SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 139

KORIŠTENJE BAZE NEO4J ZA POHRANU, OBRADU I VIZUALIZACIJU PODATAKA O IP ADRESAMA I DRUGIM MREŽNIM ARTEFAKTIMA

Marta Gračner

Zagreb, svibanj 2021.

Sadržaj

Uvod

U svijetu u kojem količina informacija i njihovi međusobni odnosi eksponencijalno rastu i gdje još uvijek prevladavaju tromi sustavi s relacijskim bazama podataka, pojavila se potreba za učinkovitijim upravljanjem podacima. Tu prazninu pokušavaju popuniti, između ostalog, graf baze podataka koje donose vidljivo bolje performanse i fleksibilnije upravljanje podacima. Graf baze podataka imaju mnogo prednosti u odnosu na relacijske baze podataka. Neke od tih prednosti su objektno orijentirano razmišljanje, ažuriranje podataka u stvarnom vremenu, fleksibilan model podataka i podatkovna infrastruktura pogodna za umjetnu inteligenciju. [1] Međutim, kao i bilo koji drugi sustav, i graf baze imaju nedostataka. Budući da su relativno nova pojava, još uvijek je teško pronaći stručnjake koji se bave upravljanjem graf bazama podataka. Također, manjkavost ovih baza leži u nedostatku standardizacije za njihove upitne jezike. Unatoč svim nedostacima, sve više programera primjećuje prednosti koje takve baze podataka nude, zbog čega njihova popularnost i zastupljenost u raznim sustavima svakim danom raste.

Korištenje graf baza podataka može biti osobito korisno u aplikacijama za obradu i analizu podataka o stanju u nekoj mreži. Kako bi se u potpunosti moglo razumjeti što se događa u pojedinoj mreži, osim poznavanja samih mrežnih artefakata, potrebno je poznavati i njihove međusobne odnose i zavisnosti. U obavještajnom radu o kibernetičkim prijetnjama glavna zadaća je otkriti ranjivosti sustava te analizom raznih događaja utvrditi moguće prijetnje. Ovdje bi upotreba graf baza podataka mogla olakšati razumijevanje sustava i njima pripadajućih mreža, jer bi identificiranje svih međuovisnosti olakšalo stručnjacima pronalaženje mogućih ranjivosti, prijetnji i puteva kojima bi napadači mogli ostvariti svoje zle namjere.

Svrha ovog rada je opisati graf baze podataka i detaljnije predstaviti svojstva Neo4j sustava za upravljanje graf bazama podataka te ukazati na prednosti korištenja graf baza podataka u sustavima koji rade s promjenjivim podacima. Kasnija poglavlja detaljnije će opisati aplikaciju za obradu datoteka dnevnika zapisa koja u pozadini koristi graf bazu podataka i Neo4j sustav. Budući da se izvori podataka u aplikaciji mogu lako mijenjati, cilj je bio napraviti implementaciju spremanja, ažuriranja i dohvaćanja podataka koja neće biti strogo povezana s jednom određenom strukturom podataka, nego će omogućiti promjenu

izvora podataka, a samim time i promjenu strukture podataka bez potrebe mijenjanja baze podataka i implementacije upravljanja podacima.

Podaci pohranjeni u graf bazi podataka prikladni su za vizualizaciju u obliku grafa. Stoga će posljednje poglavlje opisati način prikaza podataka u obliku grafa na sučelju web aplikacije. Podaci koji će se prikazati utvrdit će se na temelju filtara koje je korisnik odabrao ili ako nije odabran nijedan filtar, prikazat će se svi podaci iz baze.

Rad je strukturiran na sljedeći način. U prvom poglavlju opisan je pojam obavještajnog rada o kibernetičkim prijetnjama u čijem kontekstu je smještena aplikacija opisana u daljnjim poglavljima. Drugo poglavlje i njegova potpoglavlja opisuju graf baze podataka, Neo4j sustav za upravljanje graf bazama podataka i upitni jezik Cypher. Detaljnije su opisane karakteristike, prednosti i nedostaci nabrojanih pojmova. Treće poglavlje opisuje aplikaciju za obradu IP adresa. Naglasak je na graf bazi podataka koja se koristi u aplikaciji i na vizualizaciji podataka na sučelju web aplikacije. Na samom kraju trećeg poglavlja predloženi su daljnji koraci u nastavku rada na aplikaciji, a nakon toga slijedi zaključak.

1. Obavještajni rad o kibernetičkim prijetnjama

Kako raste veličina i složenost kibernetičkog prostora tako raste i broj ranjivosti sustava koji se u njemu nalaze i količina prijetnji kojima su ti isti sustavi izloženi. U obranu tih sustava staju stručnjaci u području obavještajnog rada o kibernetičkim prijetnjama. "Obavještajni rad o kibernetičkim prijetnjama je znanje utemeljeno na dokazima, koje uključuje kontekst, mehanizme, pokazatelje, implikacije i djelotvorne savjete o postojećoj ili novonastaloj prijetnji ili opasnosti za imovinu koja se može koristiti za donošenje odluka u vezi s reakcijom subjekta na tu prijetnju ili opasnost." [2] Zadaća obavještajnog rada o kibernetičkim prijetnjama ili kraće CTI-ja (od Cyber Threat Intelligence) je prikupljanje i analiza podataka o mogućim prijetnjama i napadima te smanjenje utjecaja prijetnji na razne sustave u kibernetičkom prostoru i posljedično stvaranje sigurnijih sustava.

Obavještajni rad uključuje pripremu, točnije identificiranje potencijalnih napadača, njihovih motivacija i akcija koje pomažu u sprječavanju napada, prikupljanje relevantnih informacija o napadima i napadačima, procesiranje prikupljenih podataka i transformiranje u prikladniji oblik, analizu podataka te dijeljenje dobivenih rezultata i zaključaka. [3]

Obavještajni rad se može podijeliti u tri kategorije. Prva kategorija, taktički obavještajni rad, fokusiran je na prikupljanje podataka o tehnikama, taktikama, procedurama i tehničkim alatima kojima se napadači mogu služiti. Drugu kategoriju čini operativni obavještajni rad kojem je zadaća identificirati moguće napadače i njihove motive te akcije koje mogu poduzeti u napadu na neki sustav. Posljednja kategorija je strateški obavještajni rad koji obuhvaća informiranje svih donositelja odluka o tome kako kibernetičke prijetnje i određeni globalni događaji utječu na sigurnost organizacije. [4]

Jedna od potencijalnih prijetnji na internetu su distribuirani napadi uskraćivanjem usluge (DDoS napadi) kojima je cilj privremeno onemogućiti legitimnim korisnicima pristup određenim sustavima ili resursima na internetu tako što se pokušava preopteretiti poslužitelj velikom količinom uzastopnih zahtjeva s raznih računala. Takvi napadi se mogu identificirati analizom zahtjeva prema tom poslužitelju. Na primjer, ako velika količina zahtjeva dolazi od jedne IP adrese ili manje skupine njih, ako puno zahtjeva pristupa samo

jednom resursu ili ako zahtjevi dolaze u neočekivano doba dana može se posumnjati na napad. [5]

Stručnjacima u području CTI-ja su na raspolaganju razni alati, formati, izvori informacija i radni okviri za dijeljenje informacija koji im pomažu u identificiranju ranjivost, prijetnji i sigurnosnih propusta u sustavima te napada i samih napadača, a ponekad čak i organizacija ili tvrtki koje su potencijalne mete napada. Opsežan popis dostupnih alata može se pronaći na web stranici [2]. Mnogi od njih mogu se koristi za dobivanje informacija o IP adresama, osobito o onima koje su potencijalno maliciozne ili kompromitirane. Neki od zanimljivijih su C&C Tracker, DNSTrails, FireHOL IP Lists i Ransomware Tracker.

S ciljem da se pomogne stručnjacima u prikupljanju podataka i procesiranju istih razvijena je aplikacija koja će biti detaljnije opisana u četvrtom poglavlju. Aplikacija ima funkcionalnost prikupljanja podataka o pojedinim IP adresama te dovođenja u određene odnose IP adrese i resurse kojima je ona pristupala na određenim poslužiteljima. S obzirom na to da su izvori podataka o pristupanim resursima i IP adresama datoteke dnevnika s određenih poslužitelja, prikupljeni podaci se dalje mogu koristiti za utvrđivanje i analizu distribuiranih napada uskraćivanjem usluge na tim poslužiteljima i za identificiranje potencijalnih napadača ili kompromitiranih računala. Aplikacija nastoji iz datoteka dnevnika izvući sve IP adrese koje su pristupale nekom poslužitelju te pomoću raznih resursa prikupiti što više informacija o tim adresama. Dobivene informacije se potom pohranjuju u bazu, a njihovom analizom se neke IP adrese mogu eliminirati s liste sumnjivih adresa. Graf baza podataka olakšava posao analize time što eksplicitno čuva sve odnose koji postoje između pojedinih mrežnih artefakata.

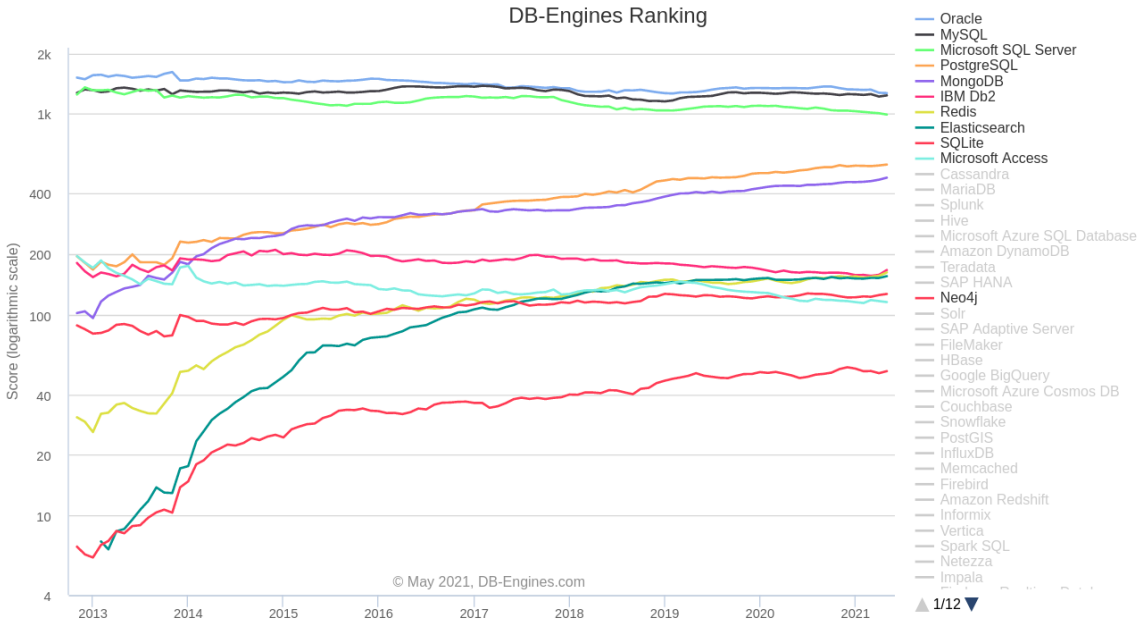
2. Graf baze podataka

Uz nagli porast količine podataka koja se koristi u različitim sustavima diljem interneta te nastajanjem novih odnosa između podataka i razvojem njihove kompleksnosti, pojavila se potreba za učinkovitijim pohranjivanjem i obradom te velike količine međusobno isprepletenih podataka. U pokušaju rješavanja navedenog problema i ubrzanja cjelokupnog poslovanja sve češće se u velike sustave uvode nerelacijske baze podataka kao na primjer graf baze podataka.

Graf baze podataka su vrsta NoSQL baza koje su nastale kao alternativa relacijskim bazama podataka u nadi efikasnijeg rješavanja specifičnih problema. Stoga je nastalo nekoliko vrsta NoSQL baza koje se mogu podijeliti u skupine na temelju struktura podataka koje koriste za čuvanje podataka, a koje nikako nisu tablice. Glavne vrste struktura podataka su dokument, parovi ključ-vrijednost, spremište širokog stupca i graf. Glavna prednost NoSQL baza nad relacijskim bazama je mogućnost čuvanja nestrukturiranih ili polustrukturiranih podataka te u određenim slučajevima brže spremanje i dohvaćanje podataka. Svaka vrsta NoSQL baza napravljena je na specifičan način za rješavanje specifičnih problema.

Najpopularniji sustavi za upravljanje graf bazama podataka su Neo4j, Microsoft Azure Cosmos DB, Oracle NoSQL Database, OrientDB, HyperGraphDB, GraphBase, Infinite Graph i AllegroGraph. [6] Od svih nabrojanih, najčešće je korišten sustav Neo4j. To je sustav otvorenog kôda dostupan široj javnosti koji se lako postavlja i koristi te ima dobro opisanu dokumentaciju s mnogo složenih i nesvakidašnjih primjera što omogućava bolje razumijevanje koncepata te na kraju bolje modeliranje složenih sustava. Upravo je iz tih razloga Neo4j sustav odabran za korištenje u aplikaciji koja će detaljnije biti opisana u trećem poglavlju.

Na slici Slika 2.1 prikazan je grafički prikaz popularnosti sustava za upravljanje bazama podataka u koji su uključene sve vrste baza. Iako je Neo4j kao najveći predstavnik sustava za upravljanje graf bazama daleko ispod najpopularnijih sustava i to tek na 19. mjestu, bitno je uočiti da graf njegove popularnosti raste puno brže od grafova prva tri sustava koji su svi relacijski. Međutim, ako nešto drastično ne promjeni ovakav trend rasta popularnosti proći će još dugo vremena dok graf baze ne dostignu relacijske baze.

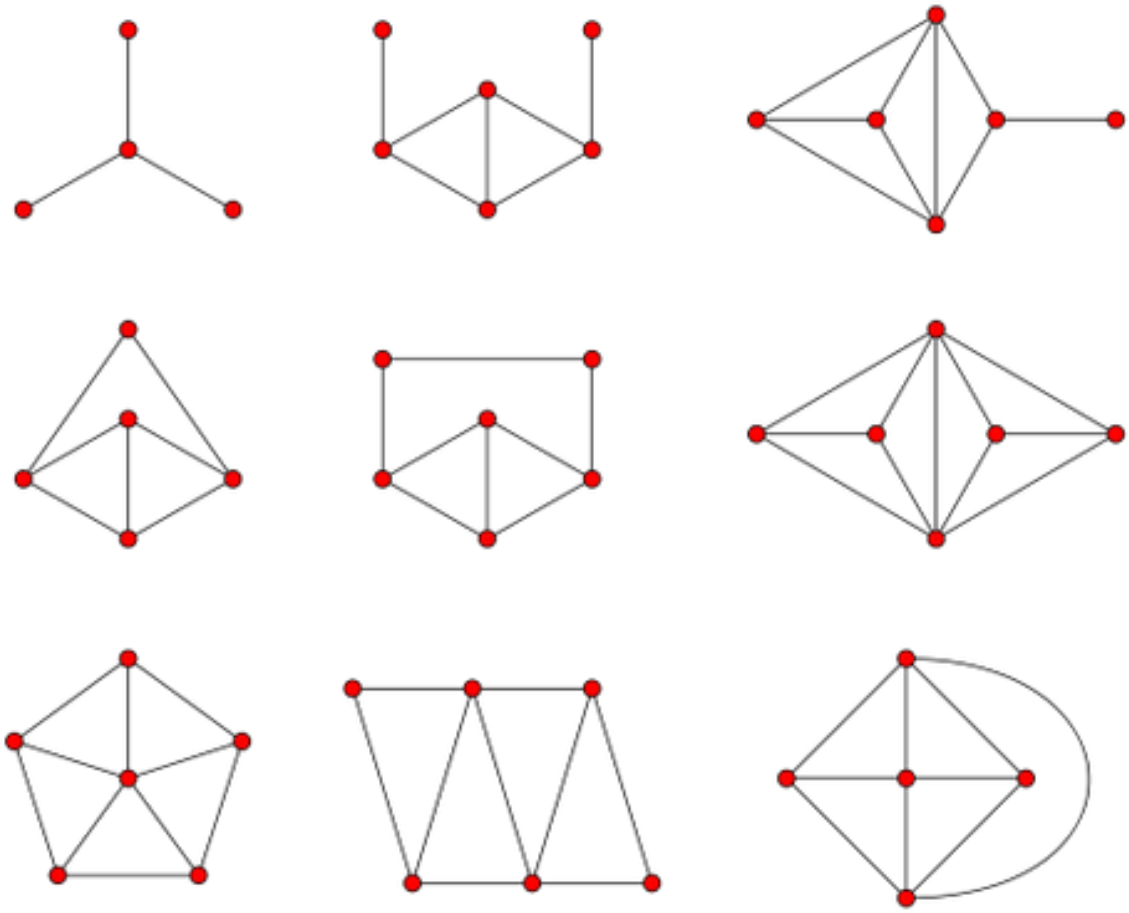


Slika 2.1 Popularnost Neo4j sustava u odnosu na druge sustave za upravljanje bazama podataka [7]

Graf baze podataka temelje se na teoriji grafova koja definira graf kao "uređeni par (V(G),E(G)) gdje je V(G) konačan neprazan skup elemenata koji se zovu čvorovi, a E(G) je konačan skup različitih neuređenih parova različitih elemenata skupa V(G) koji se zovu grane." [8] Graf se može predstaviti crtežom gdje su čvorovi prikazani kao točke ili krugovi, a grane kao linije između tih točaka. Ako grane imaju početni i završni čvor onda se radi o usmjerenom grafu, a u suprotnom riječ je o neusmjerenom grafu. Primjer nekoliko neusmjerenih grafova prikazan je na slici Slika 2.2.

Graf baze podataka se mogu koristiti u mnogim sustavima za pohranu raznih podataka. Neki od slučajeva kada je korištenje graf baza prikladno su: preporuke u stvarnom vremenu, reprezentacija društvenih mreža, otkrivanje pronevjera i lanaca pranja novca, pretraživanje zasnovano na grafovima, upravljanje računalnim mrežama, upravljanje identitetima, pristupima i dozvolama te mnogi drugi. [9] Graf baze su najprikladnije za spremanje podataka kod kojih povezanost ima jednaku ako ne i veću važnost od samih podataka, tamo gdje je potrebno znati kako i s čime su povezani određeni čvorovi te kakva svojstva imaju same veze i u kojem smjeru se podaci kreću.

Da graf bazama podataka raste popularnost i da njihove prednosti uočava sve više programera svjedoče radovi [10] i [11] slični ovome na temu graf baza podataka i usporedbe relacijskih i graf baza podataka.



Slika 2.2 Primjeri grafova [12]

2.1. Karakteristike i prednosti graf baza podataka

Graf baze podataka su vrsta NoSQL baza podataka koje se temelje na teoriji grafova. Stoga njihov model podataka čine čvorovi, veze i svojstva. Čvorovi predstavljaju objekte iz realnog svijeta te sadrže podatke koji opisuju te objekte. Veze služe za povezivanje čvorova te predstavljaju različite odnose u kojima se čvorovi mogu naći. Veze mogu biti dvosmjerne ili jednosmjerne te u tom slučaju imaju drugačije značenje ovisno koji čvor se gleda kao početni, a koji kao ciljni čvor. Između svaka dva čvora može postojati nijedna, jedna ili više veza. Svojstva se najčešće zadaju u obliku parova ključ-vrijednost te pobliže opisuju čvorove ili veze.

U odnosu na relacijske baze podataka graf baze podataka nude bolje performance kod dohvaćanja podataka i veću fleksibilnost kod unosa novih podataka. "Izvedba je konstantna jer je obilaženje susjednih čvorova određenog čvora prateći veze neovisno o veličini grafa." [1] Za razliku od relacijskih baza podataka, kod graf baza podataka nije

potrebno unaprijed poznavati strukturu spremljenih podataka što omogućava lagano dodavanje novih podataka bez kompromitiranja starih i bez narušavanja postojeće funkcionalnosti.

Odnosi između podataka su jednako važni kao i sam podaci, stoga je u graf bazama naglasak stavljen na veze te se one trajno spremaju u bazu. Suprotno tome kod relacijskih baza podataka, kod svakog upita veze se ponovno izračunavaju na temelju stranih i primarnih ključeva. S obzirom na to da su čvorovi međusobno direktno povezani vezama, uklanja se potreba primarnih i stranih ključeva. Ovo svojstvo graf baza podataka čini model baze jednostavnijim i ekspresivnijim. [13]

Podatke koji se nalaze u graf bazi podataka je lako vizualizirati pomoću modela baze u obliku neke vrste grafa te je intuitivno za razumjeti što predstavljaju pojedini čvorovi i u kakvim odnosima se oni nalaze.

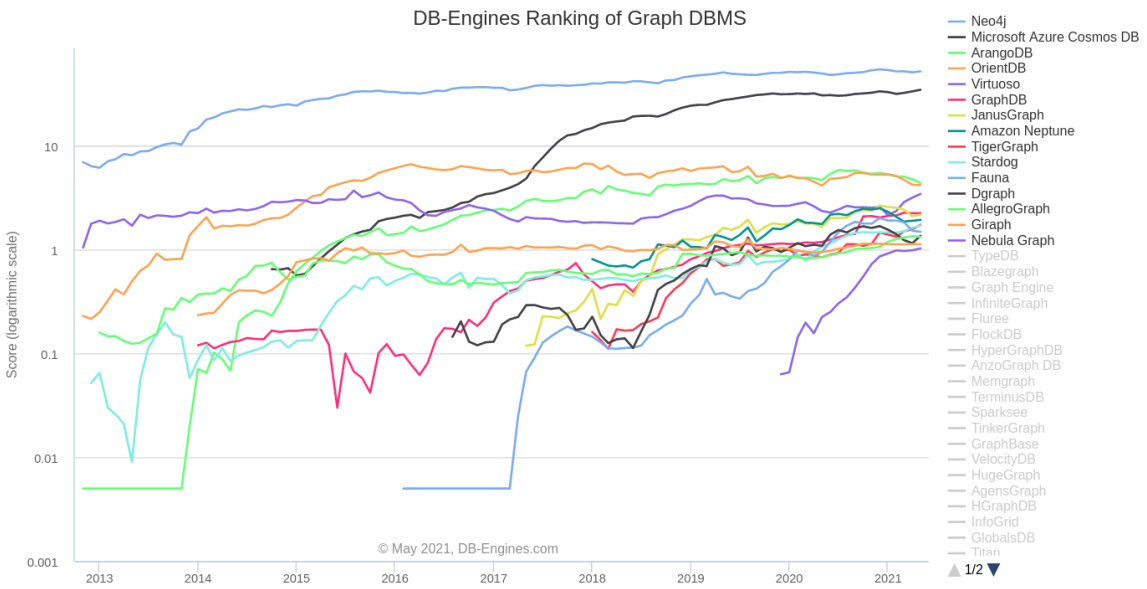
Struktura u obliku grafa omogućuje jednostavno i brzo kretanje od početnog do ciljnog čvora direktnim kretanjem između čvorova ili korištenjem nekog od algoritama za pretraživanje grafova kao što su pretraživanje u dubinu i pretraživanje u širinu. Time se uklanja dugotrajno spajanje i filtriranje tablica koje se koristi u klasičnim relacijskim bazama te se samim time štede vrijeme i računalni resursi. Također, traženje najkraćeg puta i računanje udaljenosti između pojedinih čvorova postaje trivijalno.

2.2. Sustav za pohranu podataka Neo4j

Neo4j je jedan od najpopularnijih sustava za upravljanje graf bazama podataka koji koristi upitni jezik Cypher za izvođenje upita nad bazom podataka. Nastao je početkom stoljeća zbog problema loših performansi na koje se može naići kod relacijskih baza podataka.

Da je Neo4j najpopularniji sustav za upravljanje graf bazama podataka svjedoči grafički prikaz na slici Slika 2.3. iz kojeg je jasno da je on na prvom mjestu još od 2013. godine. Od svih konkurenata najviše mu s približio sustav Microsoft Azure Cosmos DB.

Sustav oblikuje podatke u obliku grafa ne samo na konceptualnoj razini, nego i na razini memorijskog skladištenja. To omogućuju pokazivači spremljeni u svakom čvoru koji pokazuju na sve druge čvorove s kojima je on povezan što omogućuje brzu pretragu i kretanje po grafu.



Slika 2.3 Grafički prikaz popularnosti Pojedinih sustava za upravljanje graf bazama podataka [15]

Podaci u bazi su "dostupni putem softvera napisanog u drugim jezicima korištenjem Cypher-a kroz transakcijske HTTP krajnje točke ili putem binarnog protokola Bolt." [14]

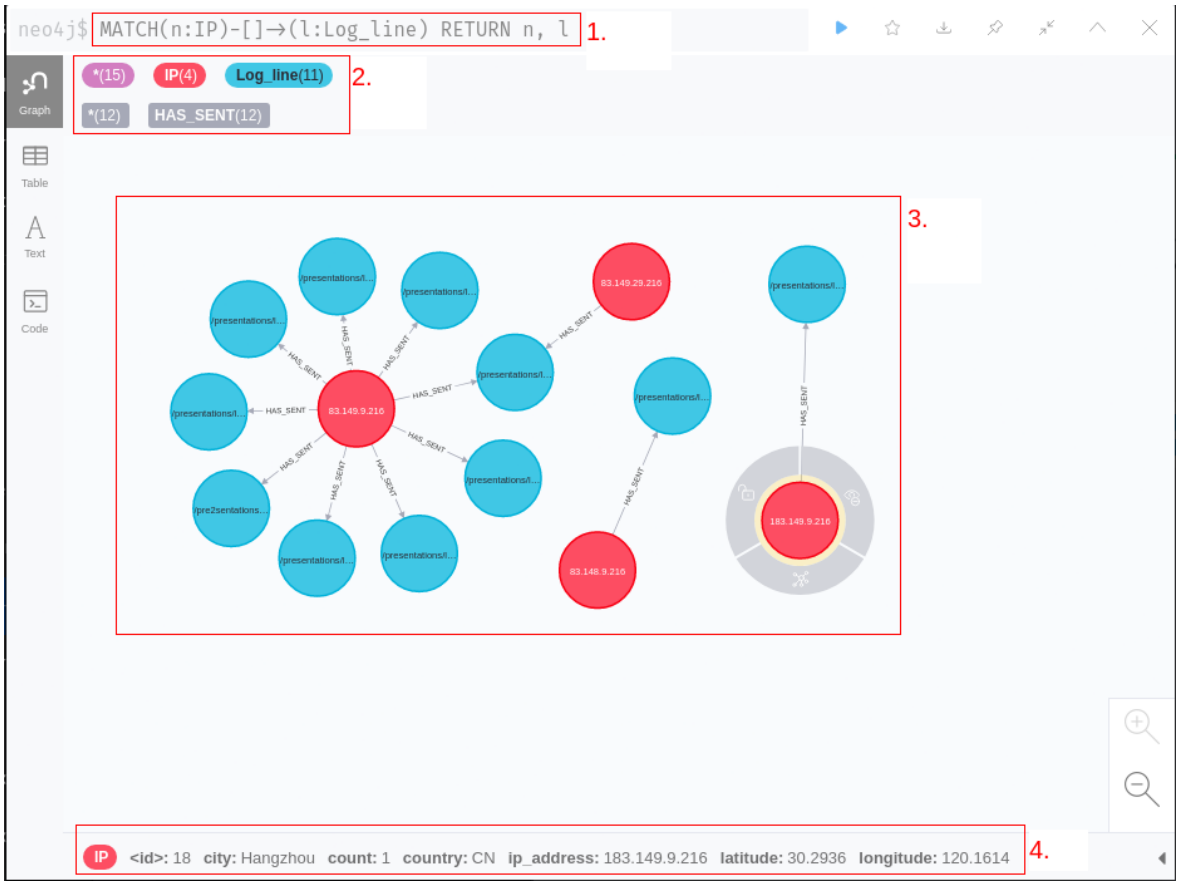
2.2.1. Karakteristike Neo4j sustava

Za izvođenje upita i pretraživanje grafa koriste se upitni jezik Cypher i algoritmi za obilazak čvorova u grafu. Cypher-om se grade upiti koji su intuitivni i lako čitljivi čak i za netehničko osoblje.

Model podataka nije unaprijed strogo definiran, nego se čvorovi i veze mogu dodavati ili brisati po potrebi bez mijenjanja neke sheme. Također, dodavanjem novih čvorova i veza nema znatnog usporenja sustava graf baze podataka, a podaci i promjene nad njima su vidljive instantno u stvarnom vremenu.

Nadalje, čvorovi i veze mogu imati svojstva koja pobliže opisuju objekte i odnose između njih te se zapisuju u obliku parova ključ-vrijednost. Ključ je ime svojstva te je tipa znakovni niz dok vrijednost može biti tipa broj, znakovni niz ili logička vrijednost. Čvorovi istog tipa ne moraju imati ista svojstva. Također, svaki čvor može imati više uloga te se shodno tome mogu označiti različitim labelama. Labele služe za grupiranje čvorova u skupine gdje svaki čvor može biti pripadnik više skupina. Sve veze obavezno imaju usmjerenje i tip s jasno definiranim početnim i ciljnim čvorovima. Unatoč tome vezama se može efikasno navigirati u oba smjera. [9]

Neo4j dolazi s ugrađenim Neo4j preglednikom koji ima ugrađeno korisničko sučelje u kojem se mogu dodavati, mijenjati, brisati i pregledavati podaci. Nakon svake operacije izvedene kroz Neo4j preglednik korisnik dobiva informaciju u promijenjenom stanju baze. Također, Neo4j preglednik omogućava vizualizaciju dohvaćenih podataka u obliku čvorova i veza te nudi mogućnost pregledavanja svih spremljenih svojstava za svaki pojedini čvor. Preglednik i službeni Neo4j pokretački programi koriste Bolt protokol za komunikaciju s Neo4j bazom podataka. [19] Na slici Slika 2.4 se može vidjeti sučelje Neo4j preglednika nakon izvođenja upita. Prvi crveni kvadrat označava upit izveden nad bazom podataka. Drugo označeno područje prikazuje brojnost pojedinih čvorova i veza koje je upit vratio te ukupan broj vraćenih čvorova i veza. Područje numerirano brojem tri označava vizualni prikaz čvorova i veza koji su rezultat izvršavanja upita. Čvorovi s različitim labelama ispunjeni su različitim bojama. Svaki od čvorova se može označiti nakon čega se u četvrtom području ispišu pripadna svojstva i njihove vrijednosti.



Slika 2.4 Sučelje Neo4j preglednika

"Neo4j podržava sva ACID pravila" [6] što osigurava konzistentnost podataka. Atomarnost znači da se cijela transakcija odvija odjednom ili se ništa ne događa. [16] Konzistentnost označava da su novo uneseni podaci validni te da je baza ostala u konzistentnom stanju.

Izolacija osigurava da transakcije koje se odvijaju istovremeno ne utječu jedna na drugu. [17] Izdržljivost znači da se nijedna transakcija ne može izgubiti čak ni u slučaju kvarova.

Pored ostalog, podacima u bazi se može pristupiti iz raznih programskih jezika korištenjem REST aplikacijskog korisničkog sučelja. Neo4j podržava pokretačke programe za programske jezike .Net, Java, JavaScript, Go i Python. Također, razni vanjski programeri održavaju pokretačke programe koje mogu koristiti PHP, Ruby, R, Erlang, Clojure, C i C++. [18]

2.2.2. Upitni jezik Cypher

Cypher Query Language je upitni jezik Neo4j sustava koji gradi upite ulančavanjem klauzula s međusobnim prosljeđivanjem međurezultata. Napravljen je tako da je iz samog teksta upita jasno kakvi podaci se traže i kako bi trebao izgledati rezultat. Ulančavanjem traženih čvorova i veza može se s relativno malo teksta postići veoma precizan upit nad veoma kompleksno povezanim podacima.

Sintaksa upita se sastoji od klauzula, čvorova i veza koji mogu biti dodatno precizirani svojstvima i labelama. Par okruglih zagrada predstavlja čvor. Unutar zagrada se ne mora nalaziti ništa, to onda označava anonimni čvor. Ako se želi referencirati određeni čvor ili skupina čvorova onda se unutar zagrada mogu navesti labele i vrijednosti za pojedina svojstva koja imaju traženi čvorovi. Veza je predstavljena parom crtica (`--`) te opcionalno strelicom u nekom smjeru (`<--`, `-->`) i uglatim zagradama koje mogu sadržavati labele i svojstva slična kao i kod grafova.

```
(:Person {name: "John"})-[:WORKS_IN {role: "developer"}]->
(:Company {name: "Google"})
```

U prethodnom primjeru navedena su dva čvora, jedan s labelom *Person* koji predstavlja osobu imena John i drugi s labelom *Company* koji predstavlja tvrtku naziva Google. Između njih nalazi se veza koja spaja osobu i tvrtku u odnos: John radi u tvrtki Google gdje ima ulogu programera.

Najvažnije klauzule koje se u Neo4j sustavu koriste za manipulaciju podacima su CREATE, MATCH, WHERE i RETURN.

Za stvaranje novih podataka se koristi klauzula CREATE. U najčešćem slučaju nakon ključne riječi CREATE dolazi čvor te opcionalno veza i drugi čvor s kojim je prvi povezan. U slučaju stvaranja više objekata odjednom, može se koristiti više CREATE

klauzula ili se objekti mogu odvojiti zarezima. [20] U sljedećem primjeru stvara se jedan čvor s labelom Person i jednim svojstvom zadanim u obliku ključ-vrijednost.

```
CREATE (:Person {name: "Mary"})
```

Za dohvaćanje podataka se koristi klauzula MATCH kojoj se predaje uzorak čvorova i veza koji tražimo.

```
MATCH (p:Person)-[:WORKS_IN]->(:Company {name: "Google"})
RETURN p.
```

Ovdje se koristi `p` kao varijabla koja je vidljiva samo u ovoj transakciji i označava sve čvorove koji zadovoljavaju ostale uvjete. Ovaj upit dohvaća sve čvorove koji predstavljaju osobe koji rade u tvrtki naziva Google.

Kod dodavanja novih informacija u postojeće podatke koristi se kombinacija prethodno navedenih klauzula. Nakon ključne riječi MATCH dolazi uzorak koji označava čvorove i veze koje se želi dohvatiti, iza toga dolazi ključna riječ CREATE i uzorak čvorova i veza koje se želi stvoriti. U sljedećem primjeru se dohvaća osoba imena John Smith te se stvara čvor koji predstavlja grad imena London te u konačnici stvara se veza između prethodno dohvaćenog i novostvorenog čvora koja označava da John Smith živi u Londonu.

```
MATCH(p:Person {name: "John Smith"})
CREATE(c:City {name: "London"})
CREATE(p)-[:LIVES_IN]->(c)
```

Podaci koje se dohvaća se mogu filtrirati upotrebom klauzule WHERE u kombinaciji s klauzulom MATCH. Filtrirati se može po jednom ili više svojstava.

```
MATCH(p:Person) WHERE p.name="John" RETURN p.
```

Iz same sintakse upita može se zaključiti da upit dohvaća sve čvorove koji predstavljaju osobe s imenom John. Isti rezultat može se dobiti izvođenjem sljedećeg upita:

```
MATCH(p:Person {name: "John"}) RETURN p
```

Uz prethodno navedene klauzule, često se koriste i sljedeće: ORDER BY, LIMIT, MERGE i DELETE. Potpuni popis podržanih klauzula u upitnom jeziku Cypher dostupne su na poveznici [21].

Cyper podržava izvođenje predefiniranih funkcija nad dijelovima grafa. Funkcije se mogu podijeliti u skupine kao na primjer predikatne funkcije, skalarne funkcije, matematičke funkcije i mnoge druge.

Neke od zanimljivijih i češće korištenih funkcija su length() koja vraća dužinu zadanog puta, funkcija avg() vraća aritmetičku sredinu skupine numeričkih vrijednosti, funkcija count(*) vraća broj redova koji zadovoljavaju zadani uzorak, funkcija max() i min() vraćaju maksimalnu tj. minimalnu vrijednost iz zadane liste vrijednosti, funkcija sum() vraća zbroj zadanih vrijednosti. Ako je potrebno saznati sve labele koje su dodijeljene svim ili nekim određenim čvorovima u bazi, dovoljno je izvršiti funkciju labels(). Slično tome za dobivanje liste svih ključeva iz liste parova ključ-vrijednost potrebno je izvršiti funkciju keys().

Još jedna zanimljiva funkcionalnost koju nudi Cypher upitni jezik je pronalaženje najkraćeg puta između dva čvora. Moguće je odrediti donju i gornju granicu kod traženja duljine puta. Sljedeći upit vraća duljinu najkraćeg puta između čvorova koji predstavljaju osobu imena Kevin Bacon i osobe imena Meg Ryan gdje je donja granica duljine puta 1, a gornja granica 20.

```
MATCH p=shortestPath( (bacon:Person {name:"Kevin Bacon"})-
[*1..20]-(meg:Person {name: "Meg Ryan"}) )
RETURN p
```

Isti rezultat može se dobiti izostavljanjem donje granice. Tada bi gornji upit izgledao ovako:

```
MATCH p=shortestPath( (bacon:Person {name:"Kevin Bacon"})
-[*..20]-(meg:Person {name: "Meg Ryan"}) )
RETURN p
```

Potpuni popis funkcija u Cypher upitnom jeziku je dostupan na web stranici [22].

3. Razvoj baze za podršku aplikaciji za procesiranje podataka o IP adresama

Pozadina aplikacije je implementirana u programskom jeziku Python uz korištenje radnog okvira Django. Aplikacijsko sučelje je napisano u programskom jeziku JavaScript uz korištenje radnog okvira Angular. Za trajno spremanje podataka koristi se graf baza podataka i Neo4j sustav za upravljanje navedenom bazom. Također, za spremanje podataka o korisnicima i podataka o učitanim datotekama koristi se relacijska baza podataka i PostgreSQL sustav za upravljanje relacijskim bazama podataka. Za razvoj aplikacije su upotrebljene radne okoline PyCharm, Visual Studio Code i Neo4j Desktop.

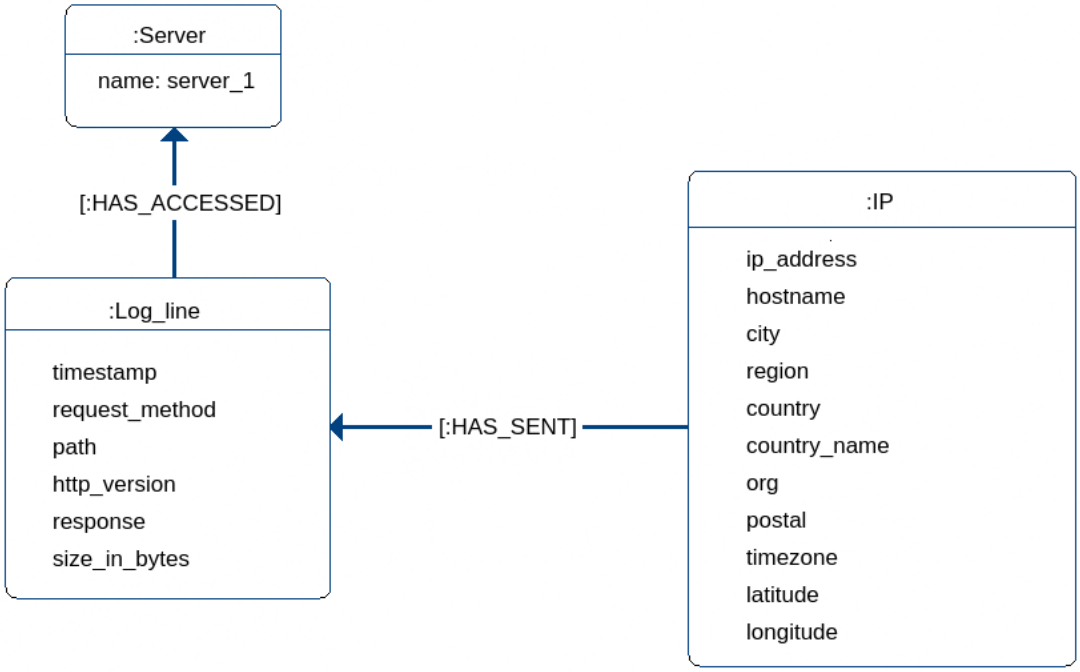
Za pokretanje aplikacije potrebno je prethodno instalirati PostreSQL, Python3, Angular i Neo4j. Također potrebno je inicijalizirati obje baze. Aplikacija zahtjeva da PostgreSQL baza ima naziv *cti_db* i da baza pokrenuta u Neo4j-u ima naziv korisnika *neo4j* i zaporku *password*. Baze trebaju biti pokrenute na predefiniranim priključcima. Samu aplikaciju moguće je pokrenuti na operacijskom sustavu Linux izvođenjem skripte *start_development_server.sh* koja se nalazi u korijenskom direktoriju projekta, a detaljniji opis potrebnih preduvjeta za pokretanje aplikacije nalazi se u datoteci README.md koja je također u korijenskom direktoriju projekta.

3.1. Model baze podataka

Podaci koji se spremaju u bazu dolaze iz dvije vrste izvora. Jedan izvor su datoteke dnevnika koje korisnik unosi kroz sučelje aplikacije zajedno s imenom po kojem se identificira poslužitelj. Aplikacija očekuje Apache datoteke dnevnika koje su oblikovane u formatu Common Log Format. Iz svake linije kôda izvlače se sljedeći podaci: IP adresa, vremenska oznaka pristupanja poslužitelju, metoda korištena u zahtjevu, putanja do traženog resursa, verzija HTTP protokola, odgovor poslužitelja i veličina odgovora u bajtovima. Drugi izvor je servis koji nudi informacije o IP adresama. Jedan od takvih servisa je IPinfo koji je dostupan kroz biblioteku u Python-u. IPinfo pruža sljedeće informacije o svakoj IP adresi: ime hosta, grad u kojem se nalazi računalo, regija u kojoj se nalazi računalo, država u kojoj se nalazi računalo, puno ime države, autonomni sustav

kojem pripada IP adresa, poštanski broj, vremenska zona, geografska širina i geografska dužina.

Navedeni podaci se mogu grupirati u tri skupine te će posljedično biti spremljeni u tri vrste čvorova. Prva vrsta sadrži sve podatke o poslužitelju, u ovom slučaju je to samo ime poslužitelja. Druga vrsta čvorova predstavlja pojedini zapis iz datoteke dnevnika te sadrži sve podatke dobivene iz tog zapisa bez vrijednosti IP adrese. IP adresa i sve informacije o njoj spremljene su u treću vrstu čvorova. Čvorovi IP adresa direktno su povezani s čvorovima zapisa iz dnevnika čije zahtjeve su slale prema nekim poslužiteljima, a čvorovi zapisa iz dnevnika su direktno povezani s čvorovima poslužitelja prema kojima su ti zahtjevi bili poslani. Opisani model baze je prikazan na slici Slika 3.1.



Slika 3.1 Model baze podataka

Upotrebom dodatnih izvora informacija moglo bi se za svaku IP adresu saznati tko je davatelj internetskih usluga, da li je poslužitelju pristupila koristeći Tor mrežu, virtualnu privatnu mrežu, Proxy poslužitelje i mnoge druge korisne informacije. Također, koristeći izvore informacija specifične za obavještajni rad o kibernetičkim prijetnjama, pojedine IP adrese iz datoteka dnevnika se može unaprijed označiti kao sumnjive ako se nalaze na popisu nekih malicioznih adresa.

3.2. Komunikacija između aplikacije i baze

Neo4j sustav podržava pokretačke programe za pojedine programske jezike, između ostalog i za programski jezik Python koji su održavani od strane samog Neo4j-a ili od strane zajednice. Za potrebe ove aplikacije korišten je pokretački program imena neo4j koji je službeno održavan od strane Neo4j-a. Pokretački program koristi binarni protokol za pristupanje bazi te omogućava izgradnju upita za dohvaćanje, spremanje, ažuriranje i brisanje podataka. [23] Osim njega u Python-u su dostupna još dva pokretačka programa, jedan imena Py2neo, a drugi neomodel. Oba su održavana od strane zajednice, a ne od samog Neo4j-a. Py2neo je u ključnim stvarima kao što je izvođenje upita sličan neo4j pokretačkom programu. Neomodel ima dosta različitosti u odnosu na ostala dva programa jer je zamišljen da mapira objekte iz Python-a u čvorove iz baze i obrnuto. Dakle, u neomodelu se upiti ne pišu eksplicitno nego se stvaraju objekti koje će on samostalno pretočiti u čvorove. Glavni nedostatak ovog pristupa je potreba poznavanja strukture podataka pri kreiranju klasa i metoda za manipuliranje podacima, a upravo to je ključni problem kojeg se nastoji zaobići.

Zbog toga što ga održava sam Neo4j, zato što za njega postoji najopsežnija dokumentacija te zbog eksplicitnog navođenja upita, za potrebe ovog projekta odabran je pokretački program neo4j koji se u aplikaciju dodaje putem Python-ovog upravitelja paketa pip-a izvođenjem sljedeće naredbe: `python -m pip install neo4j`.

Povezivanje prema bazi je ostvareno kroz klasu implementiranu u Pythonu. Programski kôd za navedenu klasu preuzet je s web stranice [24]. Napravljene su manje promjene u posljednjim linijama metode kako bi vraćeni rezultat bio u obliku liste. Pri svakom pozivanju metode za izvođenje upita, osim samog upita, metodi se predaju podaci za autentifikaciju, te se pri izlasku iz metode konekcija prema bazi zatvara. Ako je upit izvršen uspješno, metoda će vratiti listu čvorova i veza koje je dohvatila, stvorila ili ažurirala, a u suprotnom će u terminal ispisati tekst iznimke.

Definirane su metode za izvođenje raznih upita nad bazom. Sve metode koriste prethodno navedenu klasu za ostvarivanje konekcije prema bazi.

3.3. Unos i pohrana podataka

Korisnik unosi podatke u bazu kroz sučelje web aplikacije tako da odabere datoteku dnevnika čije podatke želi spremiti te navede naziv poslužitelja kojem datoteka pripada. Aplikacija se brine o tome da se datoteka do kraja obradi, točnije da se iz svake linije datoteke raščlane potrebni podaci, oblikuju u pravilan format te da se pozovu određene metode za spremanje čvorova i stvaranje veza između njih zajedno sa svim njihovim svojstvima.

Za stvaranje novih čvorova postoje dvije metode. Jedna koja stvara čvor neovisno o drugim podacima koji se tog trena nalaze u bazi, a druga koja stvara čvor samo ako takav isti čvor već ne postoji u bazi. Obje metode primaju dva argumenta: labelu koja će biti dodijeljena čvoru i niz svojstava koje će taj čvor sadržavati. S obzirom na to da se svojstva u čvorove spremaju u obliku parova ključ-vrijednost, drugi argument metode je tipa dictionary koji u programskom jeziku Python također sadrži parove podataka u formatu ključ-vrijednost te je stoga prikladan u ovoj situaciji. U nastavku je moguće vidjeti funkciju `create_node_if_not_exist` koja stvara čvorove kada isti takvi ne postoje u bazi. Većina tijela funkcije zadužena je za stvaranje Cypher upita. Na samom početku definirana je ključna riječ MERGE koja stvara čvor ako takav ne postoji u suprotnom vraća postojeći čvor. Nakon toga se kroz petlju u upit dodaje svako svojstvo zadano u dictionary-ju `node_properties`. Naposljetku upit zaključujemo ključnom riječi RETURN koja označava da će u slučaju uspješnog izvršavanja upita rezultat njegovog izvođenja biti stvoreni čvor. Upit predajemo metodi `query` koja ga izvodi nad bazom.

```
def create_node_if_not_exist(node_label, node_properties):
    first = True
    query_string = 'MERGE(node' + ':' + node_label + '{'
    for node_property in node_properties:
        if first:
            query_string += node_property + ': "'
            query_string += node_properties[node_property] + '"'
            first = False
        else:
            query_string += ', ' + node_property + ': "'
            query_string += node_properties[node_property] + '"'
    query_string += '}) RETURN node'

    db = Database(db_url, db_name, db_password)
```

```
    return db.query(query_string)
```

Za stvaranje veza između čvorova, također postoje dvije metode, te slično kao i kod metoda za stvaranje čvorova, jedna metoda stvara veze bezuvjetno, a druga ih stvara samo u slučaju kada one već ne postoje u bazi. Obje metode primaju 5 argumenata. Prva dva argumenta su labela i niz svojstava prvog čvora prema kojima se identificira početni čvor. Druga dva argumenta su labela i niz svojstava na temelju kojih će se identificirati završni čvor. Posljednji argument je labela veze koja ujedno i označava kakav odnos ona predstavlja.

3.4. Dohvaćanje podataka iz baze

Implementirane su razne metode za dohvaćanje podataka iz baze kojima je zadaća da specifičnim dijelovima aplikacije vraćaju tražene rezultate. Metoda `get_all_ips()` dohvaća sve IP adrese koje se nalaze u bazi zajedno sa svim njihovim detaljima te se rezultirajuća lista filtrira po određenim kriterijima u pozivajućoj metodi. Filtrirani podaci se prikazuju na sučelju web aplikacije u obliku tablice.

Metoda `get_all_server_names()` dohvaća imena svih poslužitelja koji se nalaze u bazi. Dobiveni podaci se prikazuju na sučelju u obliku liste te je moguće izabrati jedan ili više njih na temelju čega će se korisniku ispisati filtrirani podaci.

Metoda `get_by_ip()` prima jedan argument, a to je vrijednost IP adrese. Metoda vraća sve detalje o zadanoj IP adresi te se rezultat koristi za prikazivanje podataka na sučelju web aplikacije.

```
def get_by_ip(ip_address):
    query_string = 'MATCH(n) WHERE n.ip_address="'
    query_string += ip_address
    query_string += '" RETURN n'

    db = Database(db_url, db_name, db_password)
    response = db.query(query_string)
    return response
```

Na početku metode stvara se varijabla `query_string` te joj se dodjeljuje sadržaj upita koji će se izvesti nad bazom. Nakon toga se stvara instanca klase zadužene za komunikaciju s bazom te se poziva metoda `query` za izvođenje upita. Metoda `query`

vraća listu čvorova sa zadanom IP adresom te je to i konačan rezultat izvođenja metode `get_by_ip()`.

Posljednja metoda koja se koristi za dohvaćanje podataka iz baze naziva se `get_ips_by_server_name()`. Ova metoda za zadanu listu imena poslužitelja vraća sve IP adrese koje su pristupile barem jednom od zadanih poslužitelja. Dohvaćeni podaci se prikazuju na sučelju web aplikacije.

3.5. Vizualizacija podataka

Podaci iz baze podataka se mogu bez problema prikazati na sučelju web aplikacije u obliku tablica. Međutim, kada tih podataka ima puno te osobito ako su oni kompleksni, snalaženje u tim podacima postaje zamorno i dugotrajno. Jedna od prednosti spremanja podataka u obliku grafa je lakša vizualizacija podataka, a samim time i jednostavnije snalaženje.

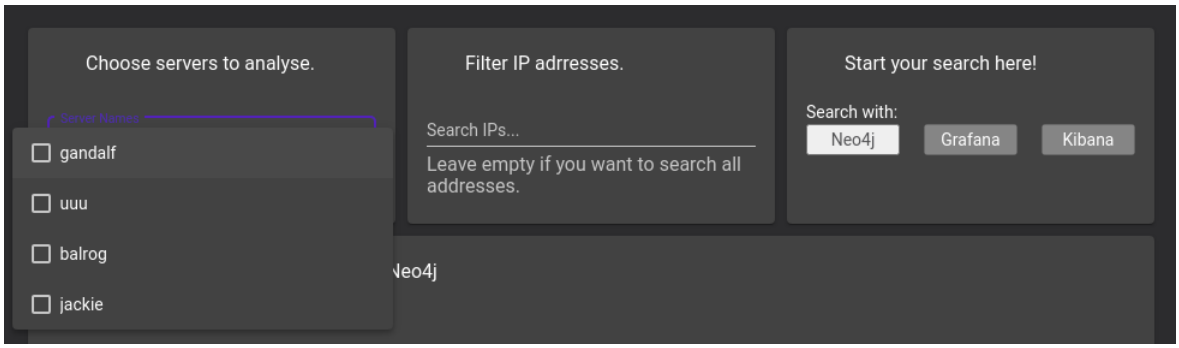
Postoji nekoliko JavaScript biblioteka za vizualizaciju podataka. Neke od njih su D3.js, Vis.js i Sigma.js. Također postoje biblioteke koje su napravljene specifično za vizualizaciju podataka iz Neo4j baze. Najpopularnije od njih su Neovis.js i Popoto.js. Njihova prednost je to što pružaju već gotovu funkcionalnost za dohvaćanje i formatiranje podataka iz baze, međutim veoma su ograničavajuće kod oblikovanja izgleda grafa te su i najmanje promjene kompleksne za implementirati. Najveći nedostatak tih dviju biblioteka je oskudna dokumentacija. Ono malo dokumentacije koje se može pronaći na internetu je ili zastarjelo ili opisuje najosnovnije primjere korištenja tih biblioteka. Nepogodne su za stvaranje kompleksnih vizualizacija i za bilo kakve naprednije prilagodbe. Osim navedenih postupaka, za vizualizaciju se mogu koristiti i samostalne aplikacije. Jedna od njih je već spomenuti Neo4j preglednik. Međutim, unošenje podataka kroz jednu aplikaciju, a vizualizacija tih istih podataka kroz drugu je poprilično nepraktično, osobito ako se treba izvršiti specifičan upit za prikaz željenih podataka. Iz tog razloga samostalne aplikacije za vizualizaciju podataka ovdje ne dolaze u obzir.

Od svih navedenih, D3.js je najpopularnija biblioteka koja nudi najviše mogućnosti i potpunu slobodu nad oblikovanjem vizualizacije. Ona nudi različite načine prikaza informacija, a jedna od njih je i prikaz informaciju u obliku grafa točnije u obliku kružića koji predstavljaju čvorove i u obliku linija koje predstavljaju veze. S obzirom na to da je sučelje aplikacije pisano uz pomoć radnog okvira Angular, D3.js biblioteka je implementirana koristeći Angular modul za D3.js. Modul se može instalirati putem

JavaScript-ovog upravitelja paketima npm-a izvođenjem naredbe: `npm install d3 && npm install @types/d3 --save-dev`.

Biblioteka očekuje dvije liste podataka. Prva lista sadrži objekte koji će biti mapirani u čvorove, a druga lista sadrži objekte koji će biti mapirani u veze. U objektima su sadržani svi detalji o čvorovima i vezama koji trebaju biti vidljivi korisnicima. Za dohvaćanje podataka koje biblioteka koristi implementirane su dvije metode. Metoda `create_d3_nodes()` dohvaća sve čvorove koje treba prikazati na sučelju zajedno sa svim njihovim detaljima, te ih pretvara u format koji D3.js biblioteka može razumjeti. Metoda `create_d3_links()`, slično kao i prethodna metoda, dohvaća sve veze koje će biti iscrtane na sučelju i formatira dobivene podatke u prikladan oblik. Obje metode primaju dva argumenta po kojima će se filtrirati podaci. Prvi argument je lista poslužitelja, a drugi lista IP adresa.

Na sučelju web aplikacije na stranici analyse nalaze se redom: padajuća lista za odabir nijednog, jednog ili više poslužitelja, okvir za unos teksta gdje se mogu unijeti vrijednosti IP adresa i gumbi. Klikom na Neo4j gumb iz baze se dohvaćaju podaci koji će biti filtrirani prema vrijednostima unesenim kroz padajuću listu i okvir za tekst. Dohvaćeni podaci se predaju D3.js biblioteci te ih ona iscrtava na sučelje u obliku čvorova i veza. Isti ti podaci ispisani su u tablici ispod nacrtanog grafa. Podaci u tablici se mogu filtrirati po vrijednostima svih svojstava. Ako se u padajućoj listi ne odabere nijedan poslužitelj rezultat će biti istovjetan kao i da su se odabrali svi poslužitelji u listi. Slično tome ako se ne upiše nijedna IP adresa smatrat će se da korisnik želi prikazati sve IP adrese za odabrane poslužitelje. Dio sučelja aplikacije gdje se nalaze prethodno objašnjeni filtri prikazan je na slici Slika 3.2.

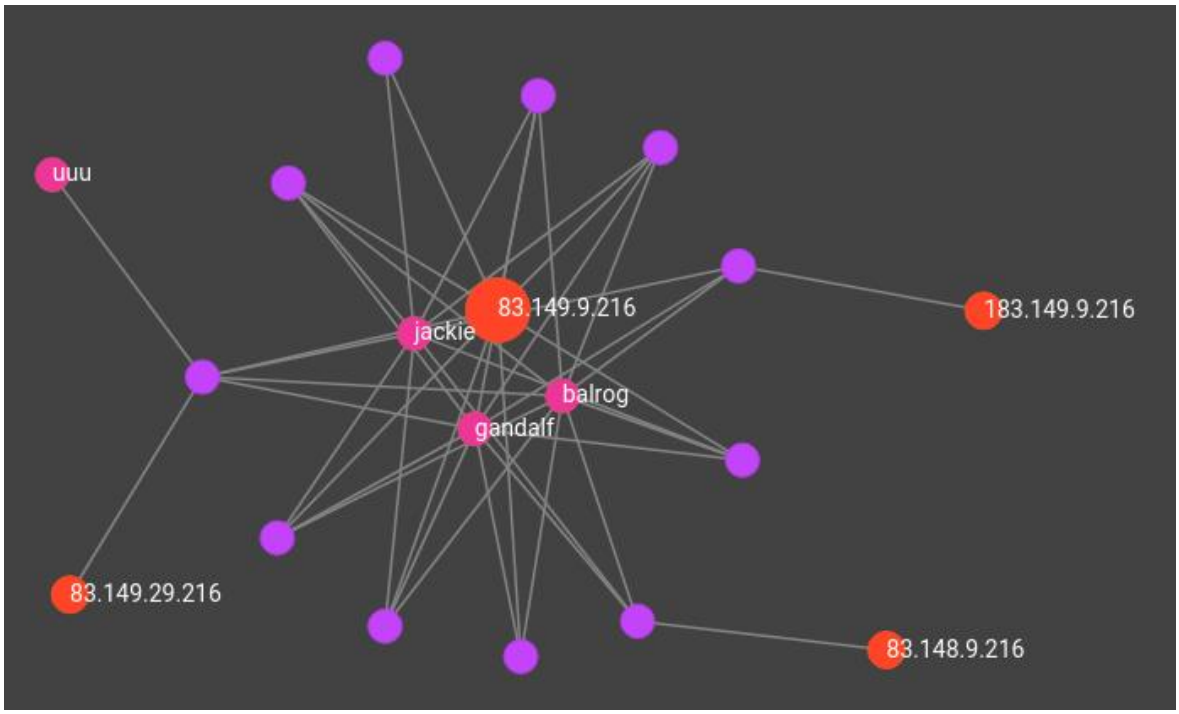


Slika 3.2 Dio sučelja web aplikacije s filtrima za vizualizaciju

Prvim filtrom se odabiru željeni poslužitelji. Klikom na polje *"Server names"* pojavljuje se padajuća lista s imenima svih poslužitelja koji se trenutno nalaze u bazi. Drugi filtar

dodatno izdvaja prethodno filtrirane podatke na temelju vrijednosti IP adresa. Konačno klikom na gumb "Neo4j" izvršit će se upit nad bazom sa zadanim filtrima te će se rezultantni graf nacrtati na sučelju ispod filtara i ti isti podaci će se prikazati u tabličnom obliku ispod grafa.

Rezultat dohvaćanja podataka nad jednom instancom baze prikazan je na slici **Error! Reference source not found.** gdje su različite vrste čvorova obojene različitom bojom. IP adrese su crvene, zapisi iz dnevnika ljubičaste, a poslužitelji ružičaste boje. Ovdje su čvorovi grupirani i obojeni prema labelama, međutim grupiranje se može izvesti prema nekom drugom argumentu. Na primjer čvorovi se mogu grupirati prema njihovoj pripadnosti nekom poslužitelju, tada bi na primjer svi zapisi iz dnevnika koji su proizašli iz nekog poslužitelja bili obojeni jednom bojom dok bi zapisi iz dnevnika nekog drugog poslužitelja bili obojeni drugom bojom.



Slika 3.3 Vizualizacija podataka iz Neo4j baze

Nadalje, iznad poslužiteljskih i IP čvorova nalazi se tekst koji označava ime poslužitelja ili vrijednost IP adrese. Taj tekst može biti vidljiv kao što je kod poslužiteljskih i IP čvorova ili se ne mora uopće prikazivati kao što je to slučaj kod čvorova zapisa iz dnevnika. Također, tekst koji se prikazuje je moguće definirati putem funkcije što znači da tekst može biti bilo što. Čvorovi se mogu povlačenjem po ekranu prerasporediti, a lebdenjem mišem nad pojedinim čvorom prikazuje se popis svih svojstava koje taj čvor posjeduje u

obliku parova ključ-vrijednost. D3.js također nudi mogućnost promjene veličine čvorova i veza. Na slici 7. vidljiv je središnji IP čvor koji je očito veći od ostalih IP čvorova. Ovdje je veličina IP čvorova definirana na temelju broja veza koje proizlaze iz njih. Međutim, veličinu je moguće definirati na temelju nekog drugog argumenta ili čak na temelju cijele funkcije.

3.6. Budući rad

Kao i u svakom drugom sustavu i u ovoj aplikaciji se može pronaći mjesta za napredak. Jedno od mogućih poboljšanja je implementacija ažuriranja podataka u bazi kroz sučelje aplikacije. Za sada je moguće samo stvarati nove čvorove i veze, ali nije moguće ažurirati već postojeće dodavanjem novih svojstava. U slučaju kada korisnik želi dodati novo svojstvo čvoru koji se već nalazi u bazi, korisnik će morati stvoriti potpuno novi čvor sa svim željenim svojstvima ili će morati direktno pristupiti bazi i ručno izmijeniti podatke. Ovakva funkcionalnost bi se mogla ostvariti implementiranjem nove metode koja bi za navedene podatke pronašla odgovarajuće čvorove te bi dodala svojstva koja je korisnik unio kroz aplikaciju.

Također, aplikacija je u mogućnosti stvarati samo tri vrste čvorova: IP adresu, zapise iz dnevnika i poslužitelje. Ako bi korisnik u bazu htio unijeti novu vrstu informacije, morao bi to napraviti, kao i u prethodnom slučaju, direktnim pristupanjem bazi i izvođenjem pravilnog upita. Funkcionalnost za stvaranje proizvoljnog tipa čvorova s proizvoljnim brojem svojstava je već implementirana u aplikaciju, međutim korisniku nije omogućeno iskoristiti tu funkcionalnost kroz sučelje aplikacije. Ova značajka bi se mogla ostvariti definiranjem obrasca kroz koji bi korisnik mogao puniti podatke. Ako se ovo omogući, trebalo bi također izmijeniti implementaciju vizualizacije podataka tako da su vidljive i informacije koje korisnik unosi kroz obrazac. Uz to, bilo bi zanimljivo implementirati dodatne filtre prema kojima korisnik može preciznije odrediti što želi vidjeti.

Promjena koja ne bi nužno bila vidljiva korisnicima, ali bi olakšala posao programerima je preoblikovanje modela podataka koji je prikazan na slici 1. Mogućnost promjene izvora podataka o IP adresama otvara mogućnost za bolje grupiranje podataka u specifičnije čvorove koji se onda posljedično mogu logičnije povezati u razne odnose.

Uključivanje informacija iz raznih izvora dovelo bi do najvećeg napretka u korisnosti aplikacije. Osobito ako se ti izvori specijaliziraju za objavljivanje informacija vezanih uz

CTI. Kada bi se podaci o raznim znanim prijetnjama i napadima mogli usporediti i povezati s podacima iz baze, učinkovitost analize i uporabljivost aplikacije bi znatno porasla. Slično tome, važno je uključiti razne izvore specijalizirane za objavljivanje informacija o IP adresama i pripadnim mrežnim artefaktima, jer što je više informacija o IP adresama dostupno to će biti lakše uočiti nepravilnosti.

Zaključak

Relacijske i graf baze podataka i sve vrste NoSQL baza imaju svoje prednosti i mane. Za svaku vrstu mogu se navesti neki primjeri gdje će upravo ta vrsta biti najbolji izbor za tražene zahtjeve, možda zbog najboljih performansi ili zbog najmanje složenosti. Pri odabiru vrste baze podataka koja će se koristiti u nekom sustavu, treba imati na umu zahtjeve koje moraju ispunjavati aplikacija i baza podataka, kako su strukturirani podaci koji se pohranjuju u bazu podataka i kako se karakteristike svake baze uklapaju u kontekst sustava. Na kraju će odabrana baza biti ona koja će imati najbolje performanse i koju će biti najlakše modelirati.

U kontekstu aplikacije za obradu IP adresa u kojoj se potiče upotreba što više različitih izvora informacija, koji će zajedno prikupiti puno podataka o svakoj pojedinoj IP adresi, prikladno je koristiti graf bazu podataka, jer je u tom slučaju lako pohraniti sve te podatke bez intervencije programera. Osim toga, za potrebe obrade IP adresa i ostalih mrežnih artefakata korisno je podatke spremati u bazu koja daje veliki značaj odnosima između podataka. Glavni nedostatak upotrebe relacijskih baza podataka u ovom sustavu je potreba ze ručnom promjenom modela baze podataka s pojavom svake nove vrste podatka.

Vizualizacija podataka na sučelju aplikacije, osobito u obliku grafa, ima smisla samo kada su ti isti podaci već pohranjeni u strukturi koja nalikuje grafu, što je još jedna prednost korištenja graf baza podataka. Bez njih vizualizacija podataka ne bi bila moguća jer je pretvaranje tablica u graf besmislen posao.

Literatura

[1] Wu M., *What Are the Major Advantages of Using a Graph Database?*, TigerGraph, (2019, studeni). Poveznica: https://www.tigergraph.com/blog/what-are-the-major-advantages-of-using-a-graph-database/; pristupljeno 25. svibnja 2021.

[2] Balaji P, *Cyber Threat Intelligence Tools For Security Professionals – 2021*, Soc Investigation, (2021, veljača). Poveznica: https://socinvestigation.com/cyber-threat-intelligence-tools-for-security-professionals-2021/; pristupljeno 27. svibnja 2021.

[3] Baker K., *What is Cyber Threat Intelligence?*, CrowdStrike, (2021, veljača). Poveznica: https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/; pristupljeno 24. svibnja 2021.

[4] CREST, *What is Cyber Threat Intelligence and how is it used?*, (2019), str. 10. Poveznica: https://www.crest-approved.org/wp-content/uploads/CREST-Cyber-Threat-Intelligence.pdf; pristupljeno 24. svibnja 2021.

[5] *What is a DDoS attack?*, Cloudflare. Poveznica: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/; pristupljeno 24. svibnja 2021.

[6] Tutorialspoints, *Neo4j - Overview*, Tutorialspoint, (2019, prosinac). Poveznica: https://www.tutorialspoint.com/neo4j/neo4j_overview.htm; pristupljeno 24. svibnja 2021.

[7] *DB-Engines Ranking - Trend Popularity*, DB-Eengines. Poveznica: https://db-engines.com/en/ranking_trend; pristupljeno 27. svibnja 2021.

[8] Aleksić Lampert T., *Diskretna Matematika*, (2014). Poveznica: https://imi.pmf.kg.ac.rs/moodle/file.php/17/vezbe_grafovi_1_1.pdf; pristupljeno 27. svibnja 2021.

[9] Neo4j, *What is a Graph Database?*, Neo4j. Poveznica: https://neo4j.com/developer/graph-database/; pristupljeno 23. svibnja 2021.

[10] Šestak, M. *Graf baze podataka*. Završni rad. Sveučilište u Zagrebu Fakultet organizacije i informatike Varaždin, 2014.

[11] Brica, M. *Usporedba graf i relacijskih baza podataka*. Diplomski sat. Sveučilište Josipa Jurja Strossmayera u Osjeku Fakultet Elektrotehnike, računarstva i informacijskih tehnologija, 2018.

[12] Eppstein D, *Forbidden line subgraphs.svg*, Wikipedia, (2007, siječanj). Poveznica: https://en.wikipedia.org/wiki/File:Forbidden_line_subgraphs.svg; pristupljeno 27. svibnja 2021.

[13] Merkl Sasaki, B., *Graph Databases for Beginners: Why Graph Technology Is the Future*, Neo4j, (2018, srpanj). Poveznica: https://neo4j.com/blog/why-graph-databases-are-the-future/; pristupljeno 18. svibnja 2021.

[14] *Neo4j*, Wikipedia, (2021, svibanj). Poveznica: https://en.wikipedia.org/wiki/Neo4j; pristupljeno 24. svibnja 2021.

[15] *DB-Engines Ranking - Trend of Graph DBMS Popularity*, DB-Eengines. Poveznica: https://db-engines.com/en/ranking_trend/graph+dbms; pristupljeno 27. svibnja 2021.

[16] GeeksforGeeks , *ACID Properties in DBMS*, GeeksforGeeks, (2021, travanj). Poveznica: https://www.geeksforgeeks.org/acid-properties-in-dbms/; pristupljeno 24. svibnja 2021.

[17] Chapple M., *The ACID Database Model*, Lifewire, (2021, siječanj). Poveznica: https://www.lifewire.com/the-acid-model-1019731; pristupljeno 24. svibnja 2021.

[18] Neo4j. Poveznica: https://neo4j.com/; pristupljeno 24. svibnja 2021.

[19] Neo4j. Poveznica: https://neo4j.com/docs/java-reference/current/java-embedded/bolt/; pristupljeno 24. svibnja 2021.

[20] *Patterns in practice*, Neo4j, (2021, svibanj). Poveznica: https://neo4j.com/docs/getting-started/current/cypher-intro/patterns-in-practice/#cypher-intro-patterns-in-practice; pristupljeno 24. svibnja 2021.

[21] Neo4j. Poveznica: https://neo4j.com/docs/cypher-manual/current/clauses/#header-reading-clauses; pristupljeno 27. svibnja 2021.

[22] *Functions*, Neo4j. Poveznica: https://neo4j.com/docs/cypher-manual/current/functions/; pristupljeno 26. svibnja 2021.

[23] *Using Neo4j from Python*, Neo4j. Poveznica: https://neo4j.com/developer/python/; pristupljeno 24. svibnja 2021.

[24] Yang S, *How to query Neo4j from Python*, Towards Data Science, (2020, srpanj). Poveznica: https://towardsdatascience.com/neo4j-cypher-python-7a919a372be7; pristupljeno 25. svibnja 2021.

Sažetak

Korištenje baze Neo4j za pohranu, obradu i vizualizaciju podataka o IP adresama i drugim mrežnim artefaktima

Svaki koristan sustav treba bazu podataka za čuvanje informacija. U te se svrhe najčešće koriste relacijske baze podataka, gdje se podaci pohranjuju u tabličnom obliku. Uz nagli porast količine podataka koje sustavi koriste, potrebna je učinkovitija pohrana podataka, osobito ako se radi o podacima čija se struktura često mijenja. Ove zahtjeve pokušavaju zadovoljiti graf baze podataka u kojima su podaci pohranjeni u obliku grafa. Ovaj će rad detaljnije opisati trenutno najpopularniji sustav za upravljanje graf bazama podataka Neo4j i upitni jezik Cypher koji se koristi u tom sustavu za dohvaćanje i manipuliranje podacima. Također, bit će opisana web aplikacija za obradu IP adresa u kojoj je korišten sustav Neo4j te o ostvarenoj vizualizaciji podataka na sučelju aplikacije.

Ključne riječi: Obavještajni rad o kibernetičkim prijetnjama, graf baze podataka, Neo4j, Cypher, vizualizacija, sustav za upravljanje bazom podataka

Summary

Using the Neo4j database to store, process, and visualize data about IP addresses and other network artifacts

Every useful system needs a database to store information. Relational databases are most commonly used for these purposes, where data is stored in tabular form. With the sharp increase in the amount of data that systems use, more efficient data storage is needed, especially if it is data whose structure changes frequently. These requirements try to satisfy the graph databases where the data is stored in the form of a graph. This paper will describe in more detail the currently most popular graph database management system Neo4j and the Cypher query language used in this system to retrieve and manipulate data. Also, a web application for processing IP addresses in which the Neo4j system was used will be described, as well as the realized visualization of data on the application interface.

Keywords: Cyber Threat Intelligence, graph databases, Neo4j, Cypher, visualization, database management system

Skraćenice

CTI	*Cyber Threat Intelligence*	obavještajni rad o kibernetičkim prijetnjama
DDoS	*Distributed Denial-of-service*	distribuirani napad uskraćivanjem usluge
IP	*Internet Protocol*	internetski protokola
HTTP	*Hypertext Transfer Protocol*	protokol za prijenos hiperteksta
ACID	*atomicity, consistency, isolation, durability*	atomarnost, konzistentnost, izolacija, izdržljivost
REST	*Representational state transfer*	reprezentacijski prijenos stanja