NoSQL baze podataka i usporedba njihovih performansi s relacijskim bazama podataka

Sabljo, Josipa

Sveučilište J.J. Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni diplomski studij Matematika i Računarstvo

Josipa Sabljo

NoSQL baze podataka i usporedba njihovih performansi s relacijskim bazama podataka

Diplomski rad

Osijek, 2023.

Sveučilište J.J. Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni diplomski studij Matematika i Računarstvo

Josipa Sabljo

NoSQL baze podataka i usporedba njihovih performansi s relacijskim bazama podataka

Diplomski rad

Mentor: izv. prof. dr. sc. Domagoj Matijević

Komentor: dr. sc. Mateja Đumić

Osijek, 2023.

Sadržaj

Uvod

Danas je gotovo nemoguće zamisliti život bez baza podataka. Podaci i informacije se gomilaju velikom brzinom iz raznih izvora i nije ih lako razumjeti zbog čega brojne tvrtke koriste sustave za upravljanje informacijama. Korištenje prikladnih baza podataka za prikupljanje i spremanje podataka o vlastitom poslovanju znatno im pomaže donositi pravovremene i korektne odluke kako bi napredovale i opstale na tržištu u ovom dobu kontinuiranih promjena, tehnoloških napredaka i velikog broja konkurenata.

Već su u doba starih Egipćana baze podataka korištene na danas primitivan, ali tada jedini mogući način. Egipćani su u to vrijeme provodili popisivanje stanovništva i zapisivali podatke o zemljištima na starim spisima. Unatoč postojanju baza podataka dugog niza godina, naziv baze podataka se tek pojavio 60-ih godina prošlog stoljeća zajedno s razvojem računala i njihove memorije.

Danas pod nazivom baza podataka se većinski misli na relacijsku, odnosno na bazu u kojoj su podaci modelirani relacijama. Upravo tom vrstom baze podataka bavimo se u prvom poglavlju. Imena kao što su PostgreSQL, MySQL, Oracle, Microsoft SQL Server i IBM DB2 dobro su poznata u IT zajednici. Ova vrsta baza podataka podržava velik broj poslovnih sustava današnjih organizacija. Pouzdanost i funkcionalnost ovakvih baza dokazana je u mnogim sustavima kroz dugi vremenski period. Međutim, pojava i masovno korištenje interneta, kao i intenzivni razvoj drugih tehnologija, dovelo je do povećanja količine i tipova podataka zbog čega se javila potreba za optimiziranjem prihvaćanja i obrade te velike količine podataka. Kao rješenje tog problema razvila se nova vrsta baze podataka sa kojom se bavimo u drugom poglavlju. U NoSQL bazi podaci nisu modelirani relacijama, nego je riječ o "slobodnijem" modelu podataka koji ne zahtijeva definiciju sheme. Takva baza koristi razne tipove pohranjivanja podataka, kao što su dokumenti, grafovi, vrijednost-ključ parovi i stupci. Određeni modeli imaju bolje performanse za određene vrste problema, stoga treba pripaziti pri odabiru samog modela.

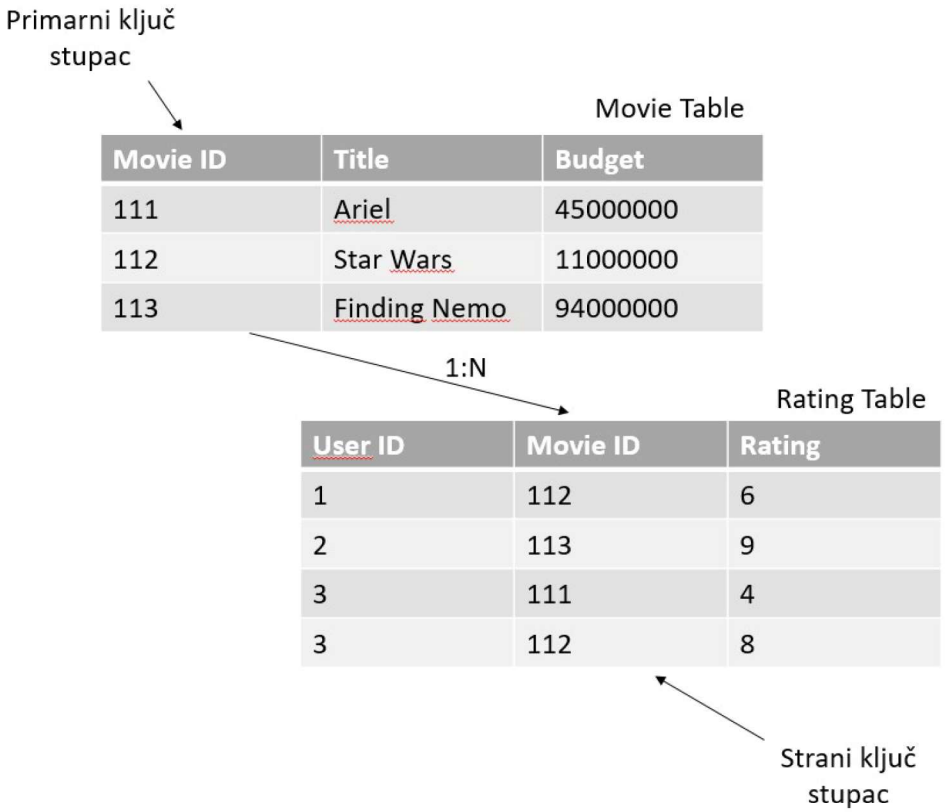
Posljednjih godina grafovska NoSQL baza podataka se primjenjuje u mnogim područjima, od društvenih aplikacija do znanosti. Velike tvrtke poput Googlea, Netflixa, Facebooka, Microsofta i drugih, koriste upravo ovaj tip NoSQL baza podataka kako bi poboljšale svoje performanse. Korištenjem ove baze, ove tvrtke su uspjele poboljšati korisničko iskustvo, pružiti preciznije preporuke, poboljšati tražilice, upravljanje sadržajem, financijske usluge, maloprodajne usluge i sl. Iako su danas relacijske baze podataka i dalje najpopularnije, postavlja se pitanje da li ih u budućnosti mogu zamijeniti NoSQL baze podataka. U trećem poglavlju ćemo provesti detaljnu usporedbu performansi SQL i NoSQL baza podataka na dva primjera te pokušati potvrditi ili opovrgnuti danu tvrdnju.

1 Relacijske baze podataka

Relacijske baze podataka se nazivaju "relacijskim" jer podaci unutar njih imaju unaprijed određene međusobne odnose. Podaci se u relacijskoj bazi podataka pohranjuju u dvodimenzionalne tablice koje se sastoje od redaka i stupaca, a povezane su jedinstvenim ID-ovima ili "ključevima". Sjecište stupca i retka ima samo jednu vrijednost iz ograničene domene dopuštenih vrijednosti. Svaki red u tablici je zapis s jedinstvenim ID-om koji se naziva primarni ključ, a stupci tablice sadrže atribute podataka. Primarni ključevi se postavljaju kako bi se mogao identificirati svaki red u tablici, a strani se ključevi koriste za međusobno povezivanje više tablica (slika 1.). Shema, odnosno veza između tablica, mora biti jasno i unaprijed definirana kako bi se podaci mogli unositi. Loše dizajnirana shema može dovesti do redundancije podataka i nesklada među tablicama. Zbog ovakve strukture relacijske baze podataka su okomito skalabilne, što znači da najbolje rade na jednom računalu.

Sustav za upravljanje relacijskim bazama podataka (engl. Relational Database Management Systems, RDBMS) je softver baze podataka koji korisnicima omogućuje stvaranje, ažuriranje i održavanje relacijske baze podataka. Uobičajeni primjeri sustava za upravljanje relacijskim bazama podataka su MySQL, Microsoft SQL Server i Oracle Database. U RDBMS-u korisnici unose SQL upite kako bi dohvatili podatke potrebne za određene funkcije posla. SQL je skraćenica za Structured Query Language. To je jezik implementiran za traženje informacija iz relacijskih baza podataka (prema [19]).

S pojavom interneta, ograničenja relacijskih baza podataka postala su sve problematičnija. Tvrtke kao što su Google, Facebook, Amazon shvatile su da rad s velikim brojem korisnika na internetu se znatno razlikuje od dosadašnjeg rada u velikim tvrtkama s nekoliko tisuća korisnika. Uočena je potreba za visokom dostupnošću, niskom latentnošću odgovora i bržom obradom velike količine podataka što je s relacijskom bazom teško ostvariti. Ona se može nadograditi s više CPU-a ili dodatnom memorijom ako radi sporo, no to je skupa radnja i ne djeluje uvijek jer postoje ograničenja koliko memorije i CPU-a može biti podržano u jednom poslužitelju. Druga mogućnost je korištenje više poslužitelja s relacijskom bazom podataka, ali tada upravljanje takvim sustavom je vrlo složena radnja. Stoga, ako relacijske baze ne zadovoljavaju potrebe preporuča se korištenje NoSQL baza podataka (prema [18]).



Slika 1: Primjer tablica Movie i Rating te relacije među njima

2 NoSQL baze podataka

2.1 Kratka povijest

Naziv NoSQL baza podataka prvi put je upotrijebio Carlo Strozzi 1998. godine kada je svojoj „relacijskoj" bazi podataka koja nije koristila SQL dao naziv „Strozzi NoSQL“. Pojam NoSQL može značiti ili „Bez SQL sustava" ili prijevod „Ne samo SQL", odnosno neki sustavi mogu podržavati upitne jezike slične SQL-u ili uopće ne koristiti SQL([21]). Do kasnih 2000-ih, SQL baze podataka još uvijek su bile iznimno popularne, ali za one koji su trebali podržavati skaliranje, NoSQL je postao bolja opcija. Naziv NoSQL se ponovno pojavio 2009. kada su ga Eric Evans i Johan Oskarsson upotrijebili za opisivanje nerelacijskih baza podataka. Budući da trenutni NoSQL pokret iz 2009. u potpunosti odstupa od relacijskog modela, Strozzi je predložio da se ipak zove 'NoREL' ([11]).

2.2 ACID i BASE svojstva

Za razliku od relacijskih baza podataka, gdje ACID (engl. atomicity, consistency, isolation, durability) svojstva su primijenjena, nerelacijske baze podataka temelje se na svojstvima BASE (engl. basically available, soft state, eventually consistent). Nepodržavanje ACID svojstva od strane nerelacijskih baza podataka ne implicira da su one nedosljedne i nepouzdane baze, nego će zbog eventualne dosljednosti sustav postati konzistentan s vreme-

nom. ACID svojstva jamče pouzdanost transakcije baze podataka, a BASE visoku dostupnost i skalabilnost.

ACID svojstva su :

- **Atomarnost** (engl. *Atomicity)* - Transakcije se moraju izvšiti u potpunosti ili nikako. Atomarni sustavi moraju biti pripremljeni za sve potencijalne probleme. Ako dođe do hardverskog, softverskog ili nekog drugog problema, sve buduće operacije moraju biti obustavljene.

- **Konzistentost** (engl. *Consistency)* - Svojstvo koje osigurava da samo validni podaci mogu biti u bazi podataka. Ako se neka transakcija protivi konzistentnošću baze, ona se mora prekinuti. Drugim riječima, svaka valjana transakcija je ona koja očuva konzistentnost.

- **Izolacija** (engl. *Isolation)* - Transakcije koje se izvršavaju u isto vrijeme ne smiju utjecati jedna na drugu. Točnije, ako dvije transakcije izvršavaju neke operacije nad bazom podataka u isto vrijeme, sustav mora jamčiti da su transakcije međusobno izolirane kako bi se očuvala konzistentnost.

- **Trajnost** (engl. *Durability)* - Svaka uspješno završena transakcija u bazi podataka mora biti trajna, odnosno sve promjene nad podacima moraju ostati pohranjene na disku, ili nekom drugom trajnom mediju, pa i u slučaju neočekivanih događaja poput nestanka struje.

BASE svojstva su prema [18] :

- **"Basically Available"** - Sustav baze podataka uvijek treba biti dostupan za odgovor na zahtjeve korisnika, čak i ako trenutno ne može garantirati pristup svim podacima u bazi. Točnije, baza podataka može biti nedostupna tijekom kratkog razdoblja, ali bi trebala minimizirati vrijeme prekida rada i brzo se oporaviti od kvarova.

- **"Soft state"** - Stanje baze podataka se može mijenjati, čak i kada nema eksplicitnih upita korisnika nad bazom podataka. To može biti posljedica pozadinskih procesa, ažuriranja podataka itd. Baza podataka treba lako podnijeti ovu promjenu i osigurati da ne dođe do oštećenja i gubitka podataka.

- **"Eventually consistent"** - Baza podataka u konačnici treba konvergirati u konzistentno stanje. Točnije, može se dogoditi da je baza u nekom trenutku nekonzistentna. Npr. neka NoSQL baza podataka čuva više kopija podataka na više računala. Te višestruke kopije mogu neko vrijeme biti nedosljedne, ali mehanizam replikacije će na kraju ažurirati sve kopije i time osigurati konzistentnost.

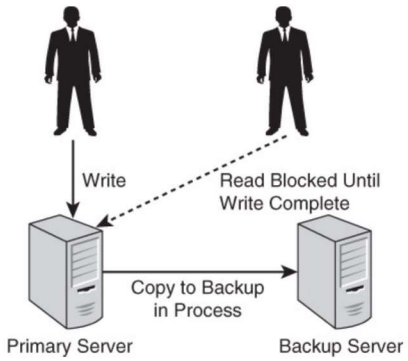
Uz BASE svojstva, za NoSQL bazu podataka vežemo i CAP teorem (Brewerov teorem). Ovaj teorem navodi kako je nemoguće u distribuiranom sustavu s mehanizmom replikacije

jamčiti sva tri poželjna svojstva - konzistentnost, dostupnost i toleranciju particije u isto vrijeme. Mogu se odabrati samo dva od sljedeća tri svojstva [18]:

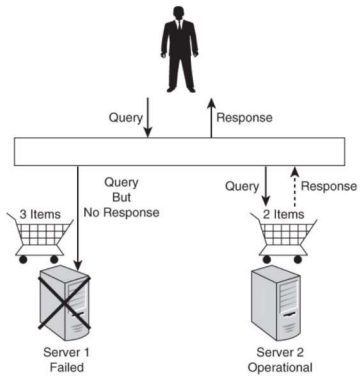
- Konzistentnost - Svi čvorovi imaju dostupne iste kopije podataka, odnosno svaki klijent ima isti pogled na podatke.
- Dostupnost - Svaki korisnički zahtjev za čitanje ili pisanje bit će uspješno obrađen ili će korisnik biti obaviješten s porukom da se zahtjev ne može ispuniti.
- Tolerancija particioniranja - U ovom slučaju, sustav nastavlja raditi unatoč prekidu veze između čvorova koja rezultira stvaranjem dviju ili više particiji, pri čemu je komunikacija moguća samo među čvorovima u istoj particiji.

Sada možemo nabrojati sve moguće opcije CAP teorema i koje baze podataka ih koriste prema [20]:

- CA (konzistentnost i dostupnost) - Sustav može stati ako dođe do prekida veze među čvorovima. Inače, odgovora s najnovijim ažuriranim podacima u kratkom vremenu. Primjeri baza podataka: Cassandra, CouchDB, Riak, Voldemort i dr.
- AP (dostupnost i tolcrancija particijc) - Sustav dajc prednost dostupnosti nad konzistentnošću zbog čega može odgovoriti s možda zastarjelim podacima (slika 3.). Sustav se može distribuirati na više čvorova i dizajniran je za pouzdan rad čak i u slučaju mrežnih particija. Podaci s čvora u kvaru su dostupni i dalje, ali nekonzistentni s ostalim podacima. Primjeri baza podataka: Amazon DynamoDB, Google Cloud Spanner i dr.
- CP (tolerancija particije i konzistentnost) - Sustav daje prednost konzistentnosti nad dosljednošću (slika 2.). Ako dođe do kvara particije ili čvora, tada se nekozistentni čvorovi isključe i sekundarni čvorovi preuzimaju odgovornost. Primjeri baza podataka: Apa



Slika 2: CP - podaci su konzistenti, ali nisu dostupni, Izvor: [18]



Slika 3: AP - podaci su dostupni, ali nisu konzistentni, Izvor: [18]

2.3 Karakteristike NoSQL baza podataka

Osnovne karakteristike NoSQL baza podataka prema [12] su:

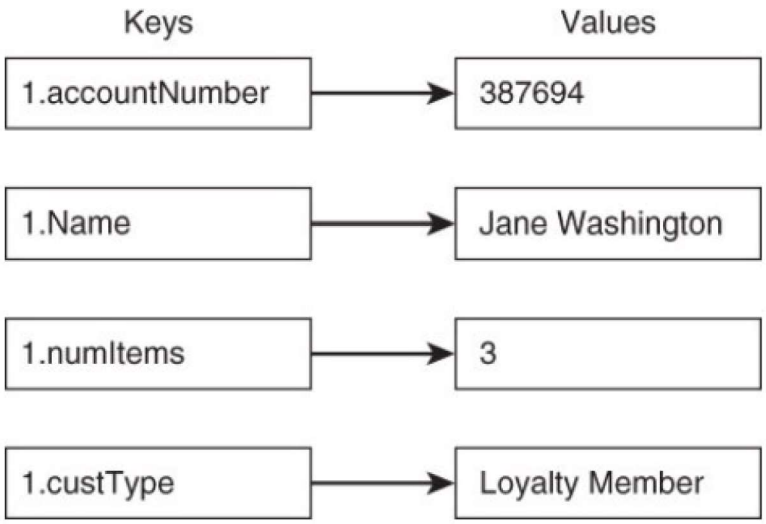
- Distribuirani sustav (skalabilnost, pouzdanost, dijeljenje resursa) - NoSQL baze podataka su dizajnirane za korištenje više poslužitelja, horizontalno su skalabilne i omogućuju rukovanje s velikim količinama podataka. Uspješno upravljanje bazom podataka s povećanim brojem korisnika i količine podataka postižu korištenjem sljedeća dva načina prema [2]:
  - Skaliranje pomoću resursa - podrazumijeva dodavanje resursa nekom od čvorova na način da stavimo još jedan procesor ili povećamo memoriju za pohranu. Takav oblik skaliranje nazivamo vertikalno skaliranje koje je često u relacijskim bazama podataka. Instalacija i održavanje velikih poslužitelja složen je zadatak koji podrazumijeva velike troškove.
  - Skaliranje pomoću čvorova - podrazumijeva dodavanje novih čvorova u sustav (dodavanje novih računala u distribuirani sustav). Takav oblik skaliranja zovemo horizontalno skaliranje koje koriste NoSQL baze podataka. One se najbolje nose s naglim skokovima u aktivnostima novih korisnika tako što se lako može dodati novi poslužitelj baze podataka kako bi se proširio trenutni klaster poslužitelja.
- Fleksibilnost - Postoje četiri različite vrste modela koje NoSQL baza podataka može koristiti:
  - ključ-vrijednost baza podataka,
  - dokument baza podataka,
  - grafovska baza podataka,
  - stupčana baza podataka.
- Jezik upita - Ne podržavaju SQL, ali umjesto njega imaju svoje alternativne jezike za upite. Neki od njih su Javascript, Cypher, SparQL...
- Jednostavna implementacija - Ne zahtijevaju definiranje i strogo praćenje sheme pri unosu podataka za razliku od relacijskih baza podataka.
- BASE/CAP svojstva
- Prikladne za rad s i nestrukturiranim i polustrukturiranim podacima .

Postoje četiri uobičajene vrste NoSQL baza podataka koje pohranjuju podatke na drugačije načine, a to su ključ-vrijednost, grafovska, stupčana i dokument baza podataka. Svaka vrsta pruža različite prednosti i nedostatke ovisno o skupu podataka. U nastavku će biti rečeno nešto više o pojedinoj vrsti.

2.4 Ključ - vrijednost baza podataka

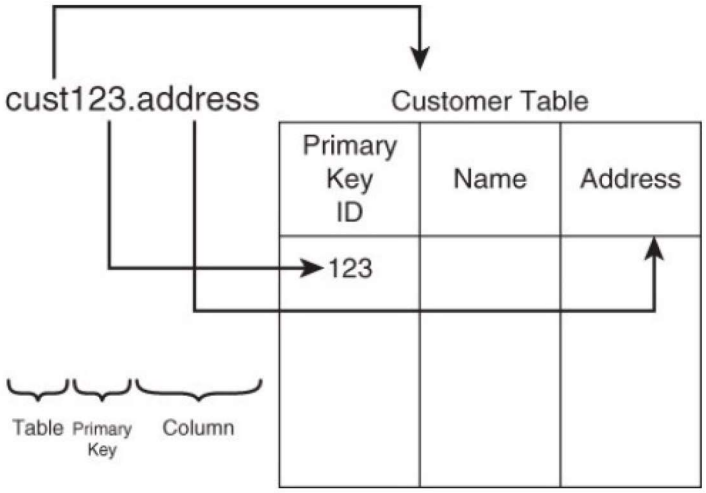
Ključ-vrijednost baza podataka je najjednostavniji i najintuintivniji oblik NoSQL baza podataka. Ona se obično implementira kao hash tablica s određenim ključem i pokazivačem na određenu vrijednost, odnosno podaci su organizirani i pohranjeni u obliku ključ-vrijednost parova. Ključ se sastoji od niza znakova koji predstavlja jedinstveni kod, dok vrijednost može biti bilo koja vrsta podataka, čak i u obliku audio datoteka, dokumenata i slično (Slika 4.). Za ovaj tip baze podataka vrijede dva pravila:

- Različiti ključevi - Svaki ključ mora biti jedinstven.
- Nema upita baziranih na vrijednost - Upit može biti baziran samo na ključu, odnosno ova baza podataka ne može se pretraživati prema određenoj vrijednosti, nego moguć je samo upit kojim se traži određeni ključ. Upravo zbog takvih upita ključ mora biti indeksiran kako bi se poboljšala performansa upita.



Slika 4: Primjer ključ-vrijednost tablice, Izvor: [18]

Ovakve baze podataka nemaju vlastiti jezik upita, samo operacije pronalaženja, unosa i uklanjanje parova na temelju odabranog ključa, odnosno operacije GET, PUT i DELETE, koje imaju jedan od ključeva kao parametar za izvršavanje. Neke baze podataka ključ-vrijednost podržavaju imenski prostor (engl. namespace), odnosno kolekcije parova ključa i vrijednosti. Ovo svojstvo se može iskoristiti za implementaciju nečeg sličnog relacijskoj shemi u kombinaciji s konvencijom imenovanja ključeva [18]. Primjer imenovanja ključeva vidljiv je na slici 5.



Slika 5: Imenovanje ključeva, Izvor: [18]

Slučajevi kada se ključ-vrijednost baza podataka preporuča koristiti prema [16] su:

- Preporuke proizvoda i personalizirane liste:

  Ovaj tip baze podataka idealan je za pohranjivanje osnovnih informacija o kupcima e-trgovine, kao što su detalji o kupcima, kategorije proizvoda, sadržaj košarice i detalji proizvoda e-trgovine. Svaki podatak može biti povezan s jedinstvenim ključem što omogućuje jednostavno dohvaćanje i ažuriranje na temelju vrijednosti ključa. Prilagođene oglase i preporuke u stvarnom vremenu često pokreću pohrane ključeva i vrijednosti koje prate korisnika koji se kreće kroz web mjesto i odgovaraju predstavljanjem novih oglasa i preporuka za korisnika.

- Upravljanje sesijom u stvarnom vremenu - igre i financije: Baze podataka ključ-vrijednost su iznimno prikladne kada je potreban nasumični pristup podacima u stvarnom vremenu zbog brzog čitanja i pisanja te brzog pristup memoriji. Na primjer, u financijama ili igrama za generiranje atributa korisničke sesije tijekom upravljanja velikim sesijama u online aplikaciji poput ranga i rezultata igrača.

- Aplikacije s rijetkim ažuriranjima i jednostavnim upitima: Baze podataka ključ-vrijednost dobar su izbor za aplikacije koje ne zahtijevaju složene upite ili česta ažuriranja. Ako se unutar aplikacije koristi samo jednostavne CRUD operacije te dohvaćaju rijetko ažurirani podaci, onda ključ-vrijednost baze podataka nude jednostavno i učinkovito rješenje.

- Caching: Baze podataka ključ-vrijednost poput Redis-a često se koriste za predmemoriju što pomaže u poboljšanju performansi aplikacije privremenim pohranjivanjem podataka kojima se često pristupa.

Neke popularne baze podataka ključ-vrijednost [15] su Redis, Riak, LevelDB, MemcacheDB, Amazon DynamoDB i dr. Pohrana para ključ-vrijednost može biti privremena, trajna ili njihova kombinacija.

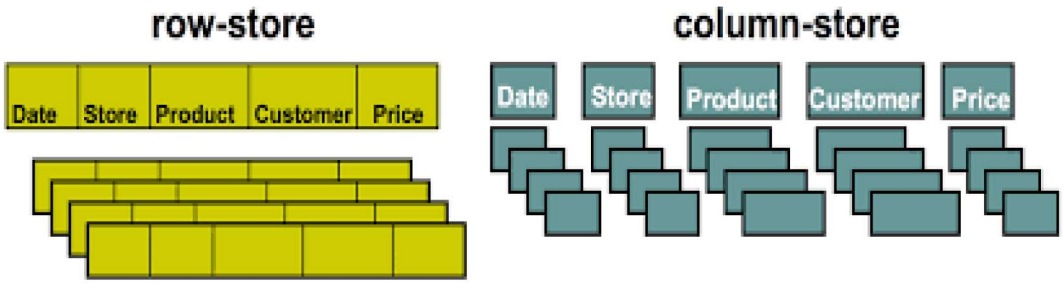
Memcached je NoSQL baza podataka s privremenim parovima ključ-vrijednost. Memcached čuva sve podatke koje čita i sprema u memoriji, ali kada se Memcached zaustavi, podaci više ne postoje.

Tokyo Tyrant pripada NoSQL bazi podataka s trajnom pohranom para ključ-vrijednost. Za razliku od privremene pohrane, kod trajne pohrane ne spremaju se podaci u memoriju, nego na tvrdi disk. Budući da se IO operacija tvrdog diska mora dogoditi kada se podaci spremaju na tvrdi disk dolazi do sporije izvedbe, ali podaci neće biti izgubljeni.

Redis kombinira prednosti privremenih parova ključ-vrijednost i trajnih parova ključ-vrijednost. Prvo sprema podatke u memoriju i zapisuje ih na tvrdi disk kada se ispune određeni uvjeti. Ovo ne samo da osigurava brzinu obrade podataka u memoriji, već jamči i trajnost podataka upisivanjem na tvrdi disk.

2.5 Stupčane baze podataka

Relacijske baze podataka organiziraju podatke prema zapisu tako što su svi podaci povezani s jednim zapisom u bazi spremljeni jedan pored drugog u memoriji. S druge strane, stupčane baze podataka organiziraju podatke po stupcu, odnosno sve podatke povezane unutar jednog stupca spremaju kao jedan blok u memoriji. Ovaj tip baze podataka također pruža brzo pohranjivanje podataka, ali i brzo izvođenje upita nad podacima zbog čega postaje sve češći izbor pri modeliranju podataka. Na slici 6. vizualno je prikazana razlika u spremanju podataka pomoću redaka i stupaca.



Slika 6: Relacijska u odnosu na stupčanu bazu podataka,
Izvor:http://www.primarydigit.com/blog/-a-brief-introduction-to-column-oriented-databases

Relacijske baze podataka brzo dohvaćaju redak ili skup redaka, ali pri agregaciji unose se dodatni, ali nerelevantni podaci (stupci) u memoriju što usporava rad baze podataka. Osim toga, baza podataka orijentirana na retke često treba pristupiti većem broju diskova. Dakle, unatoč jednostavnosti te brzini spremanja i dohvaćanja podataka u relacijskoj bazi podataka, izvođenje upita pak može biti sporije zbog korištenja dodatne memorije i pristupa višestrukim diskovima.

U stupčanoj bazi podataka tablica se pohranjuje jedan po jedan stupac redom red po red. Pritom je moguće pohranjivanje stupaca na zasebnim diskovima što pri određenim upitima smanjuje broj diskova kojima treba pristupiti i količinu dodatnih, nerelevantnih podataka u memoriji zbog čega se brzina izvođenja povećava. Svaki se stupac može komprimirati kao jedna datoteka što znači da baza zauzima manje prostora na disku.

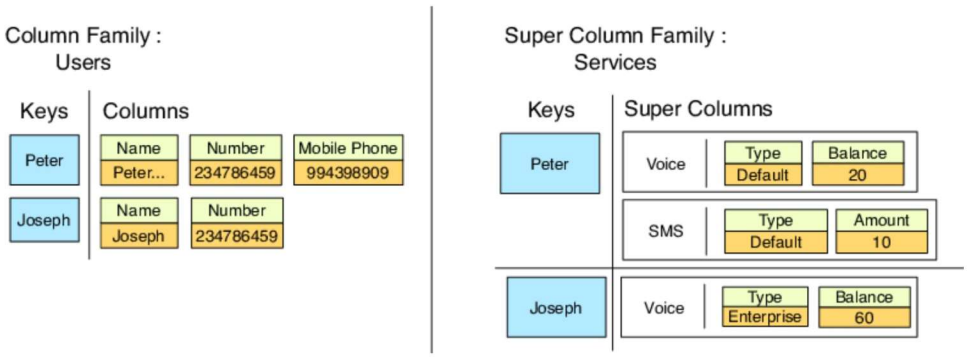
Slaba točka ovakve baze podataka su modifikacije podataka (dodavanje i brisanje redaka) zbog organizacije podataka u memoriji. Za dodavanje novog retka podataka, svaka vrijednost iz novog retka mora biti dodana u ispravan blok postojeće baze podataka. Ovo može postati kompleksna radnja kada unutar baze postoji mnogo stupaca. Relacijske baze podataka rade ovo iznimno brzo jer su svi atributi jednog retka pohranjeni zajedno [6].

Svaki stupac sadrži naziv, vremensku oznaku te pripadajuće vrijednosti. Vremenske oznake se spremanju zajedno s ostalim podacima radi održavanja poretka vrijednosti unutar stupca, Ako postoje slični (povezani) stupci, oni tada pripadaju istoj obitelji stupaca koja se sprema odvojeno od ostalih obitelji stupaca. To omogućuje fleksibilniji i učinkovitiji model podataka, budući da svaka obitelj stupaca može imati vlastiti skup stupaca i može se optimizirati za različite vrste upita. Prvi stupac u svakoj obitelji stupaca je ključ retka te on služi kao identifikator retka. Broj stupaca koji se odnose na svaki red ili njihov naziv može varirati [18].
Vrste obitelji stupaca u stupčanoj bazi podataka prema [22] su:

- Standardna familija stupaca (*engl. standard column family)* - Kao i tablica, sadrži par ključ-vrijednosti gdje su ključu retka pridružene vrijednosti pohranjene u odabranim, grupiranim stupcima.

- Super familija stupaca (*engl. super column family)* - U ovom slučaju, vrijednost stupca sadrži listu drugih stupaca. Takve stupce nazivamo super-stupcima. Treba napomenuti da super-stupci ne sadrže vremenske oznake.

Razlika u standardnoj i super familiji stupaca prikazana je na slici 7. Stupčane baze podataka stupaca koriste prostor ključeva (engl. keyspace) unutar kojega moraju biti definirane sve obitelji stupaca. [18]



Slika 7: Vrste familija stupaca,
Izvor: https://www.researchgate.net/figure/Example-of-column-and-super-column-families_fig1_265890153

Baze podataka orijentirane na stupce idealne su za analitičku obradu zbog svoje optimiziranosti rukovanja velikom količinom podataka te brže agregacije podataka. Ova vrsta baze podataka je lako skalabilna, odnosno lako se mogu dodati novi stupci [1]. Jedna od najpopularnijih dostupnih stupčanih baza podataka je Apache Cassandra. To je stupčana baza podataka koja obrađuje velike količine podataka na brojnim poslužiteljima čineći podatke visoko dostupnima [3]. Još neka od imena na ovom popisu uključuju Apache HBase, Hypertable i Druid [15].

Relacijske baze podataka	**Stupčane baze podataka**
Lakše dodavanja i brisanje podataka	Modifikacija podataka može biti spora
Idealne za OLTP (obrada transakcija)	Idealne za OLAP (analitička obrada)
Spora agregacija podataka	Brza agregacija podataka
Sporo komprimiranje	Brzo komprimiranje
Zahtijeva više prostora za pohranu podataka	Zahtijeva manje prostora za pohranu podataka

Tablica 1: Usporedba relacijskih i stupčanih baza podataka, Izvor: [1]

2.6 Dokument baze podataka

Dokument baza podataka je vrsta NoSQL baze podataka koja pohranjuje podatke unutar JSON, BSON, ili XML dokumenata. Dokument obično pohranjuje informacije o jednom objektu i sve njegove povezane metapodatke koji su pohranjeni u obliku parova ključeva s pridruženim vrijednostima. Vrijednosti mogu biti različite vrste podataka poput brojeva, nizova, datuma ili objekata. Svaki dokument ima jedinstveni ključ radi lakše organizacije što ih čini sličnim bazama podataka ključ-vrijednosti. ID-ovi se obično indeksiraju u bazi podataka kako bi se ubrzalo dohvaćanje podataka.

```
{
    firstName: “Alice”,
    lastName: “Johnson”,
    position: “CFO”,
    officeNumber: “2-120”,
    officePhone: “555-222-3456”,
}
```

Slika 8: Primjer dokumenata u .json formatu, izvor:[18]

Radi lakše rukovanja s velikim brojem dokumenata, dokumenti se grupiraju u kolekcije. Unutar jedne kolekcije se nalaze dokumenti koji imaju sličan sadržaj, no ne moraju svi

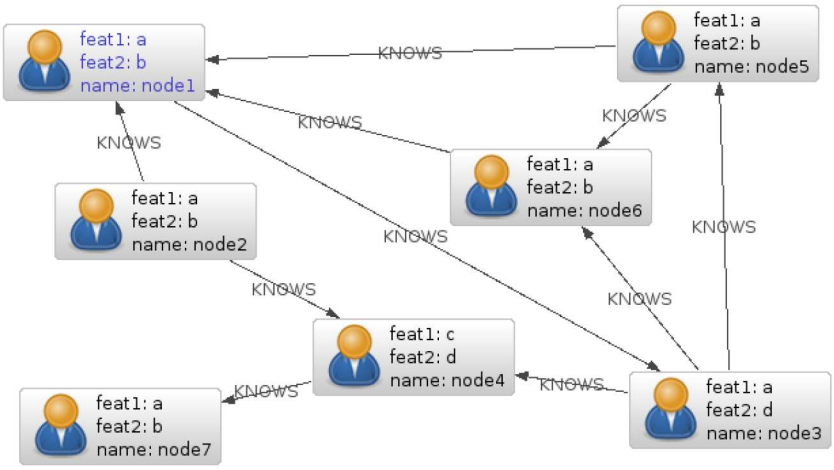
dokumenti u kolekciji imati istu strukturu jer dokument baze podataka imaju fleksibilnu shemu [18]. Za razliku od relacijskih baza podataka, u dokument bazi podataka naglasak je na podacima, a ne vezama između njih. Relacije se reprezentiraju pomoću ugnježđenih podataka, odnosno u slučaju 1:N ili N:M veza, podaci se spremaju unutar jednog dokument kao polje. Ako su podaci polustrukturirani ili nestrukturirani, odnosno ne postoji fiksna shema za pohranu, tada su dokument baze podataka prikladne za upotrebu. One su često horizontalno skalabilne, što znači da mogu rukovati velikom količinom podataka i velikim brojem istodobnih korisnika. Dokument baze podataka prikladne su za razne slučajeve upotrebe, uključujući profile korisnika, baze podataka knjiga, kataloge i slično.

Dostupne su mnoge popularne dokument baze podataka, a neke od najpoznatijih su MongoDB, RavenDB, CouchDB i brojne druge [23]. Najčešće korištena baza podataka dokumenata je MongoDB koji je poznat po svojoj jednostavnosti, skalabilnosti i performansama. Za CouchBase bazu podataka je specifično podržavanje SQL i NoSQL upita [8].

2.7 Grafovske baze podataka

Grafovska baza podataka je vrsta NoSQL baze podataka koja, na temelju teorije grafova, pohranjuje podatke u obliku čvorova i bridova. Podaci se u njoj pohranjuju na vrlo fleksibilan način bez praćenja unaprijed strogo definirane sheme.

Čvorovi služe za spremanje entiteta baze podataka, odnosno svaki čvor sadrži skup atributa koji daju informacije o samom čvoru. Također, svaki čvor ima jedinstveni identifikator. S druge strane, brid pohranjuje odnos između dva čvora ili entiteta. On uvijek mora imati početni i završni čvor te vrstu i smjer. Bridovi su jednako važni kao i čvorovi jer sadrže vitalne informacije kao što su vlasništvo, odnos roditelj-dijete, radnje i slično. Oni također mogu imati jedinstvene identifikatore, baš kao i čvorovi. Osim toga, broj i vrsta odnosa koje čvor može imati je neograničen [18]. Na slici 9. prikazan je primjer grafa društvene mreže gdje se s obzirom na osobe (čvorove) i njihove odnose (bridove) lako vide međusobna poznanstva.

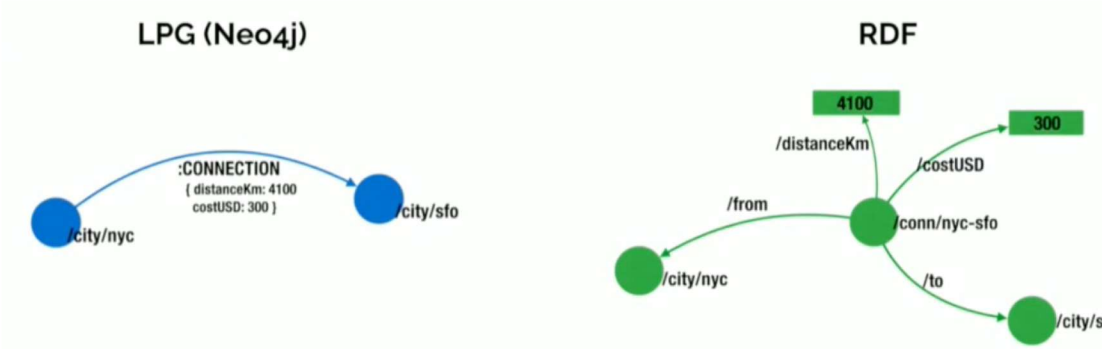


Slika 9: Primjer grafa društvene mreže,
Izvor: https://neo4j.com/blog/social-networks-in-the-database-using-a-graph-database/

Postoje dva popularna modela grafova: grafovi svojstava i RDF (engl. Resource Description Framework) grafovi. Ovi modeli su slični, ali su izgrađeni za korištenje u različite svrhe. Graf svojstava fokusiran je na analitiku i postavljanje upita, dok se RDF graf fokusira na integraciju podataka iz više izvora i njihovo dijeljenje. Obje vrste grafova se sastoje od skupa točaka (vrhova) i veza između tih točaka (brdova), ali postoje i razlike.

RDF grafovi sastoje se od RDF trojke, tj. dva čvora i brida koji ih povezuje. Drugim rječima, RDF trojka sadrži subjekt, predikat i objekt. RDF-ove često koriste zdravstvene tvrtke, statističke agencije i sl. Grafovi svojstava mnogo su deskriptivniji i sastoje se od čvorova koji mogu sadržavati detaljne informacije o subjektu i bridova koji označavaju odnos između čvorova. Čvorovi i bridovi mogu imati atribute koji se nazivaju svojstva, a odakle i dolazi naziv grafa. Dok graf svojstava jedinstveno identificira svaki čvor i bird, RDF graf ne koristi jedinstvenu identifikaciju niti sprema atribute u obliku para ključa i vrijednosti. Budući da se atributi u RDF grafu ne mogu spremiti u vezama, dodaje se međučvor. Zbog neidentifikacije čvorova u RDF grafu nije moguće između dva čvora napraviti brid iste vrste više puta. Grafovi svojstava koriste se u širokom rasponu industrija i sektora, kao što su financije, proizvodnja, društvene mreže, maloprodaja i mnogi drugi. Točnije, u slučaju velike količine podataka za koje treba s vremena na vrijeme napraviti dubinski obilazak grafa [7].

Na slici 10. je prikazan jednostavan primjer iz baze aviokompanije modeliran kao graf svojstava i kao RDF graf.



Slika 10: Usporedba grafa svojstava i RDF grafa, Izvor: [7]

Pri analizi grafova, korisnicima je omogućeno izvođenje upita na temelju relacija i primjena algoritama koji istražuju putove i udaljenosti između vrhova, klastere i važnost pojedinih vrhova. Moguće je primjenom nekog algoritma identificirati pojedinca koji je najviše povezan s drugima u nekoj društvenoj mreži ili poslovnim procesima. Algoritmi mogu identificirati zajednice, anomalije, zajedničke obrasce i putove koji povezuju pojedince ili povezane transakcije [14]. Jedan od algoritama je PageRank koji mjeri važnost svakog čvora unutar grafa na temelju broja dolaznih bridova. S druge strane, pomoću Dijkstra algoritma se lako pronalaze najkraći putovi između dva čvora, što je vrlo korisno kod aplikacija s prometnim mrežama s rutama prijevoznih sredstava između lokacija. Više o algoritmima na grafovima može se pronaći na [14]. Budući da grafovske baze podataka eksplicitno pohranjuju veze između čvorova, korisnici ne moraju izvršavati bezbrojne join-operacije kako se bi pokrenuli upiti i algoritmi, već se slijedi uzorak u grafu [10]. Neke od najpopularnijih baza podataka

s grafovima [15] su Neo4j, HyperGraphDB, OrientDB, ArangoDB i dr.
Neke od primjena grafovski baza podataka prema [14] su:

- Otkrivanje prijevare: Poslovni događaji i podaci o korisnicima, kao što su novi računi, zahtjevi za kredit i transakcije kreditnim karticama mogu se modelirati pomoću grafa te upotrebom tehnike "Analize veze entiteta" za prepoznavanje sumnjivih veza otkriti prijevare.

- Društvene mreže: Tvrtke društvenih medija koriste se grafovskim bazama podataka kako bi pronašle „prijatelje prijatelja" ili proizvode koji se sviđaju korisnikovim prijateljima te prema tome poslali prijedloge korisniku.

- Pogled kupca od 360 stupnjeva: Ovaj tip NoSQL baza podataka koristan je za kompanije kojima su potrebne prodajne strategije koje se kontinuirano prilagođavaju potrebama kupaca. Korištenje relacijske baze podataka, u tim situacijama, zahtijeva mukotrpno modeliranje i ogromna spajanja tablica kako bi se dobili korisni uvidi. S odabirom grafovske baze podataka omogućeno je bolje razumijevanje ponašanja kupaca kako bi im se pružile točne preporuke radi povećanja prodaje i njihovog zadovoljstva.

- Opskrbni lanci: Glavna stvar u upravljanju opskrbnim lancem je smanjenje rizika od prekida rada, poboljšanje odnosa s dobavljačima i učinkovito troškovno upravljanje operacijama postrojenja. Mnoge organizacije mogu prikupiti većinu potrebnih podataka, ali su njihove tradicionalne analitičke tehnologije prespore i preskupe. Većina tradicionalnih analitičkih rješenja opskrbnog lanca izgrađena je na relacijskim bazama podataka, ali se grafovske baze podataka počinju sve češće koristiti.

3 Praktični zadatak

3.1 Opis i motivacija

Danas je vrlo važno na samom početku projekta odabrati odgovarajuću bazu podataka. Trenutno je na tržištu dostupno više od 300 sustava za upravljanje bazama podataka što može otežati odabir. Programeri se moraju odlučiti za relacijsku, nerelacijsku ili čak kombinaciju dviju baza podataka. Naravno, svaka od njih ima svoje prednosti i nedostatke. Zato pri izboru treba pokušati odabrati onu čiji nedostatci će se najslabije osjetiti pri radu. Treba vidjeti hoće će biti potrebno korištenje složenih upita ili samo dohvaćanje po ključu. Pri odabiru je važno primijetiti je li potrebna jaka konzistenost i koliki je kapacitet skladišta dovoljan. Naravno, i budžet projekta igra veliku ulogu. Ako je potrebno rješenje u oblaku, treba pripaziti na količinu troškova i ograničenja. U ovom praktičnom zadatku bazirat ćemo se na testiranju performansi izvođenja upita triju baza podataka. Odabrali smo jednu relacijsku i dvije nerelacijske baze podataka, a to su MySQL, Neo4j i MongoDB. Odabrana je baza podataka filmova i mali dio baze podataka društvene mreže kako bismo mogli testirati i rekurzivne upite. Provjerit ćemo performanse baza pri unosu i brisanju podataka te upita sa stvaranjem tablica.

Izvor podataka za bazu filmova je skup .csv datoteka koje su dostupne na Kaggle web stranici. Ukupna veličina podataka je 1GB i sastoji se od 7 datoteka. Nisu korištene sve .csv datoteke, nego movie_metadata.csv, credits.csv i rating.csv. Iz movie_metadata odabrali smo samo naziv filma, trajanje filma, budžet, zarada, datum izlaska i žanr, no mogli smo i koristiti i ostale podatke u datoteci. Podaci za socijalnu mrežu slučajno su generirani pomoću Python skripte. Generirane su dvije tablice - korisnici i prijatelji korisnika.

3.2 Srodni radovi

Postoje brojni radovi koji su uspoređivali performanse baza podataka Neo4j, MySQL i MongoDB na različitim podacima i upitima. Osim toga, neki radovi su uspoređivali i njihovo korištenje memorije. Naravno, nisu u svakom radu uspoređivane sve tri baze istovremeno, nego njihove kombinacije. Na kraju samog testiranja neki su zaključili da Neo4j brže izvodi upite od MySQL-a, a neki da je MySQL brži. Istražena je primjena tih baza podataka u sustavima upravljanja odnosima s kupcima, analizi društvenih mreža i dr. Sholichah, Imrona, i Alamsyah su 2020. godine u [17] usporedili MySQL i Neo4j korištenjem četiri upita, odnosno tri upita s 10-10.000 zapisa i jedan upit s nestrukturiranim podacima. Na kraju istraživanja je zaključeno da je MySQL brži pri izvođenju tih upita te troši manje memorije, no treba napomenuti da su se u ovom istraživanju koristili upiti nad samo 2 tablice s najviše 10 000 zapisa. U [9] testirana je baza podataka za informacijski sustav web aplikacije na četiri vrste upita. Neo4j je bio za sve četiri vrste upita brži, a najveća razlika u brzini je primijećena pri rekurzivnim upitima, čak do 146 puta je brži od MySQL-a. MongoDB i MySQL zajedno su testirane na supermarket aplikaciji. Svaki upit unosa i pretraživanja je izveden na 100 - 25000 zapisa. Na kraju se zaključilo da je MongoDB brži

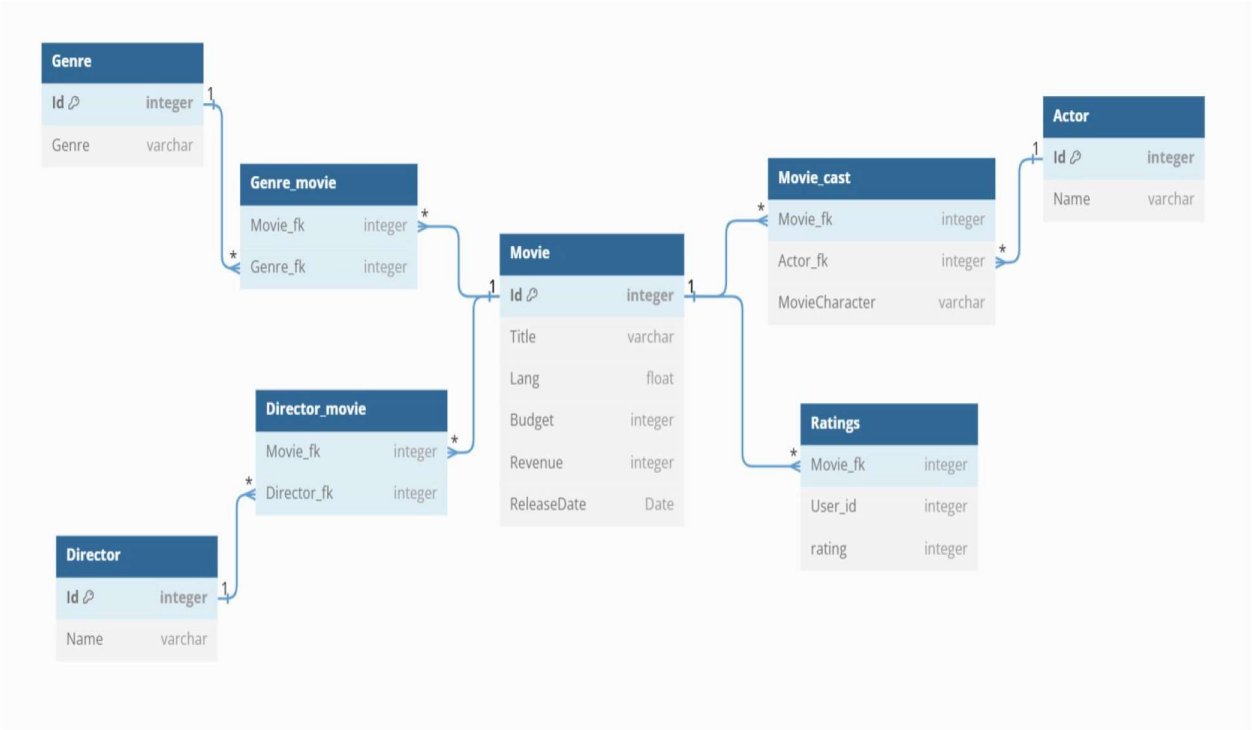
od MySQL-a [4].

3.3 Modeliranje baza podataka

3.3.1 MySQL

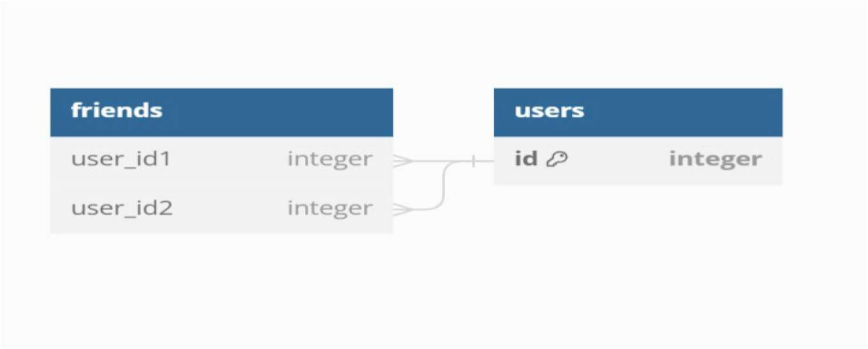
MySQL je vrlo rasprostranjen sustav za upravljanje relacijskim bazama podataka otvorenog koda. Objavljen je 1995. godine, no u to vrijeme je bio vrlo skup. Ovaj sustav je dobio ime po kćeri suosnivača Montyja Wideniusa čiji je ime My. Razvija ga i distribuira Oracle Corporation. Kao upitni jezik koristi SQL, Pri unosu podataka, potrebno je kreirati tablice i deklarirati tipove podataka unutar tablica. Najčešća naredba u MySQL-u je naredba SELECT koja se koristi za dohvaćanje i prikaz podataka iz tablice [13].

Relacijska baza filmova u MySQL-u ima 8 tablica koje odgovaraju filmovima, glumcima, direktorima, žanrovima te ocjenama filmova. U njima se nalaze osnovni podaci o filmovima kao što su naziv filma, budžet, zarada, datum izlaska, duljina trajanja filma, imena glumaca i direktora te ocjene filmova korisnika aplikacije. Slika 11. prikazuje shemu relacijske baze filmova s prikazanim vezama između tablica i vrstama podataka u svakoj tablici.



Slika 11: Relacijski model baze podataka filmova

Na slici 12. prikazan je isječak sheme relacijske baze socijalne mreže koja sadrži samo dvije tablice. Tablica users koja sadrži trenutno samo identifikacijski broj korisnika, ali inače sadrži i druge osnovne podatke o korisniku, poput imena, prezimena, korisničkog imena, datuma rođenja i slično. Tablica Friends sprema sve podatke o prijateljstvu korisnika na socijalnoj mreži, odnosno identifikacijski broj korisnika i identifikacijski broj drugog korisnika s kojim je prijatelj na društvenoj mreži.



Slika 12: Relacijski model prijateljstva na socijalnim mrežama

3.3.2 MongoDB model

MongoDB je dokument NoSQL baza podataka čije ime dolazi od riječi "humongous". Napisana je u C++ programskom jeziku 2007. godine s namjerom da se olakša skaliranje podataka. Podaci se pohranjuju na fleksibilniji način od podataka u SQL-u. Svaka kolekcija može biti drugačija. Dokumenti unutar MongoDB baze su spremljeni u BSON formatu koji je sličan JSON formatu, a njegova prednost je to što prihvaća više vrsta podataka. Ova baza podataka ima svoj vlastiti upitni jezik nazvan Mongo Query Language. Pri upitu se kreira dokument upita s traženim agregacijama i poljima. MongoDB podržava programske jezike C, C++, C#, Go, Java, Python i mnoge druge [5].

MongoDB baza podataka filmova modelirana je i sastoji se od dvije kolekcije: movie i ratings (slika 14.). U kolekciji movie nalaze se svi osnovni podaci o pojedinom filmu poput naziva, budžeta, vrijeme trajanja, ali i niza žanrova kojima film pripada te niz direktora i glumaca. U kolekciji ratings se nalaze identifikacijski brojevi korisnika i filma te ocjena koja je dana tom filmu.

U MongoDB bazi podataka prijateljstava na socijalnim mrežama sve informacije se spremaju u jednu kolekciju. U njoj se nalazi identifikacijski broj korisnika i niz identifikacijskih brojeva drugih korisnika s kojim je prijatelj (slika 13.).

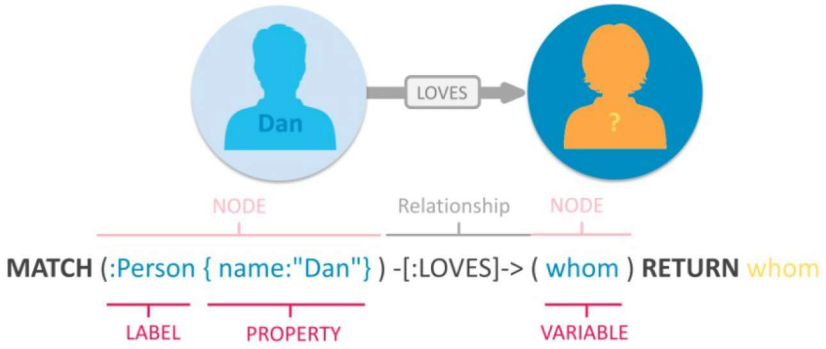
Slika 13: MongoDB model prijateljstva na socijalnim mrežama

(a) Kolekcija movie (b) Kolekcija ratings

Slika 14: MongoDB model baze podataka filmova

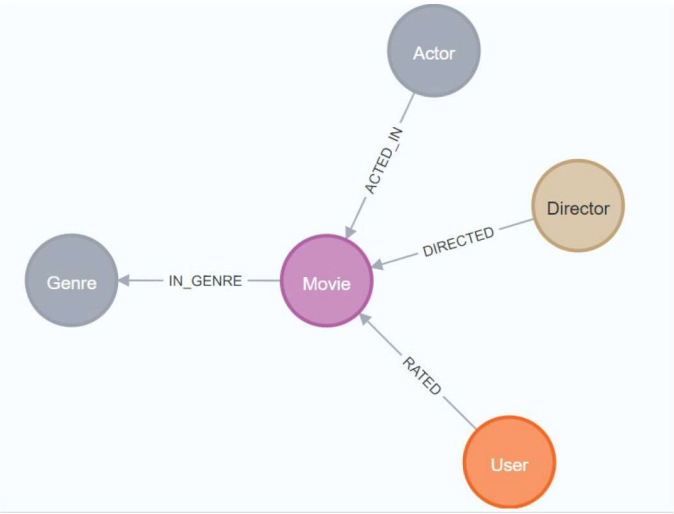
3.3.3 Neo4j model

Neo4j je grafovska NoSQL baza podataka koju razvija Neo Technology. U Neo4j se koristi upitni jezik imena Cypher. To je upitni jezik namijenjen za izvršavanje upita na grafovima. Sličan je s drugim jezicima i vrlo intuitivan pa se lako uči. Koristi ASCII-art sintaksu gdje oble zagrade predstavljaju čvorove, a uglate predstavljaju relacije, odnosno bridove među čvorovima. Svi čvorovi i bridovi unutar baze imaju ime, svojstva i grupirani su u skupove. Pri izvođenju upita, prati se napisani uzorak kroz graf. Naredba MATCH u Cypheru je slična naredbi SELECT u SQL-u, a naredbom RETURN se specificiraju vrijednosti koje će upit vratiti. Na slici 15 nalazi se opis jednostavnog upita u Cypher jeziku gdje se traži osoba koju Dan voli [14].

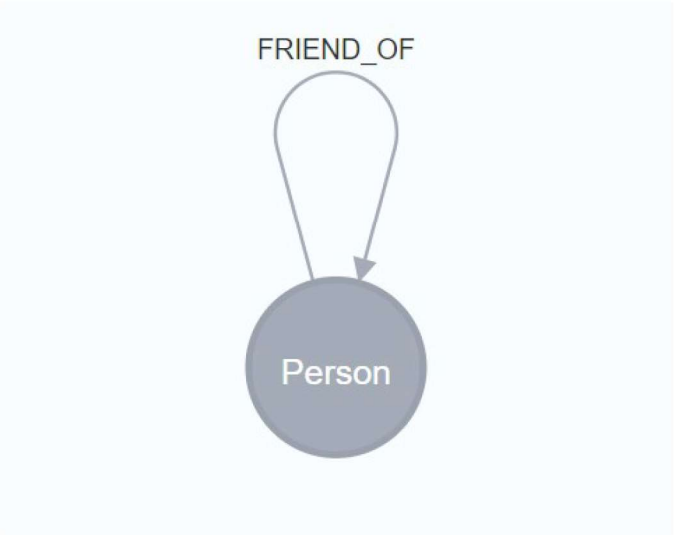


Slika 15: Opis upita u Cypher jeziku, Izvor: [14]

Naša shema Neo4j baze podataka filmova sastoji se od 5 čvorova koji odgovaraju filmu, glumcima, direktoru, korisniku i žanru. Postoje četiri brida između tih čvorova, od kojih jedino brid s nazivom RATED sadrži atribut koji daje informaciju o ocjeni korisnika za neki film (slika 16.).
U shemi baze podataka grafova za socijalne mreže korisnici predstavljaju čvorove, a bridovi predstavljaju prijateljstva korisnika. Korisnik je u ovom slučaju čvor s nazivom Person, a prijateljstvo brid naziva FRIEND_OF (slika 17.).



Slika 16: Grafovski model prijateljstva na socijalnim mrežama



Slika 17: Grafovski model prijateljstva na socijalnim mrežama

3.4 Test postavke

Testovi su izvršeni na Windows poslužitelju sa sljedećim tehničkim specifikacijama:

- OS: Windows 10 22H2, 64-bit
- Memorija: 8GB DDR4
- Procesor: Intel Core i7-8550U 1.8GHz
- Grafika: NVIDIA GeForce MX150 s 2GB VRAM-a
- Disk: 512GB SSD

MySQL WorkBench 8.0.34, Neo4j Community Edition 4.4.23 te MongoDB Shell 4.4.3 su instalirane na ovom računalu. Navedene baze su testirane izvođenjem upita preko Python

skripte. Veza s MySQL bazom podataka uspostavljena je pomoću MySQL Connector Python, veza s MongoDB bazom podataka uspostavljena je pomoću MongoClient i veza s bazom podataka Neo4J uspostavljena je korištenjem biblioteke Neo4j Driver. Svaki upit sa stvaranjem tablice pokrenut je šest puta, ali prvo pokretanje ne uzimamo u obzir. Kada prvi put pokrenemo upit, a podaci nisu u predmemoriji, poslužitelj čita podatke s diska što oduzima puno vremena. Drugo izvršavanje upita je puno brže jer su podaci već u predmemoriji. Vrijeme je mjereno korištenjem biblioteke time u Pythonu te se na kraju uspoređuju prosječna dobivena vremena.

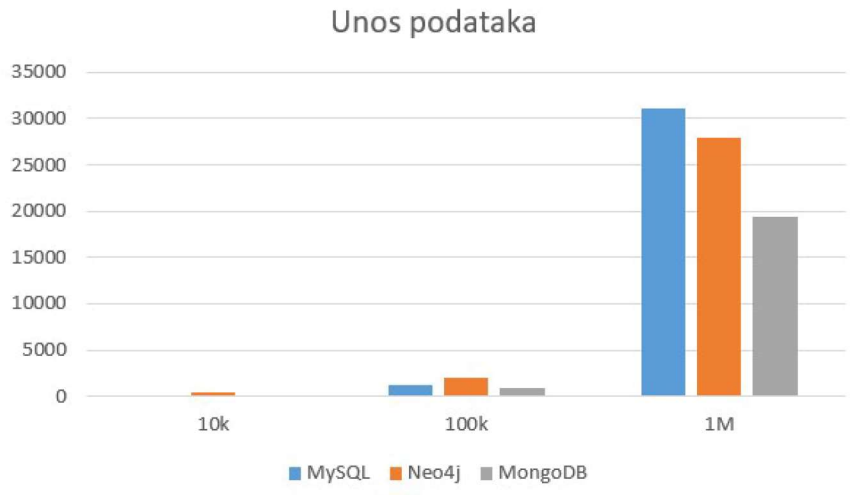
3.5 Usporedba performansi baze podataka filmova

U ovom poglavlju izvršene su različite operacije na bazi podataka filmova u sustavu MySQL, Neo4j i MongoDB uz postavke iz prethodnog poglavlja. Ove operacije su:

- upit unosa,
- upit brisanja,
- upita sa stvaranjem tablice.

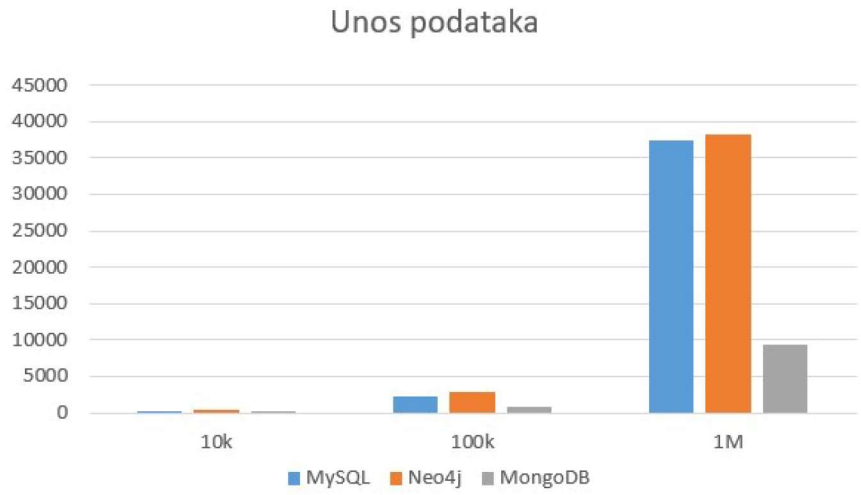
3.5.1 Upit unosa

Brzinu izvođenja upita unosa testirali smo na podacima o ocjenama filma koje dodjeljuju korisnici. Slučajno generirane su tri .csv datoteke s 10 tisuća, 100 tisuća i 1 milijun zapisa. Dobiveni rezultati mjerenja su prikazani na slikama 18. i 19. Na prvom grafikonu je prikazano vrijeme koje je potrebno da upit pošalje, izvrši na bazi te dobije odgovor natrag u aplikaciji, a na drugom grafikonu je vrijeme izvršavanja upita direktno na bazi.



Slika 18: Vrijeme izvršenja upita na bazi

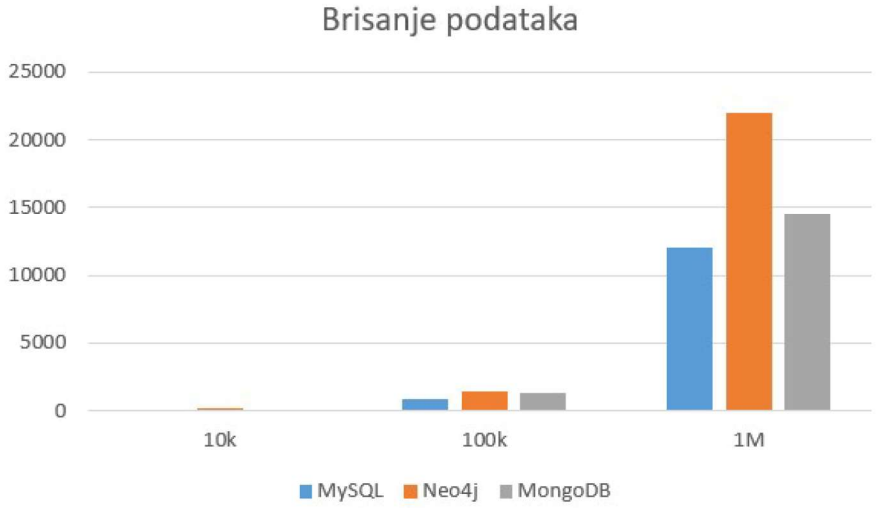
Slike pokazuju da je MongoDB najbrži pri unosu podataka. S druge strane, Neo4j je brži od MySQL-a pri unosu većeg broja podataka, ali zbog neo4j-driver biblioteke sporije vraća odgovor natrag za razliku od MySQL-a.



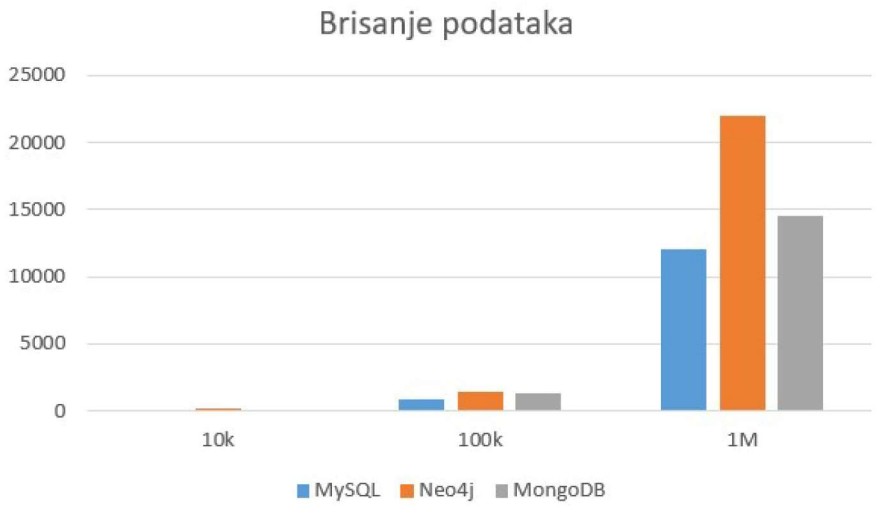
Slika 19: Vrijeme dobiveno python bibliotekom time

3.5.2 Upit brisanja

Upit kojeg testiramo briše sve podatke o ocjenama filmova koje dodjeljuju korisnici. U MySQL ovaj upit briše sve podatke iz tablice rating. U Neo4j taj upit briše sve čvorove korisnika i bridove između korisnika i filmova. U MongoDB ovim upitom brišemo sve dokumente iz kolekcije rating. Dobiveni podaci su prikazani na slikama 20. i 21. Vrijeme dobiveno pomoću python biblioteke time gotovo pa je i jednako vremenu izvršavanja upita direktno na bazi. Razlika je samo 1 - 2 milisekunde. Neo4j je najsporiji u brisanju podataka, a najbrže MySQL izvodi upite brisanja.



Slika 20: Vrijeme izvršavanja upita brisanja na bazi



Slika 21: Vrijeme izvršavanja upita brisanja dobiveno pomoću biblioteke time

3.5.3 Upit sa stvaranjem tablice

U ovom poglavlju testiramo brzinu izvođenja pet upita na bazi podataka filmova. Odabrani su upiti koji bi se koristili u aplikaciji sa svrhom traženja filmova za gledanje po određenim kategorijama i ocjenama. U tablicama 2. - 6. se nalaze upiti i njihova sintaksa u MySQL, Neo4j i MongoDB sustavu.

SQL
SELECT movie.Title FROM movie JOIN genre_movie ON genre_movie.Movie_fk = movie.ID JOIN genre ON genre.ID = genre_movie.Genre_fk WHERE genre.Genre LIKE "%Comedy%"
Neo4j
MATCH (m: Movie) - [r: IN_GENRE]→(g: Genre{name: "Comedy"}) RETURN m.name;
MongoDB
{ "$match": { "Genres.name": "Comedy", }, }, { "$project": { "Title": 1, } }

Tablica 2: Upit 1 - naslovi komedija

SQL
SELECT director.Name, COUNT(Movie_fk) as directed FROM movie JOIN director_movie ON director_movie.Movie_fk = movie.ID JOIN director on director_movie.Director_fk=director.ID GROUP BY director.name ORDER BY directed desc, director.Name asc
Neo4j
MATCH (u: Director) - [r: DIRECTED] → (z: Movie) RETURN distinct u.name, COUNT(r) as directed ORDER BY directed desc, u.name;
MongoDB
{ "$unwind": { "path": "$Directors", "includeArrayIndex": "string", "preserveNullAndEmptyArrays": False, }, }, { "$group": { "_id": "$Directors.name", "ArrId": { "$addToSet": "$id", }, }, }, { "$addFields": { "directed": { "$size": "$ArrId", }, }, }, { "$sort": { "directed": -1, } },

Tablica 3: Upit 2 - broj režiranih filmova za pojedinog filmskog redatelja

SQL
SELECT movie.ID, movie.Title, m.avgRating FROM ((SELECT movie.ID as movie_id, avg(rating.rating) AS avgRating FROM movie JOIN rating ON movie.ID = rating.Movie_fk GROUP BY movie.ID) AS m JOIN movie ON movie.ID = m.movie_id);
Neo4j
MATCH (m: Movie) ← [r: RATED] - (ac: User) RETURN m.id, m.name, AVG(r.rating)
MongoDB
{ "$lookup": { "from": "ratings", "localField": "id", "foreignField": "id_movie", "as": "res", }, }, { "$unwind": { "path": "$res", "includeArrayIndex": "string", "preserveNullAndEmptyArrays": False, }, }, { "$group": { "_id": "$id", "avg_rating": { "$avg": "$res.rating" }, "title": { "$first": "$Title" }, }, },

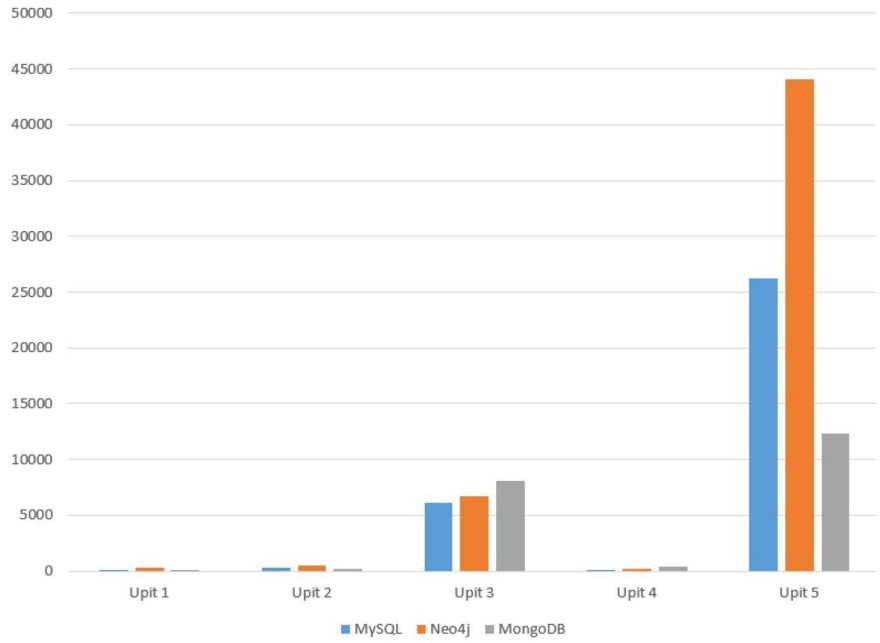
Tablica 4: Upit 3 - prosječna ocjena za svaki naslov filma i njegov ID

SQL
SELECT * FROM movie WHERE year(movie.ReleaseDate)=2005;
Neo4j
MATCH (m: Movie) WHERE m.released.year = 2005 RETURN m;
MongoDB
{ "$project": { "id": 1, "year": { "$year": "$Release Date", }, "Title": 1, "Budget": 1, "Release Date": 1, "Lang": 1, "Revenue": 1, }, }, { "$match": { "year": { "$eq": 2005, }, }, },

Tablica 5: Upit 4 - filmovi koji su izašli 2005. godine

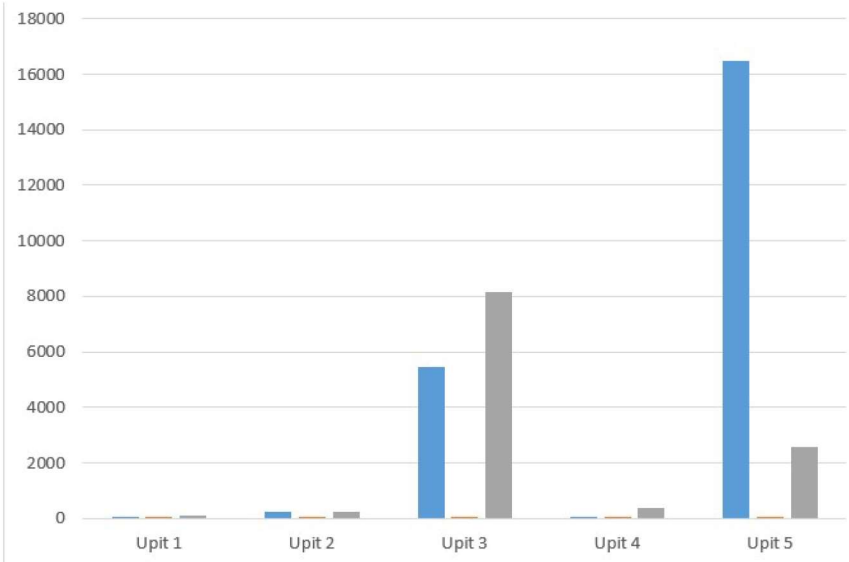
SQL
SELECT movie.Title, genre.Genre, actor.Name, director.Name FROM movie JOIN movie_cast ON movie.ID = movie_cast.Movie_fk JOIN actor ON actor.ID= movie_cast.Actor_fk JOIN genre$_{m}ovieONgenre_movie.Movie_fk = movie.ID$ JOIN genre ON genre.ID = genre_movie.Genre_fk JOIN director_movie ON director_movie.Movie$_{f}k = movie.ID$ JOIN director ON director.ID = director_movie.Director_fk;
Neo4j
MATCH (m:Movie)-[gg:IN$_{G}ENRE$]→(g:Genre), (d:Director)-[dr:DIRECTED]→(m:Movie), (m:Movie)←[ac:ACTED_IN]-(a:Actor RETURN m.name, g.name,a.name,d.name
MongoDB
{ "$unwind": { "path": "$Directors", "includeArrayIndex": "string", "preserveNullAndEmptyArrays": False, }, }, { "$unwind": { "path": "$Genres", "includeArrayIndex": "string", "preserveNullAndEmptyArrays": False, }, }, { "$unwind": { "path": "$Actors", "includeArrayIndex": "string", "preserveNullAndEmptyArrays": False, }, }, { "$project": { "Title": 1, "Directors.name": 1, "Actors.name": 1, "Genres.name": 1, } }

Tablica 6: Upit 5 - izvješće koje sadrži naslov filma, ime glumca, žanrove filma i redatelja



Slika 22: Vrijeme izvršavanja upita dobiveno python bibliotekom time

Na slici 22. gledajući vrijeme dobiveno pomoću python biblioteke time možemo zaključiti da je MongoDB brži u izvođenju svih uputa, osim upita 3. To je jedini upit koji koristi obje kolekcije u MongoDB, odnosno zahtijeva korištenje lookup operacije. S druge strane, na grafikonu s vremenom koje je potrebno da se upit izvrši samo na bazi, odnosno bez slanja poziva upita i dobivanje odgovora natrag, vidimo da je Neo4j najbrži (slika 23.).



Slika 23: Vrijeme izvršavanja upita

3.6 Usporedba performansi baze podataka socijalne mreže

U ovom poglavlju izvršen je upit na malom isječku baze podataka socijalnih mreža u sustavu MySQL, Neo4j i MongoDB uz prethodno navedene postavke. U sljedećim upitima tražimo prijatelje prijatelja do određene dubine. U grafovskoj bazi to predstavlja dubinski obilazak stabla. Takvi upiti su vrlo važni za socijalne mreže. Pomoću njih se preporučuju prijatelji korisniku, a i druge stvari poput proizvoda, usluga i slično. To su upiti koji zahtijevaju nekoliko JOIN operacija nad tablicama s velikim brojem korisnika. Testirana je brzina izvođenja upita na tablicama s tisuću, 10 tisuća te 100 tisuća korisnika društvene mreže od kojih svaki ima 20 prijatelja. U tablici 7. se nalazi upit kojeg testiramo, odnosno tražimo ukupan broj ili ID-jeve prijatelja korisnika s ID-jem 159 do dubine 3.

SQL

```
SELECT cf3.user_id2 FROM Friends f1
JOIN Friends f2 ON f1.user_id2 = f2.user_id1
JOIN Friends f3 ON f2.user_id2 = f3.user_id1
WHERE f1.user_id1 =159
```

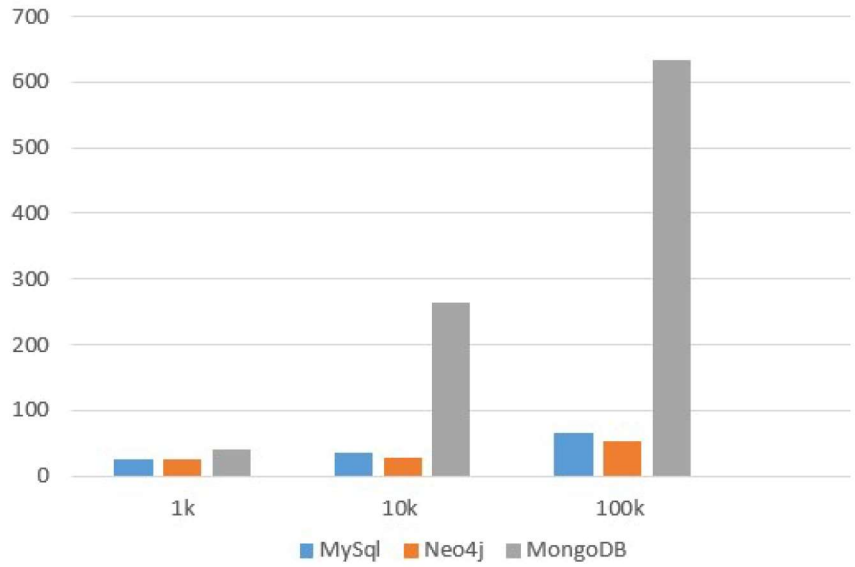
Neo4j

```
MATCH (c:Person{id:"159"}) - [:FRIEND_OF] → (c1) - [:FRIEND_OF] → (c2)
WITH DISTINCT c2
MATCH (c2) - [:FRIEND_OF] → (c3)
WITH DISTINCT c3
RETURN c3.id
```

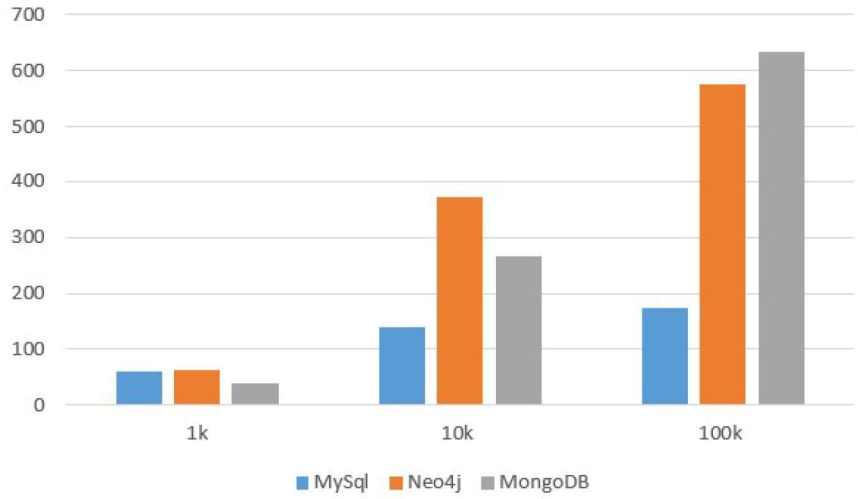
MongoDB

```
{"$match":
      { "Id": 159, },
},
{"$graphLookup":
      {
      "from": "friends7",
      "startWith": "$Friends",
      "connectFromField": "Friends",
      "connectToField": "Id",
      "as": "res",
      "maxDepth": 2,
      "depthField": "dep",
      "restrictSearchWithMatch": {},
      },
},
{"$unwind":
      {
      "path": "$res",
      "includeArrayIndex": "string",
      "preserveNullAndEmptyArrays": False,
      },
},
{ "$group":
      {
      "_id": "null",
      "uniqueFriends": { "$addToSet": "$res.Id", },
      }
}
```

Tablica 7: Upit traženja prijatelja do dubine 3



Slika 24: Upit koji vraća broj prijatelja



Slika 25: Upit koji vraća prijatelje do dubine 3

Na slici 24. vide se da je Neo4j najbrži u izvođenju upita kada se uz pronalazak prijatelja mora i izračunati njihov broj. Dakle, razlika u vremenima izvršavanja upita za Neo4j i MySQL bazu podataka se povećava porastom broja korisnika. MongoDB mnogo sporije izvodi ovaj upit. S druge strane, na slici 25. se vidi da MySQL treba kraće vremena da pošalje tražene podatke na klijenta. Točnije, Neo4j najbrže izvodi upit ako gledamo vrijeme koje je potrebno samo za izvršavanje upita, a ne i vrijeme za slanje upita bazi i slanje odgovora natrag na klijenta. Dakle, interakcija s Neo4j instancom putem Python aplikacije je znatno sporija nego interakcija s MySQL instancom.

3.7 Zaključak

Rezultati dobiveni testiranjem upita iz tri kategorije, odnosno upit unosa, brisanja te upit sa stvaranjem tablice, su pokazali da je MongoDB najbrži pri unosu podataka. S druge strane, MySQL najbrže briše podatke, a Neo4j najsporije. Također, rezultati dobiveni testiranjem pet upita sa stvaranjem tablica su pokazali da je Neo4j brži od ostalih baza u izvođenju ovih upita sa stvaranjem tablica, ali ne uključujući vrijeme koje je potrebno da se traženi podaci pošalju klijentu. Dakle, interakcija s Neo4j instancom putem Python aplikacije je znatno sporija nego interakcija s MySQL instancom. S druge strane, interakcija s MongoDB instancom ne zahtijeva mnogo vremena. Kod upita s više JOIN operacija Neo4j je brži od MySQL ako vraća samo broj prijatelja. Inače je sporiji pri slanju odgovora klijentu. MongoDB znatno sporije izvodi takvu vrstu upita.

Sažetak

Tema ovog diplomskog rada su NoSQL baze podataka te njihova usporedba s relacijskim bazama podataka. Na samom početku rada ukratko su objašnjene osnove relacijskih baza podataka, odnosno pojam primarnog i stranog ključa, pojam RDBMS-a i SQL-a. Zatim u drugom poglavlju bavimo se NoSQL bazama podataka. Opisana su ACID i BASE svojstva te glavne karakteristike NoSQL baza podataka. Također, navedena je podjela NoSQL baza podataka na ključ-vrijednost, grafovske, dokumentne i stupčane baze podataka. Za svaku vrstu navedena su svojstva i njihova primjena. U praktičnom dijelu rada dizajnirali smo bazu podataka filmova i mali isječak baze podataka socijalne mreže u MySQL, Neo4j i MongoDB sustavu. Na bazi podataka filmova testirana je brzina izvođenja unosa, brisanja i selektiranja podataka, a na bazi podataka socijalne mreže testirani su upiti koji zahtijevaju dubinski obilazak stabla u grafovskim baza podataka.

Ključne riječi: relacijska baza podataka, nerelacijska baza podataka, brzina upita, MySQL, Neo4j, MongoDB

NoSQL databases and comparison of their performance with relational databases

Summary

The topic of this master thesis is NoSQL databases and their comparison with relational databases. At the very beginning of this thesis, the basics of relational databases are briefly explained, that is, the concept of primary and foreign keys, the concept of RDBMS and SQL. Then in the second chapter we deal with NoSQL databases. ACID and BASE properties and the main characteristics of NoSQL databases are described. Also, the division of NoSQL databases into key-value, graph, document and column-based is specified. For each type of NoSQL database, the properties and their application are listed. In the practical part of the thesis, we designed a movie database and a small snippet of a social network database in MySQL, Neo4j and MongoDB system. The speed of inserting, deleting and selecting data was tested on the movies database, and on the social networks database queries that require an in-depth tour of the tree in the graph database.

Key words: relational database, non-relational database, query speed, MySQL, Neo4j, MongoDB

Literatura

[1] ". "How to choose between a columnar database vs. row database". " (2023.). URL: https://www.fivetran.com/learn/columnar-database-vs-row-database.

[2] Ahmed Ali i Mahmood Abdullah. "A Survey on Vertical and Horizontal Scaling Platforms for Big Data Analytics". *International Journal of Integrated Engineering* (2019.).

[3] *Apache Cassandra dokumnetacija.* URL: https://cassandra.apache.org/_/index.html.

[4] Dipina B, Shirin Salim i Surekha Mariam Varghese. "Mongodb Vs Mysql: A Comparative Study of Performance in Super Market Management System". *International Journal of Computational Science and Information Technology* 4 (svibanj 2016.).

[5] Kyle Banker i dr. *MongoDB in Action: Covers MongoDB Version 3.0.* 2nd. USA: Manning Publications Co., 2016.

[6] Blake Barnhill i Matt David. "Row vs Column Oriented Databases". (2020.). URL: https://dataschool.com/data-modeling-101/row-vs-column-oriented-databases/.

[7] Jesús Barrasa. "RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?": (2017.). URL: https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/.

[8] *CouchBase dokumentacija.* URL: https://www.couchbase.com/.

[9] Thi-Thu-Trang Do i dr. "Query-Based Performance Comparison of Graph Database and Relational Database". *Proceedings of the 11th International Symposium on Information and Communication Technology.* Hanoi, Vietnam: Association for Computing Machinery, 2022.

[10] Emil Eifrem, Ian Robinson i Jim Webber. *Graph Databases.* 2nd. O'Reilly Media, 2015. URL: https://web4.ensiie.fr/~stefania.dumbrava/OReilly_Graph_Databases.pdf.

[11] Keith D. Foote. "A Brief History of Non-Relational Databases". *DataVersity* (2018.). URL: https://www.dataversity.net/a-brief-history-of-non-relational-databases/.

[12] Sourav Mazumder. "NoSQL in the Enterprise". (2010.). URL: https://www.infoq.com/articles/nosql-in-the-enterprise/.

[13] *MySQL Tutorial.* URL: https://www.w3schools.com/mysql/mysql_intro.asp.

[14] *Neo4j dokumentacija.* URL: https://neo4j.com/docs/graph-data-science/2.5-preview/.

[15] *NoSql baze podataka.* URL: https://hostingdata.co.uk/nosql-database/.

[16] *Redis dokumentacija.* URL: https://redis.com/nosql/key-value-databases/.

[17] Rahmatian Sholichah, Mahmud Imrona i Andry Alamsyah. "Performance Analysis of Neo4j and MySQL Databases using Public Policies Decision Making Data". Rujan 2020.

[18] Dan Sullivan. *NoSQL for Mere Mortals.* 1st. Addison-Wesley Professional, 2015.

[19] S. Sumathi i S. Esakkirajan. *Fundamentals of Relational Database Management Systems.* 1st. Springer Publishing Company, Incorporated, 2010.

[20] "The CAP Theorem in DBMS". (2023.). URL: https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/.

[21] Wikipedia contributors. *NoSQL — Wikipedia, The Free Encyclopedia*. [Online; accessed 24-August-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=%20NoSQL&oldid=1169506180.

[22] Alex Williams. "NoSQL database types explained: Column-oriented databases". "(2021.). URL: https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases.

[23] Alex Williams. "What is a Document Database?": "(2021.). URL: https://phoenixnap.com/kb/document-database.

Životopis

Josipa Sabljo, rođena 31.12.1999. godine u Osijeku.
Pohađala je Osnovnu školu Vladimira Nazora u Čepinu te III. Gimnaziju u Osijeku. Obrazovanje nastavlja na preddiplomskom studiju Matematike i računarstva na Odjelu za matematiku 2018. godine te ga završava 2021. godine s temom završnog rada "SIR model širenja epidemije" kod mentora prof. dr. sc. Dragana Jukića. Dobitnica je Pročelnikove nagrade i pohvale za uspješnost za akademsku godinu 2019/20. Preddiplomsku razinu studija završila je s summa cum lude pohvalom. Odmah po završetku preddiplomskog studija upisuje diplomski studij Matematike i računarstva na Odjelu za matematiku. Akademske 2022/2023. godine nagrađena je Rektorovom nagradom za izvrstan seminarski rad iz područja Matematički modeli pod naslovom "SIR model širenja epidemije". Na diplomskom studiju odradila je stručnu praksu u firmi BEterna.