgraph

Kao odgovor na slabe performanse i nedostatke dosadašnjih sustava za upravljanjem grafovskih baza podataka, 2016. godine nastaje novi sustav, Memgraph. Inicijalno stvoren kao in-memory grafovska baza podataka s odličnim performansama jednostavnih upita nad grafom, napredovanjem se ostvaruje i kao sustav za provođenje različitih analitika nad grafom u stvarnom vremenu. Pisan je u C++ programskom jeziku te je dostupan u besplatnom i komercijalnom izdanju. Podržava upitni jezik Cypher, a razni algoritmi nad grafovima opisani su procedurama koje proširuju Cypher te čiji kod je otvoren i smješten u javni repozitorij. Memgraph je mlad i moderan sustav u razvoju.

Poglavlje 4

Migracija podataka

Popularizacijom grafovskih baza podataka od interesa postaje migrirati podatke iz neke druge baze podataka u grafovsku kako bi se koristile sve njene prednosti. Mnogo sustava dugi niz godina koristi relacijske baze podataka za pohranu svojih podataka i poslovnih procesa. No, pojavom grafovskih baza podataka, brojne domene efikasnu pohranu pronalaze upravo u njima. Kako konvertirati relacijske podatke u graf postao je novi izazov i zadatak.

Postoje dvije vrste migracije podataka: generička i specifična. Koncept svake ponaosob opisan je u sljedećim poglavljima.

4.1 Generička migracija podataka

Generička migracija podataka odnosi se na definiranje skupa pravila prema kojima bi se odvijala migracija bilo koje relacijske baze podataka u grafovsku, neovisno o specifičnostima domene i karakteristikama same baze podataka. Definirati generalni skup pravila može se ostvariti na više načina, no ono što je svima zajedničko je težnja za postizanjem neovisnosti o semantici baze podataka te migracija bez gubitka podataka. Temeljem tako unaprijed definiranih pravila, podaci iz relacijske baze podataka konvertiraju se u podatke grafa te je konverzija neovisna o promjenama sheme relacijske baze podataka. S druge strane, mogućnost je da zbog neovisnosti o semantici generirani graf neće biti optimiziran za željene primjene i/ili sama migracija možda neće biti i najbrže rješenje (primjerice u slučajevima kada je relevantan samo dio podataka), no migraciju je moguće provesti za svaku relacijsku bazu podataka.

Kako postoji više načina za migraciju, tako postoje već neki predloženi generički algoritmi za kreiranje grafova koji podržavaju različite modele grafa.

Najčešće se prate sljedeća tri pravila:

1. Svaki redak u tablici relacijske baze podataka predstavlja vrh u grafu.
2. Naziv tablice u relacijskoj bazi podataka postaje oznaka vrha/brida u grafu.
3. Strani ključevi u relacijskoj bazi podataka postaju bridovi grafa.

No, različiti modeli grafa nastaju iz različitih načina prikaza atributa entiteta te podržavanju svojstava vrhova i bridova. Primjeri nekih od njih su sljedeći:

- Model grafa bez svojstava:

  Svaki redak tablice i svaki pripadni atribut postaju međusobno povezani vrhovi u grafu, a svaki strani ključ postaje brid. Ovakav se model koristi u pristupu poput RDB2Graph [38] te kod ovakvih modela vrhovi i bridovi nemaju svojstva. Algoritmi koji ih podržavaju, ne podržavaju kompozitne strane i primarne ključeve.

- Model grafa sa svojstvima vrha:

  Svaki redak postaje vrh, a svaki pripadni atribut svojstvo tog vrha. Svaki strani ključ postaje brid. Ovakav model koristi se u pristupu poput R2G [33] te kod ovakvih modela bridovi nemaju svojstva.

- Model grafa sa svojstvima vrhova i bridova:

  Svaki redak postaje vrh, a svaki pripadni atribut svojstvo tog vrha. Svaki redak vezne tablice postaje brid i svaki pripadni atribut svojstvo tog brida. Također, svaki strani ključ postaje brid. Ovakav se model koristi u pristupu poput [37] te kod ovakvih modela potpuno je izložen model grafa sa svojstvima.

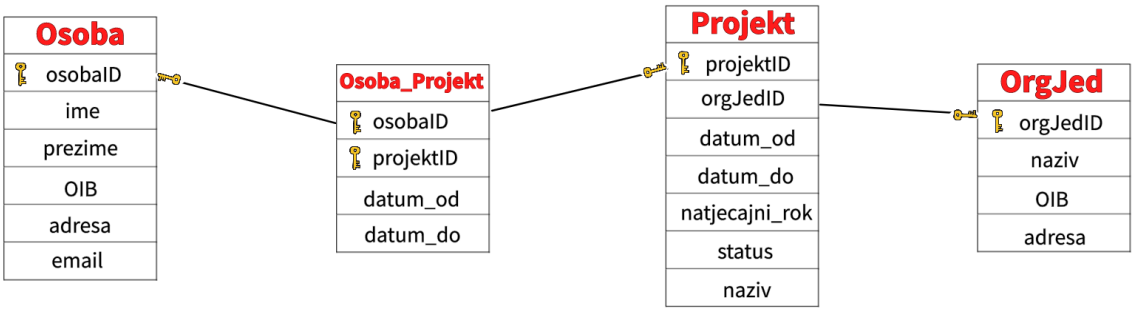
4.2 Specifična migracija podataka

Za razliku od generičke, specifična migracija podataka uglavnom se koristi za specifičnu upotrebu i točno određenu domenu. Poznavanjem domene relacijske baze podataka, točno se određuje što i kako će se konvertirati u graf te je moguće izvršiti određene naredbe kojima se podaci izvlače iz relacijske i unose u grafovsku bazu podataka. Definiranjem i kontrolom takvih naredbi otvara se prostor za optimiziranje i modeliranje grafa po željenom izgledu, ali ostaje prisutna ovisnost o promjenama sheme relacijske baze podataka. Ova vrsta migracije može biti znatno brža od generičke, no zbog velike ovisnosti o domeni, ne predstavlja rješenje za migriranje bilo koje relacijske baze podataka.

Određivanje načina prikaza tablica i redaka kao vrhova i bridova jako može ovisiti o poslovnim potrebama, no često se slijede neke općenite smjernice navedene i u poglavlju 4.1:

1. Redak je vrh.
2. Naziv tablice je oznaka.
3. Retci u veznoj tablici i strani ključevi su bridovi.

Kao primjer kojim ćemo prikazati koncept specifične migracije uzeti ćemo malo izmjenjeni relacijski model sa slike 3.3 - slika 4.1.

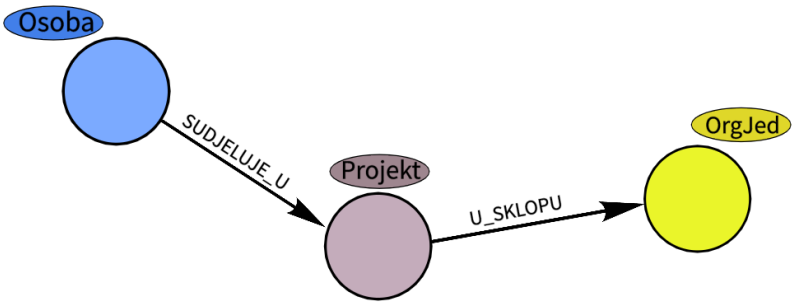


Slika 4.1: Relacijski model

Konvertirati ovakav model u model grafa moguće je na sljedeći način:

- Svaki redak tablice Osoba postaje vrh oznake ”Osoba”, a pripadni atributi postaju svojstva tog vrha.
- Svaki redak tablice Projekt postaje vrh oznake ”Projekt”, a pripadni atributi postaju svojstva tog vrha.
- Svaki redak tablice OrgJed postaje vrh oznake ”OrgJed”, a pripadni atributi postaju svojstva tog vrha.
- Svaki redak tablice Osoba_Projekt postaje brid oznake ”SUDJELUJE_U”, a pripadni atributi postaju svojstva tog brida.
- Veza između tablica Projekt i OrgJed (strani ključ orgJedID u tablici Projekt) postaje brid oznake ”U_SKLOPU”.

Izgled modela grafa bez popisa pripadnih atributa (radi jednostavnosti slike), prikazan je na sljedećoj slici:



Slika 4.2: Model grafa

Nakon određivanja načina konverzije, migracija se može jednostavno provesti nizom prikladnih naredbi. Primjerice, u [28] podaci se iz relacijske baze podataka najprije izvoze u više CSV datoteka (za svaki tip vrhova odnosno bridova po jedna datoteka), a potom se Cypher naredbama čitaju njihovi podaci i mapiraju u odgovarajuće vrhove i bridove, kreiraju se indeksi i postavljaju ograničenja.

Poglavlje 5

Vizualizacije grafova

Promatranjem grafovske baze podataka kao izvještajnog sustava, uočavamo neosporan utjecaj korisničkih iskustava, interakcija i zahtjeva nad samim izvještajnim sustavom. S obzirom na to da većinu informacija opažamo vizualno te po prirodi težimo korištenju i lakšem razumijevanju vizuala, stvorio se interes za izgradnjom raznih alata za vizualizaciju. Kako sâm koncept grafa kao skupa vrhova i bridova intuitivno stvara vizualnu predodžbu, a s ciljem ostvarenja pristupačne, brze i jednostavne pretrage i analize podataka, razvili su se i brojni interaktivni alati za vizualizaciju grafova i grafovskih baza podataka.

5.1 Alati za vizualizaciju

Većina alata za vizualizaciju grafova i grafovskih baza podataka može se svrstati u tri arhitekturalne kategorije: alati bez direktnog spajanja na bazu podataka, alati s direktnim spajanjem na bazu podataka i samostalni alati.

1. **Alati bez direktnog spajanja na bazu podataka**

   Pod ovu vrstu alata najčešće spadaju biblioteke koje omogućuju ugradnju vizualizacije grafa unutar aplikacije bez direktnog spajanja na grafovsku bazu podataka.

   Vizualizacija može biti sačinjena od podataka poslanih preko aplikacijskog programskog sučelja aplikacije koja je spojena na bazu podataka. Tako se osigurava da klijentska aplikacija, dakle ona koja sadržava vizualizaciju, ne komunicira direktno s bazom podataka pa se time implicitno ostvaruje dodatan zaštitni sloj.

   Također često je potrebno transformirati i izvoditi podatke iz grafovske baze podataka u format kojeg biblioteka prihvaća.

Velika većina takvih biblioteka su upravo JavaScript biblioteke poput: D3.js, Vis.js, Sigma.js, Vivagraph.js, Cytoscape.js i druge.

2. **Alati s direktnim spajanjem na bazu podataka**

   Ove se vrste alata mogu uključiti u aplikaciju kao dodatna biblioteka (engl. *dependency*) te konfigurirati na način da su spojene direktno na instancu grafovske baze podataka. Sukladno tome, vizualizacija može biti i dio korisničkog sučelja aplikacije, a zbog direktnog spajanja, aktualnost podataka nije upitna.

   Posljedično, treba imati na umu da se klijentska aplikacija spaja direktno na bazu podataka, što nije uvijek poželjna arhitektura. Naime, dozvoljavajući pristup samoj bazi podataka "izvana", korisnicima aplikacije, dozvoljeno je izvršavanje raznih upita na bazu podataka koji između ostalog mogu biti i destruktivni.

   Neki od takvih alata, također JavaScript biblioteke, su primjerice Neovis.js i Popoto.js.

3. **Samostalni alati**

   Alati osmišljeni kao samostalne aplikacije mogu se povezati s grafovskom bazom podataka i operirati pohranjenim podacima bez potrebe za pisanjem koda.

   Ovakvi su alati osmišljeni ponajviše za poslovne analitičare, znanstvenike koji se bave podacima, menadžere i druge korisnike u svrhu interakcije sa grafovima i grafovskim bazama podataka.

   Neki od takvih alata su: Neo4j Bloom, Gephi, GraphXR, yFiles, Linkurious, Keylines.

Od širokog izbora alata, u ovom ćemo se radu orijentirati na one koji su najprikladniji kriterijima sukladnim temi samog rada, kao i inicijalnim željama i očekivanjima od same vizualizacije. Prvo su izloženi određeni kriteriji za odabir alata za vizualizaciju, a potom slijedi pregled i opis nekih od alata otvorenog koda (engl. *open source*) koji su detaljnije istraženi i isprobani.

5.1.1 Kriteriji za odabir alata za vizualizaciju

Definiramo kriterije temeljem kojih ćemo odabrati adekvatne alate za vizualizaciju, koje ćemo istražiti u ovom radu, a to su: izgled vizualizacije, slojevitost i performanse.

1. **Izgled vizualizacije** - mogućnosti oblikovanja i bojanja vrhova, bridova, oznaka i svojstava, podešavanje veličine vrha odnosno debljine brida u odnosu na određeno svojstvo, izbor dostupnih algoritama, primjerice za grupiranje (klasteriranje, engl. *clustering*) i slično.

2. **Slojevitost** - kompleksnost povezivanja baze podataka s alatom za vizualizaciju, odnosno broj slojeva potrebnih za prijenos podataka iz baze podataka u alat.

3. **Performanse** - količina podataka koju alat može podnijeti, brzina i kvaliteta iscrtavanja grafa.

Alati za vizualizaciju često pružaju razne formate za oblikovanje danih podataka poput vremenskih traka, mreže, 2D, 3D grafova i drugih. U slučaju grafovske baze podataka, format mreže je najčešći izbor, stoga se često govori o mrežnoj vizualizaciji (engl. *network visualization*) koja predstavlja mrežu sastavljenu od vrhova i bridova.

S obzirom da se bavimo grafovskim bazama podataka i format mreže nam vizualno i konceptualno najviše odgovara, posebno ćemo se osvrnuti na mrežne vizualizacije. Kako nam je u ovom radu cilj izgraditi i web aplikaciju sa prikazom vizualizacije, od interesa su oni alati koji omogućuju prikaz vizualizacije u preglednicima. To su najčešće JavaScript biblioteke.

Važno je naglasiti da se alati često razlikuju i u podržavanim tehnologijama za prikaz i animiranje grafika, a to su SVG [1], Canvas [2] i WebGL [3]. Njihove performanse ovise o veličini i uređenosti podataka koje treba vizualizirati pa predstavljaju važan kriterij za dobru vizualizaciju, u ovisnosti o danim podacima. Pogotovo ako promatramo velike grafove (više od 1.000 vrhova i više od oko 10.000 bridova), ove su tehnologije gotovo isključive. Na slici 5.1 nalaze se karakteristike rada i specifikacije svake pojedine tehnologije za prikaz nad velikim grafom koji ima više od 1.000 vrhova. Potom na slici 5.2 slijedi prikaz nekih od alata uz tehnologije za prikaz koje podržavaju.

[1] SVG (engl. *Scalable Vector Graphics*) je standard za predstavljanje 2D vektorske grafike.

[2] Canvas je HTML5 element koji se koristi za crtanje grafike.

[3] WebGL (engl.*Web Graphics Library*) je standard, točnije, JavaScript aplikacijsko sučelje za prikazivanje interaktivnih 2D i 3D grafika.

	SVG	Canvas	WebGL
Kompatibilnost sa preglednicima	Svi (čak i stariji preglednici)	Svi	Noviji preglednici
Faze prikazivanja	Stvaranje SVG-a -> parsiranje u pregledniku -> prikazivanje u pregledniku	JS kod -> prikazivanje u pregledniku	JS kod -> OpenGL prikazivanje
Izvođenje - za prikaz velikih grafova (više od 1000 vrhova)	Sporo	Normalno	Efikasno

Slika 5.1: Karakteristike SVG, Canvas i WebGL tehnologija za prikaz velikog grafa

Ime	Prikaz
d3.js	SVG
Vis.js	Canvas
Neovis.js	Canvas
Vivagraph.js	SVG, WebGL
Sigma.js	SVG, Canvas, WebGL
Cytoscape.js	SVG, WebGL

Slika 5.2: Prikaz nekih alata za vizualizaciju zajedno s tehnologijama za prikaz i animiranje grafika koje podržavaju

Slijedeći navedene kriterije, brojni su alati upali u uži izbor za odabir. Velika većina njih ima dobre specifikacije i nudi dobra rješenja za opisane kriterije. Kako bismo ipak odabrali nekoliko njih koje ćemo detaljnije istražiti, odlučili smo pokriti što različitije specifičnosti alata pa smo tako odabrali sljedeće:

- Neovis.js zbog svoje jednostavnosti uz cilj testiranja njegovih performansi
- Vis.js jer predstavlja bazu za Neovis.js

- Sigma.js koja nudi veći izbor tehnologija za prikaz
- Gephi koji kao samostalan alat podržava velik broj podataka

Primijetimo pritom da odabrani alati pripadaju različitim arhitekturalnim kategorijama, čime smo pokrili i specifičnosti pojedine kategorije te time ostvarili bolji pregled i uvid u mogućnosti širokog izbora alata.

U narednim ćemo poglavljima dati detaljniji prikaz odabranih alata za vizualizaciju.

5.1.2 Vis.js

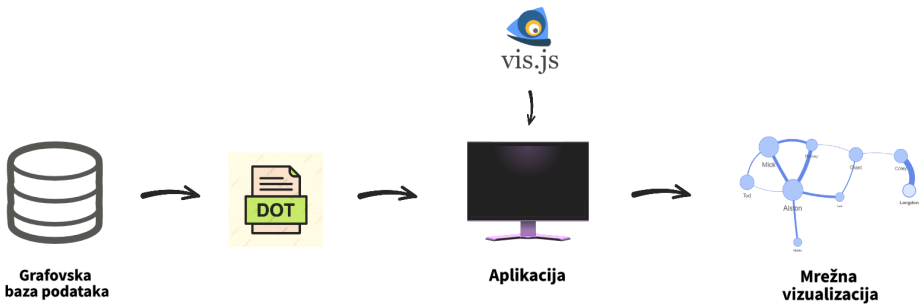
Kao što je navedeno, Vis.js je alat za vizualizaciju skupova podataka bez direktnog spajanja na bazu podataka. Štoviše, to je JavaScript biblioteka koja pruža razne vizualizacije namijenjene kompleksnim i dinamičnim skupovima podataka. Sadržava razne formate za oblikovanje danih podataka pa tako postoje vremenske trake, 2D i 3D grafovi, mreže i drugi.

Mrežna vizualizacija Vis biblioteke radi dobro na svim modernim preglednicima i to za nekoliko tisuća vrhova i bridova, a za prikaz odnosno iscrtavanje koristi se HTML Canvas. U slučaju jako velikih grafova, postoje opcije grupiranja podataka kako bi prikaz bio pregledniji i analiza podataka jednostavnija. Također, mnogo je opcija za interaktivno korištenje mrežne vizualizacije pa je lako implementirati bilo kakvu radnju vezanu uz događaj primjerice klika mišem. Tu su još i razne mogućnosti za oblikovanje vrhova i bridova, poput bojanja, mijenjanja veličine, dodavanja animacija i slično.

Kao i kod ostalih alata iste arhitekturalne kategorije, Vis prihvaća samo određene formate datoteka u koje je potrebno izvesti podatke iz grafovske baze podataka. Tako za mrežnu vizualizaciju podaci za uvoz mogu doći u obliku DOT [4] jezika ili u obliku JSON [5] datoteke izvezene iz Gephi programa (Gephi je također jedan od alata za vizualizaciju grafova).

[4] DOT je opisni jezik za grafove.

[5] JSON (engl. *JavaScript Object Notation*) je standardni tekstualni format za predstavljanje strukturiranih podataka na temelju sintakse JavaScript objekta.



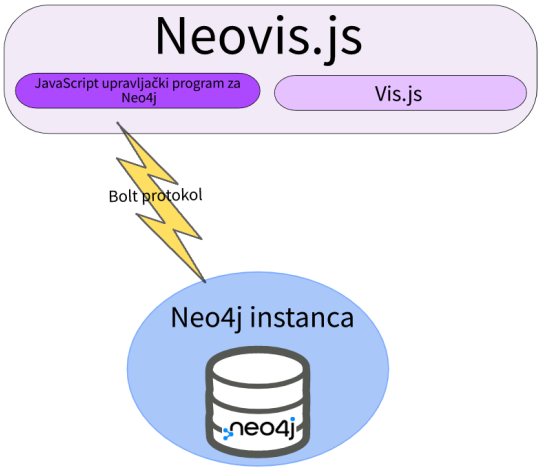
Slika 5.3: Shema postupka kreiranja mrežne vizualizacije pomoću Vis.js biblioteke s podacima u DOT formatu

5.1.3 Neovis.js

Kako bi vizualizacije grafovskih baza podataka učinili što jednostavnijim, Neo4j tvrtka je, kao jedan od svojih projekata, stvorila biblioteku Neovis. Neovis je osmišljen kao integracija JavaScript vizualizacije s Neo4j grafovskom bazom podataka. Kao što samo ime govori, Neovis je u suštini kombinacija Vis.js i Neo4j-a, točnije JavaScript upravljačkog programa za Neo4j (engl. *Neo4j driver*). Pomoću upravljačkog programa za Neo4j se obavlja spajanje s bazom podataka (Bolt protokolom) i dohvat podataka, a Vis.js služi za prikaz mrežne vizualizacije (drugi formati nisu podržani). No, iako je temeljen na Vis.js biblioteci, nije moguće koristiti sve njene funkcionalnosti.

Direktno spajanje na Neo4j bazu podataka jednostavno je i zauzima svega nekoliko linija koda, dok je kroz jedan konfiguracijski objekt moguće oblikovati i uređivati vrhove, oznake vrhova, svojstva i bridove. Vizualizacija nastaje uz minimalno korištenje JavaScript koda pa se lako može ukomponirati u neku već postojeću aplikaciju, a jezik Cypher se može i ne mora koristiti u kodu.

Neovis podrazumijeva Neo4j-ev model grafa sa svojstvima pa su imenovanje i formati podataka usklađeni s onima u bazi podataka. Također, moguće je izvršavati razne algoritme nad podacima poput centralnosti (engl. *centrality*), grupiranja, rangiranja stranica (engl. *page rank*) i drugih. Oni se izvršavaju pomoću Neo4j-eve biblioteke za algoritme nad grafovima, dakle direktno unutar Neo4j-a, a rezultati tih algoritama se onda mogu koristiti unutar konfiguracijskog objekta Neovis-a kako bi prikaz i analiza podataka bio pregledniji, funkcionalniji i jednostavniji. Prema Vis biblioteci, za prikaz i iscrtavanje mreže koristi se



Slika 5.4: Shema komponenti Neovis biblioteke i povezanost s bazom podataka

HTML Canvas, pa postoje izazovi za prikaz većih grafova.

Neo4j tvrtka i dalje radi na unapređivanju biblioteke, iako u manjem obujmu, pa nije za očekivati velike promjene u funkcionalnostima u odnosu na one do sada implementirane. S druge strane, manja dokumentacija napisana u sklopu izvornog GitHub projekta, uz nekoliko dobro napisanih blogova, dobar su i jednostavan uvod u korištenje ovog alata.

5.1.4 Sigma.js

Među alate za vizualizaciju bez direktog spajanja na bazu podataka spada i JavaScript biblioteka Sigma.js. Poznata po svojoj vrlo proširivoj okolini, nudi razne dodatke i biblioteke koje je moguće uključiti za stjecanje dodatnih mogućnosti. Korištenjem velikog izbora dodataka moguće je od jednostavne vizualizacije stvoriti bogatu interaktivnu vizualizaciju. Posljedično, to uključuje više programerskog znanja i kodiranja.

Za iscrtavanje vrhova i bridova mrežne vizualizacije, podržava korištenje SVG, Canvas i WebGL tehnologije pa pokriva veći spektar korisničkih potreba. Oblikovanje vrhova, bridova, svojstava i oznaka, interaktivnost u smislu reagiranja na događaje poput klika mišem, podržavanje algoritma za grupiranje podataka i drugih, slične su pogodnosti poput onih u Vis biblioteci.

Podaci za uvoz moraju biti pohranjeni u jedan od dva podržavana formata: JSON ili GEXF [6]. Za svaki format postoji pripadni dodatak koji služi za uvoz, obradu i spremanje podataka unutar objekata Sigma biblioteke. Naknadno je implementiran i dodatak za

[6]GEXF (engl. *Graph Exchange XML Format*) je jezik za opisivanje struktura složenih mreža, njihovih povezanih podataka i dinamike.

Neo4j grafovsku bazu podataka koji omogućuje uvoz podataka iz Neo4j-a, no on u pozadini također dohvaća podatke u JSON formatu koji se onda obrađuje.

Za probleme velikih (više od 1.000 vrhova i više od oko 10.000 bridova) povezanih grafova, izvršavanjem raznih algoritama i korištenjem WebGL tehnologije može se postići relativno brz i pregledan prikaz. S druge strane, oskudnija dokumentacija vezana uz korištenje WebGL tehnologije, zahtjeva više koncentracije i proučavanje samog izvornog koda. Za razliku od SVG i Canvas tehnologija, Sigma ima ograničene funkcionalnosti ako se koristi WebGL.

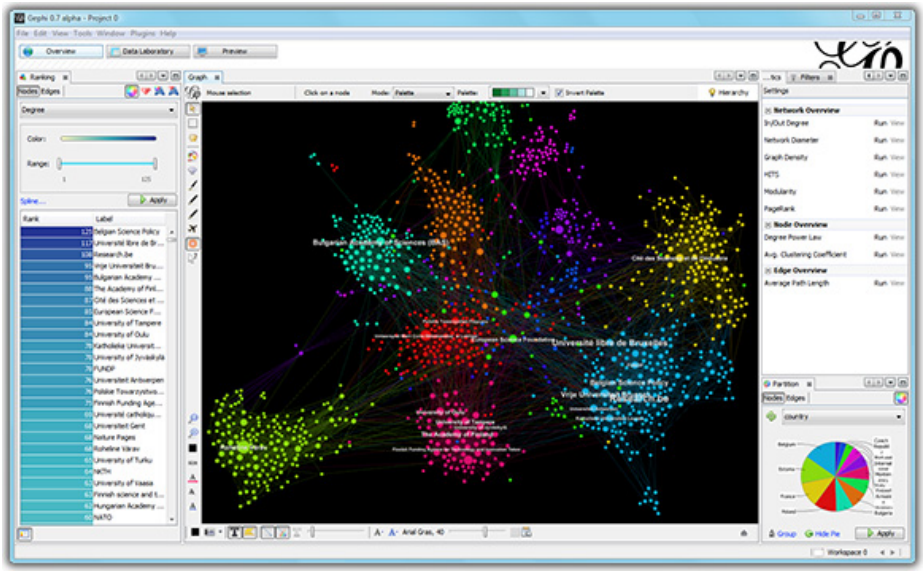
5.1.5 Gephi

Gephi je softver za vizualizaciju i istraživanje svih vrsta grafova i mreža. Kao bogata samostalna aplikacija, pomaže analitičarima da otkrivaju obrasce i trendove, ističu odstupanja i vizualiziraju svoje podatke. Pruža brojne mogućnosti za manipulaciju i analizu podataka, a samo neki od njegovih funkcionalnosti su:

- interakcija s podacima
- manipulacija strukture
- oblikovanje i bojanje vrhova i bridova
- izvršavanje algoritama za grupiranje
- filtriranje podataka
- provođenje poznatih metrika i statistika nad mrežom/grafom podataka
- manipulacija i analiza podataka prikazanih u tablici
- manipulacija i analiza podataka prikazanih vizualno u obliku mreže
- uvoz podataka u različitim formatima (GEXF, CSV, Pajek NET, GraphViz DOT, GraphML i brojni drugi)
- direktno spajanje na relacijsku bazu podataka
- direktno spajanje na Neo4j bazu podataka koristeći dodatak, štoviše radi se o strujanju podataka (engl. *streaming*)
- izvoz podataka ili mrežne vizualizacije u različitim formatima (CSV, GEXF, GraphML, PDF, SVG i drugi)
- vizualiziranje razvoja mreže/grafa podataka manipulacijom vremenske trake

Gephi je pisan u programskom jeziku Java i izgrađen je nad Netbeans platformom. Za uspješnu instalaciju potrebno je imati instaliranu Javu isključivo u verziji 7 ili 8. Instalirati se može na Windows, Linux ili Mac OS X operacijskom sustavu. Prvo izdanje Gephi-a izašlo je 2008. godine, a posljednja verzija je 0.9.2 izdana 2017.godine te od tada nema većih promjena nad svojim funkcionalnostima. Dapače, u posljednje vrijeme može se primijetiti da podrška ide znatno sporije. Za iscrtavanje i prikaz mrežne vizualizacije koristi 3D grafički standard - OpenGL, a može podnijeti mreže od oko 100.000 vrhova i 1.000.000 bridova.

Gephi je poprilično poznat alat i iza njega stoje brojni korisnici. Unatoč pojavi brojnih novih alata, i dalje je dobra opcija za velike grafove, a neki alati, poput Vis-a, za uvoz podataka traže format datoteke baš izvezen iz Gephi-a.



Slika 5.5: Izgled radnog prostora Gephi aplikacije

5.1.6 Osvrt na izbor alata za vizualizaciju

Nakon realizacije kriterija za dobru vizualizaciju, u prethodnim smo se poglavljima bazirali na četiri alata, Vis.js, Neovis.js, Sigma.js i Gephi. Svaki od njih u pravilu opravdava prvi kriterij, svi dakle imaju mogućnosti oblikovanja vrhova i bridova, posjeduju razne algoritme i imaju mogućnost podešavanja debljine i veličine bridova odnosno vrhova prema određenim svojstvima.

Krenuvši potom od drugog kriterija, nedvojbeno je da Neovis ima ”najmanju” slojevitost, odnosno nudi direktno i jednostavno spajanje na grafovsku bazu podataka. Što se tiče njegovih performansi, korištenjem HTML Canvas-a upitna je podržavana količina podataka, no jednostavno stvaranje web aplikacije ovim alatom dao je dovoljno opravdanja za odabir. U sljedećem ćemo poglavlju detaljno ćemo opisati implementaciju i izgled vizualizacije izrađene pomoću Neovis biblioteke.

Nakon Neovis-a, isprobali smo također i Sigma biblioteku. Sigma se pokazala kao alat sa mnogo dodatnih mogućnosti i funkcionalnosti, no to je zahtijevalo znatno više kodiranja. Također, WebGL učinio se kao odličan za iscrtavanje i prikaz grafa sa Sigma bibliotekom, no nailaženjem potom na znatno smanjene mogućnosti uz jako slabu dokumentaciju, bili su dovoljan razlog za prelazak na drugi alat.

Naposljetku, odlučili smo se dodatno isprobati Gephi alat koji, kao samostalna aplikacija, inicijalno nije osmišljen za uklapanje u web aplikaciju, no pruža razne mogućnosti uvoza podataka, dodatnih biblioteka te obećava dobro ”ponašanje” za velike grafove. Više o Gephi alatu govoriti ćemo u poglavlju 7.

Poglavlje 6

Implementacija izvještajnog sustava na Neo4j + Neovis.js platformi

S obzirom da je jedan od ciljeva ovog rada dati prikaz grafovskih baza podataka te način za migraciju iz relacijskih u grafovske baze podataka, prije korištenja alata za vizualizaciju, obavljena je konverzija podataka iz jedne u drugu bazu podataka na domeni pogodnoj za isticanje posebnosti i prednosti grafovskih baza podataka. Iz tog razloga implementacija izvještajnog sustava na Neo4j + Neovis.js platformi sastoji se od nekoliko koraka:

1. Određivanje ideje vizualizacije i modela grafa
2. Izvoz podataka iz relacijske baze podataka
3. Uvoz podataka u grafovsku bazu podataka (Neo4j)
4. Spajanje na bazu podataka i implementacija vizualizacije (korištenje Neovis.js biblioteke)

Svaki od koraka detaljnije su opisani u narednim poglavljima.

6.1 Tema izvještajnog sustava i ideja vizualizacije

Tema izvještajnog sustava kojeg je u cilju ovog rada implementirati, temelji se na znanstvenim suradnjama na projektima. Znanstvene suradnje predstavljaju dobar primjer povezane domene nad kojom se detaljno mogu uočiti prednosti grafovske baze podataka, a inicijalna ideja same vizualizacije bila je sljedeća:

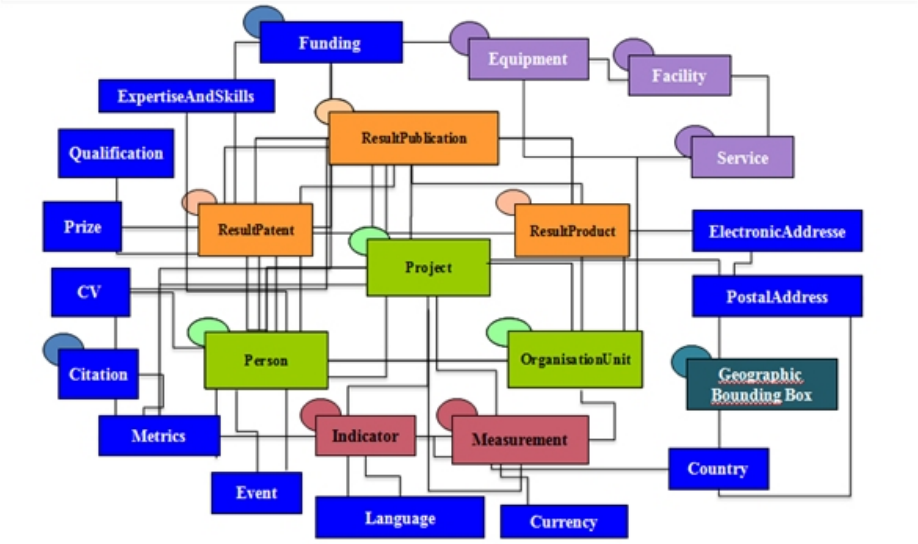
- prikazati osobe (znanstvenike) zajedno sa svojih nekoliko osnovnih podataka te veze među njima određene suradnjama na projektima

- vrhovi grafa predstavljaju osobe i njihova veličina ovisi o faktoru odjeka (engl. *Impact Factor*) znanstvenika
- bridovi grafa predstavljaju suradnju između dvije osobe i njihova debljina ovisi o broju projekata na kojima su osobe zajedno surađivale
- boja vrha ovisi o znanstvenom području kojim se znanstvenik bavi

Zamišljeno je da izvještajni sustav bude interaktivan te da prikazuje jasnu sliku domene varirajući vizualnim detaljima.

6.2 Izvor podataka

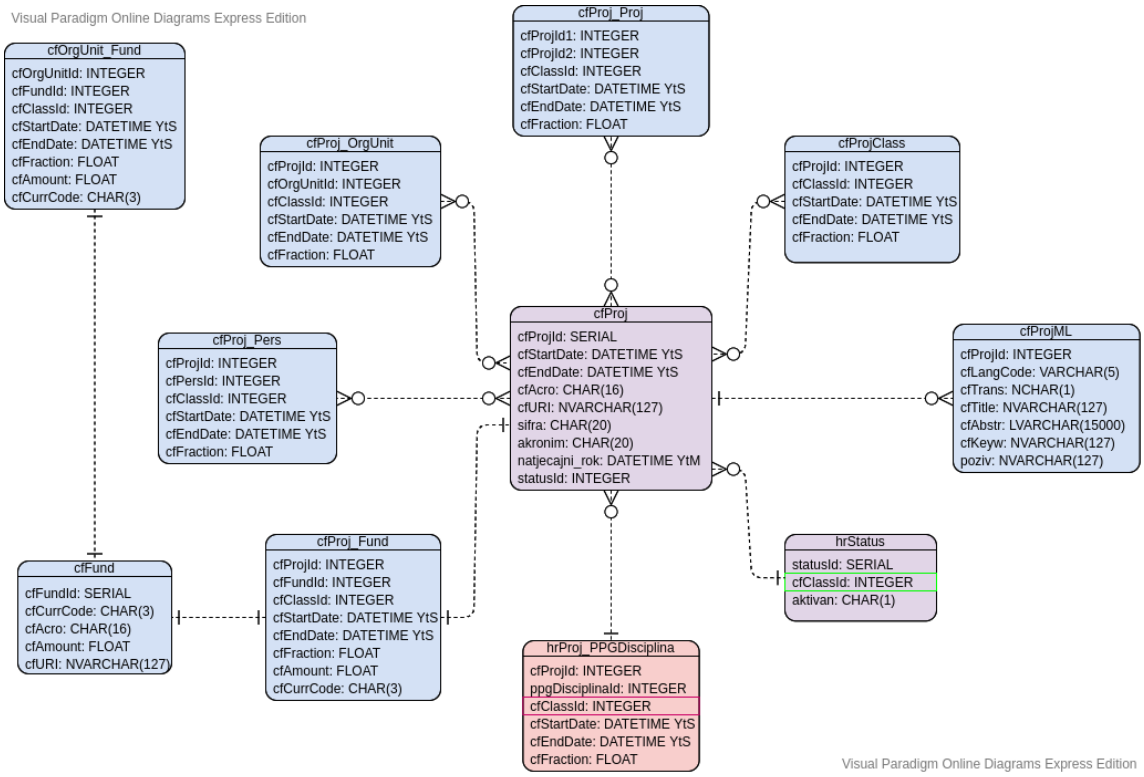
Kao izvor podataka koristi se testna baza podataka Informacijskog sustava znanosti Republike Hrvatske, skraćeno nazvanog CroRIS (engl. *Croatian Research Information System*). CroRIS je sustav u izgradnji koji objedinjuje informacije o znanstveno-istraživačkom radu u Hrvatskoj. Svoj podatkovni model CroRIS temelji na CERIF (engl. *Common European Research Information Format*) modelu. CERIF je standardni model podataka za znanstveno istraživački prostor prvobitno u vodstvu Europske unije, a potom predan euroCRIS-u - neprofitnoj organizaciji posvećenoj interoperabilnosti istraživačkih informacijskih sustava.



Slika 6.1: Shema s osnovnim entitetima CERIF modela podataka

Jedno od glavnih svojstava CERIF modela podataka relativno je mali broj entiteta, a veliki broj veza među entitetima, što je odličan temelj za korištenje grafovske baze podataka.

CroRIS koristi relacijsku bazu podataka te Informix sustav za upravljanje relacijskom bazom podataka. Za potrebe ovog rada, koristi se dio testne baze podataka CroRIS-a vezan uz projekte (slika 6.2) i osobe. Testna baza podataka sastoji se od podataka generiranih slučajnim odabirom, pri čemu je generirano 999 vrhova i 25.442 bridova.

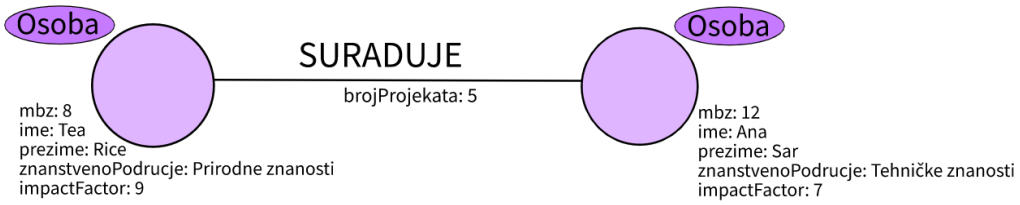


Slika 6.2: Model podataka sustava CroRIS vezan uz projekte

6.3 Migracija podataka

Provodimo specifičnu migraciju podataka čiji koncept je opisan u poglavlju 4.2 i temelji se na definiranju točno određenog načina konverzije podataka, kao i izvršavanju naredbi za migraciju za točno određenu domenu opisanu u prijašnjim poglavljima. Stoga, prije pisanja i izvršavanja naredbi za migraciju podataka, potrebno je odrediti željenu strukturu

grafa kako bi se znalo koji su podaci potrebni i na koji ih način konvertirati. No, referirajući se na inicijalnu ideju vizualizacije (poglavlje 6.1) lako dolazimo do izgleda modela grafa predstavljenog sljedećom slikom:



Slika 6.3: Struktura grafa (1)

Dakle, postoji jedna vrsta vrhova grafa - Osoba (znanstvenik) koja ima svojstva: *mbz* (matični broj znanstvenika), *ime*, *prezime*, *znanstvenoPodručje* (znanstveno područje kojim se znanstvenik bavi) i *impactFactor* (faktor odjeka), te jednu vrstu bridova - SURADUJE, s jednim svojstvom koje predstavlja broj projekata na kojima su dvije osobe zajedno surađivale.

Imajući sada na umu strukturu grafa jasno se može odrediti koje podatke iz relacijske baze podataka je nužno izvući te u koji oblik ih konvertirati pa složenost naredbi za migraciju ovisi samo o složenosti modela relacijske baze podataka.

Važno je napomenuti da specifična migracija podataka koju implementiramo usko slijedi odredbe Neo4j-og načina uvoza podataka preko CSV datoteka, stoga ćemo u narednim poglavljima opisati i prikazati naredbe prikladne pravilima tog načina migracije.

6.3.1 Izvoz iz relacijske baze podataka

Izvoz podataka iz relacijske baze podataka odvija se na način da se podaci spremaju u CSV datoteke oblika definiranog od Neo4j-a. Postoji nekoliko pravila koje je za ovu migraciju nužno slijediti:

1. Podaci koji predstavljaju vrhove određene vrste smješteni su zajedno unutar iste CSV datoteke.

2. Podaci koji predstavljaju bridove smještaju se unutar posebne CSV datoteke uz obavezno postojanje identifikatora koji predstavljaju početni i krajnji vrh.

3. Svi podaci smješteni u CSV datoteke čitaju se kao nizovi znakova, stoga je prilikom konverzije nužna pretvorba podatka u prikladni tip.

4. Atributi se odvajaju zarezom, a svaki redak datoteke predstavlja jedan vrh ili vezu (osim eventualno prvog koji predstavlja nazive atributa)

S obzirom da u našem modelu grafa postoji samo jedna vrsta vrhova i jedna vrsta bridova, podaci se izvoze u dvije CSV datoteke: *znanstvenici.csv* koja predstavlja vrhove i njihova svojstva, te *suradnje.csv* koja predstavlja bridove i njihova svojstva. Koristimo komandno-linijski alat Informix Db-access. Njega pozivamo naredbom iz ljuske operacijskog sustava (engl. *shell*) koja se periodički pokreće koristeći alat cron, a koja zapravo izvršava sljedeće naredbe za izvoz podataka:

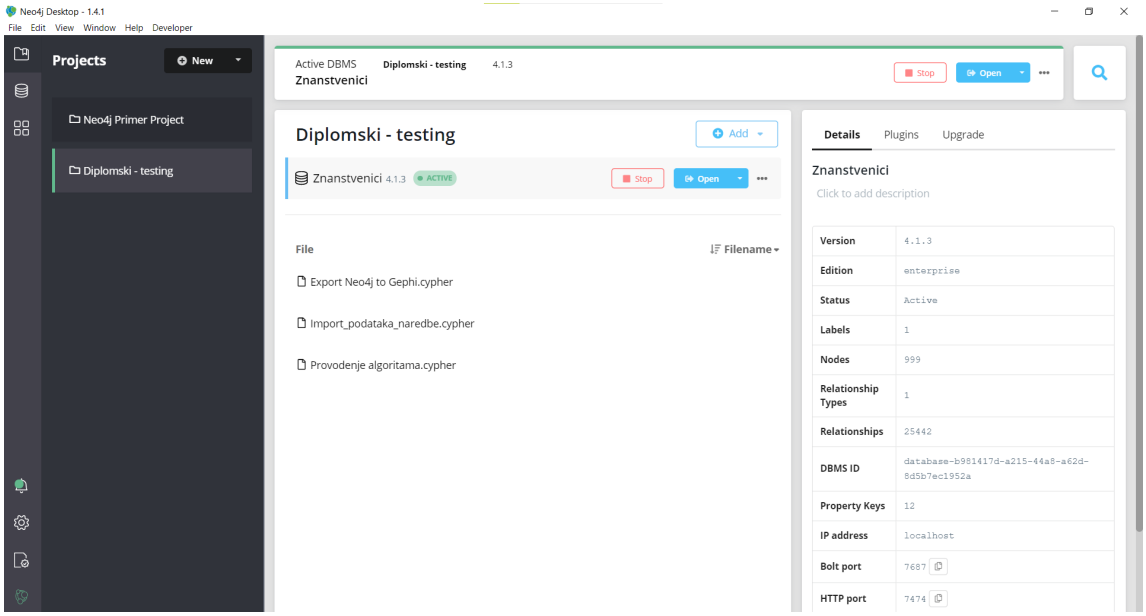
```
-- vrhovi
UNLOAD TO "znanstvenici.csv" DELIMITER ","
SELECT MBZ                        AS mbz,
       cfFirstNames               AS ime,
       cfFamilyNames              AS prezime,
       (SELECT ppgDisciplinaNaziv
          FROM hrPPGDisciplinaML
         WHERE ppgDisciplinaId = hrPers_Zvanje.ppgDisciplinaId1
           AND cfLangCode = 'hr') AS znanstvenoPodrucje,
       impactFactor
 FROM cfPers, cfPersName_Pers, cfPersName, hrPers_Zvanje
WHERE cfPers.cfPersId = cfPersName_Pers.cfPersId
  AND cfPersName_Pers.cfPersNameId = cfPersName.cfPersNameId
  AND cfPersName_Pers.cfStartDate < TODAY
  AND cfPersName_Pers.cfEndDate IS NULL
  AND hrPers_Zvanje.cfPersId = cfPers.cfPersId;


-- bridovi
UNLOAD TO "suradnje.csv" DELIMITER ","
SELECT pers1.MBZ AS mbz1,
       pers2.MBZ AS mbz2,
       COUNT(*)  AS brojProjekata
  FROM cfPers pers1, cfPers pers2, cfProj_Pers proj1, cfProj_Pers proj2
 WHERE proj1.cfProjId = proj2.cfProjId
   AND proj1.cfPersId < proj2.cfPersId
   AND pers1.cdPersId = proj1.cfPersId
   AND pers2.cdPersId = proj2.cfPersId
GROUP BY 1,2;
```

6.3.2 Uvoz u grafovsku bazu podataka

Za uvoz podataka zapisanih u CSV datotekama (znanstvenici.csv, suradnje.csv), korišten je Neo4j Desktop - sučelje prilagođeno korisniku preko kojeg je moguće jednostavno stvoriti i pokrenuti Neo4j instancu, dodavati i uklanjati razna proširenja i biblioteke, uvesti podatke te upravljati konfiguracijom same instance. Neo4j Desktop dolazi u sklopu besplatne razvojne licence Neo4j komercijalnog izdanja, a upravljati može proizvoljnim brojem projekata i baza podataka lokalno, kao i spojiti se na udaljene Neo4j servere.

Kako bi se uvezli podaci, potrebno je stvoriti novu instancu Neo4j baze podataka, zadati joj korisničko ime i lozinku te smjestiti pripadne CSV datoteke na adekvatno mjesto navođeno kroz sučelje Neo4j Desktop aplikacije. Nakon toga, instancu je moguće pokrenuti te izvršiti naredbe za uvoz odnosno konverziju podataka iz CSV datoteka u vrhove i bridove.



Slika 6.4: Sučelje Neo4j Desktop aplikacije i aktivna instanca baze podataka

Uvoz se može odraditi na više načina, no odabrano je izvršavanje naredbi za uvoz preko Neo4j Browser-a [1]. No, prije samog uvoza podataka, dobro je osigurati jedinstvenost podataka uvođenjem ograničenja. U našem slučaju jedina nužna provjera je unos osobe već postojećeg matičnog broja znanstvenika (atribut *mbz*) i to se ostvaruje naredbom:

[1] Neo4j Browser je alat za vizualizaciju i izvršavanje naredbi u jeziku Cypher nad Neo4j bazom podataka.

```
CREATE CONSTRAINT ON (o:Osoba) ASSERT o.mbz IS UNIQUE;
```

Uvoz podataka obavlja se korištenjem Cypher naredbe LOAD CSV za čitanje CSV datoteka i naredbe MERGE koja osigurava da se u slučaju postojanja vrha obavlja ažuriranje, a ne ponovni unos, čime se (radi dodatne sigurnosti) još jednom izbjegavaju duplikati. Kao što je već ranije rečeno, svi podaci iz CSV datoteka su nizovi znakova, stoga je potrebna konverzija u prikladni tip podatka.

Naredbe za uvoz podataka su sljedeće:

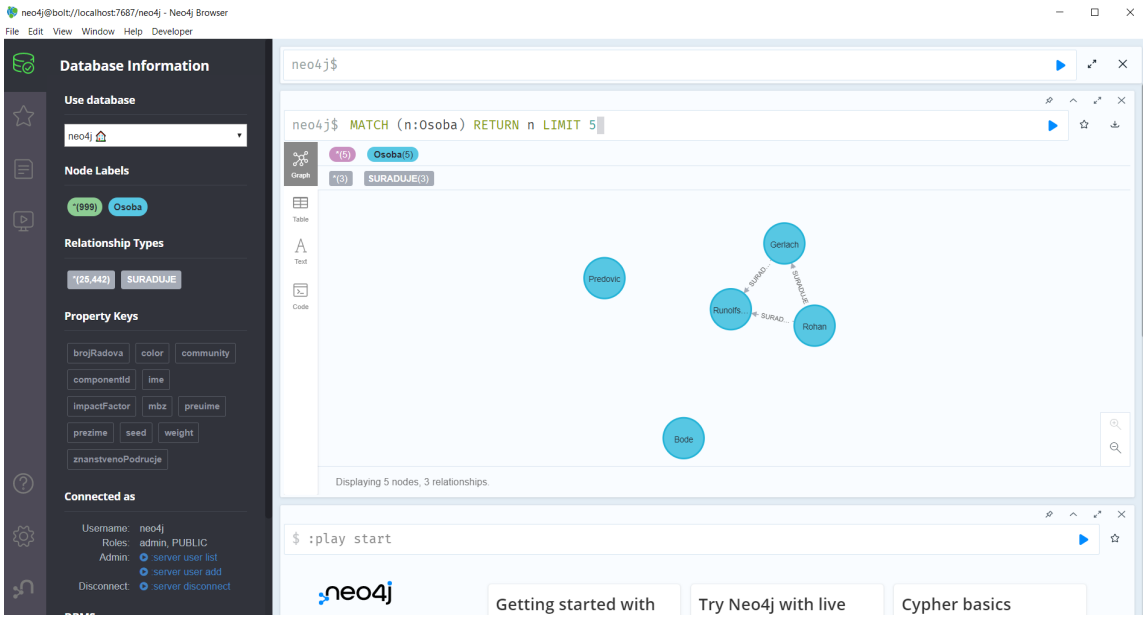
1. **Stvaranje vrhova - osobe (znanstvenici)**

```
        LOAD CSV
WITH HEADERS FROM 'file:///import-data/znanstvenici.csv' AS row
             WITH toInteger(row.mbz)                     AS mbz,
                  row.ime                                AS ime,
                  row.prezime                            AS prezime,
                  row.znanstvenoPodrucje                 AS znanstvenoPodrucje,
                  toInteger(row.impactFactor)            AS impactFactor
            MERGE (o:Osoba {mbz:mbz})
              SET o.ime                = ime,
                  o.prezime            = prezime,
                  o.znanstvenoPodrucje = znanstvenoPodrucje,
                  o.impactFactor       = impactFactor
```

2. **Stvaranje bridova - suradnje**

```
        LOAD CSV
WITH HEADERS FROM 'file:///import-data/suradnje.csv' AS row
             WITH toInteger(row.mbz1)                AS mbz1,
                  toInteger(row.mbz2)                AS mbz2,
                  toInteger(row.brojProjekata)       AS brojProjekata
            MATCH (o1:Osoba {mbz: mbz1})
            MATCH (o2:Osoba {mbz: mbz2})
            MERGE (o1)-[rel: SURADUJE {brojProjekata: brojProjekata}]-(o2)
           RETURN count(rel)
```

Izvršavanjem ovih naredbi, Neo4j baza podataka je napunjena podacima i spremna za daljnje korištenje.



Slika 6.5: Sučelje Neo4j Browser aplikacije i vizualni prikaz podataka dohvaćenih jednostavnim upitom

Alternativno, za potrebe produkcijske okoline, cijeli uvoz podataka moguće je automatizirati na način da se navedene naredbe za uvoz smjeste u skripte te periodički izvršavaju na prikladnim serverima.

6.4 Vizualizacija podataka s Neovis.js bibliotekom

Jednom kada imamo spremnu Neo4j bazu podataka, preostaje ju povezati i prikazati uz pomoć alata za vizualizaciju. Kao alat za vizualizaciju koristimo JavaScript biblioteku Neovis.js, detaljnije opisanu u poglavlju 5.1.3.

Kao što je već rečeno, Neovis.js koristi JavaScript upravljački program za Neo4j kako bi se spojila na Neo4j bazu podataka i pretraživala podatke te Vis.js biblioteku za prikaz mrežne vizualizacije. Pomoću Neovis.js biblioteke do vizualizacije dolazimo u svega nekoliko koraka, a temelj implementacije je popunjavanje konfiguracijskog objekta parametrima potrebnim za spajanje na bazu i definiranje izgleda vizualizacije. Vizualizaciju prikazujemo u sklopu web stranice pa je kod smješten unutar HTML datoteke, a sama biblioteka uključena je kao skripta:

```
<script src="https://cdn.neo4jlabs.com/neovis.js/v1.5.0/neovis.js">
</script>
```

Konfiguracijski objekt definira sljedeće:

1. Parametre potrebne za spajanje na Neo4j bazu podataka: poveznica za konekciju na bazu podataka, korisničko ime i lozinka baze podataka. Definiranje svih navedenih parametara je obavezno.

   U našem su slučaju korisničko ime i lozinka Neo4j baze podataka definirani prilikom kreiranja instance Neo4j-a preko Neo4j Desktop aplikacije gdje se, također, među detaljima pokrenute instance mogu pronaći sve poveznice za konekciju (Neovis zahtjeva onu koja koristi bolt protokol).

   ★ Dodatno, važno je naglasiti postojeću mogućnost za kreiranjem korisničkih uloga kroz Neo4j Browser, pri čemu se može ograničiti rad ulogiranog korisnika nad bazom podataka. Ovom mogućnošću neposredno se ostvaruje nedostajući zaštitni sloj alata ove arhitekturalne kategorije.

2. Parametre za stiliziranje vizualizacije vrhova i bridova, kao što su:

   - *size* parametar kojemu se pridružuje naziv numeričkog svojstva vrhova prema kojemu će im se odrediti veličina i kao takva prikazivati.

     U našem slučaju veličine vrhova ovise o svojstvu *impactFactor*.

   - *community* parametar kojemu se pridružuje naziv numeričkog svojstva vrhova prema kojemu se obavlja grupiranje (najčešće prikazivanjem vrhova u pripadajućoj boji). Vrijednosti koje se pridružuju mogu biti rezultati provođenja algoritama za grupiranje.

     U našem slučaju osobe (znanstvenici) koje pripadaju istom znanstvenom području članovi su iste grupe i pripadajući su vrhovi obojani istom bojom.

   - *thickness* parametar kojemu se pridružuje naziv numeričkog svojstva bridova prema kojemu im se određuje debljina i kao takva prikazuje.

     U našem slučaju debljina bridova ovisi o broju projekata na kojima su dvije osobe zajedno surađivale.

   - *title_properties* parametar kojemu se pridružuje niz svojstava vrhova koje je u cilju prikazivati prelaskom miša preko iscrtanog vrha.

     U našem slučaju prikazuju se sva svojstva koja predstavljaju osobne podatke neke osobe.

   i brojni drugi.

3. Inicijalni Cypher upit kojim dohvaćamo podatke za prikaz.

4. DOM [2] element u sklopu kojeg će vizualizacija biti prikazana. Navođenje DOM elementa je obavezno.

Sljedećim odsječkom koda predstavljena je definicija našeg konfiguracijskog objekta:

```
var config = {
    container_id: "viz",

    server_url:"bolt://localhost:7687",
    server_user:"vanjski",
    server_password:"12345",

    labels: {
      "Osoba": {
        caption: "prezime",
        size: "impactFactor",
        community: "seed",
        title_properties: ["ime", "prezime", "znanstvenoPodrucje",
                           "impactFactor", "mbz"]
      }
    },

    relationships: {
        "SURADUJE": {
          caption: false,
          thickness: "brojProjekata"
        }
    },

    initial_cypher:
        " MATCH (o1:Osoba)-[rel:SURADUJE]-(o2:Osoba)" +
        " WHERE o1.znanstvenoPodrucje = 'Podrucje prirodnih znanosti'" +
        " OR o1.znanstvenoPodrucje    = 'Umjetnicko podrucje' " +
        " RETURN o1, rel, o2" +
        " ORDER BY rel.brojProjekata, o1.ime" +
        " LIMIT " + n + ""
      }
```

Rad programa temelji se na pozivu *draw()* funkcije prilikom učitavanja stranice. U sklopu te funkcije definira se gore spomenuti konfiguracijski objekt uz pomoć kojega se potom stvara novi Neovis objekt. Dodatno je implementirana i interaktivnost vrhova tako da se klikom na pojedini vrh reducira graf i prikazuju samo vezani vrhovi. Funkcija, zatim, nad Neovis objektom poziva funkciju *render()* za iscrtavanje vizualizacije.

[2] DOM (engl. *Document Object Model*) je objektni model HTML dokumenta koji predstavlja hijerarhijski prikaz pripadne web stranice.

Implementacija navedenih funkcionalnosti prikazana je sljedećim odsječkom koda:

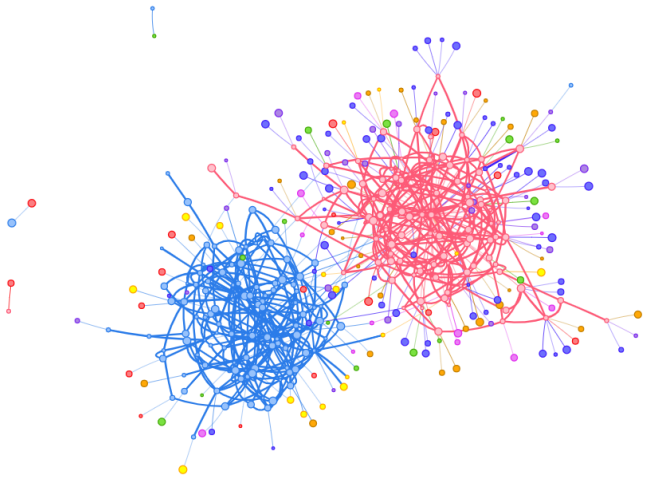
```
var viz = new NeoVis.default(config);

viz.registerOnEvent
    ("completed", (e)=>{
        viz["_network"].on
            ("selectNode", (e1) => {
                var node = e1.nodes[0];
                viz.renderWithCypher
                    (" MATCH (o1:Osoba)-[rel:SURADUJE]-(o2:Osoba)" +
                     " WHERE o1.mbz = " + node +
                     " RETURN o1, rel, o2" );
    });
});

viz.render();
```

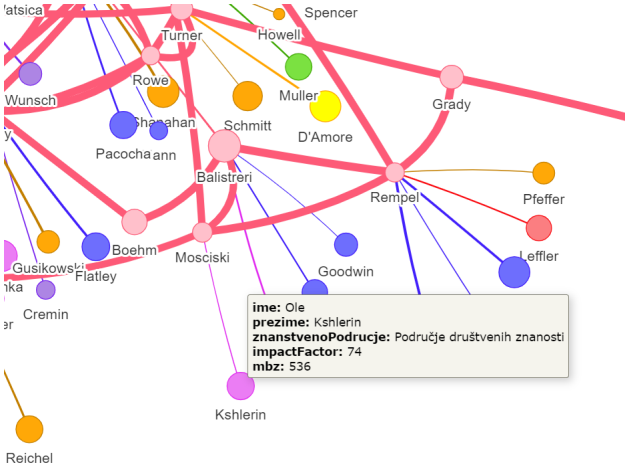
Na slici 6.6 nalazi se prikaz grafa podataka dohvaćenih Cypher upitom:

```
   MATCH (o1:Osoba)-[rel:SURADUJE]-(o2:Osoba)
   WHERE o1.znanstvenoPodrucje = 'Područje prirodnih znanosti'
      OR o1.znanstvenoPodrucje = 'Interdisciplinirana područja umjetnosti'
  RETURN o1, rel, o2
ORDER BY rel.brojProjekata, o1.ime
   LIMIT 1200;
```



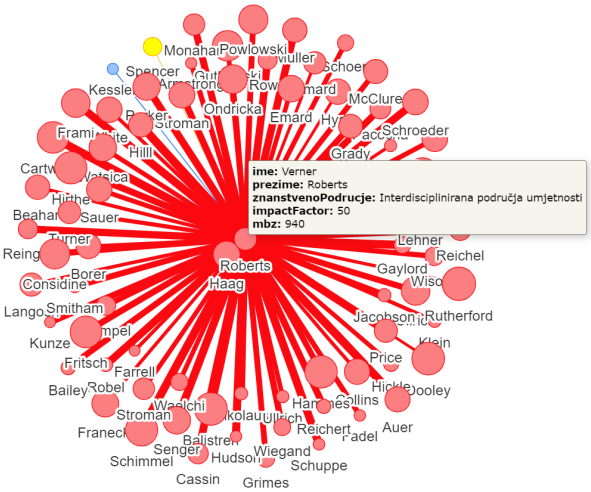
Slika 6.6: Vizualizacija reduciranog grafa sa prikazom 1.200 veza

Prelaskom miša preko bilo kojeg vrha, otvara se okvir sa informacijama dotičnog vrha. Informacije koje se prikazuju su svojstva vrha navedena u definiciji konfiguracijskog objekta pod parametrom *title_properties*. Primjer je prikazan sljedećom slikom:



Slika 6.7: Prikaz detalja vrha prelaskom miša

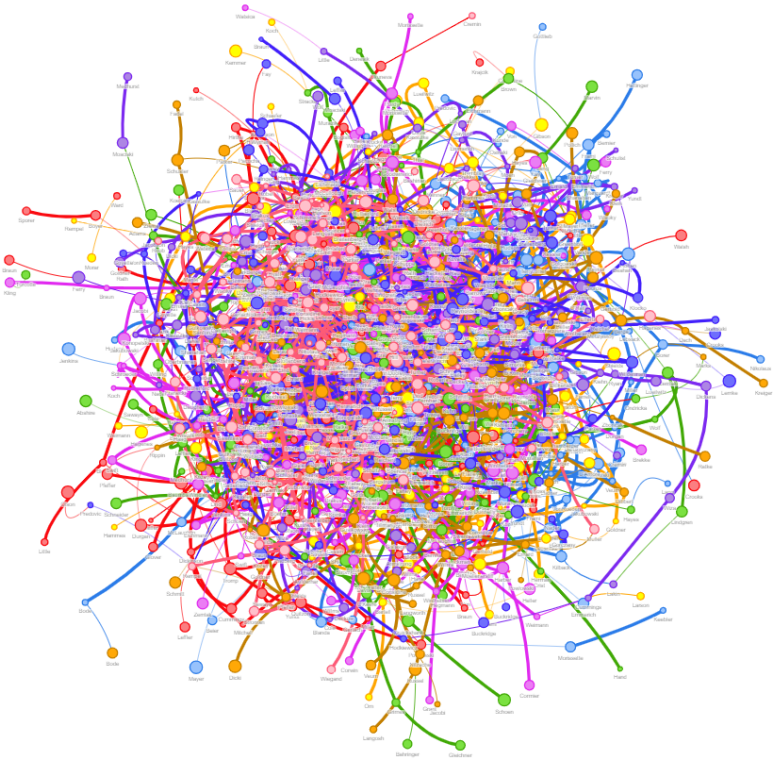
Klikom na pojedini vrh, graf se reducira i prikazuje se pripadni vrh zajedno sa svim onima koji su s njim povezani - slika 6.8.



Slika 6.8: Prikaz znanstvenika zajedno sa svim njegovom suradnicima

6.5 Ograničenja Neo4j + Neovis.js platforme

Neovis.js, prema Vis biblioteci, za prikaz i iscrtavanje mreže koristi HTML Canvas, stoga postoje izazovi za prikaz velikih grafova. Takve se poteškoće mogu ublažiti provođenjem raznih algoritama, no u slučaju jače povezanih podataka, izostaje dobro grupiranje podataka pa je teško otkloniti preklapanje vrhova, što utječe na čitljivost i preglednost podataka. Iz tog razloga, implementirali smo još jednu vizualizaciju uz pomoć Gephi alata čiji prikaz slijedi u idućem poglavlju.



Slika 6.9: Graf bez redukcije s prikazom 3.000 bridova nakon dužeg učitavanja stranice

Poglavlje 7

Implementacija izvještajnog sustava za veće količine podataka

Najčešći problem web aplikacija koje vizualiziraju veće količine podataka predstavlja iscrtavanje grafa, odnosno mreže prilikom učitavanja stranice. U takvim slučajevima, neke bibliteke nude korištenje WebGL tehnologije za prikaz, koja u odnosu na SVG i Canvas tehnologije, nudi najefikasnije rješenje.

No, važno je imati na umu što kojem alatu, odnosno tehnologiji prikaza koju podržava, predstavlja veliki graf te koliko se vrhova i bridova pod tim pojmom podrazumijeva. Prema slici 5.1, može se zaključiti da sve biblioteke koje koriste neke od prikazanih tehnologija pod velikim grafom podrazumijevaju one koji imaju više od 1.000 vrhova. No, što je sa grafovima koji imaju od oko 50.000 vrhova i 800.000 bridova, mogu li se i oni prikazati ili su takvi grafovi ipak ekstremne veličine? S druge strane, ako bi se podaci mogli unaprijed pripremiti na način da se svakom vrhu unaprijed odredi lokacija i izgled, te onda kao takvi prikazati, očekivalo bi se učitavanje web stranice bez poteškoća.

Sličnu ideju otkrili smo uz pomoć Gephi alata. Gephi, detaljnije opisan u poglavlju 5.1.5, samostalna je aplikacija kroz koju je moguće istraživati, analizirati, manipulirati i vizualizirati podatke raznih vrsta grafova, a podnijeti može od oko 100.000 vrhova i 1.000.000 bridova. S obzirom da se radi o samostalnoj aplikaciji, grafovi se vizualiziraju unutar aplikacije (lokalno na računalu na kojem je instalirana), ali postoje razna proširenja kojima je vizualizaciju moguće izvesti uz očuvanje interaktivnosti (primjerice u PDF format). Ono što je korisno svrsi ovog rada jest da postoji proširenje, Sigma Exporter, kojim je moguće vizualizirati podatke (unaprijed pripremljene u Gephi aplikaciji) na webu i to korištenjem Sigma.js biblioteke. Štoviše, proširenje stvara izvještajni sustav dozvoljavajući korisnicima određeno pretraživanje podataka i interakciju sa grafom. Korištenjem tog proširenja, automatski se stvaraju sve datoteke potrebne za prikaz izvještajnog sustava unutar web stranice, a detalje vizualizacije moguće je mijenjati direktno u kodovima

stvorenih datoteka. Stvorene datoteke, dakle, sačinjavaju web aplikaciju.

Stvaranje i upravljanje vizualizacijom te izvoz u web aplikaciju može se ručno obaviti preko Gephi aplikacije, no za potrebe ovog rada, cilj je bio automatizirati cijeli proces stvaranja izvještajnog sustava pa se sve obavlja programski. U tom slučaju, za razvojne inženjere postoji Gephi Toolkit - Java biblioteka koju je moguće koristiti u sklopu bilo kojeg projekta kako bi se programski mogla obavljati većina operacija koje nudi Gephi aplikacija (osim proširenja koja se moraju posebno integrirati).

Više o Gephi Toolkit-u i proširenju Sigma Exporter, kao i implementaciji izvještajnog sustava, nalazi se u idućim poglavljima.

Koristi se isti izvor podataka kao kod implementacije izvještajnog sustava nad Neo4j + Neovis platformom, opisan u poglavlju 6.2.

7.1 Migracija podataka

Migraciju podataka ne možemo potpuno poistovjetiti s migracijom opisanom u poglavlju 6.3 iz dva razloga: prvi je razlog što se podaci iz relacijske baze podataka izvoze u CSV datoteke sada drugačijeg izgleda, propisanog od strane Gephi alata, a drugi je razlog što se podaci uvoze pomoću naredbi Gephi Toolkit-a direktno u strukturu grafa u memoriji [1]. Postoje mogućnosti za uvoz podataka posredstvom grafovske baze podataka (primjerice Neo4j), no u ovom slučaju time se komplicira arhitektura sustava pa se taj dio izostavlja.

Slično kao i prije, podaci koji predstavljaju vrhove i bridove (zajedno s njihovim svojstvima), smještaju se u dvije različite CSV datoteke za koje vrijede sljedeće odredbe:

1. CSV datoteka koja predstavlja vrhove:
    - nužno postojanje zaglavlja u kojem se nalaze nazivi atributa (svojstava)
    - nužno postojanje dvaju atributa, u zaglavlju nazvanim *Id* i *Label* (redom), pri čemu *Id* predstavlja jedinstveni identifikator vrha, a *Label* oznaku vrha
    - nazivi ostalih atributa su proizvoljni

[1] Grafovske baze podataka (najčešće one koje koriste nativnu pohranu i obradu grafa) pohranjuju graf lokalno na disk ili u klaster te su optimizirane za rad sa velikim grafovima i većim brojem istovremenih korisnika. Fokusirane su na brze odgovore prilikom pretraživanja podataka pri čemu se pristupa određenom podgrafu, a ne cijelom grafu.

Gephi alat, s druge strane, kao analitički softver zahtjeva pristup cijelom grafu u svakom trenutku. Pohranjujući graf u memoriju računala omogućuje brzi pristup koordinatama svakog vrha, u svakom trenutku, što je nužno, ne samo za vizualizaciju, već i za provođenje analiza kojima raspolaže. Implementacijom strukture grafa u memoriji (engl. *in-memory graph structure*), analize se vrše u stvarnom vremenu.

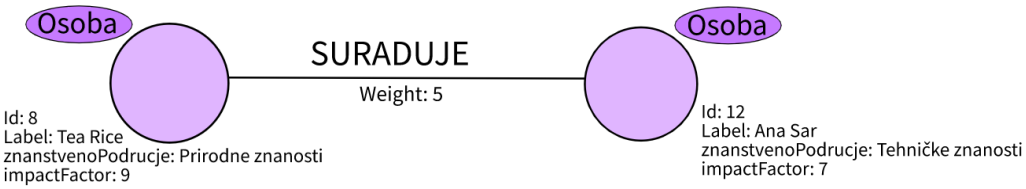
2. CSV datoteka koja predstavlja bridove:

   - nužno postojanje zaglavlja uz obavezna dva atributa: *Source* i *Target* (redom), pri čemu *Source* predstavlja identifikator početnog vrha, a *Target* završnog
   - atribut koji predstavlja težinu ili debljinu brida, u zaglavlju mora imati naziv *Weight*
   - moguće je postaviti i tip brida vrijednostima *"Directed"* i *"Undirected"* uz *Type* naziv atributa u zaglavlju
   - nazivi ostalih atributa su proizvoljni

Pripadne dvije datoteke, analogno kao i prije, nazivaju se *znanstvenici.csv* i *suradnje.csv*.

Strukturu grafa možemo poistovjetiti sa strukturom prikazanom na slici 7.1 uz nekoliko manjih izmjena. Izmjene su određene po uzoru na gore navedena pravila izgleda CSV datoteka, a u modelu grafa odnose se samo na promjenu naziva svojstava. Naime, određeno je sljedeće: matični broj znanstvenika sada je predstavljen svojstvom *Id*, ime i prezime osobe zajedno predstavljaju svojstvo *Label*, a broj projekata na kojima su dvije osobe zajedno surađivale predstavlja težinu brida i pripadno svojstvo *Weight*.

Slijedeći navedeno, analogna struktura grafa predstavljena je sljedećom slikom:



Slika 7.1: Struktura grafa (2)

Za izvoz podataka iz relacijske baze podataka koristimo isti alat opisan u poglavlju 6.3.1, a naredbe za izvoz su sljedeće:

```
-- vrhovi
UNLOAD TO "znanstvenici.csv" DELIMITER ","
SELECT MBZ                                  AS Id,
       cfFirstNames  ||' '|| cfFamilyNames AS Label,
       (SELECT ppgDisciplinaNaziv
```

```
          FROM hrPPGDisciplinaML
         WHERE ppgDisciplinaId = hrPers_Zvanje.ppgDisciplinaId1
           AND cfLangCode = 'hr') AS znanstvenoPodrucje,
       impactFactor
 FROM cfPers, cfPersName_Pers, cfPersName, hrPers_Zvanje
WHERE cfPers.cfPersId = cfPersName_Pers.cfPersId
  AND cfPersName_Pers.cfPersNameId = cfPersName.cfPersNameId
  AND cfPersName_Pers.cfStartDate < TODAY
  AND cfPersName_Pers.cfEndDate IS NULL
  AND hrPers_Zvanje.cfPersId = cfPers.cfPersId;


-- bridovi
UNLOAD TO "suradnje.csv" DELIMITER ","
SELECT pers1.MBZ AS Source,
       pers2.MBZ AS Target,
       COUNT(*)  AS Weight,
       'directed' AS Type
  FROM cfPers pers1, cfPers pers2, cfProj_Pers proj1, cfProj_Pers proj2
 WHERE proj1.cfProjId = proj2.cfProjId
   AND proj1.cfPersId < proj2.cfPersId
   AND pers1.cdPersId = proj1.cfPersId
   AND pers2.cdPersId = proj2.cfPersId
GROUP BY 1,2;
```

Naredbe za uvoz podataka navedene su u idućem poglavlju u sklopu prikaza Gephi Toolkit biblioteke te implementacije izvještajnog sustava.

7.2 Vizualizacija podataka sa Gephi Toolkit bibliotekom

Gephi Toolkit je Java biblioteka koja sadrži glavne module Gephi alata, poput modula za strukturu grafa, modula za upravljanje izgledom vizualizacije, modula za filtriranje i drugih. Pomoću naredbi Gephi Toolkit-a moguće je koristiti funkcionalnosti Gephi alata u sklopu bilo kojeg Java projekta te na taj način automatizirati proces izgradnje vizualizacije i analize grafa. Posljednja podignuta verzija je 0.9.2.

Na samom početku, kako bismo koristili Gephi Toolkit biblioteku, izgradili smo Java projekt uz pomoć Maven alata. Maven je alat za za upravljanje Java projektima koji služi kao pomoć pri razvoju izvješća te praćenja i postavljanja automatizacije, a glavna značajka

je jednostavno uključivanje i preuzimanje raznih biblioteka. Maven projekt, kao i cijela implementacija izvještajnog sustava, izgrađen je u sklopu integriranog razvojnog okruženja za Javu - IntelliJ IDEA.

Postoji nekoliko načina na koji se Gephi Toolkit biblioteka može uključiti u Java projekt, a u slučaju korištenja Maven alata, uključivanje se obavlja u konfiguracijskoj datoteci *pom.xml* navođenjem sljedećeg koda:

```
<dependency>
    <groupId>org.gephi</groupId>
    <artifactId>gephi-toolkit</artifactId>
    <version>0.9.2</version>
</dependency>
```

U sljedećim poglavljima nalaze se koraci implementacije željene vizualizacije koju će se Sigma Exporter-om izvesti u cjelovit izvještajni sustav.

7.2.1 Stvaranje radnog prostora

Jednom kada je biblioteka uključena, potrebno je stvoriti radni prostor (spremnik) u kojeg se smještaju svi podaci. Radni prostor često se predaje modulima kako bi nad njim obavljali svoje funkcionalnosti, a po potrebi, moguće je imati i nekoliko neovisnih radnih prostora. Moduli svoje objekte spremaju u zajednički spremnik *Lookup*, odakle ih drugi moduli mogu pozivati navodeći ime klase.

Stvaranje novog radnog prostora prikazano je sljedećim kodom:

```
ProjectController pc = Lookup.getDefault().lookup(ProjectController.class);
pc.newProject();
Workspace workspace = pc.getCurrentWorkspace();
```

7.2.2 Uvoz podataka

Uvoz podataka ostvariv je na više načina, a najčešći su: uvoz direktnim spajanjem na podržane relacijske baze podataka i uvoz preko podržanih vrsta datoteka. Također, podatke je moguće programski kreirati i spremiti u strukturu grafa.

U našem slučaju, podaci se uvoze iz CSV datoteka pomoću modula za uvoz podataka te pozivom klase *ImportController*. Uvoz podataka preko podržanih datoteka općenito se obavlja u dva koraka:

1. Spremanje podataka u klasu *Container* (također jedna vrsta spremnika).

2. Prilaganje podataka iz spremnika u strukturu grafa i pripadne atribute pozivom funkcije *process(Container container, Processor processor, Workspace workspace)* modula za uvoz podataka.

Naredbe potrebne za uvoz podataka iz ranije stvorenih CSV datoteka *znanstvenici.csv* i *suradnje.csv*, prikazane su sljedećim odsječcima kodova:

1. Uvoz vrhova iz *znanstvenici.csv* datoteke:

```
Container container;
try {
    File file_nodes =
        new File(getClass().getResource("znanstvenici.csv").toURI());

    container = importController.importFile(file_nodes);
    // Zabrana automatskog kreiranja nedostajucih vrhova
    container.getLoader().setAllowAutoNode(false);
    container.getLoader().setAutoScale(true);

} catch (Exception e) {
    e.printStackTrace();
    return;
}

importController.process(container, new DefaultProcessor(),
                         workspace);
```

2. Uvoz bridova iz *suradnje.csv* datoteke:

```
try {
    File file_edges =
        new File(getClass().getResource("suradnje.csv").toURI());

    container = importController.importFile(file_edges);
    container.getLoader()
        .setEdgeDefault(EdgeDirectionDefault.DIRECTED);
    // Ukoliko postoje paralelni (isti) bridovi, uzeti jednog od njih
    container.getLoader()
        .setEdgesMergeStrategy(EdgeMergeStrategy.FIRST);
    container.getLoader().setAutoScale(true);

} catch (Exception e) {
```

```
    e.printStackTrace();
    return;
}

importController.process(container, new DefaultProcessor(),
                         workspace);
```

Izvršavanjem ovih naredbi, svi podaci su smješteni u strukturu grafa, a dohvaćati ih se može korištenjem modula za dohvat modela grafa, pozivanjem klase *GraphModel*.

7.2.3 Izgled vizualizacije

Gephi nudi provođenje raznih algoritama, statistika, filtera i metrika nad podacima te razne akcije koje utječu na prikaz tih podataka vizualizacijom grafa. U ovom praktičnom radu provedeni su samo neki od algoritama i akcija kako bismo prikazali graf u svjetlu inicijalne ideje vizualizacije, opisane u poglavlju 6.1.

Slijedi njihov kratki pregled.

Veličina vrhova

Postavljanje veličine vrhova u ovisnosti o numeričkoj vrijednosti svojstva *impactFactor*, provodi se transformacijom inicijalno postavljenih veličina vrhova. Transformaciju obavlja funkcija *transform(Function function)* modula za prikaz modela grafa (klasa *AppearanceModel*). Ključna klasa koja izvodi transformaciju je *RankingNodeSizeTransformer*.

Transformacija se izvodi sljedećim naredbama:

```
Column columnForSize =
    graphModel.getNodeTable().getColumn("impactFactor");
Function ranking =
   appearanceModel.getNodeFunction(graph, columnForSize,
                                   RankingNodeSizeTransformer.class);
RankingNodeSizeTransformer transformer =
   (RankingNodeSizeTransformer) ranking.getTransformer();
transformer.setMinSize(3);
transformer.setMaxSize(10);
appearanceController.transform(ranking);
```

Boja vrhova

Postavljanje boje vrha s obzirom na svojstvo *znanstvenoPodrucje* provodi se na sličan način. Boja vrha i pripadnog brida (koji iz njega izlazi) je ista, a boje su birane slučajnim odabirom pomoću klase *Pallete*. Ključna klasa koja izvodi transformaciju , naziva se *PartitionElementColorTransformer*.

Bojanje vrhova prikazano je sljedećim odsječkom koda:

```
Column column =
    graphModel.getNodeTable().getColumn("znanstvenoPodrucje");
Function func =
    appearanceModel.getNodeFunction(graph, column,
                                    PartitionElementColorTransformer.class);
Partition partition = ((PartitionFunction) func).getPartition();
Palette palette =
    PaletteManager.getInstance().generatePalette(partition.size());
partition.setColors(palette.getColors());
appearanceController.transform(func);
```

Grupiranje vrhova

Grupiranje vrhova provedeno je Fruchterman Reingold algoritmom [2]. Instanci FruchtermanReingold klase predaje se model grafa, a potom parametri poput brzine i duljine izvođenja, blizine vrhova i drugih koji predstavljaju tehničke detalje izvedbe. Grupiranjem podataka mijenjaju se koordinate vrhova i kao takve ostaju spremljene. Ovim se ispunjava primarna želja da se vizualizacija unaprijed pripremi za iscrtavanje na web stranicu ili neki drugi oblik izvoza.

Stvaranje instance Fruchterman Reingold algoritma i predaja modela grafa obavlja se sljedećim naredbama:

```
FruchtermanReingold frLayout =
    new FruchtermanReingold(new FruchtermanReingoldBuilder());
frLayout.setGraphModel(graphModel);
```

Grupiranje podataka posljednja je napravljena akcija nad grafom podataka i vizualizacija grafa spremna je za prikaz.

[2] Fruchterman Reingold algoritam se temelji na sili koja vrhove tretira kao opruge koje ih pomiču bliže ili dalje jedan od drugog, često u ovisnosti o jačini odnosno težini bridova (veća težina brida jače privlači vrhove jedan prema drugom), kako bi se pronašlo ravnotežno stanje sustava.

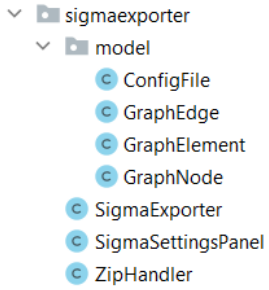
7.3 Sigma Exporter

Sigma Exporter dodatak je Gephi alatu kojim se omogućuje prikaz interaktivne vizualizacije mreže u web pregledniku. Inicijalna implementacija nastala je u sklopu projekta InteractiveVis pod organizacijskom jedinicom Oxford Internet Institute, University of Oxford, 2012. godine. Temelji se na korištenju Sigma.js biblioteke, web tehnologija HTML5 i CSS3 te Canvas tehnologije za prikaz vizualizacije. S obzirom na sigurnosne restrikcije preglednika, vizualizacije nije moguće pokrenuti lokalno sa računala bez web servera.

Sigma Exporter preuzima vizualizaciju izrađenu pomoću Gephi alata te stvara ZIP direktorij u kojem se nalazi niz datoteka potrebnih za prikaz interaktivne vizualizacije mreže. Niz datoteka čini web aplikaciju kojom se vizualizira mreža odnosno graf unaprijed obrađen Gephi alatom. Interaktivnu vizualizaciju moguće je dorađivati i mijenjati.

Kako bismo nastavili s automatizacijom procesa stvaranja izvještajnog sustava, nakon što je vizualizacija spremna za prikaz, preostalo je priključiti potrebne dijelove implementacije Sigma Exporter-a u projekt. Izvorni kod skinut je sa stranice [6].

Popis preuzetih klasa prikazan je sljedećom slikom:



Slika 7.2: Klase preuzete iz izvorne implementacije Sigma Exporter dodatka

Ukratko, klase implementiraju sljedeće:

- GraphEdge, GraphElement i GraphNode klase implementiraju osnovne funkcionalnosti ”rukovanja” elementima grafa.
- ConfigFile klasa implementira postavljanje određenih vrijednosti i informacija o izgledu izvještajnog sustava, a sve potrebne informacije unose se u sklopu funkcije *setup()* klase SigmaSettingsPanel.
- ZipHandler klasa implementira potrebne funkcije za upravljanje ZIP direktorijem.

- SigmaExporter klasa implicitno koristi sve ostale klase i izvršava izvoz vizualizacije u izvještajni sustav, prosljeđivanjem grafa i svih potrebnih informacija u datoteke ZIP direktorija.

Uz preuzete klase, postoji i kostur ZIP direktorija, koji se Sigma Exporter-om puni potrebnim podacima, stvara nekoliko novih datoteka te tako cjelovit izvozi u novi ZIP direktorij. Izostavljene klase većinom su usko vezane uz dijaloški okvir preko kojeg korisnici ručno unose informacije o izgledu izvještajnog sustava.

Stvaranje novog ZIP direktorija odvija se na način da se instanci Sigma Exporter-a preda cijeli radni prostor (dakle spreman graf), posredstvom SigmaSettingsPanel-a predaju sve potrebne informacije za željeni izgled izvještajnog sustava te naposljetku izvrši funkcija *execute()*. Funkcija *execute()* proslijeđene informacije o izgledu izvještajnog sustava sprema u *config.json* datoteku, a podatke u datoteku *data.json*.

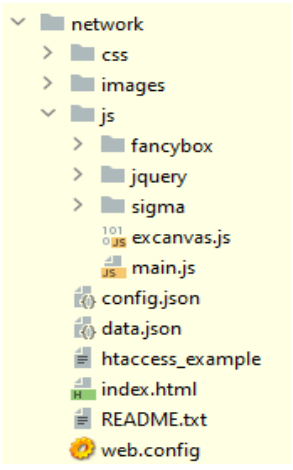
Pokretanje Sigma Exporter-a izvršava se sljedećim naredbama:

```
final SigmaExporter exporter = new SigmaExporter();
exporter.setWorkspace(workspace);

SigmaSettingsPanel settingPanel = new SigmaSettingsPanel();
settingPanel.setup(exporter);

exporter.execute();
```

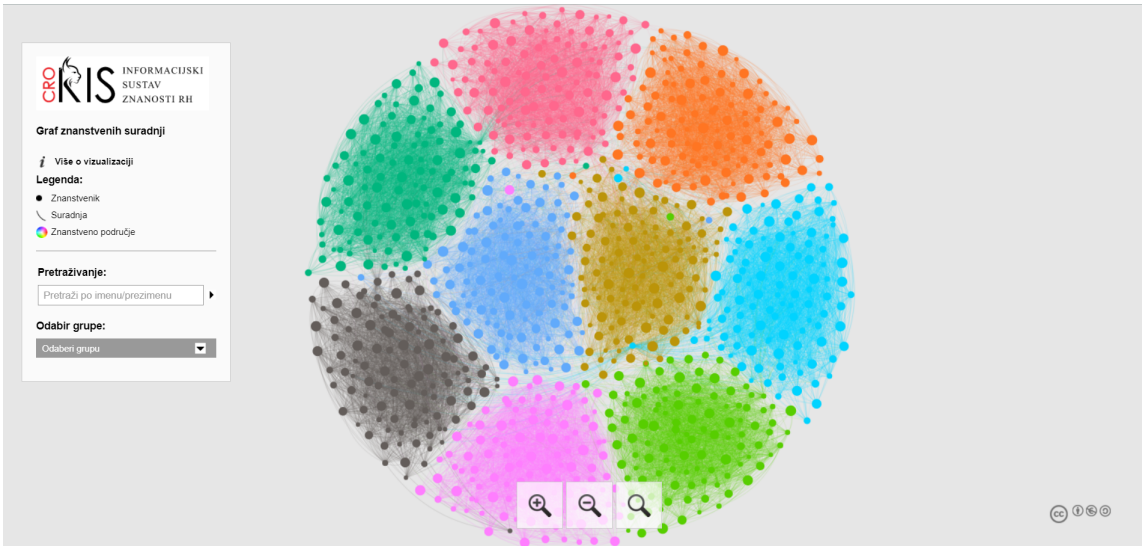
Sadržaj stvorenog ZIP direktorija prikazan je sljedećom slikom:



Slika 7.3: Sadržaj raspakiranog ZIP direktorija *network.zip*

7.4 Završni izgled

Kao što je već rečeno, izvještajni sustav prikazuje se posredstvom web servera. U našem slučaju, na lokalnom računalu, korišten je Python web server i naredba za njegovo pokretanje nad direktorijem *network* u komandnom prozoru: **python -m http.server**. Ovom naredbom, prikaz izvještajnog sustava nalazi se na web adresi: http://localhost:8000/. Dobivamo izvještajni sustav prikazan na sljedećoj slici (prikaz svih 999 vrhova i 25.442 bridova):

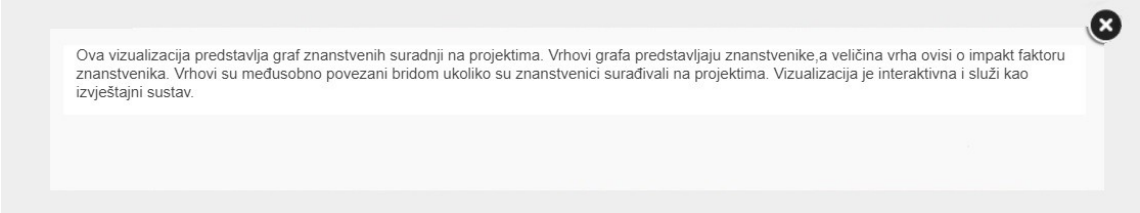


Slika 7.4: Izvještajni sustav - početni izgled

Informacijski panel prikazan na lijevoj strani slike 7.4 sastoji se od osnovnih informacija vezanih uz vizualizaciju te interaktivnog dijela koji služi za pretraživanje grafa.

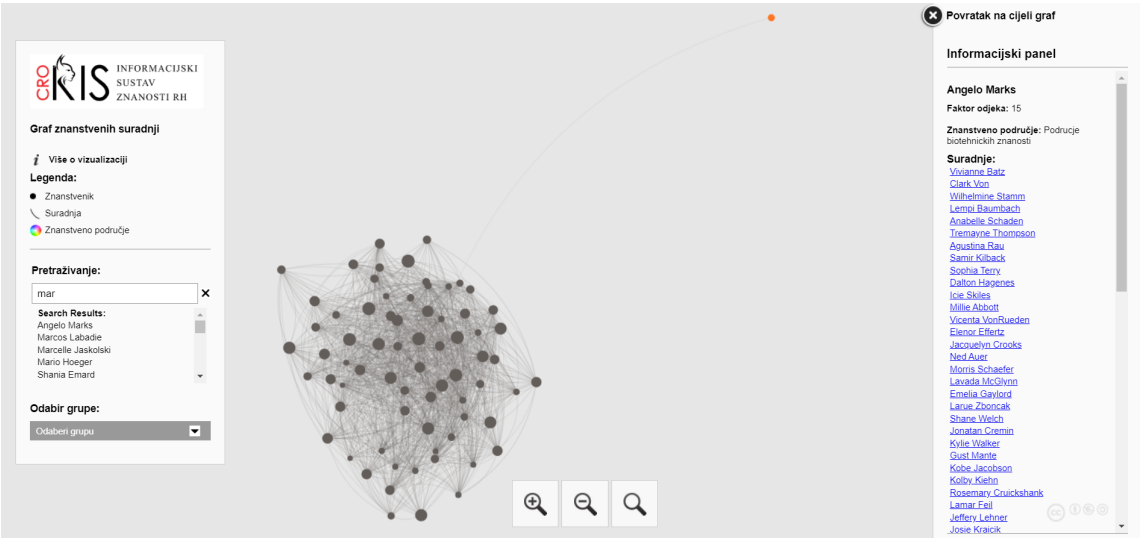
Funkcionalnosti su sljedeće:

- Osnovne informacije o vizualizaciji prikazuju se klikom na tekst ”Više o vizualizaciji”, otvaranjem skočnog prozora prikazanog slikom 7.5.
- Prikazana je i legenda simbola vizualizacije: vrh predstavlja znanstvenika, brid predstavlja suradnju, a boje ovise o znanstvenom području kojim se pojedini znanstvenik bavi.
- Pretraživati graf je moguće po imenu i/ili prezimenu znanstvenika. Unosom niza znakova od najmanje tri slova, prikazuje se popis znanstvenika koji u imenu i/ili prezimenu imaju dani niz znakova. Paralelno, graf se reducira i ukoliko nema drugog



Slika 7.5: Skočni prozor s informacijama o vizualizaciji

odabira, prikazuje se prvi znanstvenik s dobivenog popisa, zajedno sa znanstvenicima s kojima je surađivao. Također, otvara se novi informacijski panel s desne strane grafa. Primjer je prikazan sljedećom slikom:



Slika 7.6: Pretraživanje po imenu i/ili prezimenu

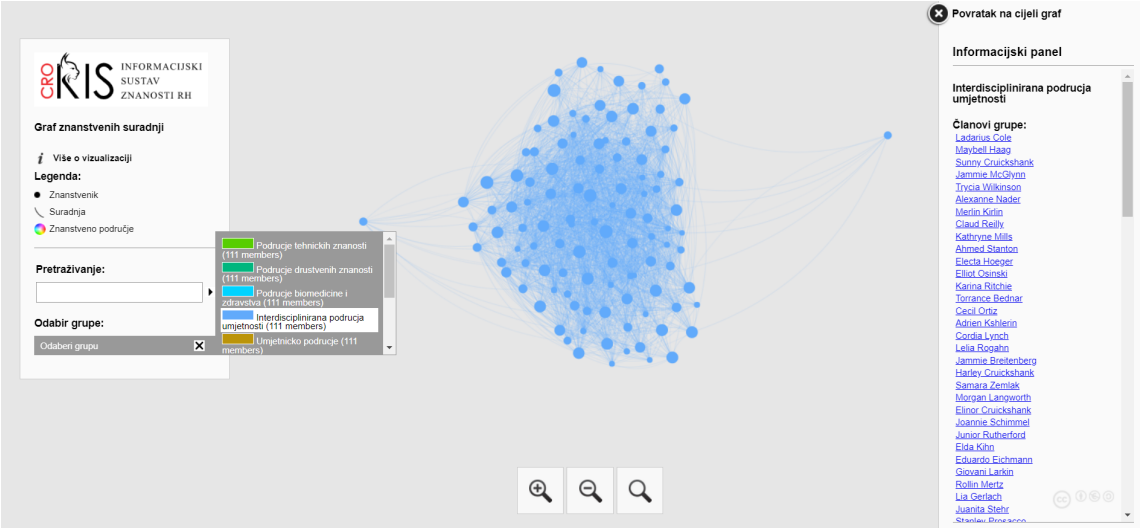
Novootvoreni informacijski panel sadrži informacije o znanstveniku kojeg se prikazuje. Prikazano je ime i prezime znanstvenika, faktor odjeka i znanstveno područje kojim se bavi, a potom i popis svih znanstvenika s kojima je surađivao. Svakog od popisanih znanstvenika odabirom je moguće analogno prikazati.

- Pretraživanje po grupi odnosi se na odabir znanstvenog područja pri čemu se reducira graf i prikazuju samo oni znanstvenici koji pripadaju odabranom znanstvenom području. Popis pipadnih znanstvenika nalazi se u informacijskom panelu koji se

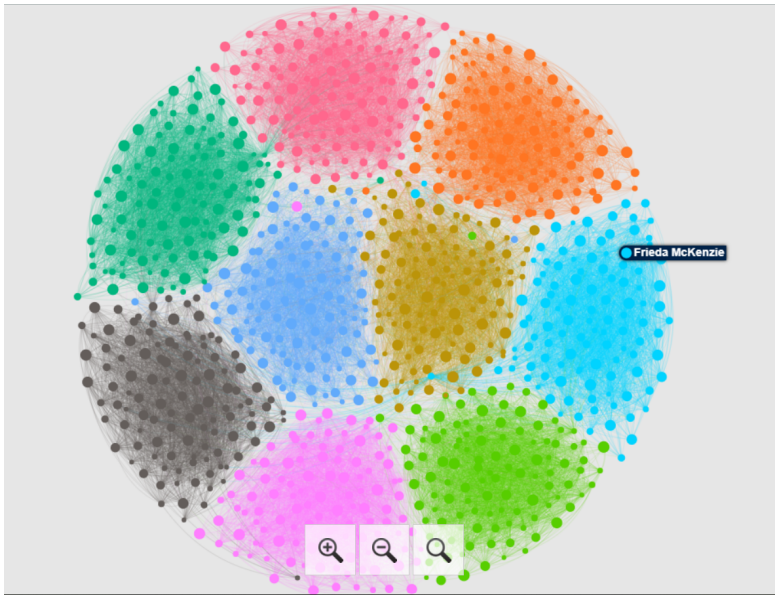
paralelno otvara desno od grafa.
Primjer je prikazan slikom 7.7.

- Prelaskom miša preko bilo kojeg vrha na grafu prikazuje se ime i prezime znanstvenika, a klikom na vrh prikazuju se informacije o znanstveniku u informacijskom panelu, analogno kao pretragom po imenu i/ili prezimenu.
  Primjer prikaza imena i prezimena znanstvenika prelaskom miša preko vrha, prikazan je slikom 7.8.

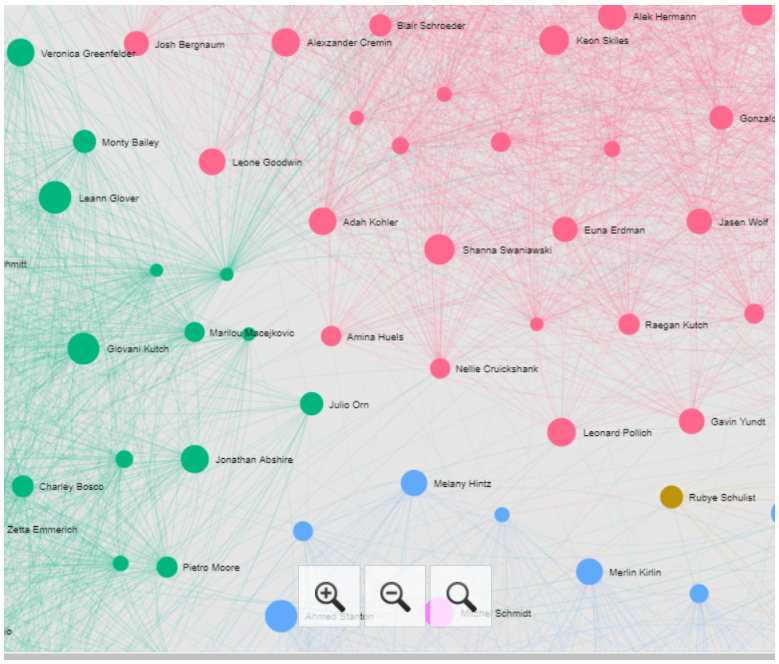
- Graf je moguće zumirati za to predviđenim gumbima, a dovoljno velikim približavanjem, prikazuju se imena i prezimena znanstvenika - slika 7.9.



Slika 7.7: Pretraživanje po znanstvenom području



Slika 7.8: Prelazak miša preko vrha



Slika 7.9: Zumiranje grafa

Poglavlje 8

Zaključak

Otkrivanjem prednosti grafovskih NoSQL baza podataka usmjerava se pažnja na njihovo korištenje. S obzirom da je velika količina do sada prikupljenih podataka smještena u relacijskim bazama podataka, predloženi su različiti načini za njihovu migraciju u svrhu korištenja izvještajnih sustava. Izvještajni sustavi, posebno oni interaktivni, mogu dati dobar uvid u poslovne procese različitih sustava, a korisnicima omogućuju jednostavnije upravljanje podacima i lakše shvaćanje domene.

Osim teorijske podloge, implementirali smo cijeli proces izgradnje izvještajnog sustava počevši od podataka smještenih u relacijskoj bazi podataka, njihovom migracijom u grafovsku, a potom implementacijom izvještajnog sustava alatima za vizualizaciju.

Koristili smo podatke koji predstavljaju znanstvene suradnje na projektima i stvorili grafovsku bazu podataka s 999 vrhova i 25.442 bridova. Podatke smo iz testne relacijske baze podataka naprije migrirali u Neo4j grafovsku bazu podataka, a onda uz pomoć JavaScript biblioteke Neovis implementirali vizualizaciju. Time smo pokazali kako se implementira izvještajni sustav grafovskih baza podataka.

No, unatoč brojnim prednostima Neovis.js biblioteke, poput jednostavnosti implementacije, dobrog izgleda vizualizacije te kvalitetne dokumentacije, naišli smo na poteškoće s prikazom većih količina podataka. Naime, kako bismo ostvarili dobar izgled vizualizacije trebali smo reducirati graf jer za prikaz većeg broja podataka (već od oko 3.000) izostaje dobro grupiranje, a web stranica ima poteškoća s učitavanjem. Razlog tome je prevelika količina podataka koja se u stvarnom vremenu povlači iz Neo4j-a i potrebno ju je grupirati i iscrtati za vrijeme učitavanja web stranice.

S idejom da cjeloviti izgled graf pripremimo unaprijed, implementirali smo dodatno izvještajni sustav i Gephi alatom. Ideja pripreme vizualizacije unaprijed, izložena je u svrhu sprječavanja grupiranja i dohvaćanja podataka u vrijeme učitavanja web stranice. Gephi alat ima mogućnost spajanja na grafovsku bazu podataka, no radi jednostavnosti

arhitekture podatke smo iz relacijske baze podataka izvozili u CSV datoteke te učitavali direktno u strukturu grafa u memoriji.

S obzirom da je Gephi alat samostalna aplikacija, koristili smo biblioteku Gephi Toolkit kako bismo automatizirali proces izgradnje vizualizacije te uz pomoć dodatka Sigma Exporter-a stvorili web aplikaciju, odnosno izvještajni sustav. Sigma Exporter nudi unaprijed pripremljeni izvještajni sustav koji je po potrebi moguće doraditi, ali uz malo više programerskog znanja i kodiranja. Ovim izvještajnim sustavom cijeli graf prikazuje se bez poteškoća kao i sva implementirana interakcija s podacima.

Zaključujemo da unatoč prednostima grafovskih baza podataka i prirodi želje za izgradnjom izvještajnog sustava nad njom, još uvijek nisu dovoljno razvijeni alati koji nude dobre performanse za prikaz većih količina podataka. Problemi i dalje ostaju u području prikaza vizualizacije koja podatke učitava i prikazuje u stvarnom vremenu. Iz tog razloga, potrebno je pribjegavati unaprijed pripremljenim vizualizacijama i izvještajnim sustavima kao što je to primjer implementacije uz pomoć Gephi alata.

Bibliografija

[1] *"Build your own interactive network"*, http://blogs.oii.ox.ac.uk/vis/2012/11/15/build-your-own-interactive-network/, dohvat: 2021-08-26.

[2] *"CERIF-CRIS"*, https://openaire-guidelines-for-cris-managers.readthedocs.io/en/v1.1.1/introduction.html#cerif-cris, dohvat: 2021-08-10.

[3] *"CRORIS"*, https://www.srce.unizg.hr/croris/, dohvat: 2021-08-10.

[4] *"Cypher Query Language"*, https://neo4j.com/developer/cypher/, dohvat: 2021-07-22.

[5] *"Gephi boosts its performance with new "GraphStore" core"*, https://gephi.wordpress.com/tag/architecture/, dohvat: 2021-08-24.

[6] *"Gephi plugins - Sigma Exporter"*, https://github.com/oxfordinternetinstitute/gephi-plugins/tree/sigmaexporter-plugin, dohvat: 2021-03-24.

[7] *"Gephi Toolkit Tutorial"*, https://gephi.org/tutorials/gephi-tutorial-toolkit.pdf, dohvat: 2021-08-26.

[8] *"Gephi"*, https://gephi.org/, dohvat: 2021-05-14.

[9] *"Graph Visualization Tools"*, https://neo4j.com/developer/tools-graph-visualization/, dohvat: 2021-05-14.

[10] *"Graph Visualization With Neo4j Using Neovis.js"*, https://medium.com/neo4j/graph-visualization-with-neo4j-using-neovis-js-a2ecaaa7c379, dohvat: 2021-05-14.

[11] *"Graph Visualization With Neo4j Using Neovis.js"*, https://medium.com/neo4j/graph-visualization-with-neo4j-using-neovis-js-a2ecaaa7c379, dohvat: 2021-08-17.

[12] *"Gremlin Query Language"*, https://docs.janusgraph.org/basics/gremlin/, dohvat: 2021-07-22.

[13] *"Hands on Graph Data Visualization"*, https://medium.com/neo4j/hands-on-graph-data-visualization-bd1f055a492d, dohvat: 2021-05-14.

[14] *"History of graph databases"*, https://bitnine.net/blog-graph-database/history-of-databases-and-graph-database/?ckattempt=1/, dohvat: 2021-06-17.

[15] *"How-To: Import CSV Data with Neo4j Desktop"*, https://neo4j.com/developer/desktop-csv-import/, dohvat: 2021-08-10.

[16] *"How to use Sigmajs to display your graph ?"*, https://medium.com/neo4j/how-to-use-sigmajs-to-display-your-graph-3eedd75275bb, dohvat: 2021-05-14.

[17] *"Hrvatski Memgraph osigurao investiciju od 6,7 milijuna dolara predvođenu Microsoftovim fondom!"*, https://www.netokracija.com/memgraph-investicija-174007, dohvat: 2021-07-21.

[18] *"Importing csv data in Gephi"*, https://seinecle.github.io/gephi-tutorials/generated-html/importing-csv-data-in-gephi-en.html, dohvat: 2021-08-24.

[19] *"Importing CSV Data into Neo4j"*, https://neo4j.com/developer/guide-import-csv/, dohvat: 2021-08-10.

[20] *Materijali iz kolegija Napredne baze podataka 2019./2020.*, https://www.pmf.unizg.hr/math/predmet/nbp.

[21] *"Memgraph - overview"*, https://docs.memgraph.com/memgraph/overview/, dohvat: 2021-07-21.

[22] *"Most popular graph databases"*, https://www.c-sharpcorner.com/article/most-popular-graph-databases/, dohvat: 2021-07-21.

[23] *"Neo4j Browser User Interface Guide"*, https://neo4j.com/developer/neo4j-browser/, dohvat: 2021-08-10.

[24] *"Neo4j Desktop User Interface Guide"*, https://neo4j.com/developer/neo4j-desktop/, dohvat: 2021-08-10.

[25] *"Neovis.js"*, `https://github.com/neo4j-contrib/neovis.js`, dohvat: 2021-08-17.

[26] *"Network visualization"*, `https://visjs.github.io/vis-network/docs/network/`, dohvat: 2021-05-14.

[27] *"OrientDB - overview"*, `https://www.tutorialspoint.com/orientdb/orientdb_overview.htm`, dohvat: 2021-07-21.

[28] *"Tutorial: Import Relational Data Into Neo4j"*, `https://neo4j.com/developer/guide-importing-data-and-etl/`, dohvat: 2021-07-28.

[29] *"Tutorial: Import Relational Data Into Neo4j"*, `https://neo4j.com/developer/guide-importing-data-and-etl/`, dohvat: 2021-08-10.

[30] *"vis.js"*, `https://visjs.org/`, dohvat: 2021-05-14.

[31] *"What is Neo4j?"*, `https://neo4j.com/developer/graph-database/`, dohvat: 2021-06-17.

[32] E. Brewer, *Towards robust distributed systems*, PODC, sv. 7, Portland, OR, 2000, str. 343477–343502.

[33] R. De Virgilio, A. Maccioni i R. Torlone, *Converting relational to graph databases*, First International Workshop on Graph Data Management Experiences and Systems, 2013, str. 1–6.

[34] R. Manger, *Baze podataka*, Zagreb, Element (2012).

[35] E. Murljačić, *Rudarenje podataka o znanstvenoj suradnji iz baze google scholar (Diplomski rad)*, (2020).

[36] O. Orel, *Otkrivanje moguće zlouporabe u informacijskim sustavima temeljeno na odstupanjima u vremenskim grafovima (Disertacija)*, (2017).

[37] O. Orel, S. Zakošek i M. Baranović, *Konverzija relacijskih u grafovske baze podataka orijentirana na svojstva*, Automatika : časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije (2017), br. 57, 836–845.

[38] S. Palod, *Transformation of relational database domain into graphs based domain for graph based data mining*, (2004).

[39] I. Robinson, J. Webber i E. Eifrem, *Graph Databases: New Opportunities for Connected Data, 2nd Edition*, O'Reilly Media, 2015.

[40] N. Tomić, *NoSQL tehnologije i primjene (Diplomski rad)*, (2016).

[41] A. Vukotic, N. Watt, T. Abedrabbo, D. Fox i J. Partner, *Neo4j in action*, sv. 22, Manning Shelter Island, 2015.

Sažetak

Razlozi nastanka NoSQL baza podataka ponajviše se temelje na ograničenjima relacijskih baza podataka. Nudeći fleksibilnu shemu, efikasnu pohranu i obradu velikih količina podataka te financijski prihvatljivu cijenu, sve više šire svoju popularnost. Postoje četiri kategorije, a svaka je prikladna za različite vrste i značajke domena. Među njima, posebno su specifične grafovske baze podataka. Razlikujući se prvenstveno u modelu podataka i načinu pohrane, najveću svrhu pronalaze prikazivanjem veza među podacima. Postoje razni modeli grafa, a među njima najprimjenjeniji je model grafa sa svojstvima. S obzirom na vizualnu prirodu modela grafovskih baza podataka pojavljuje se potreba za korištenjem istih u izvještajnim sustavima.

Kako velike količine podataka brojni poslovni i informacijski sustavi pohranjuju u relacijskim bazama podataka, uviđajući prednosti grafovskih, stvoreni su razni načini migracije podataka u grafovske baze podataka. Migracija podataka može se podijeliti u dvije skupine: generička i specifična. Generička migracija ista je za sve relacijske modele, dok specifična implementira konverziju podataka za točno određenu domenu. Šire gledajući, automatizacijom procesa migracije otvara se mogućnost periodičnog pretakanja podataka za izvještajne sustave.

U svrhu vizualizacije grafova razvijeni su različiti alati. Možemo ih svrstati u tri arhitekturalne kategorije: alati bez direktnog spajanja na bazu podataka, alati s direktnim spajanjem na bazu podataka i samostalni alati. Često razliku među alatima čini i skup podržavanih tehnologija prikaza o kojima ovisi kvaliteta vizualizacije većih količina podataka. Svojom jednostavnošću ističe se JavaScript biblioteka Neovis.js. Neovis biblioteka temeljena je na Neo4j grafovskoj bazi podataka i Vis.js biblioteci za prikaz vizualizacije. Ovim se alatom jednostavno i brzo može implementirati izvještajni sustav, no efikasan prikaz veće količine podataka postaje upitan. Gephi alat, s druge strane, samostalna je aplikacija kojom je vizualizaciju moguće stvoriti i obraditi lokalno na računalu te kao takvu prikazati u sklopu neke web stranice.

U sklopu ovog rada, prikazana je implementacija izvještajnog sustava nad podacima koji predstavljaju znanstvene suradnje na projektima. Ova domena pogodna je za prikaz specifičnosti grafovske baze podataka, a podaci se migriraju iz relacijske baze podataka. Implementacija izvještajnog sustava izložena je na dvama primjerima: pomoću Neo4j +

Neovis platforme te automatizacijom naredbi Gephi alata. Na svakom od primjera prikazan je adekvatan način za migraciju podataka te opis implementacije izvještajnog sustava, a na kraju svakog primjera priložene su slike izgleda.

Summary

The reasons for the emergence of NoSQL databases are mostly based on the limitations of relational databases. Due to their flexible scheme, cost-efficiency and ability to efficiently store and process large amounts of data, NoSQL databases have grown in popularity. There are four categories of NoSQL databases, each suitable for different types and characteristics of domains. Among them, graph databases are particularly specific. Differing primarily in the data model and storage method, graph databases are widely appreciated for their application in displaying the connections between the data. Given their visual nature, graph database models are becoming essential in reporting systems.

As large amounts of data are stored in relational databases, various techniques of migrating to graph databases have been created. Data migration can be divided into two groups: generic and specific. Generic migration is the same for all relational models, while a specific one implements data conversion for a specific domain. Automating the migration process opens the possibility of periodic data streaming for the reporting systems.

Various tools have been developed for graph visualization. We can classify them into three architectural categories: tools without direct connection to a database, tools with direct connection to a database, and standalone tools. Frequently, the difference between the tools is the set of supported display technologies, on which depends the quality of visualization of large amounts of data. The JavaScript library Neovis.js stands out due to its simplicity. The Neovis library is based on the Neo4j graph database and the Vis.js visualization library. This tool can easily and quickly implement a reporting system, but it is not reliable for the efficient display of larger amounts of data. On the other hand, Gephi tool is an independent application that allows the user to create and process a visualization locally on a computer and display it as such within a website.

This paper presents the implementation of a reporting system on data representing scientific project collaborations. This domain is suitable for displaying the specifics of a graph database, while the data is migrated from a relational database. The implementation of the reporting system is presented in two manners: using the Neo4j + Neovis platform and by automating the commands of the Gephi tool. Each of them contains an adequate way to migrate data and a description of the implementation of the reporting system. Images of the layouts are attached at the end of both examples.

Životopis

Rođena sam 06.04.1995. u Splitu. Pohađala sam OŠ "Don Lovre Katića" u Solinu te Prirodoslovno-matematičku gimnaziju u Splitu. Akademske godine 2014./2015. upisala sam preddiplomski studij Matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu, kojeg sam završila 2018. godine. Iste godine upisujem diplomski studij Računarstvo i matematika na istom fakultetu.