Implementacija baze podataka u sustavu MongoDB

Orejaš, Marin

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:295743

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2024-12-03**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marin Orejaš

IMPLEMENTACIJA BAZE PODATAKA U SUSTAVU MONGODB

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marin Orejaš

JMBAG: 0016154376

Studij: Informacijski i poslovni sustavi

IMPLEMENTACIJA BAZE PODATAKA U SUSTAVU MONGODB

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, rujan 2024.

Marin Orejaš

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Završni rad prikazuje implementaciju i upravljanje MongoDB bazom podataka, ističući prednosti iste u odnosu na tradicionalne relacijske sustave koji se koriste za upravljanje podacima. Naglasak rada je na opisu efikasnih i skalabilnih modela podataka, praćenju performansi i sigurnosti te testiranju i optimizaciju sustava. Zaključci navode MongoDB kao fleksibilno rješenje za moderne zahtjeve upravljanja podacima. Rad pruža vrijedne uvide za programere i administratore baza podataka te preporučuje daljnje istraživanje i optimizaciju implementacija MongoDB-a za maksimalnu učinkovitost.

Ključne riječi: MongoDB; NoSQL; baza podataka; implementacija; upravljanje podacima; performanse; testiranje i evaluacija;

Sadržaj

1. Uvod

Ovaj završni rad demonstrira praktičnu implementaciju baze podataka s pomoću sustava MongoDB-a. MongoDB, kao jedna od najboljih „NoSQL baza podataka“, nudi fleksibilno, skalabilno rješenje visokih performansi za upravljanje velikim i složenim skupovima podataka. Rad se fokusira na dizajn, razvoj i upravljanje NoSQL bazom podataka s MongoDB-om kao temeljnom tehnologijom.

Ova tema je od neizmjerne važnosti. Dizajn bez shema, horizontalna skalabilnost i robusne mogućnosti postavljanja upita čine MongoDB prilično snažnim alatom za moderne aplikacije koje zahtijevaju agilnost i performanse. Stoga, ovaj završni rad istražuje funkcionalnosti MongoDB-a kako bi dao uvid u to kako bi NoSQL baze podataka mogle mijenjati pravila u praksi upravljanja podacima nudeći rješenja koja su visoko skalabilna, ali lako prilagodljiva dinamičkim potrebama.

Ovaj završni rad posvećuje pozornost nekim od kritičnih aspekata implementacije MongoDB baze podataka. Prvo, razmatra identifikaciju zahtjeva aplikacije i definiranje potreba u pogledu pohrane i upravljanja podacima. Drugo, ulazi u dizajn podatkovne sheme, uključujući najbolje prakse za dizajniranje učinkovitih, skalabilnih modela podataka. Nadalje, razrađuje detaljnu fazu implementacije, odnosno instalaciju, stvaranje baze podataka i CRUD operacije pomoću MongoDB Compass-a. Isto tako, rad pokriva tehnike upravljanja podacima, indeksiranje, sigurnosne prakse i praćenje performansi. Posljednji odjeljci posvećeni su različitim metodama testiranja i evaluacije, mjerenju performansi i strategijama optimizacije. Zaključak daje retrospektivu svega viđenog u ovom radu te izdvaja glavnu misao odnosno ključ dobre implementacije baze podataka u sustavu MongoDB.

2. MongoDB

„MongoDB je moćna, fleksibilna i skalabilna baza podataka opće namjene. Kombinira sposobnost horizontalnog skaliranja s funkcijama kao što su sekundarni indeksi, složeni upiti, sortiranje, agregacije i geolokacijski indeksi. Ovo poglavlje pokriva glavne dizajnerske odluke koje su učinile MongoDB onim što jest.“ [1]

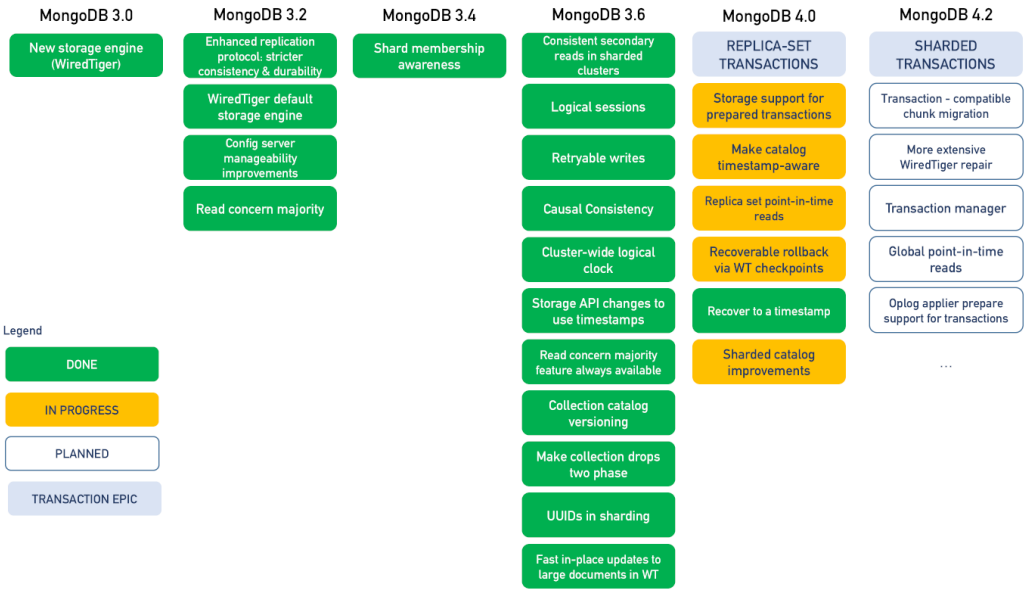
Izvor [1] navodi kako je MongoDB dokumentno orijentirana, a ne relacijska baza podataka. Primarni razlog za odmicanje od relacijskog modela je olakšavanje skaliranja, ali postoje i neke druge prednosti. Baze podataka orijentirane na dokumente koriste mnogo fleksibilniji model, umjesto „redova“, koriste „dokumente“. Dokumenti u dokument-orijentiranom pristupu dopuštaju ugrađeni dokument i niz, čime se modeliraju složeni hijerarhijski odnosi s jednim zapisom. To se prirodno uklapa u način na koji programeri u modernim objektno-orijentiranim jezicima razmišljaju o svojim podacima. Također nema unaprijed definiranih shema gdje ključevi i vrijednosti dokumenta nisu fiksne vrste niti fiksne veličine. To znači da, budući da ne postoji shema koje se treba pridržavati, dodavanje ili uklanjanje polja (atributa) postaje znatno jednostavnije. Također je lakše eksperimentirati, deseci modela mogu se isprobati za podatke, a zatim se može odabrati koji od tih modela slijediti.

2.1. Povijest

Padmanabhan [2] navodi da su MongoDB 2007. razvili Dwight Merriman, Eliot Horowitz i Kevin Ryan, svi osnivači softverske tvrtke 10gen, koja se sada zove MongoDB Inc. Osnivači su u početku počeli raditi na MongoDB-u kao podsustavu za veću platformu koju su gradili za podršku web aplikacijama. Međutim, dok su još radili na projektu, postalo je očito da baze podataka, kakve su tada bile, ne mogu skalirati niti zadovoljiti zahtjeve fleksibilnosti za moderne web aplikacije. To je zapravo ono što ih je natjeralo da u potpunosti preusmjere fokus na razvoj nove vrste baze podataka, koja je kasnije postala poznata kao MongoDB.

Službeno, MongoDB je objavljen kao open source projekt 2009. Mnoge je kasnije privukla ova inovacija pohrane podataka, koja se odvojila od strukture tradicionalnih relacijskih baza podataka. Za razliku od tabularne strukture SQL baza podataka, MongoDB je koristio podatkovni model orijentiran na dokumente gdje su podaci pohranjeni u fleksibilnim dokumentima sličnim JSON-u. Omogućio je puno prirodnije predstavljanje podataka i također olakšava rukovanje složenim, evoluirajućim strukturama podataka tipičnim za web aplikacije. Što je više vremena prolazilo i što je postajao zreliji, MongoDB je dodavao više značajki svojoj

ponudi što je sustav činilo još zanimljivijim. Prešao je s podržavanja horizontalnog skaliranja putem dijeljenja i skupova replika za visoku dostupnost na implementaciju vrlo moćnog jezika upita koji podržava bogate mogućnosti manipulacije podacima. Sve je to učinilo MongoDB savršenim za sve programere koji žele izgraditi aplikacije velikih razmjera koje zahtijevaju fleksibilnost i performanse.



Slika 1. Evolucija MongoDB sustava kroz povijest (Izvor: [2])

Padmanabhan [2] također navodi da je MongoDB sljedećih godina samo povećao svoju popularnost, postavši jedna od najčešće korištenih NoSQL baza podataka u svijetu. Korisnička baza ove tehnologije obuhvaća organizacije svih veličina, od startupa do vrlo velikih poduzeća, a primjenjuje se u nizu industrija. MongoDB Inc. izašao je na burzu 2017., čime se čvrsto učvrstio kao lider na tržištu baza podataka. Danas MongoDB nastavlja s inovacijama, uz kontinuirani razvoj usmjeren na poboljšanje performansi, sigurnosti i jednostavnosti korištenja, uz kontinuirano širenje usluga temeljenih na oblaku s MongoDB Atlasom. Ovo putovanje od eksperimenta u startupu do jednog od kamena temeljaca modernog upravljanja podacima naglašava transformacijski utjecaj koji je MongoDB imao na način na koji pohranjujemo podatke i upravljamo njima u ovom digitalnom dobu.

2.2. Metode manipulacije baze podataka

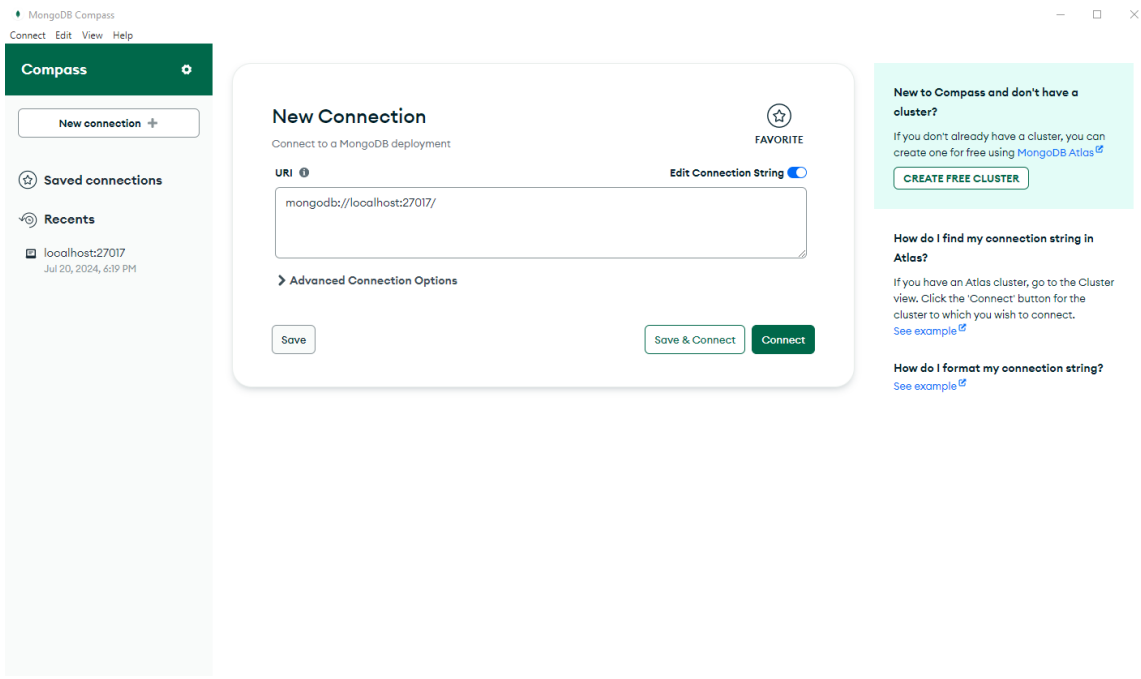
MongoDB serveru možemo pristupiti na nekoliko načina, za početak preko službenog „Compass“ programa koji je dostupan na službenoj stranici te se može preuzeti na računalo također poput CLI-a (Command Line Interface-a), dok je treća opcija web aplikacija. Bitno je

napomenuti da se CLI te Compass postoji za nekoliko verzija Linuxa te jedna verzija za sve Windows operativne sustave.

2.2.1. MongoDB Compass

„MongoDB Compass je GUI za istraživanje, analizu i interakciju sa sadržajem pohranjenim u MongoDB bazi podataka bez poznavanja ili korištenja upita.“ [3]

Izvor [3] izdvaja MongoDB Compass kao mnogo boljim izborom od Mongo Shell-a. Compass ima sposobnost da obavi sve što Mongo Shell može, a uz to nudi i mnoge dodatne funkcionalnosti, uključujući pregled i istraživanje podataka pohranjenih u bazi podataka, kreiranje baza podataka, umetanje, ažuriranje i brisanje podataka, pregled statistike poslužitelja u stvarnom vremenu, razumijevanje problema s performansama s pomoću vizualnih planova objašnjenja, upravljanje indeksima, validaciju podataka s pomoću JSON shema, te proširivost putem dodataka. Instaliranjem pune verzije Compass-a, moguće je iskoristiti bogate značajke MongoDB-a. Compass je besplatan za sve i olakšat će rad s MongoDB-om više nego bilo koji drugi alat.



Slika 2. Korisničko sučelje MongoDB Compass-a (Izvor: [24])

Slika 2 prikazuje korisničko sučelje MongoDB Compass-a koje se prikazuje po završetku instalacije istoimenog programa. Riječ je o alatu temeljenom na GUI-u za manipuliranje MongoDB-om. MongoDB Compass dizajniran je kao prijateljsko sučelje za rad s MongoDB bazama podataka. Sučelje je postavljeno na takav način da su neke od središnjih značajki i funkcionalnosti lako dostupne te su veoma uočljive i intuitivne što pojednostavljuje manipulaciju bazom podataka novim korisnicima.

Gumb „New connection“ u gornjem lijevom kutu omogućuje korisniku stvaranje nove veze s MongoDB instancom. Korisnik će se moći povezati sa svojom implementacijom MongoDB-a putem niza veze koji se nalazi unutar URI polja za unos navedenog u ovom odjeljku. Također uključuje opcije za uređivanje niza veze i pregled naprednih postavki veze.

Na lijevoj ploči nalaze se kartice za spremljene veze i nedavne veze, koje prikazuju prethodno spremljene veze odnosno nedavne veze za brzi pristup.

Na krajnjoj desnoj strani sučelja nalazi se odjeljak za pomoć koji sadrži upute o tome kako pronaći i formatirati niz za povezivanje i kako stvoriti besplatni klaster u slučaju da zatreba. Ovaj odjeljak je jako koristan kao pomoć novim korisnicima koji možda žele dodatnu pomoć pri postavljanju svojih veza.

U cjelini, MongoDB Compass pruža i sveobuhvatnu i intuitivnu vidljivost u MongoDB bazama podataka, potičući tako korisnika da pregledava, istražuje i učinkovito upravlja, odnosno manipulira, svojim podacima što je nešto zahtjevnije u CLI-u.

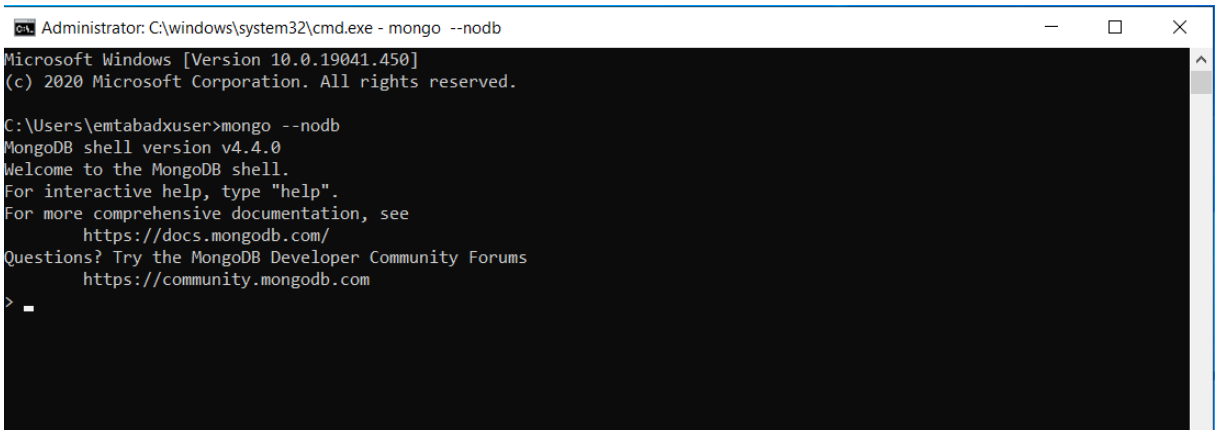
Završni se rad bazira na prikazu implementacije baze podataka u sustavu MongoDB te će sama implementacije u sljedećim sekcijama biti implementirana kroz korisničko sučelje MongoDB Compass-a.

2.2.2. MongoDB shell

Mongo DB Shell, ili jednostavno Mongo shell, moćan je uslužni program naredbenog retka za interakciju s MongoDB-om. Programerima i administratorima baza podataka daje sveobuhvatno okruženje za manipuliranje podacima unutar MongoDB-a.

Izvori [1] i [4] navode da u osnovi, MongoDB Shell pomaže korisnicima da se povežu s njihovim instancama MongoDB-a kako bi izvršili razne operacije, kao što su upiti u bazama podataka, umetanje, ažuriranje ili brisanje dokumenata i upravljanje indeksima. Bogat skup naredbi i funkcija ugrađenih u ljusku čini ga vrlo izvedivim za detaljne i složene interakcije s bazom podataka. Skriptiranje je jedna od temeljnih prednosti MongoDB Shell-a. Bilo koji JavaScript koji je netko napisao može se jednostavno pokrenuti unutar ljuske. Ovo korisnicima omogućuje automatizaciju zadataka koji se ponavljaju, izvršavanje skupnih operacija i

implementaciju prilagođene logike u obradi podataka. Ovu fleksibilnost iznimno cijene programeri koji obično moraju obavljati napredne manipulacije podacima i administratori koji traže učinkovite alate za upravljanje.



Slika 3. Naredbeni redak MongoDB-a (Izvor: [23])

Marcon [4] navodi da MongoDB Shell pruža interaktivne sesije koje korisnicima omogućuju da isprobaju naredbe i odmah vide rezultate. Ovo je vrlo korisno za rješavanje problema i fino podešavanje upita, budući da takve povratne informacije u stvarnom vremenu otkrivaju što će skup naredbi koje se izvršavaju unutar baze podataka učiniti. Iako se grafički alati poput MongoDB Compass-a sve više koriste, MongoDB Shell je uvijek tu za nekoliko korisnika zbog svoje snage i fleksibilnosti funkcionalnosti naredbenog retka. Osobito je preferiran među ljudima koji vole sučelja temeljena na tekstu i žele vrlo preciznu kontrolu nad svojim operacijama baze podataka.

Konačno, MongoDB Shell, je vrlo sveobuhvatan alat sa svim značajkama za upravljanje bazom podataka i razvoj NoSQL baza podataka. Izvršava složene naredbe, skripte za prilagođene operacije i omogućuje interakciju u stvarnom vremenu, što ga čini uistinu nezamjenjivim svakom profesionalcu koji se bavi razvojem NoSQL baza podataka.

2.2.3. MongoDB Atlas

MongoDB Atlas potpuno je upravljana usluga baze podataka u oblaku koju nudi MongoDB, Inc. Omogućuje implementaciju, upravljanje i skaliranje MongoDB baza podataka bez problema u okruženjima oblaka. Stoga Atlas omogućuje organizaciji da se usredotoči na izradu aplikacija bez zamršenosti administracije baze podataka.

Možda je najveća prednost MongoDB Atlasa njegova potpuno upravljana priroda. Brine se o svemu što je povezano s bazama podataka, od pružanja i konfiguracije do krpanja i

sigurnosnih kopija, ostavljajući po strani skaliranje. To omogućuje programerima i administratorima baza podataka da se usredotoče na razvoj aplikacija umjesto na tehničke detalje koji se tiču rada baze podataka.

MongoDB Atlas podržava globalnu implementaciju oblaka preko glavnih pružatelja usluga oblaka kao što su AWS, GCP i Microsoft Azure. To omogućuje organizaciji da odabere bilo kojeg pružatelja usluga oblaka koji joj se sviđa, a zatim ima fleksibilnost za jednostavnu distribuciju baza podataka u različitim geografskim regijama kako bi se osigurala veća izvedba i pouzdanost.

MongoDB Atlas ima napredne sigurnosne mogućnosti u obliku end-to-end enkripcije, mrežne izolacije i finih kontrola pristupa. Uvedene su odgovarajuće kontrole kako bi se podaci zaštitili od neovlaštenog pristupa ili kršenja, pružajući bezbrižnost organizacijama koje rade s osjetljivim informacijama.

Još jedna velika prednost MongoDB Atlasa je njegova skalabilnost. Omogućuje besprijekorno horizontalno skaliranje, što olakšava povezivanje baza podataka s aplikacijom. Uz automatsko dijeljenje i praćenje performansi u stvarnom vremenu, MongoDB Atlas osigurava da baze podataka mogu podnijeti povećana radna opterećenja bez gubitka performansi.

2.3. Tipovi podataka u MongoDB bazi

Kao što je u uvodu spomenuto, MongoDB podržava dosta tipova podataka te ćemo u nastavku navesti neke koji su biti konkretno za našu implementaciju baze podataka, odnosno najkorištenije tipove podataka u MongoDB-u. Za početak bitno je spomenuti da je BSON format koji koristi MongoDB za pohranu podataka i mrežni prijenos. BSON je kratica za Binary JSON i dizajniran je da bude lagan te učinkovit.

Chodorow [1] navodi da su dokumenti u MongoDB-u poput JSON objekata u JavaScriptu, no u binarnom zapisu. JSON, kao jednostavan i samoopisni, ima samo šest tipova podataka: null, Boolean, numerički, niz, polje i objekt. Međutim, postoji nekoliko tipova podataka koji su se pokazali prilično važnima kada radite s bazama podataka, a koje JSON ne podržava. Naime, tipovi datuma, brojevi s pomičnim udjelom u odnosu na cijele brojeve i tipovi za regularne izraze ili funkcije. Ima više tipova podataka podržanih u MongoDB-u nego u JSON-u, iako također održava format para ključ-vrijednost. Sljedeće su uobičajene vrste podataka koje MongoDB obično podržava:

- Null – koristi se za prikazivanje stanja null ili nepostojećeg polja

```
{"x": null}
```

- Boolean – koristi se za vrijednosti true i false:

```
{"x": true}
```

- Number – zadano koristi 64-bitne floating-point brojeve:

```
{"x": 3.14}

{"x": 3}
```

- Number – za cijele brojeve koriste se klase NumberInt (4-bajtni) i NumberLong (8-bajtni):

```
{"x": NumberInt("3")}

{"x": NumberLong("3")}
```

- String – bilo koji UTF-8 niz znakova može biti prikazan pomoću string tipa:

```
{"x": "foobar"}
```

- Date – datumi su pohranjeni kao milisekunde od epohe (UNIX epoch):

```
{"x": new Date()}
```

- Regular expression – regularni izrazi se koriste za napredna pretraživanja:

```
{"x": /foobar/i}
```

- Array – skupovi ili liste vrijednosti mogu biti prikazani kao nizovi:

```
{"x": ["a", "b", "c"]}
```

- Embedded document – dokumenti mogu sadržavati druge dokumente kao vrijednosti unutar glavnog dokumenta:

```
{"x": {"foo": "bar"}}
```

2.4. Dokumenti i zbirke

Koristeći MongoDB, većinu vremena u centru pažnje je upravljanje dokumentima na ovaj ili onaj način. Bilo to stvaranje novih dokumenata i dodavanje u zbirke, dohvaćanje dokumenata ili ažuriranje podataka, dokumenti su u središtu MongoDB modela.

2.4.1. Dokumenti

Izvori [5] i [6] navode da su dokumenti osnovna jedinica za pohranu podataka u MongoDB-u. Oni odgovaraju redovima relacijskih baza podataka, ali su mnogo fleksibilniji.

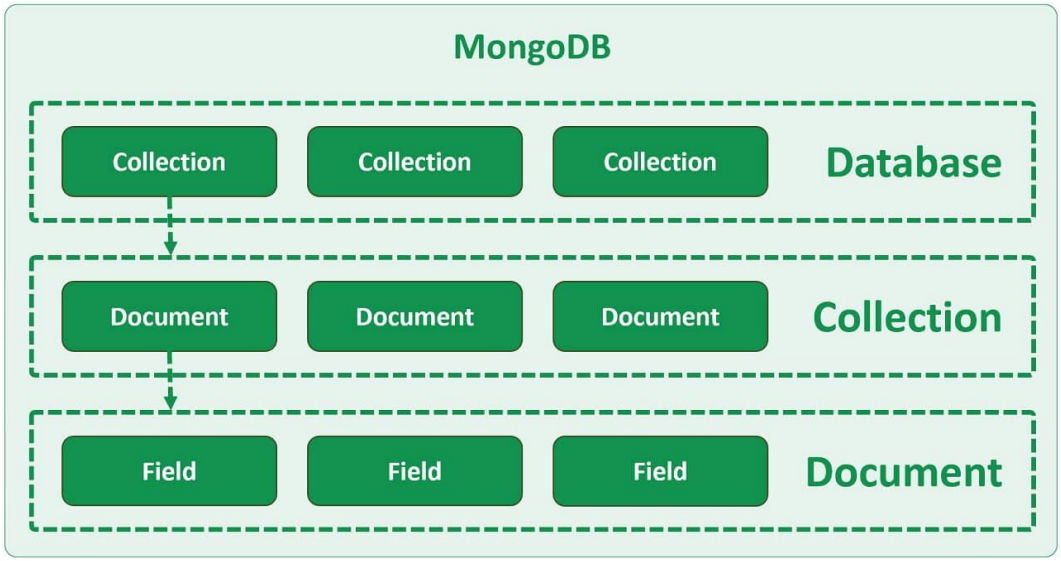
Dokumenti se pohranjuju u formatu koji se zove BSON, za binarni JSON kao što je već spomenuto. Predstavlja binarni oblik JSON-a koji podržava različite vrste podataka, uključujući nizove i ugniježđene objekte. Ovaj format čini MongoDB vrlo prilagodljivim različitim strukturama podataka.

Prema [6] vrijedi da svaki dokument u MongoDB-u je prikaz skupa parova ključ-vrijednost gdje su ključevi nizovi, ali vrijednosti mogu biti većina tipova podataka uključujući nizove, brojeve, datume, nizove, pa čak i druge dokumente. Dokumenti stoga mogu sadržavati različita polja; to jest, budući da su dokumenti po prirodi bez sheme, dokumenti unutar iste zbirke ne moraju se pridržavati nekog strogog formata, što čini pohranu podataka dinamičnom i heterogenom. Na primjer, jedan bi mogao predstavljati korisničke informacije s poljima za ime, e-poštu i adresu, a drugi bi mogao uključivati polja za telefon ili postavke.

2.4.2. Zbirke

Dokumenti su organizirani u zbirke u MongoDB-u. Zbirka se može usporediti s tablicom relacijske baze podataka, no međutim, nema unaprijed definiranu shemu. Zbirke sadrže dokumente i olakšavaju organiziranje i dohvaćanje. Razlog za ovu veliku fleksibilnost u načinu na koji se podaci pohranjuju i kako im se pristupa je činjenica da svaka zbirka može sadržavati dokumente različite strukture.

Chauhuan [5] navodi da zbirke u MongoDB-u stvaraju se u hodu. Kada se dokument umetne u zbirku, on automatski stvara zbirku ako već ne postoji. To je način na koji, za razliku od relacijskih baza podataka s krutim shemama, stvaranje u hodu omogućuje programeru da dizajnira svoj model podataka na vrlo fluidan način.



Slika 4. Arhitektura zbirki i dokumenata (Izvor: [6])

2.4.3. Prednosti i nedostaci

„MongoDB je moćna baza podataka koja je izvrsna u rukovanju s mnogo podataka i osiguravanju da su uvijek dostupni. Koristi fleksibilan način pohranjivanja podataka koji se zove dokumenti i zbirke. Kako tvrtka raste – može obraditi sve više i više podataka bez usporavanja. Njegove visoke performanse, dostupnost i skalabilnost čine ga dostupnim mnogim tvrtkama koje traže robustan pogon baze podataka.“ [7]

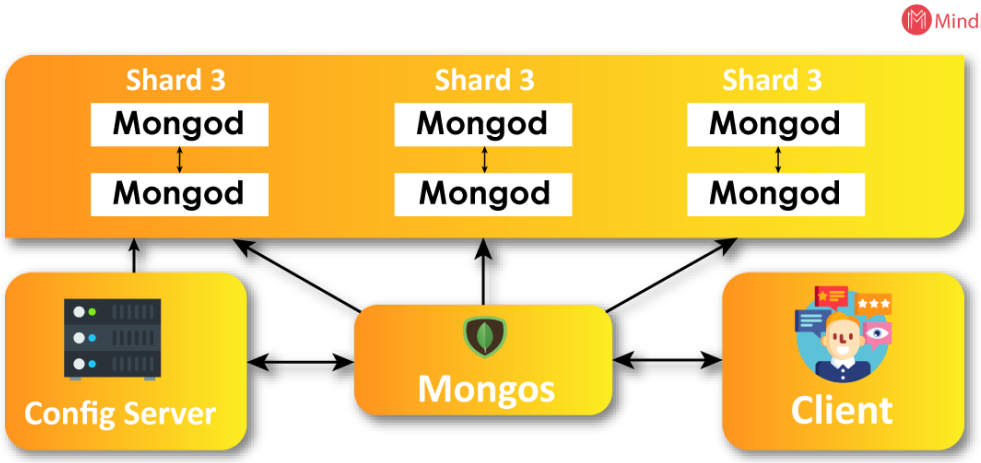
Prema izvoru [7] slijedi da je jedna važna prednost MongoDB-a je u fleksibilnosti koju donosi dizajnu sheme. Ima zbirke koje mogu sadržavati dokumente s različitim strukturama, za razliku od relacijskih baza podataka gdje je shema fiksna. To daje fleksibilnost podatkovnom modelu i jednostavnost mijenjanja za različite zahtjeve. Sljedeća važna prednost je skalabilnost. MongoDB omogućuje horizontalno skaliranje kroz dijeljenje, koje distribuira podatke na različite poslužitelje. Ovo se vrlo dobro uklapa u aplikacije koje imaju velike količine podataka ili veliki promet, budući da se dodatni poslužitelji uvijek mogu dodati kako bi se nosili s povećanjem opterećenja. Indeksiranje i agregacija značajke su koje dodatno poboljšavaju performanse MongoDB-a. Indeksi podržavaju brzo izvršavanje operacija upita. Okvir agregacije omogućuje obradu složenih podataka unutar baze podataka, što osigurava bolju učinkovitost.

Nadalje izvor [7] tvrdi da su neki od nedostataka MongoDB-a sljedeći: Jedan je da pune ACID transakcije nisu podržane. Dok novije verzije podržavaju transakcije s više dokumenata, MongoDB prema zadanim postavkama ne nudi sveobuhvatna ACID svojstva kao u relacijskim bazama podataka; ovo može stvoriti probleme u aplikacijama s visokim zahtjevima dosljednosti. Dosljednost podataka: Ovo je još jedan izazov. Budući da je shema fleksibilna, dosljednost podataka mora se provoditi na aplikacijskom sloju. Osim ako se pažljivo ne kontrolira, to može dovesti do nedosljednosti. Ipak, ova fleksibilnost MongoDB-a rezultira složenošću u modeliranju podataka. Zahtijeva mnogo razmišljanja o stvaranju učinkovitih struktura dokumenata i strategija indeksiranja za izvedbu.

2.5. Arhitektura

Visoke performanse, fleksibilnost i skalabilnost ono su za što karakterizira MongoDB arhitekturu. Za razliku od tradicionalnih relacijskih baza podataka, MongoDB je NoSQL baza podataka koja pohranjuje podatke u binarnom JSON formatu i stoga postaje prilagodljiviji različitim potrebama modernih aplikacija. U korijenu MongoDB arhitekture je model podataka

orijentiran na dokumente. Dokumenti u MongoDB-u donekle su slični JSON-u, koji sadrži podatke.



Slika 5. Arhitektura MongoDB sustava (Izvor: [8])

Vaishnavi [8] navodi da u osnovi, arhitektura MongoDB-a se temelji na tri glavne komponente: MongoDB poslužitelju, samoj bazi podataka i kolekcijama u toj bazi podataka. Poslužitelj je zadužen za upravljanje podacima, obradu upita prema tim podacima i jamčenje visoke dostupnosti i konzistentnosti podataka. Svaka instanca MongoDB-a radi pomoću poslužitelja. Poslužitelji su vodoravno skalirani korištenjem dijeljenja. Sharding znači distribuciju podataka na više poslužitelja te na ovaj način MongoDB učinkovito rukuje velikim količinama podataka i propusnim operacijama.

Vaishnavi [8] također navodi da je, osim toga, u poslužitelj ugrađen snažan mehanizam skupa replika koji pomaže visokoj dostupnosti i toleranciji grešaka. Može se zamisliti skup replika kao grupa MongoDB poslužitelja koji održavaju isti skup podataka, gdje je jedan poslužitelj primarni čvor, a drugi su sekundarni čvorovi. Sve operacije pisanja odvijaju se na primarnom čvoru, a podatke na primarnom repliciraju sekundarni čvorovi. U slučaju kvara primarnog čvora, jedan od sekundarnih čvorova automatski će biti promoviran kao primarni čvor, a sve će i dalje biti dostupno.

Osim snažnog podatkovnog modela i skalabilnosti, MongoDB podržava napredno postavljanje upita, indeksiranje i agregaciju, što korisnicima omogućuje izvršavanje složenih upita izravno u bazi podataka i analizu podataka. Indeksiranje je ključno za omogućavanje baze podataka za brzo lociranje i dohvaćanje podataka, čime se poboljšava izvedba upita. Cjevovodi za agregaciju, s druge strane, obrađuju podatke kroz niz faza, čime ih transformiraju i filtriraju kako bi prikazali rezultate.

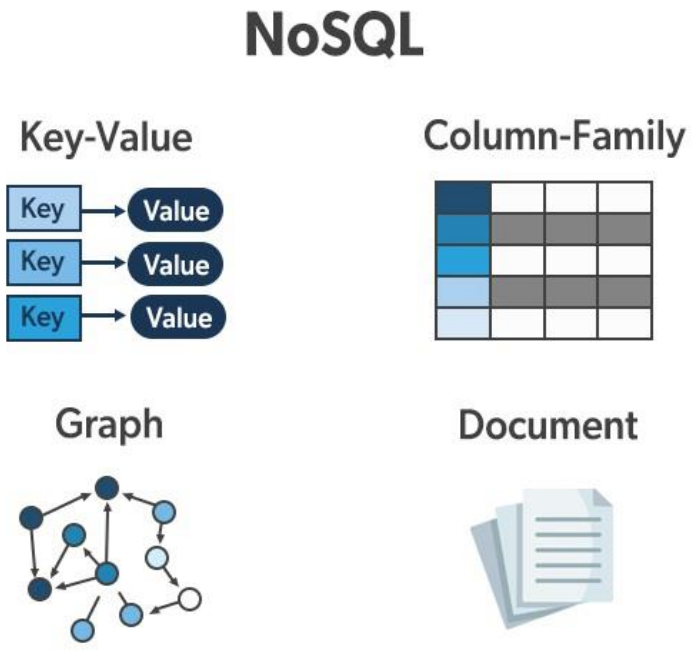
2.6. NoSQL

Molina, Sánchez i Pagán [9] navode da NoSQL (Not only SQL) odražava skupinu sustava za upravljanje bazama podataka koji odstupaju od uobičajenih modela relacijskih baza podataka. Takve baze podataka imaju za cilj rad s velikim količinama podataka, brzo unošenje i različite vrste podataka, čime se osigurava fleksibilnost, skalabilnost i visoka izvedba.

NoSQL baze podataka razvijene su kako bi pomogle u rješavanju slabosti tradicionalnih relacijskih baza podataka za rukovanje raznolikošću i dinamikom podataka koji dolaze iz današnjih aplikacija. Za razliku od relacijskih DB-ova, NoSQL baze podataka koriste različite modele podataka i fleksibilne sheme, umjesto SQL-a s fiksnom shemom, kako bi pružile agilnije i skalabilnije upravljanje podacima.

2.6.1. Osnovne značajke NoSQL baza podataka

Izvor [10] tvrdi da postoji nekoliko varijanti NoSQL baza podataka, od kojih svaka cilja na različite slučajeve upotrebe. Baze podataka orijentirane na dokumente poput MongoDB-a pohranjuju podatke u dokumente slične JSON-u. Svaki dokument može sadržavati parove ključ-vrijednost koji također mogu uključivati ugniježđene strukture, nizove, mnoge druge različite tipove podataka, te su stoga idealni za složene podatke s hijerarhijskom strukturom.



Slika 6. Osnovne značajke NoSQL baza podataka (Izvor: [10])

Vjerojatno najosnovnija vrsta NoSQL baza podataka su pohrane ključ-vrijednosti, koje uključuju Redis i Amazon DynamoDB. Njihov pristup pohrani podataka je jednostavan, čuvaju se u skupu parova ključ-vrijednost, pri čemu su ključevi jedinstveni i mapiraju se u određenu vrijednost. Iznimno su visokih performansi i stoga su prikladni za aplikacije koje zahtijevaju stvarno brz pristup podacima, kao što je predmemorija i upravljanje sesijom.

Pohrane obitelji stupaca, kao što su Apache Cassandra i HBase, pohranjuju podatke u stupce, a ne u retke. Rukovanje širokim i rijetkim skupovima podataka, prilično uobičajenim u aplikacijama s velikim podacima, trebalo bi biti prilično učinkovito s obzirom na čitanje i pisanje ogromnih skupova podataka.

Prema Molina, Sánchez i Pagán [9] baze podataka s grafikonima, kao što su Neo4j i Amazon Neptune, dizajnirane su na strukturama grafikona s čvorovima, rubovima i svojstvima za predstavljanje podataka i njihovo pohranjivanje. Mogu pokretati vrlo čvrsto međusobno povezane podatke i složene odnose; stoga su od najveće važnosti za aplikacije koje uključuju društvene mreže, mehanizme za preporuke i analizu mreže.

2.6.2. Usporedba SQL-a i NoSQL-a

NoSQL i SQL baze podataka dva su osnovna načina za upravljanje bazom podataka, a svaki karakteriziraju prilično različite karakteristike, prednosti i slučajevi korištenja. Poznavanje njihove razlike omogućuje odabir prave vrste baze podataka za specifične zahtjeve aplikacije.

NoSQL vs. SQL

	NoSQL	SQL
INTENDED PURPOSE	Specific use cases	General purpose
API	SQL not required	SQL required
DATA MODEL	Various, depending on NoSQL database type	Tables with fixed rows and columns
SCHEMA	Flexible	Rigid
SCALABILITY	Horizontal scale out	Vertical scale up
DATA INTEGRITY	BASE	ACID

Slika 7. Razlike NoSQL-a i SQL-a (Izvor: [11])

Izvor [11] navodi SQL baze podataka imaju strukturirane modele podataka s tablicama, redovima i stupcima, plus imaju unaprijed definiranu shemu. Ova instanca jamči da su podaci visoko organizirani, a odnos različitih podatkovnih entiteta dobro definiran korištenjem vanjskih

ključeva. Zbog ovih krutosti, SQL najbolje odgovara aplikacijama koje uključuju složene upite i transakcije, kao što su financijski sustavi i sustavi za planiranje poslovnih resursa.

Nasuprot tome, postoji niz podatkovnih modela na kojima NoSQL baze podataka rade, a samo su neki od modela orijentiranih na dokumente, modela ključ-vrijednost, obitelji stupaca i grafovske baze. U ovim bazama podataka nije potrebno imati unaprijed definiranu shemu, stoga je dobra u pružanju fleksibilne i dinamičke pohrane podataka. Ova je fleksibilnost vrlo korisna u aplikacijama koje sadrže nestrukturirane ili polustrukturirane podatke, kao što su platforme društvenih medija, sustavi za upravljanje sadržajem i analitičke aplikacije u stvarnom vremenu.

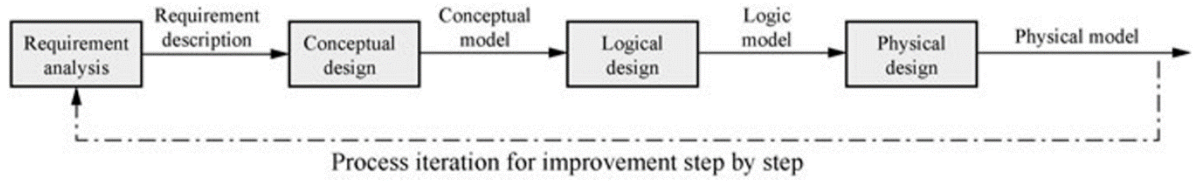
Također prema izvoru [11] možemo zaključiti da NoSQL i SQL baze podataka imaju različite snage i usmjerene su na različite vrste aplikacija. SQL baze podataka nude čvrsto i organizirano okruženje, pogodno za složene upite i transakcijske aplikacije sa strogim zahtjevima za integritet podataka. NoSQL baze podataka nude fleksibilnost, skalabilnost i visoku izvedbu; stoga su prikladni za moderne web aplikacije, velike podatke i analitiku u stvarnom vremenu. U ovom slučaju, NoSQL naspram SQL baze podataka temeljilo bi se na potrebama i ograničenjima konkretne aplikacije.

3. Identifikacija i definiranje zahtjeva

„U stvarnom životu, cijela zgrada bez dobrog temelja je loša. Iskustvo je pokazalo da loša analiza zahtjeva može izravno dovesti do pogrešnog dizajna. Ako se mnogi problemi ne otkriju do faze testiranja sustava, a zatim se vratite u želji da ih ispravite, to će biti skupo, tako da se fazi analize zahtjeva mora dati visoki prioritet. Faza analize zahtjeva uglavnom prikuplja informacije te ih analizira i organizira kako bi se osiguralo dovoljno informacija za sljedeće faze. Ova faza je najteža i najdugotrajnija faza, ali je i osnova cjelokupnog dizajna baze podataka. Ako analiza zahtjeva nije dobro obavljena, cijeli dizajn baze podataka može se preraditi.“ [12]

Huawei [12] navodi da u fazi analize zahtjeva potrebno je učiniti sljedeće:

- Shvatiti kako funkcionira postojeći sustav, uključujući usluge koje nosi postojeći sustav, proces usluge i njihove nedostatke.
- Odrediti funkcionalne zahtjeve novog sustava, odnosno razumjeti ideje krajnjeg korisnika, uključujući funkcionalne zahtjeve i željene rezultate.
- Identificirajte bitne podatke i povezane procese usluga suradnika koji bi postigli ciljeve, kako biste bili bolje pripremljeni za bolje razumijevanje procesa usluge i zahtjeva korisnika.



Slika 8. Proces izrade baze podataka (Izvor: [12])

3.1. Koraci u analizi zahtjeva baze podataka

Huawei [12] navodi da je analiza zahtjeva faza razvoja sustava u kojoj se cilja na razumijevanje i definiranje potreba korisnika i samog sustava. Ovo je faza u kojoj će se odvijati sustavno istraživanje ponašanja i procesa korisničkih usluga, sveobuhvatno istraživanje sustava, prikupljanje i analiza zahtjeva, definiranje opsega razvoja sustava i na kraju priprema detaljnog izvješća analize zahtjeva. Ova faza počinje analizom korisničkih ponašanja i procesa usluga. Postoji potreba za razumijevanjem percepcija korisnika o novom sustavu i njegovih

ciljeva iz njega. Iz toga proizlazi da se identificiraju glavni problemi i ograničenja postojećeg sustava. Svi ovi inputi smatraju se vrlo vrijednima za usklađivanje novog sustava s potrebama korisnika i istovremeno uklanjanje nedostataka postojećeg sustava. Nakon toga slijedi faza istraživanja sustava, gdje se izvlače i analiziraju zahtjevi.

Huawei [12] također smatra da se informacijsko istraživanje provodi kako bi se utvrdile sve vrste podataka koji će se koristiti u projektiranom sustavu baza podataka: izvori, načini prikupljanja podataka, format podataka i njihov sadržaj. Primarni cilj ovog zadatka je identificirati koji će podaci biti pohranjeni, obrađeni i korišteni od strane novog sustava. Na taj način dizajn baze podataka prihvaća sve vrste relevantnih informacija i podržava funkcionalne zahtjeve sustava. Sljedeći korak odnosi se na zahtjeve za obradu, gdje se potrebe korisnika pretvaraju u tehničke specifikacije. Odnosno, iz opisa jednostavnih za korištenje, prevedite ih u detaljne zahtjeve koje programeri sustava i računala mogu razumjeti. Definira operativne funkcije obrade podataka, redoslijed operacija i učestalost istih. Pojedinosti o tome kako se te operacije odnose na podatke također čine dio ovog koraka. Osim toga, utvrđuje vremena odgovora i metode obrade koje su korisnici potrebni da bi tvorili kritični dio specifikacije korisničkih zahtjeva.

„Naposljetku, za analizu su potrebni razumijevanje i dokumentiranje zahtjeva sigurnosti i integriteta. Osigurat će da sustav pruža potrebne značajke za zaštitu podataka od neovlaštenog pristupa i čuvanje podataka u izvornom obliku, osiguravajući njihovu točnost i pouzdanost. U osnovi, faza analize zahtjeva usmjerena je na postizanje dvije svrhe učinkovitosti i učinkovitosti novog sustava u odnosu na potrebe korisnika dovoljnim razumijevanjem i dokumentiranjem svih relevantnih podataka i zahtjeva za obradu.“ [12]

3.2. Metode analize

Huawei [12] navodi daj je ključni cilj analize zahtjeva opisati protok informacija i protok usluga korisnika. Tijek usluge odnosi se na trenutno stanje usluge, kao što je koja će se usluga pružati, koje će politike biti uključene, tko će pružati uslugu, kroz koju organizaciju, s kojim procesima itd. "Protok informacija" jednostavno se odnosi na protok podataka, uključujući izvor, kretanje i fokus podataka, kao i odnos između obrade podataka i usluge zajedno sa stopom generiranja i modifikacije podataka te učestalošću njihove modifikacije. Vanjske zahtjeve potrebno je navesti u fazi analize zahtjeva, uključujući povjerljivost podataka, vrijeme odgovora na upit i zahtjeve za izlazno izvješće itd.

Ovisno o realnosti i mogućoj korisničkoj podršci, u istraživanju zahtjeva mogu se koristiti različite metode, pregledi projektne dokumentacije i izvješća o postojećim sustavima,

razgovori s servisnim osobljem, upitnici. Ako je moguće, trebalo bi prikupiti i uzorke podataka iz postojećih uslužnih sustava za provjeru valjanosti nekih pravila usluge i ocjenjivanje kvalitete podataka u fazi projektiranja.

3.3. Rječnik podataka

Chodorow [1] opisuje da se rječnik podataka (eng. Data Dictionary) može generirati iz analize zahtjeva i analize podataka. Za razliku od rječnika podataka koji se nalazi u bazi podataka, rječnik analize ne sadrži podatke, već ih opisuje. Također sadrži opis podatkovnih stavki, njihovih tipova, duljina, raspona vrijednosti, jedinica i njihovih međusobnih odnosa s drugim podatkovnim stavkama koje će biti potrebne u optimizaciji modela tijekom faze logičkog dizajna ili ERA modeliranja.

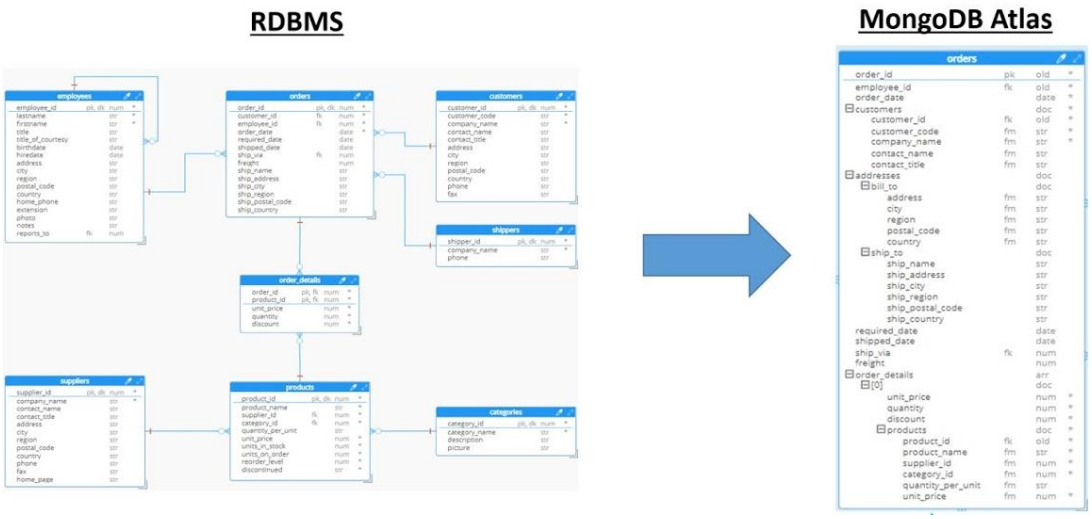
Odjeljak strukture podataka predstavlja kombinacije odnosa između podatkovnih stavki, a struktura podataka može imati mnoge stavke podataka i druge strukture. Ne postoji fiksni format za rječnik podataka i može se dokumentirati kroz razne opisne dokumente ili datoteke modela. U analizi zahtjeva, glavni izlaz će biti specifikacija zahtjeva korisnika, ali rječnik podataka često se dodaje istoj kako bi vodio dizajnere modela tijekom njihovog daljnjeg rada.

4. Projektiranje sheme podataka

Shema podataka odnosi se na strukturirani okvir koji definira organizaciju i strukturu podataka unutar baze podataka. Stoga je to nacrt o tome kako se podaci pohranjuju, organiziraju i njima upravlja te koliko se učinkovito informacije mogu dohvatiti i održavati. Također u osnovi ocrtava raspored elemenata podataka i njihove odnose, dajući tako neku vrstu glavnog plana za rukovanje podacima u sustavu baze podataka.

Općenito, podatkovna shema u relacijskoj bazi podataka će definirati strukturu tablica, stupaca i vrstu podataka koje će sadržavati, zajedno s odnosima koji postoje između različitih tablica. Objašnjava kako će podaci biti klasificirani, kakvu će vrstu ograničenja imati svaki element podataka i kako su različiti elementi podataka međusobno povezani. Na primjer, može odrediti tablicu za informacije o kupcima koja ima stupce za ID kupca, ime, adresu, telefonski broj i odnose s narudžbama i podacima o plaćanju.

Prema izvoru [13] vrijedi daj je koncept sheme još fluidniji u NoSQL bazama podataka, kao što je MongoDB. Umjesto krutog definiranja tablica i stupaca, NoSQL sheme većinu vremena podrazumijevaju definiranje dokumenata i zbirki. NoSQL dokumenti baze podataka obično se pohranjuju u formatima poput JSON (JavaScript Object Notation) ili BSON (Binary JSON), koji podržavaju jednostavne i hijerarhijske strukture podataka. Na primjer, u MongoDB-u jedna shema može definirati kolekciju korisničkih profila u kojima bi svaki dokument sadržavao različita polja: ime, e-poštu i popis narudžbi, recimo, sve se čuva unutar jednog JSON objekta.



Slika 9. Usporedba podatkovnih shema u RDBMS i MongoDB Atlasu (Izvor: [13])

Glavna svrha podatkovne sheme je organizacija podataka na takav način da promiče učinkovito upravljanje i dohvaćanje podataka. Također održava integritet podataka provođenjem pravila i ograničenja—jedinstveni identifikatori ili tipovi podataka. Osim toga, dobro dizajnirana shema dodatno će poboljšati performanse optimiziranjem izvršenja upita i indeksiranja.

Izvor [13] također navodi da u osnovi, dobru podatkovnu shemu treba dizajnirati na temelju razumijevanja podatkovnih zahtjeva aplikacije, određivanja načina na koji će se podaci koristiti i postavljati upite te definiranja odnosa između različitih podatkovnih elemenata. Ovaj je korak vrlo važan i treba ga učiniti s dužnom pažnjom, kako bi se uravnotežili konkurentski zahtjevi za fleksibilnošću, s jedne strane, i za dosljednošću i učinkovitošću podataka, s druge strane. Shema podataka općenito definira strukturu, organizaciju i odnos podataka u sustavu baze podataka. Štoviše, bilo da se radi o relacijskim ili NoSQL bazama podataka, postavljena dobra shema osigurava učinkovitu administraciju podataka i osigurava da je sustav obdaren kapacitetom da opslužuje aplikaciju na temelju svojih podataka.

4.1. Alati za izradu sheme podataka

U programu MongoDB postoji opcija za izradu sheme podataka, no postoje neke razlike u odnosu na ostale komercijalne alate za izradu sheme. Sheme su fleksibilnije u MongoDB nego u tradicionalnim relacijskim bazama podataka. Međutim, i dalje je vrlo korisno definirati shemu koja jamči dosljednost i cjelovitost podataka.

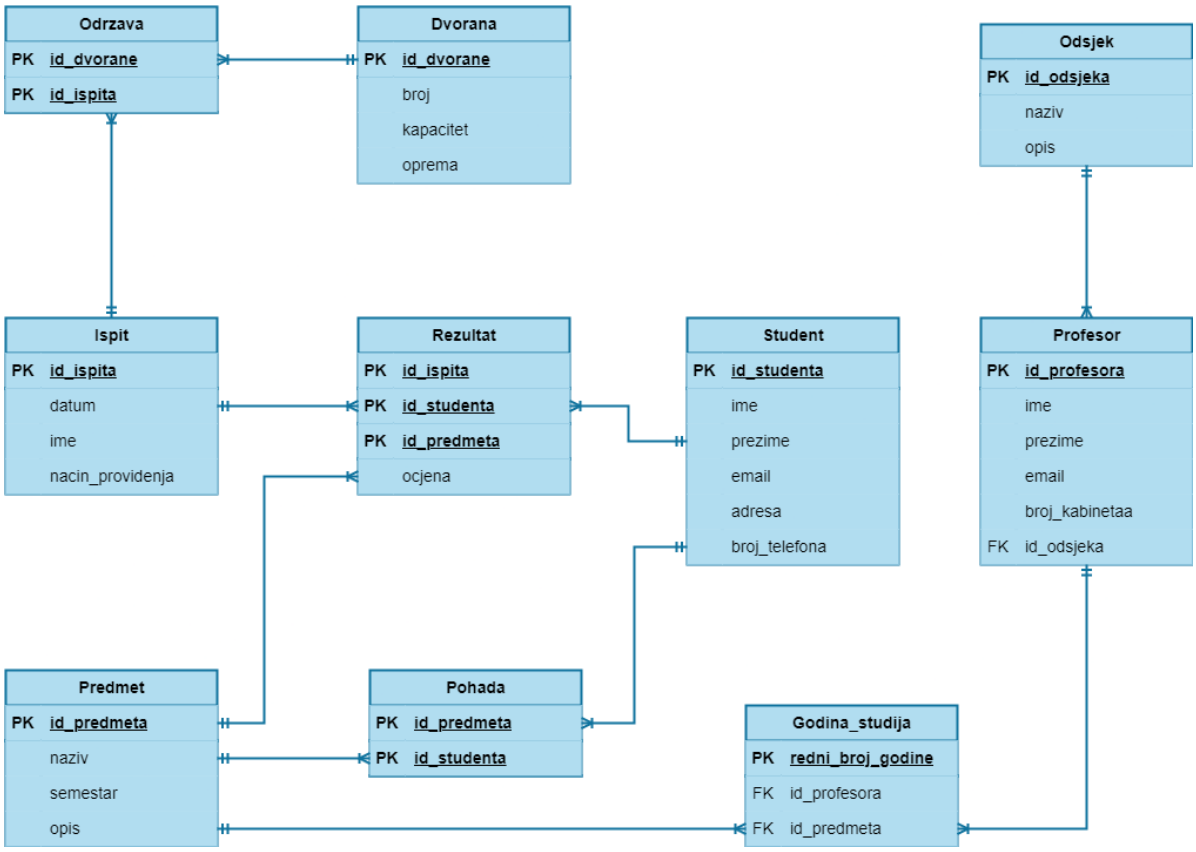
Izvor [14] navodi da je jedan od najčešće korištenih alata za projektiranje podatkovnih shema ER/Studio tvrtke IDERA. Još jedna značajka ER/Studia je izrada detaljnih dijagrama entiteta i veza. Tijekom razvoja pomaže korisnicima ciljati na veliki broj platformi baza podataka, a neke od njegovih funkcija su obrnuti inženjering, koji gradi ERD-ove iz postojećih baza podataka. Štoviše, ER/Studio podržava timski rad u razvoju projekta. Više korisnika može raditi na istoj shemi i biti sigurni u njezinu dosljednost i točnost.

Također prema izvoru [14] još jedan alat koji se široko koristi je Microsoft Visio. Alat za izradu dijagrama opće namjene Visio koristi se za izradu ER dijagrama i velikog broja drugih vrsta dijagrama. Ima prijateljsko korisničko sučelje s velikim brojem predložaka i oblika usmjerenih na dizajn baze podataka. Korisnik može jednostavno stvoriti podatkovnu shemu, definirati koji su odnosi između entiteta, pa čak i vidjeti pregled strukture baze podataka koristeći Visio. Njegova interoperabilnost s drugim Microsoft Office proizvodima dodatno dodaje značajke, što ga čini vrlo korisnim za previše organizacija.

4.2. ERA diagram

Matallah, Belalem, Bouamrane [15] opisuju ERA (eng. Entity-Relationship-Attribute) kao naprednu verziju tradicionalnog ER dijagrama jer dodaje detaljne informacije o atributima pridruženim svakom entitetu i odnosu. U usporedbi s ER dijagramima koji se u osnovi usredotočuju na identifikaciju entiteta, njihove odnose i kardinalnost među njima. Entiteti su predstavljeni kao objekti ili koncepti u sustavu, a njihovi atributi kao svojstva ili definicije značajki koje govore o entitetu u ERA dijagramima. Odnosi među entitetima također su obuhvaćeni, iako uz daljnje usavršavanje načina na koji su ti odnosi karakterizirani i ograničeni. To može obuhvatiti, primjerice, jesu li atributi obvezni ili izborni, jedinstveni ili nejedinstveni, te njihove vrste podataka i zadane vrijednosti. Pojedinosti koje sadrži ERA dijagram čine ga posebno korisnim tijekom faze implementacije dizajna baze podataka. U tom pogledu, budući da ERA dijagrami pružaju potpuni prikaz i strukturnih i opisnih elemenata podatkovnog modela, oni pomažu osigurati dobro definiranu shemu baze podataka za koju su ispunjeni svi zahtjevi sustava.

4.2.1. Primjer ERA diagrama



Slika 10. Primjer ERA diagrama za demonstraciju implementacije (Izvor: [15])

ERA diagram sa slike 8. označava strukturu sustava baze podataka unutar obrazovne ustanove, koja bi objasnila odnose između studenata i njihovih ispita, predmeta i profesora. Središnji dio dijagrama predstavlja entitet 'Ispit' u kojem su pohranjeni svi detalji o svakom pojedinom ispitu prema datumu, nazivu i načinu polaganja. Za svaki takav pregled postoji jedinstveni identifikator kroz 'id_ispita', koji je primarni ključ za svaki pojedinačni pregled.

S entitetom 'Ispit' povezan je entitet 'Rezultat' koji bilježi rezultate ispita za pojedine studente. Svaki 'Rezultat' je jedinstveno identificiran složenim ključem 'id_ispita', 'id_studenta' i 'id_predmeta' jer svaki rezultat, sam po sebi, sadrži ocjenu ili rezultat koji je postigao ovaj učenik, odnosno student.

Podaci za svakog studenta bilježe se u entitetu 'Student', s primarnim ključem 'id_studenta'. Entitet 'Pohada' povezan je sa svakim studentom, pokazujući na koje su predmeta upisani. Primarni ključ za ovaj entitet je kombinacija 'id_studenta' i 'id_predmeta', što pokazuje da student može biti upisan na nekoliko predmeta ili predmeta istovremeno.

Predmeti su opisani u entitetu 'Predmet'. Ima atribute kao što su naziv predmeta, semestar i kratak opis predmeta. Primarni ključ je 'id_predmeta'. Entitet 'Predmet' povezan je sa entitetom 'Godina_studija' koja svrstava predmete prema godinama studija i profesora na tom predmetu.

'Profesor' je entitet koji pohranjuje podatke o svakom profesoru te 'id_profesora' koji je primarni ključ. Profesori su odgovorni za predavanje na različitih godina studija i predmeta unutar tih studija. Entitet 'Odsjek' vezan je za entitet 'Profesor' na način da jedan odsjek može sadržavati više profesora, ali ne i obrnuto.

Entitet 'Odrzava' se također sastoji od dvokomponentnog primarnog ključa poput entiteta 'Pohada' te uključuje atribute kao 'id_dvorane' i 'id_ispita', budući da se jedan ispit može održavati u više dvorana isto tako u jedno se dvorani može održavati više ispita. Što se entiteta 'Dvorana' tiče, on sadrži identifikator, broj dvorane, kapacitet te opis opreme.

Kao zaključak, ER dijagram prikazuje povezanost podataka unutar akademskog sustava na sveučilištu ili nekoj drugoj obrazovnoj ustanovi, pružajući odgovarajuću organizaciju za upise studenata ili učenika, njihove rezultate ispita i upravljanje predmetima te profesorima.

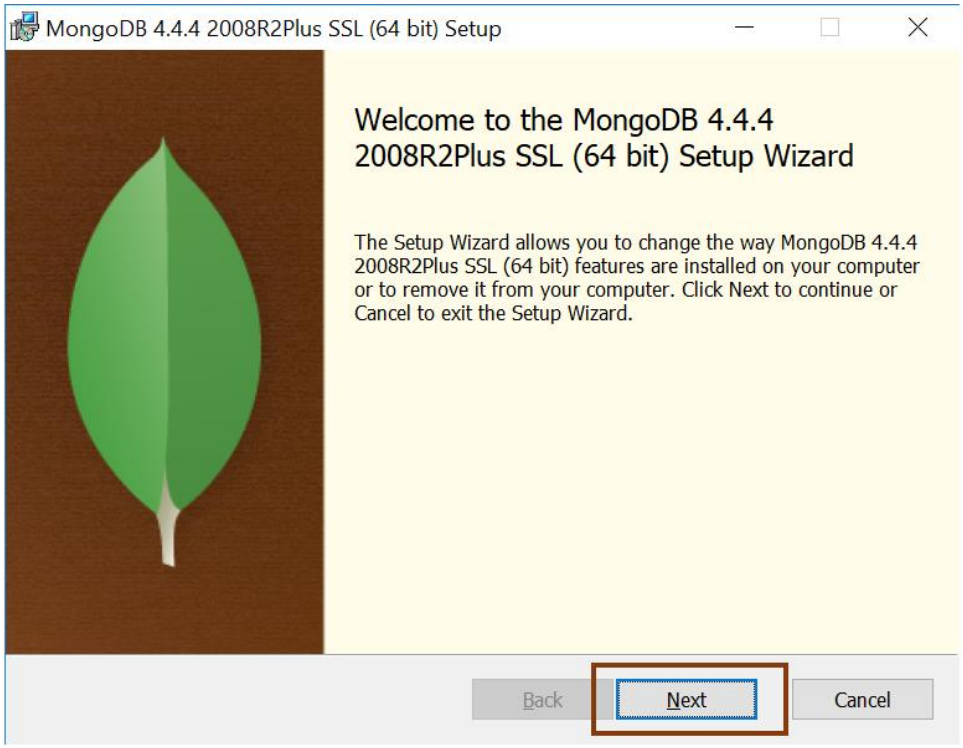
5. Implementacija

5.1. Instalacija

Instalacija kao instalacija je veoma jednostavna za MongoDB Compass, također kao i za naredbeni redak. Ovaj rad se konkretno bazira na implementaciji u operacijskom sustavu Windows.

Instalacija MongoDB-a na Windows računalo prilično je jednostavna. Korisnik treba preuzeti instalacijski program, pokrenuti čarobnjaka za instalaciju, konfigurirati varijable okoline i potvrditi instalaciju. Ovaj će vodič osigurati da je MongoDB ispravno instaliran i spreman za korištenje u zadacima upravljanja i razvoja koji se odnose na baze podataka.

Najprije ćete morati preuzeti instalacijski program za MongoDB sa službene stranice MongoDB. Najnoviju verziju MongoDB Community Servera koja odgovara Windowsu je moguće pronaći u MongoDB centru za preuzimanje. Nakon preuzimanja otvorite datoteku '.msi' koju ste preuzeli u svoju mapu za preuzimanja i ona će pokrenuti instalaciju.

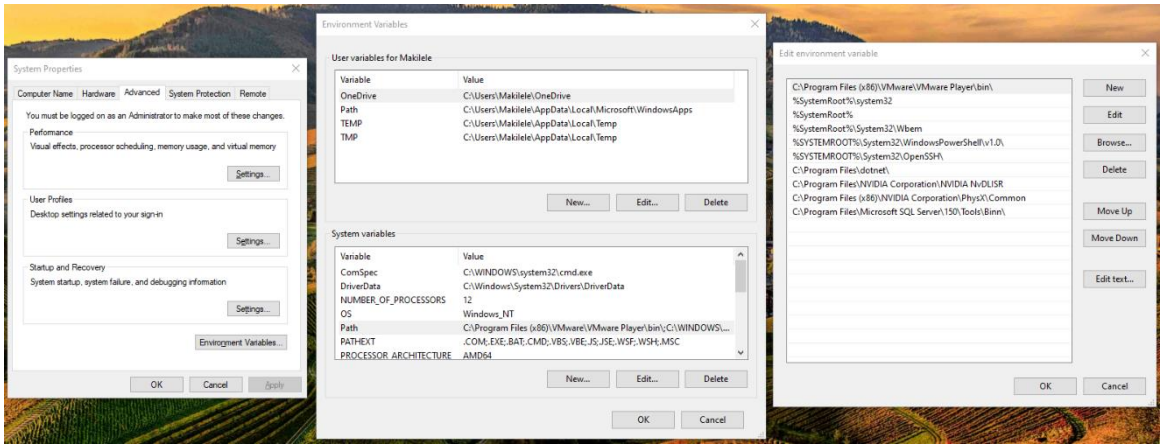


Slika 11. Čarobnjak za instalaciju MongoDB-a (Izvor: [24])

Ovdje će vas MongoDB čarobnjak za instalaciju upitati kroz nekoliko koraka. Najbolje je koristiti vrstu postavljanja „Dovršeno“ ili „Complete“ kako biste bili sigurni da su sve

relevantne komponente instalirane. Tijekom procesa instalacije dobit ćete priliku instalirati MongoDB kao Windows uslugu. Ako odaberete ovu opciju, konfigurirat će MongoDB da se automatski pokrene kad god se vaš sustav podigne i osigurati da poslužitelj baze podataka uvijek radi u pozadini. Ako želite ručno rukovati MongoDB-om, moguće je onemogućiti ovu opciju. Zatim ćete morati ručno pokrenuti MongoDB svaki put kada ga želite koristiti.

Nakon što je instalacija gotova, bilo bi dobro postaviti varijable okruženja tako da se može lako pristupiti MongoDB naredbenom retku u komandnoj liniji. Dodajte putanju do direktorija MongoDB `'bin'`, obično „C:\ProgramFiles\MongoDB\Server\<version>\bin“, u sistemsku varijablu okruženja `'PATH'`. Koristeći ovo, moguće je izvršavati MongoDB naredbe iz bilo kojeg prozora naredbenog retka bez potrebe da svaki put idete u direktorij `'bin'`.



Slika 12. Postavljanje varijable okruženja u sustavu Windows (Izvor: [24])

Prvo provjerite je li ovo ispravno instalirano otvaranjem prozora za naredbeni redak i pokretanjem naredbe 'mongod'. Naredba će pokrenuti MongoDB poslužitelj; većina poruka ispisuje da poslužitelj radi i čeka veze. Sada otvorite drugi prozor naredbenog retka i pokrenite naredbu 'mongo' da pokrenete MongoDB ljusku. To je zapravo sučelje naredbenog retka za MongoDB poslužitelj. Ako se obje naredbe uspješno izvode onda je MongoDB ispravno instaliran i spreman za korištenje.

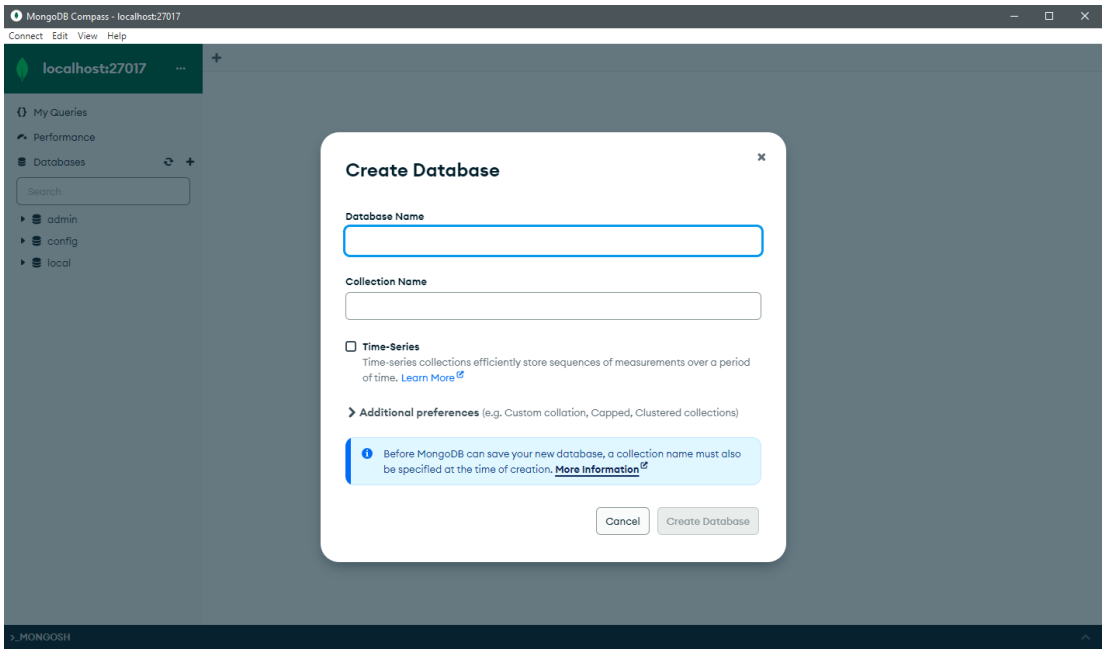
Izvor [16] navodi da je, ukratko, instalacija na Windows sustav slijedi ovu proceduru: preuzimanje instalacijskog programa, pokretanje instalacijskog čarobnjaka, postavljanje varijabli okoline i testiranje instalacije pomoću naredbi naredbenog retka. Kada se provode tim redoslijedom, ovi koraci jamče da će MongoDB biti pravilno postavljen i pokrenut, tako da je moguće učinkovito stvarati i upravljati svojim bazama podataka na Windows sustavu.

5.2. Izrada baze podataka i kolekcija

Kako biste kreirali bazu podataka u MongDB Compass-u prvo unesite detalje svog MongoDB poslužitelja. Ako MongoDB izvodite lokalno, tada se može koristiti zadani niz veze: `mongodb://localhost:27017`. Pritisnite gumb „Poveži se“ ili „Connect“ za povezivanje s MongoDB poslužiteljem.

Nakon uspješno uspostavljene veze, preusmjerit će se na glavnu nadzornu ploču MongoDB Compass-a. Na lijevoj strani pronaći ćete bočnu traku koja nabraja sve postojeće baze podataka. Gumb „Create“ obično se nalazi na vrhu bočne trake pored ikone za bazu podataka te idite na i kliknite da biste stvorili novu bazu podataka.

Pojavit će se prozor koji će tražiti naziv vaše nove baze podataka i naziv vaše prve zbirke. U MongoDB-u, zbirke su slične tablicama u relacijskim bazama podataka. Svaka baza podataka mora imati barem jednu kolekciju. Ispunite naziv svoje baze podataka i odaberite odgovarajući naziv za početnu zbirku, zatim kliknite gumb „Create database“.



Slika 13. Izrada baze podataka te početne kolekcije (Izvor: [24])

Nakon što je baza podataka stvorena, moguće je locirati na bočnoj traci. Sve zbirke prikazane su na stranici koja se otvara kada se klikne na naziv baze podataka. Putem stranice moguće je obavljati mnoge druge radnje nad zbirkama. Više zbirki može se dodati odlaskom na karticu „Collections“ unutar vaše baze podataka i klikom na gumb „Create“ odnosno plus

pokraj upravo kreirane baze podataka. Dizajner je u mogućnosti nastaviti širiti bazu podataka prema prije određenoj shemi ili kako mu odgovara.

Prema izvoru [17] se može zaključiti da MongoDB Compass također nudi razne značajke za učinkovito upravljanje bazom podataka. Može umetati, ažurirati i brisati dokumente unutar svojih kolekcija pomoću intuitivnog sučelja. Nadalje, omogućuje vizualizaciju podataka i stvaranje indeksa za optimizaciju izvedbe upita te podržava agregacije za analizu trendova u podacima i mnoštvo drugih opcija.

Izvor [17] nalaže kako izrada baze podataka se, ukratko, sastoji od slijedećih koraka: MongoDB Compassu uključuje povezivanje s vašim MongoDB poslužiteljem, klik na gumb „Stvori bazu podataka“ za navođenje baze podataka i njezine prve kolekcije, a zatim korištenje grafičkog sučelja za upravljanje bazom podataka. Svo upravljanje bazom podataka na dohvat ruke intuitivno okruženje, koje je odlično i za početnike i za iskusne administratore baza podataka.

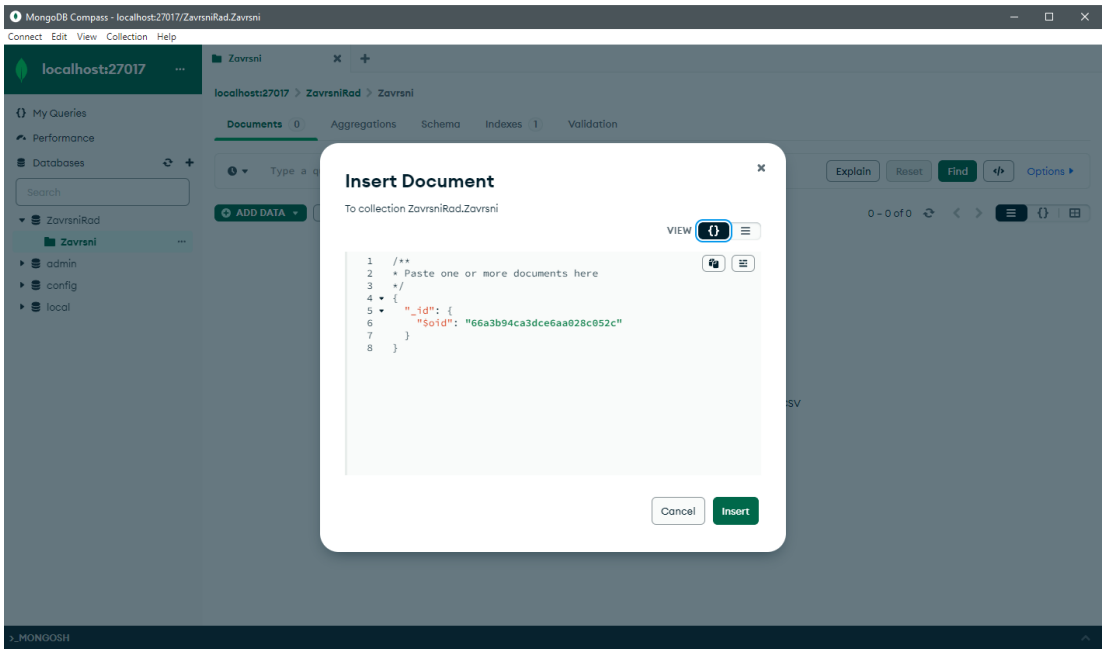
5.3. CRUD operacije

Truica, Boicea, Trifan [18] navode da je Temelj za interakciju bilo kojeg objekta s bazom podataka CRUD: stvaranje odnosno „Create“, čitanje ili „Read“, ažuriranje odnosno „Update“ i brisanje to jest „Delete“. U MongoDB-u ove radnje čine temelj učinkovitog upravljanja podacima unutar zbirki. Prva od ovih operacija, koja se odnosi na dodavanje novih dokumenata u zbirku, naziva se „Create“. Ovaj postupak je prilično jednostavan i koristi fleksibilnu shemu u MongoDB-u za prilagođavanje različitih struktura podataka.

5.3.1. Create

Za početak je potrebno imati kreiranu bazu podataka, a postupak je opisan u prošloj sekciji. Nakon što uđete u bazu podataka u koju želite umetnuti novi dokument, kliknite njegov naziv da biste ga proširili i vidjeli zbirke unutar njega. Kliknite na naziv zbirke u koju želite umetnuti dokument.

S odabranom zbirkom prikazat će se pregled dokumenata koji se trenutno nalaze u zbirci. Pronađite i kliknite gumb „Add data“, koji se obično nalazi pri vrhu prikaza popisa dokumenata za stvaranje novog dokumenta. Ovo će otvoriti novi dijaloški okvir ili prozor s uređivačem JSON za definiranje sadržaja za vaš novi dokument. Polja i vrijednosti se mogu dodati izravno u uređivač JSON za svoj dokument ili ukoliko već imate izrađen CSV ili JSON file sa podacima samo ga se može ubaciti principom „Drag and drop“ ili unos točne putanje na kojoj se nalazi dokument. Za primjer rada izraditi ćemo samostalno dokument.



Slika 14. Kreiranje novog dokumenta (Izvor: [24])

Nakon otvaranja JSON uređivača možemo pristupiti unosu podataka u naš dokument. Recimo da je riječ o nekim osobnim podacima. U nastavku se nalazi primjer jednog dokumenta:

```
{
  "name": "Ante",
  "email": "ante@example.com",
  "age": 30,
  "address": {
    "street": "Ulica tulipana 123",
    "city": "Zagreb",
    "state": "Zagrebacka",
    "zip": "10000"
  },
  "phoneNumbers": [
    {
      "type": "Kucni",
      "number": "555-333"
```

```
    },
    {
      "type": "Poslovni",
      "number": "222-333-4444"
    }
  ],
}
```

Moguće je dodati onoliko polja koliko želite, koja predstavljaju željenu strukturu podataka. Budući da MongoDB dopušta fleksibilnu shemu, svaki dokument može imati drugačiji skup polja. Dakle, nema ograničenja prema definiciji sheme.

Nakon što ispunite željene podatke, kliknite na gumb „Insert“ kako biste dodali dokument u zbirku. MongoDB Compass će umetnuti dokument jednim pokretom i tada biste ga trebali moći vidjeti na popisu dokumenata unutar zbirke. Compass će pokazati sve pogreške ili probleme s formatom vaših podataka tako da ih je moguće ispraviti i pokušati ponovno.

5.3.2. Read

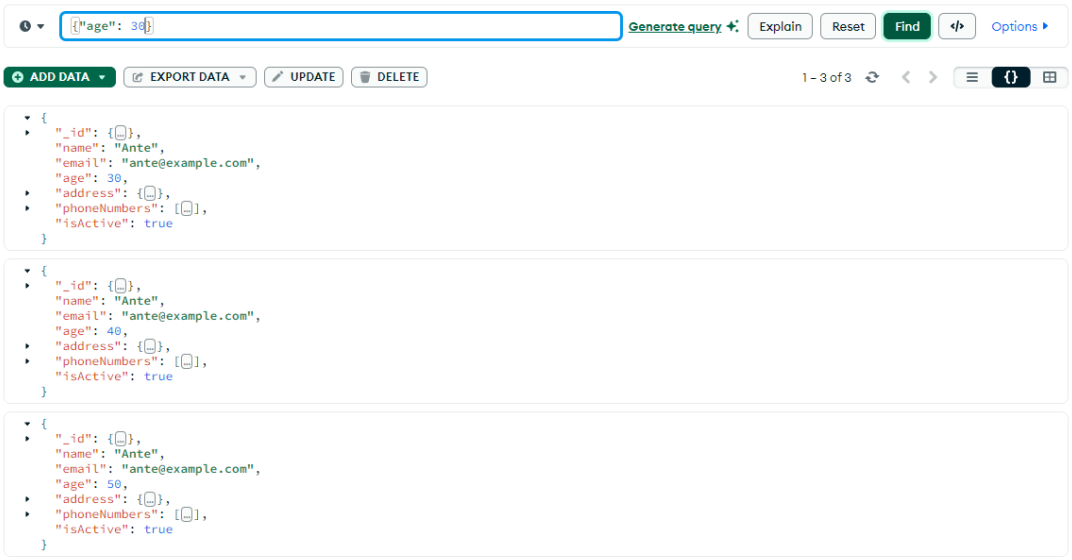
Truica, Boicea, Trifan [18] navode je Čitanje dokumenata putem MongoDB Compass-a jednostavno zbog dostupnosti grafičkog sučelja za isto. Omogućuje dohvaćanje odnosno čitanje dokumenata iz MongoDB kolekcija na prikladan način bez potrebe za aplikacijom za prevođenje koda i prikaz.

Popis svih baza podataka na poslužitelju dostupan je na bočnoj traci s lijeve strane. Potrebno je doći do baze podataka koja sadrži zbirku iz koje želite čitati podatke. Klikom na naziv baze podataka ona se proširuje kako bi se prikazale njezine sastavne zbirke. Potrebno je odabrati onaj koji je potreban klikom na naziv iste.

Nakon odabira, Compass će prikazati popis dokumenata koji su trenutno pohranjeni u ovoj zbirci. Prema zadanim postavkama, Compass prikazuje samo ograničeni broj dokumenata radi poboljšanja performansi, a moguće je promijeniti broj prikazanih dokumenata pomoću postavki na dnu prikaza popisa.

Bitno je napomenuti da postoji mogućnost da se izvršavaju se određeni upiti te se ti rezultati filtriraju u traci upita na vrhu prikaza popisa dokumenata. Unos MongoDB izraza upita za filtriranje dokumenata od interesa prema JSON standardima. Na primjer, za sve korisnike u zbirci korisnika koji imaju 30 godina imate sljedeći upit u traci upita:

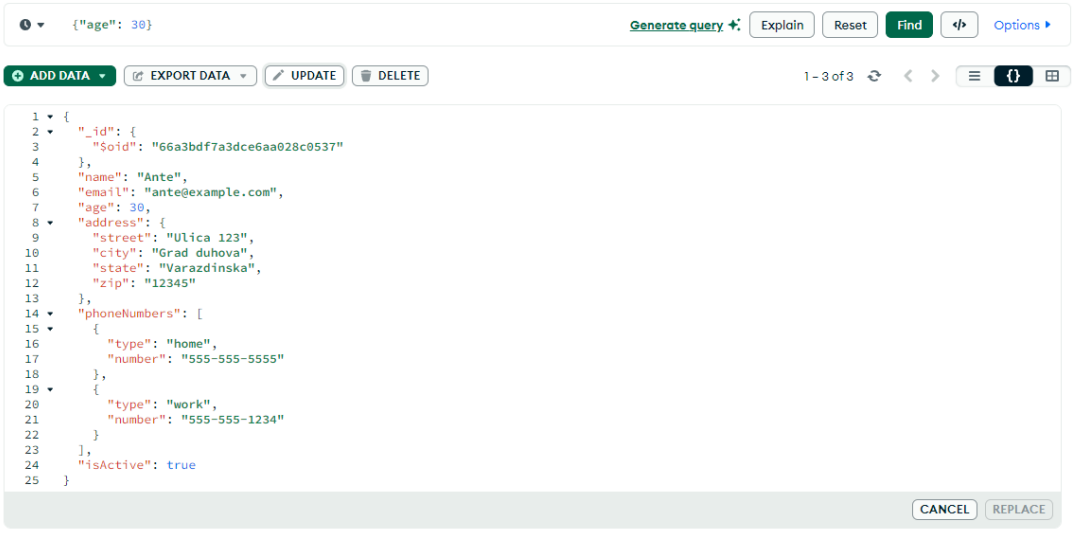
```
{ "age": 30 }
```



Slika 15. Pretraživanje uz pomoć upita (Izvor: [24])

5.3.3. Update

Za ažuriranje dokumenta, korisnik klikne gumb za uređivanje, koji je ikona olovke, pored dokumenta koji se ažurira. Ovo će otvoriti prikaz koji se može uređivati s vašim dokumentom u JSON formatu. Moguće je napraviti bilo koju promjenu koju želite izravno u ovom prikazu. Kliknite „Update“ odnosno ikonu olovke te po završetku s potrebnim izmjenama potrebno je odabrati opciju „Replace“. Ažuriranja koja ste napravili sada će biti pohranjena u bazu podataka u dokumentu.



Slika 16. Ažuriranje podataka (Izvor: [24])

Truica, Boicea, Trifan [18] navode da složenije operacije ažuriranja također se mogu izvesti korištenjem operatora ažuriranja s MongoDB Compassom. U slučaju da želite povećati neko numeričko polje ili želite postaviti bilo koje polje na potpuno novu vrijednost, može se koristiti dijaloški okvir Ažuriraj dokument. Na primjer, da biste povećali dob korisnika za 1, upotrijebili biste operator $inc:

```
{
  "$inc": { "age": 1 }
}
```

Također, za postavljanje nove vrijednosti polja koristite operator $set:

```
{
  "$set": { "email": "antonio@example.com" }
}
```

Unesite operaciju ažuriranja u dijaloški okvir i kliknite gumb „Update“ da biste primijenili svoju promjenu.

5.3.4. Delete

Prema Truica, Boicea, Trifan [18] brisanje dokumenata unutar MongoDB Compass-a uključuje povezivanje s bazom podataka, zatim navigaciju do ispravne kolekcije, korištenje trake upita za pronalaženje dokumenata koji vas zanimaju i klik na gumb za brisanje odnosno ikonu koša za smeće te će proces uklanjanja dokumenata započeti. To će omogućiti učinkovitost u vođenju i održavanju baze podataka uklanjanjem nevažnih ili zastarjelih podataka.

5.4. Povezivanje dokumenata

Kada je riječ o povezivanju dokumenata u sustavu MongoDB, ono se radi na dva različita načina. Razlikujemo referenciranje i ugradnju u dokument. Oba dvije metode imaju svoje prednosti i mane. U nastavku slijedi detaljna analiza.

5.4.1. Referenciranje

Referenciranje je proces stvaranja odnosa između dokumenata, obično preko različitih kolekcija unutar MongoDB baze podataka. Za razliku od ugrađivanja svih povezanih podataka u jedan dokument, referenciranje uključuje pohranjivanje „reference“, obično `'ObjectId'`, u

jednom dokumentu koji upućuje na drugi dokument koji se nalazi u drugoj zbirci. Ovaj je pristup sličan funkcioniranju stranih ključeva u relacijskim bazama podataka.

Primjerice uzmimo kolekciju podatak o kolegiju:

```
{
  "_id": {
    "$oid": "66c611666d19f809e43d2c18"
  },
  "id_kolegija": 1,
  "naziv": "Baze podataka 2",
  "semestar": 2
}
{
  "_id": {
    "$oid": "66c611666d19f809e43d2c12"
  },
  "id_kolegija": 2,
  "naziv": "Matematika 2",
  "semestar": 1
}
```

Ukoliko nekom studentu iz istoimene kolekcije želimo pridodati kolegij na način da ga referenciramo onda se to radi na način da se studentu uvrsti vrijednost atributa kao tip podataka array unutar uglatih zagrada te se unutar njih navode numeričke vrijednosti objekata koje želimo pridružiti tom studentu. Slijedi primjer studenta kojem su pridruženi predmeti: Baze podataka 2 te Matematika 2 preko vrijednosti `'ObjectID'`:

```
{
  "_id": {
     "$oid": "66c611546d19f809e43d2c15"
  },
  "ime": "Miljenko",
  "broj_iksice": 9012,
  "mail": "miljenko@foi.hr",
  "lozinka": "12se74",
  "predmeti": [
     {
       "$oid": "66c611666d19f809e43d2c18"
     },
```

```
        {
          "$oid": "66c611666d19f809e43d2c12"
        }
      ]
    }
```

Referenciranje u MongoDB-u je tehnika koja se koristi za povezivanje dokumenata u različitim zbirkama radi stvaranja odnosa između zasebnih dijelova podataka. Primjer njegove korisnosti je kada se podaci moraju održavati normaliziranim, bez redundancije, pohranjivanjem povezanih podataka u zasebne zbirke. Na primjer, podaci o kupcu mogu se pohraniti u jednoj zbirci, a detalji u vezi s narudžbama tog kupca pohranjuju se u drugoj zbirci, s mogućim referencama na prvu. Ova strategija ima brojne prednosti, poput većeg integriteta podataka, jer se promjene u jednoj zbirci automatski odražavaju na povezane podatke i učinkovitije pohranjivanje, budući da to ne zahtijeva ponavljanje informacija u dokumentima.

5.4.2. Ugradnja

Drugi način povezivanja dokumenata u MongoDB je ugrađivanje. To znači pohranjivanje povezanih podataka zajedno u jednom dokumentu, umjesto korištenja više zbirki za zasebne dijelove. Ova je metoda prikladna u slučajevima kada se podacima koji se odnose jedni na druge često pristupa zajedno ili je potreban odnos jedan-na-jedan ili jedan-na-nekoliko. Ugrađivanje dokumenata moglo bi sadržavati povezane podatke u jednoj strukturi koja nije nužno normalizirana, što može pomoći u lakšem dohvaćanju podataka i poboljšanju performansi.

Slijedi primjer s istim podacima kao i kod referenciranja:

```
{
        "_id": {
           "$oid": "66c611546d19f809e43d2c15"
        },
        "ime": "Miljenko",
        "broj_iksice": 9012,
        "mail": "miljenko@foi.hr",
        "lozinka": "12se74",
        "predmeti": [
           {
               "_id": {
               "$oid": "66c611666d19f809e43d2c18"
```

```
            },
            "id_kolegija": 1,
            "naziv": "Baze podataka 2",
            "semestar": 2
        },
        {
            "_id": {
            "$oid": "66c611666d19f809e43d2c12"
            },
            "id_kolegija": 2,
            "naziv": "Matematika 2",
            "semestar": 1
        }
    ]
}
```


Ugradnja u MongoDB pruža prednosti kao što su poboljšana izvedba i jednostavnost pohranjivanjem povezanih podataka unutar istog dokumenta. To rezultira bržim upitima sa samo jednim pretraživanjem baze podataka potrebnim za dohvaćanje svih informacija, što je korisno za aplikacije koje zahtijevaju puno čitanja. Međutim, nedostaci uključuju ograničenje veličine dokumenta od 16 MB, potencijalno dupliciranje podataka i povećanu složenost u upravljanju odnosima podataka. Pažljivo razmatranje ključno je pri odlučivanju između ugrađivanja i referenciranja u MongoDB-u kako bi se osigurala optimalna pohrana i dohvaćanje podataka. Unatoč njegovim prednostima, potrebno je uzeti u obzir potencijalna ograničenja kako bi se odredio najprikladniji pristup na temelju specifičnog slučaja upotrebe u MongoDB-u.

5.5. Primjer implementacije

Nakon usvajanja svih bitnih stavki same implementacije, počevši od definiranja zahtjeva, izrade sheme podataka, konkretno ERA diagrama, pa sve do izrade baze podataka i kolekcija te izrade dokumenata, možemo započeti implementaciju konkretne baze podataka uzevši u obzir sav dosadašnji rad. Primjer implementacije će biti demonstriran prema, već pokazanom, ERA diagramu koji se nalazi na slici 10.

Za početak je potrebno pristupiti već izrađenoj bazi podataka, nakon čega se možemo baciti u izradu dokumenata i kolekcija, odnosno implementaciju baze u sustavu MongoDB. Prvi korak je izrada kolekcija, u konkretno ovom slučaju entiteta, koji na sebe vežu samo vezu

jedan. Dakle to su entiteti: `'Dvorana'`, `'Ispit'`, `'Odsjek'`, `'Student'` te `'Predmet'`. Tim redoslijedom slijede isječci koda, odnosno primjer sa minimalno dva upisana podatka odnosno, u NoSQL bazama podataka, dokumenta. Dakle entitet `'Dvorana'` je istoimeni naziv kolekcije u kojoj se nalaze sljedeći dokumenti:


```
{
  "_id": {
    "$oid": "66c723256d19f809e43d2c59"
  },
  "id_dvorane": 1,
  "broj": 10,
  "kapacitet": 200,
  "oprema": "pametna_ploca"
},
{
  "_id": {
    "$oid": "66c7233e6d19f809e43d2c5b"
  },
  "id_dvorane": 2,
  "broj": 7,
  "kapacitet": 90,
  "oprema": "projektor"
}
```


Nadalje nam slijedi `'Ispit'`:


```
{
  "_id": {
    "$oid": "66c7247b6d19f809e43d2c5f"
  },
  "id_ispita": 1,
  "datum": {
    "$date": "2024-08-01T10:30:00.000Z"
  },
  "naziv": "Pisani_dio_ispita_iz_BP2",
  "nacin_provodenja": "kontaktno"
},
{
  "_id": {
```

```
    "$oid": "66c725806d19f809e43d2c60"
  },
  "id_ispita": 2,
  "datum": {
    "$date": "2024-09-05T12:40:00.000Z"
  },
  "naziv": "Usmeni_dio_ispita_iz_MAT2",
  "nacin_provodenja": "online"
}
```


Dokumenti iz kolekcije `'Odsjek'`:


```
{
  "_id": {
    "$oid": "66c72ae06d19f809e43d2c63"
  },
  "id_odsjeka": 1,
  "naziv": "Odsjek_informatike",
  "opis": "Odsjek_vezan_za_sve_informaticke_kolegije"
},
{
  "_id": {
    "$oid": "66c72b066d19f809e43d2c65"
  },
  "id_odsjeka": 2,
  "naziv": "Odsjek_matematike",
  "opis": "Odsjek_vezan_za_sve_matematicke_kolegije"
}
```


Slijedi nam `'Student'`:


```
{
  "_id": {
    "$oid": "66c72be66d19f809e43d2c68"
  },
  "id_studenta": 1,
  "ime": "Miljenko",
  "prezime": "Horvat",
  "email": "mhorvat@foi.hr",
```

```
  "adresa": "Varazdinska_ulica_80",
  "broj_telefona": "0919284710"
},
{
  "_id": {
    "$oid": "66c72c046d19f809e43d2c6a"
  },
  "id_studenta": 2,
  "ime": "Marko",
  "prezime": "Juric",
  "email": "mjuric@foi.hr",
  "adresa": "Zagrebacka_ulica_80",
  "broj_telefona": "0960172733"
}
```


Te na posljetku `'Predmet'`:


```
{
  "_id": {
    "$oid": "66c72cd66d19f809e43d2c6d"
  },
  "id_predmeta": 1,
  "naziv": "Baze_podataka_2",
  "semestar": 3,
  "opis": "Svijet_naprednih_baza_podataka"
},
{
  "_id": {
    "$oid": "66c72cf16d19f809e43d2c6f"
  },
  "id_predmeta": 2,
  "naziv": "Matematika_2",
  "semestar": 2,
  "opis": "Napredne_metode_matematike"
}
```

Nakon izrađenih 5 kolekcija odnosno entiteta sa ERA diagrama, potrebno je implementirati još toliko. Nar red je došao entitet `'Profesor'` koji je prvi u niz kod kojega ćemo koristiti veze, konkretno referenciranje, budući da se dokumenti referenciran na druge dokumente koji se nalaze u zasebnoj kolekciji. Slijedi implementacija gdje ćemo dva profesora smjestiti u isti odsjek preko njihovog identifikacijskog broja objekta:

```
{
  "_id": {
    "$oid": "66c751816d19f809e43d2c72"
  },
  "id_profesora": 1,
  "ime": "Davor",
  "prezime": "Juric",
  "email": "djuric@foi.hr",
  "broj_kabineta": 15,
  "id_odsjeka": {
    "$oid": "66c72ae06d19f809e43d2c63"
  }
},
{
  "_id": {
    "$oid": "66c752036d19f809e43d2c75"
  },
  "id_profesora": 2,
  "ime": "Ana",
  "prezime": "Mihic",
  "email": "amihic@foi.hr",
  "broj_kabineta": 70,
  "id_odsjeka": {
    "$oid": "66c72ae06d19f809e43d2c63"
  }
}
```

Dakle, nakon unesena 2 nova dokumenta za različite profesore možemo primijetiti da se oni nalaze u istom odsjeku odnosno da je identičan identifikacijski broj objekta za oba dva dokumenta te ukoliko potražimo taj identifikacijski broj u kolekciji odsjeka možemo vidjeti da su Davor i Ana zajedno u odsjeku za informatiku.

Slijede nam kolekcije odnosno entiteti koji na ERA diagramu posjeduju dvokomponentni primarni ključ, što znači da se odnos između relacija nije mogao ostvariti vezom više-na-više, već da tu stupa sekundarna tablica sa primarnim ključevima iz oba entiteta. Takvih je entiteta na primjeru slike 10 točno tri. To su: `'Odrzava'`, `'Pohada'` te `'Godina_studija'`. Slijede implementacije istim tim redoslijedom:


```
{
  "_id": {
    "$oid": "66c755456d19f809e43d2c78"
  },
  "id_dvorane": {
    "$oid": "66c7233e6d19f809e43d2c5b"
  },
  "id_ispita": {
    "$oid": "66c725806d19f809e43d2c60"
  }
},
{
  "_id": {
    "$oid": "66c7554a6d19f809e43d2c7a"
  },
  "id_dvorane": {
    "$oid": "66c723256d19f809e43d2c59"
  },
  "id_ispita": {
    "$oid": "66c725806d19f809e43d2c60"
  }
}
```


Dakle možemo iščitati da se identičan ispit održavao u dvije različite dvorane te možemo zaključiti da jedna dvorana nije imala dovoljan kapacitet sjedišta za sve studente. Budući da je ova kolekcija „glumi“ vezu više-na-više, onda možemo zaključiti da se također u

jednoj dvorani može održavati više ispita te da nije svaka dvorana striktno vezana za jedan predmet.

Slijedi implementacija kolekcije, odnosno entiteta `'Pohada'`:


```
{
  "_id": {
    "$oid": "66c759586d19f809e43d2c8a"
  },
  "id_predmeta": {
    "$oid": "66c72cd66d19f809e43d2c6d"
  },
  "id_studenta": {
    "$oid": "66c72be66d19f809e43d2c68"
  }
},
{
  "_id": {
    "$oid": "66c7595d6d19f809e43d2c8c"
  },
  "id_predmeta": {
    "$oid": "66c72cf16d19f809e43d2c6f"
  },
  "id_studenta": {
    "$oid": "66c72c046d19f809e43d2c6a"
  }
}
```


Ova se kolekcija također ponaša kao odnos dokumenata više-na-više te je moguće uvrstiti podatke na način da jedan student studira više predmeta te isto tako da jedan predmet izučava više studenata.

Na posljetku slijedi implementacija `'Godina_studija'`:


```
{
  "_id": {
    "$oid": "66c75cb56d19f809e43d2c97"
  },
  "redni_broj_godine": 1,
```

```
    "id_profesora": {
      "$oid": "66c751816d19f809e43d2c72"
    },
    "id_predmeta": {
      "$oid": "66c72cd66d19f809e43d2c6d"
    }
  },
  {
    "_id": {
      "$oid": "66c75cba6d19f809e43d2c99"
    },
    "redni_broj_godine": 2,
    "id_profesora": {
      "$oid": "66c752036d19f809e43d2c75"
    },
    "id_predmeta": {
      "$oid": "66c72cf16d19f809e43d2c6f"
    }
  }
```


Ova konkretna implementacija je nešto drugačija od posljednje dvije, budući da se ovdje ne nalazi dvokomponentni primarni ključ no odnos veza je identičan. Na istoj godini studija može biti više profesora koji predaju više predmeta i obrnuto.

Zadnje što se implementacije tiče je implementacija ternarne veze `'Rezultat'` koja je najkompleksnija od svih do sad navedenih:


```
  {
    "_id": {
      "$oid": "66c75ecc6d19f809e43d2ca1"
    },
    "id_ispita": {
      "$oid": "66c7247b6d19f809e43d2c5f"
    },
    "id_studenta": {
```

```
      "$oid": "66c72be66d19f809e43d2c68"
    },
    "id_predmeta": {
      "$oid": "66c72cd66d19f809e43d2c6d"
    },
    "ocjena": 4
  },
  {
    "_id": {
      "$oid": "66c75ed46d19f809e43d2ca3"
    },
    "id_ispita": {
      "$oid": "66c725806d19f809e43d2c60"
    },
    "id_studenta": {
      "$oid": "66c72c046d19f809e43d2c6a"
    },
    "id_predmeta": {
      "$oid": "66c72cf16d19f809e43d2c6f"
    },
    "ocjena": 5
  }
```


Ova se ternarna veza tumači na sljedeći način. Jedan student može pisati više ispita iz više predmeta. Odnosno da svaki student iz svojih nekoliko predmeta može pisati više ispita. Isto se tumači sa sve tri strane, bilo riječ o studentu, ispitu ili predmetu iz kojeg je pisan ispit.

Ovom se kolekcijom završava implementacija ERA diagrama sa slike 10 te se sada mogu poduzimati razne manipulacije, brisanja, dodavanja, pregledavanja podataka. Kada govorimo o samoj manipulaciji, ona je pobliže objašnjena u sljedećem naslovu.

6. Upravljane podacima

Chopade, Pachghare [19], Singh [20] te Sajimon, Benymol [21] Učinkovito rukovanje podacima u MongoDB-u ključno je za osiguranje učinkovitosti, sigurnosti i pouzdanosti baze podataka. To uključuje aktivnosti kao što su indeksiranje podataka, osiguranje sigurnosti podataka, izrada sigurnosnih kopija i praćenje performansi sustava. Indeksiranje poboljšava učinkovitost upita brzim pronalaženjem podataka. MongoDB nudi podršku za različite vrste indeksa kao što su indeksi temeljeni na jednom polju, složeni indeksi te geoprostorni indeksi. Sigurnost podataka uključuje ograničavanje pristupa odobrenim korisnicima pomoću kontrola provjere autentičnosti i autorizacije. Enkripcija štiti podatke sprječavajući njihovo presretanje ili krađu. Redovito korištenje programa kao što su mongodump i mongorestore jamči da se podaci mogu vratiti kada je to potrebno. Alati za praćenje kao što je MongoDB Atlas prate metriku performansi kako bi unaprijed otkrili i riješili probleme te kako bi se osiguralo optimalno funkcioniranje baze podataka.

6.1. Indeksiranje

Chopade, Pachghare [19] navode da je indeksiranje jedna od tehnika optimizacije baze podataka koja može dramatično poboljšati izvedbu operacija dohvaćanja podataka. Slično kao indeks u knjizi, gdje se dolazi izravno do informacija bez čitanja cijelog sadržaja, tako indeks u bazi podataka pomaže mehanizmu baze podataka da brzo locira i pristupi podacima potrebnim za upit. Ovo je vrlo važno za velike skupove podataka gdje bi potpuno skeniranje podataka bilo vrlo sporo.

U osnovi, indeks u bazi podataka je podatkovna struktura od kojih je većina B-stablo koje održava poredak u indeksiranim poljima, dopuštajući učinkovit način pretraživanja, umetanja i brisanja. Putem indeksa, bazi podataka je omogućeno sužavanje pretraživanja na podskup na učinkovit način, smanjujući količinu podataka koje treba skenirati.

6.1.1. Osnovni principi indeksiranja

Chopade, Pachghare [19] navode da je Prvo i najvažnije, razumijevanje obrazaca upita vrlo važno u smislu učinkovitog indeksiranja. Poznavanje vrste upita koji se često izvršavaju i onih koji su kritični za izvedbu pomaže u dizajnu indeksa koji ciljaju polja i operacije za koje se najčešće postavljaju upiti. Pretpostavimo, primjerice, da aplikacija često traži dokumente na temelju adrese e-pošte korisnika. Tada stvaranje indeksa u polju e-pošte dramatično povećava brzinu procesije upita.

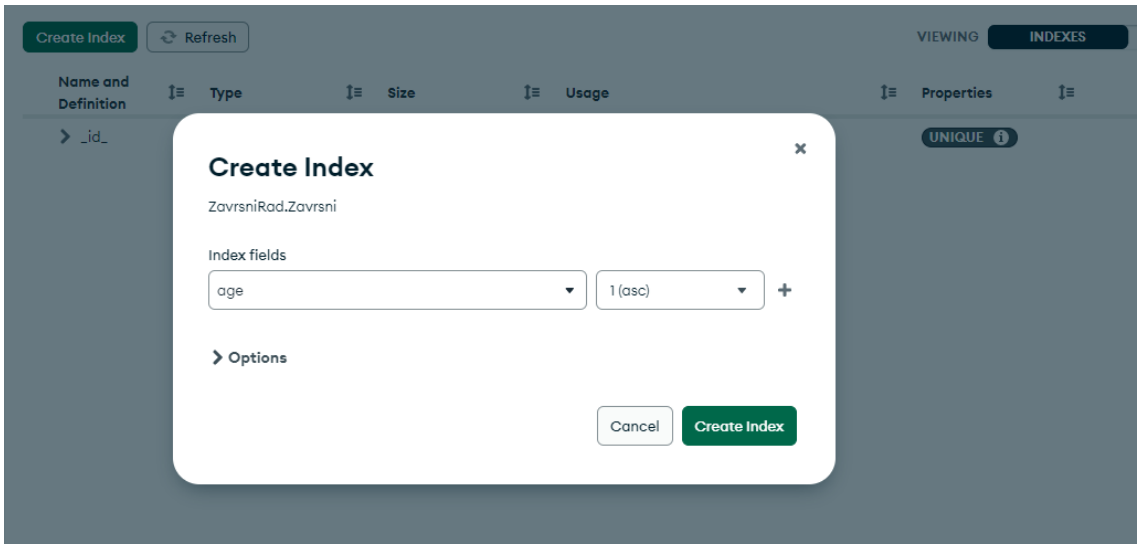
Također, Chopade, Pachghare [19] navode da je drugi važan aspekt selektivno indeksiranje. Indeksi bi se trebali postaviti na često korištena polja u upitima, posebno na ona koja sudjeluju u operacijama sortiranja i filtriranja. Stvaranje indeksa za polja za koja se rijetko postavlja upit ili za neselektivna polja, poput polja s malo mogućih vrijednosti, donosi dodatne troškove bez bitne koristi u povratu performansi. Također je važna selektivnost polja, koja označava jedinstvenost vrijednosti u polju. Selektivnija polja, poput korisničkog ID-a, obično su dobri kandidati za indeks, za razliku od manje selektivnih polja poput Booleovih oznaka.

Složeni indeksi mogu se koristiti u slučaju upita koji sadrže više polja. Složeni indeks je indeks na više polja unutar zbirke, što omogućuje MongoDB-u da učinkovito izvršava upite filtrirajući ili sortirajući prema više od jednog polja. Prilikom dizajniranja samog složenog indeksa, redoslijed polja u indeksu trebao bi se temeljiti na redoslijedu kojim se polje koristi u upitu. Recimo, primjerice, filtriranje koje se često vrši prema dobi, a zatim prema lokaciji unutar upita. U tom slučaju, složeni indeks s poljem „age“ na prvom i „address“ na drugom će biti korisniji gledajući performanse sustava.

Kreirane indekse treba redovito pratiti i pregledavati. MongoDB je implementirao alate i naredbe, poput funkcije objašnjenja koda, koji omogućuju da vidi kako upiti koriste indekse i procjenjuju njihov učinak na izvedbu. Ovo redovito praćenje će opisati koji indeksi imaju lošu izvedbu i tako prilagoditi svoju strategiju indeksiranja. Uklanjanje ili mijenjanje onih indeksa za koje se pokaže da nisu korisni poboljšat će opću izvedbu baze podataka.

6.1.2. Kreiranje indeksa

Za izradu novog indeksa kliknite gumb „Create index“ koji otvara dijaloški okvir u kojem se određuju polja koja će se indeksirati i vrstu indeksa koji će se stvoriti. Razmotrivši slučaj kada treba kreirati indeks s jednim poljem u polju „age“. Potrebno je navesti „age“ u polju za unos i odaberite uzlazni ili silazni tip indeksa, ovisno o potrebama baze podataka. MongoDB Compass također podržava složene indekse, odnosno indekse na više polja. Opcijom „Add row“, odnosno ikonom plusa u dijaloškom okviru za izradu složenog indeksa i navođenje dodatnih polja. Može se napraviti složeni indeks prema dobi i lokaciji dodavanjem oba polja i odabirom odgovarajućih vrsta indeksa. To omogućuje potpunu mogućnost učinkovite optimizacije upita koji filtriraju ili sortiraju prema više od jednog kriterija.



Slika 17. Kreiranje jednostavnog indexa (Izvor: [24])

Indeks se stvara klikom na gumb „Create index“ nakon što su navedena polja i njihove vrste. MongoDB Compass će izgraditi ovaj indeks i biti će vidljiv u kolekciji na popisu indeksa. Na kartici „Indexes“ pronaći će te novi indeks s detaljima o njegovim poljima, vrsti i statusu.

6.2. Sigurnost

Singh [20] navodi da osiguravanje sigurnosti podataka u MongoDB-u znači postavljanje mjera za izbjegavanje ulaska i kršenja pravila baze podataka od strane neovlaštenih korisnika. To uključuje postavljanje vrlo robusnog mehanizma provjere autentičnosti, primjerice, postavljanje korisničkih imena s lozinkama za sve korisnike baze podataka. Nakon toga se vrši postavljanje uloga i kontrola pristupa gdje RBAC pomaže definirati i nametnuti radnje koje svaki korisnik može učiniti. Drugi kritični aspekt je šifriranje. Informacije u mirovanju i u prijenosu bit će šifrirane, što će nekome otežati čitanje osjetljivih informacija.

6.2.1. Sigurnosne prakse

Osim što je iznimno popularna NoSQL baza podataka, MongoDB, kao i mnogi drugi, naglašava jake sigurnosne implementacije za zaštitu i integritet ključnih podataka. Važni dijelovi dobrog sigurnosnog rada uključuju autentikaciju, autorizaciju i enkripciju. Svaki od njih igra ključnu ulogu u svakom sigurnom i pouzdanom aspektu baze podataka.

Singh [20] navodi da se autentikacija može smatrati prvom linijom obrane u MongoDB-u. To je proces provjere identiteta bilo koje vrste korisnika ili aplikacije koja zahtijeva pristup bazi podataka. MongoDB ima podršku za višestruke mehanizme provjere autentičnosti kako

bi pristup podacima omogućio samo ovlaštenom korisniku. Najraširenija tehnika autentikacije je takva da se za svakog korisnika dodjeljuje korisničko ime i lozinka. Mogao bi postojati zadani mehanizam provjere autentičnosti, kao što pruža MongoDB. Nadalje, MongoDB se može povezati s vanjskim pružateljima autentikacije kao što je LDAP kako bi se postigla autentikacija poslovne klase.

Singh [20] također navodi da se autorizacija u osnovi oslanja na autentikaciju. To je proces davanja korisnicima određenih ovlasti za obavljanje određenih aktivnosti u bazi podataka. MongoDB ima ono što se naziva kontrola pristupa temeljena na ulogama koja regulira što korisnik može ili ne može učiniti. U RBAC-u, korisnik je definiran različitim ulogama koje određuju korisnikove sposobnosti i aktivnosti koje se mogu izvoditi na bazi podataka. Te se uloge mogu unaprijed odrediti kao samo za čitanje ili za čitanje-pisanje ili čak dodatno implementirane kako bi se prilagodile određenim sigurnosnim ograničenjima.

Prema Singh [20] ključna važnost MongoDB-ove enkripcije odnosi se i na podatke u mirovanju i na podatke u prijenosu. Enkripcija u mirovanju je enkripcija pohranjenih podataka na disku koja dopušta pristup samo odabranim osobama ili aplikacijama. Obično se šifriranje pohrane vrši pomoću naprednog standarda šifriranja u bazi podataka. Dodatno, MongoDB omogućuje jednostavnu integraciju sa sustavima za upravljanje enkripcijskim ključevima, koji su važni za sigurno rukovanje navedenim enkripcijskim ključevima. Enkripcija u prijenosu osigurava prijenos podataka između klijenta i poslužitelja.

6.2.2. Upravljanje korisnicima i dozvolama

Pravilno upravljanje korisnicima i dozvolama ključni su dijelovi sigurnosti MongoDB baze podataka. Adekvatno upravljanje korisnicima i njihovim pravima pristupa spriječit će neovlašteni pristup osjetljivim podacima i dopustiti da operacije unutar baze podataka izvode samo ovlaštene osobe. MongoDB podržava robusne mehanizme upravljanja korisnicima i preciznu kontrolu dopuštenja kroz svoj sustav kontrole pristupa temeljen na ulogama.

Upravljanje korisnicima u MongoDB-u odnosi se na stvaranje, modificiranje i brisanje korisničkih računa. Upravlja osobama koje imaju pristup bazi podataka i radnjama koje treba poduzeti. Korisnici koji su stvoreni u MongoDB-u dobivaju jedinstvena korisnička imena, a zatim im se dodjeljuju različite uloge koje definiraju njihove dozvole. Obično kreiranje korisnika vrši administrator baze podataka koji ima pravo upravljanja pristupom korisnika. Administratori izvršavaju naredbe poput `db.createUser()` kako bi stvorili nove korisnike navodeći njihove uloge u sustavu.

Kontrola pristupa temeljena na ulogama sastavni je dio upravljanja dopuštenjima s MongoDB-om. RBAC omogućuje vrlo preciznu kontrolu nad radnjama koje korisnici mogu izvršiti nad bazom podataka. Uloge su unaprijed definirane ili su to korisnički definirani skupovi povlastica koje se mogu dodijeliti korisnicima ili grupama korisnika. MongoDB dolazi s puno ugrađenih uloga, kao što su `read`, `readWrite`, `dbAdmin`, `clusterAdmin` itd., od kojih svaka daje određene privilegije.

Eksplicitne sigurnosne potrebe također se mogu zadovoljiti stvaranjem prilagođenih uloga. Administratori mogu odabrati koje privilegije žele pridružiti određenim zadacima ili aplikacijama prilikom stvaranja prilagođenih uloga. Takve se uloge definiraju pomoću naredbe `db.createRole()` i mogu uključivati radnje na različitim resursima, poput zbirki, baza podataka i funkcija koje upravljaju bazom podataka.

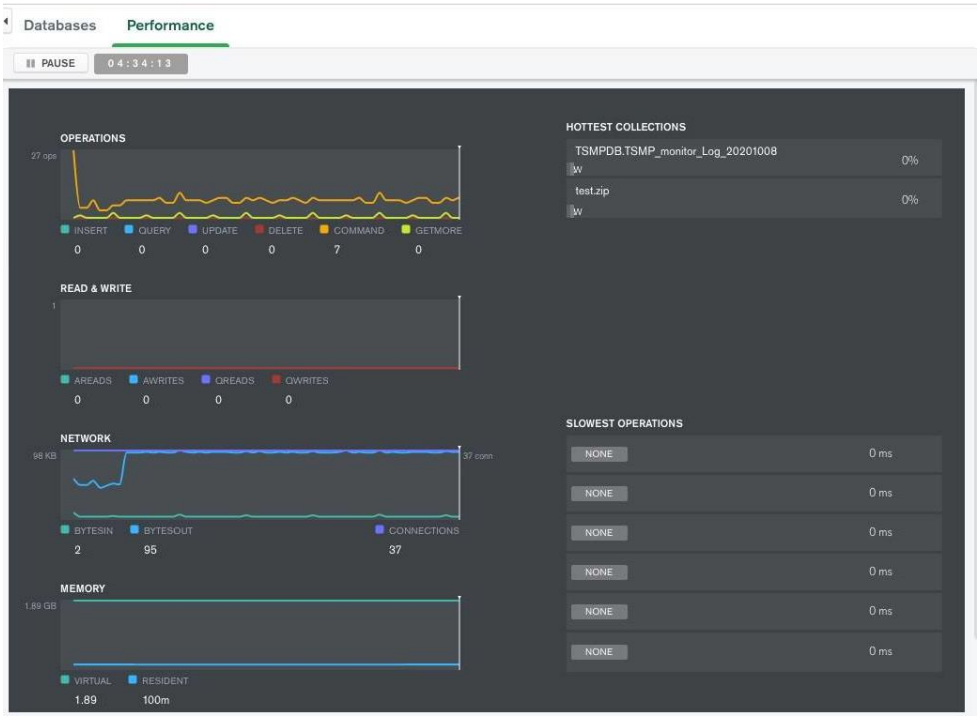
Upravljanje dopuštenjima osigurava da su korisnicima dodijeljene odgovarajuće uloge, tako da korisnici imaju pristup samo onim podacima i funkcijama koje su im potrebne. Ovo načelo najmanje privilegije znači da će neovlašteni pristup podacima ili slučajne izmjene biti minimalni. Dopuštenja se mogu dodati pri stvaranju ili modificiranju korisnika i mogu se ažurirati prema potrebi kasnije.

6.3. Performanse

Sajimon, Benymol [21] navode da su performanse su vrlo vitalan element u MongoDB-u koji određuje učinkovitost i brzinu dohvaćanja i manipulacije podacima. MongoDB je dizajniran da ponudi visoke performanse za velike količine podataka i složene upite kroz značajke kao što su indeksiranje, horizontalno skaliranje i obrada u memoriji. Glavni čimbenici performansi uključuju optimizaciju izvršavanja upita, distribuciju podataka po dijelovima i mehanizme predmemoriranja koji pridonose smanjenju latencije odnosno kašnjenja.

6.3.1. Praćenje performansi baze podataka

Sajimon, Benymol [21] navode da praćenje performansi baze podataka osigurava učinkovitost i pouzdanost. To će uključivati praćenje niza metrika i pokazatelja koji se stalno mijenjaju kako bi se znalo koliko dobro baza podataka radi. Među glavnim čimbenicima koje treba uzeti u obzir pri praćenju performansi su vrijeme izvršenja za upite, upotreba resursa, primjerice, od strane CPU-a, memorije, diska te ulaz/izlaz odziv sustava općenito. Učinkovito praćenje performansi pomaže u ranom otkrivanju problema, čime se poboljšava izvedba za glatko pokretanje baza podataka. Moguće je brzo riješiti probleme, iskoristiti resurse na najbolji mogući način i učiniti iskustvo krajnjeg korisnika optimalnim pomnim praćenjem.



Slika 18. Performanse baze podataka (Izvor: [24])

MongoDB Compass pruža vrlo jake ugrađene alate za praćenje performansi baza podataka. Kombinira sve te značajke u vrlo intuitivno vizualno sučelje. Također daje uvid u stvarnom vremenu u višestruke dimenzije MongoDB operacija, stoga pomaže administratorima s načinima praćenja metrike performansi i traženja uskih grla.

Indeksi su ključ za poboljšanje izvedbe upita, a MongoDB Compass pomaže administratorima da prate koliko se često i koliko učinkovito koriste indeksi. Samo poznavanjem načina na koji se indeksi koriste može se prepoznati koji su korisni, a koji bi mogli biti suvišni ili čak neiskorišteni. Ovo su vrlo vrijedne informacije za postizanje maksimalnih performansi uz minimalne troškove indeksiranja.

Nadalje, MongoDB Compass pruža priliku za praćenje statistike veza, što je neophodno da biste vidjeli opterećenje baze podataka i kako ona upravlja dolaznim zahtjevima. Administratoru omogućuje planiranje skupova veza, osiguravajući da će baza podataka izdržati očekivano radno opterećenje bez degradacije performansi, budući da prikazuje količinu aktivnih i dostupnih veza.

6.3.2. Alati i tehnike za optimizaciju MongoDB performansi

Profiliranje i optimiziranje performansi MongoDB-a ključni su za učinkovit rad baze podataka, zadovoljavanje potreba performansi i učinkovito skaliranje s povećanim

opterećenjima. Korištenje različitih alata i tehnika, kao što su MongoDB Profiler i Explain Plan, omogućuje administratorima da analiziraju izvršenje upita, indeksiranje i korištenje resursa. Omogućavanjem profilira mogu se prikupiti detaljni podaci o operacijama baze podataka poput vremena upita i korištenih indeksa.

Sajimon, Benymol [21] navode da Sharding u MongoDB-u omogućuje horizontalno skaliranje distribucijom podataka na više poslužitelja/klastera za poboljšane performanse čitanja/pisanja. To uključuje odabir ključa fragmenata za ravnomjernu distribuciju podataka i korištenje alata kao što su `sh.status()` i naredbe za balansiranje za praćenje distribucije fragmenata.

Izvor [14] navodi da alati za praćenje kao što su MongoDB Atlas, Prometheus i Grafana pružaju uvid u metriku performansi u stvarnom vremenu, omogućujući proaktivno otkrivanje problema. Tehnike podešavanja performansi kao što su skupljanje veze, optimizacija upita i dizajn sheme također optimiziraju performanse MongoDB-a.

6.4. Optimizacija upita

Chodorow [1] navodi da je optimizacija upita u MongoDB-u neophodna za poboljšanje performansi baze podataka. Učinkovito izvršavanje upita dovodi do bržeg vremena odgovora i poboljšanog korištenja resursa. Metoda `expand()` pruža uvid u izvršavanje upita, pomažući administratorima da identificiraju uska grla i optimiziraju izvedbu upita stvaranjem odgovarajućih indeksa. Dizajn sheme također igra ključnu ulogu u optimizaciji upita, s dobro dizajniranim shemama koje organiziraju podatke za podršku učinkovitim upitima i indeksiranju. Odabir pravog modela podataka, bilo da se radi o ugrađivanju ili referenciranju dokumenata, na temelju uzoraka upita, može značajno utjecati na izvedbu upita. Korištenjem ovih tehnika, administratori mogu osigurati da njihova MongoDB baza podataka radi učinkovito i brzo.

6.5. Složeni upiti

Složeni upiti u MongoDB-u su operacije pretraživanja koje pomažu korisniku da dobije podatke prema mnogim uvjetima i kriterijima. Dok jednostavni upiti mogu tražiti samo jedno polje ili vrijednost, složeni upiti kombiniraju nekoliko logičkih i usporednih operatora, dodatno sužavajući rezultate pretraživanja. Takvi upiti korisniku daju mogućnost filtriranja, sortiranja i analize podataka na sofisticiranije načine i stoga postaju vrlo kritični u rukovanju velikim i složenim skupovima podataka.

Uvjeti u složenim upitima kreirani su u MongoDB-u s pomoću nekoliko operatora. Oni su poznati kao logički operatori, kombinirajući nekoliko uvjeta kako bi se osiguralo točno dohvaćanje podataka. Primjerice, operator $and može se koristiti za pronalaženje dokumenata koji ispunjavaju sve navedene uvjete u isto vrijeme, $or operator dohvaća dokumente koji ispunjavaju bilo koji od nekoliko uvjeta. Podaci, odnosno rezultat, se dodatno sužava pretraživanjem operatorima usporedbe, na primjer, $gt za veće od, $lt za manje od i $eq za jednako.

Agregacijski okvir (eng. Aggregation framework) u MongoDB-u pruža cjevovod za obradu podataka u zbirkama, savršen za slučajeve u kojima korisnici žele obraditi ili analizirati podatke. Svaki stupanj ovog cjevovoda obavlja operaciju na dokumentima, moguće filtriranje, grupiranje ili sortiranje. Omogućuje korisnicima stvaranje složenih upita koji mogu transformirati i sažeti podatke na način na koji to jednostavni upiti ne mogu. Na primjer, može se koristiti agregacijski okvir za izračunavanje prosjeka, generiranje izvješća ili čak preoblikovanje dokumenata u nove formate. Omogućuje fleksibilan i učinkovit način za rješavanje složenih zadataka obrade podataka unutar baze podataka lančanim povezivanjem više faza.

Cjevovod u MongoDB-u vrlo je snažna značajka unutar agregacijskog okvira koji obrađuje podatke kroz niz faza u cjevovodima, transformirajući ih. Svaki stupanj u cjevovodu uzima ulaz, izvodi operaciju na njemu, a zatim prosljeđuje izlaz u sljedeći stupanj, slično pokretnoj traci. Omogućuje složenu manipulaciju podacima, pomažući u filtriranju, grupiranju, sortiranju i preoblikovanju dokumenata unutar zbirke. Ovi su cjevovodi vrlo korisni u generiranju izvješća, a mogu se također koristiti za izračun i generiranje sažetih podataka izravno iz baze podataka bez potrebe za izvozom podataka za vanjsku obradu.

Koristeći agregacijski okvir u sustavu MongoDB Compass mogu se izvršavati složeni upiti te za to postoji nekoliko načina. Za početak ručni unos parametara za filtriranje, grupiranje i pretragu podataka. Slijedi primjer složenog upita koji je ručno upisan u agregacijski okvir:

```
[
  {
    "$lookup": {
      "from": "Student",
      "localField": "id_studenta",
      "foreignField": "_id",
      "as": "student"
```

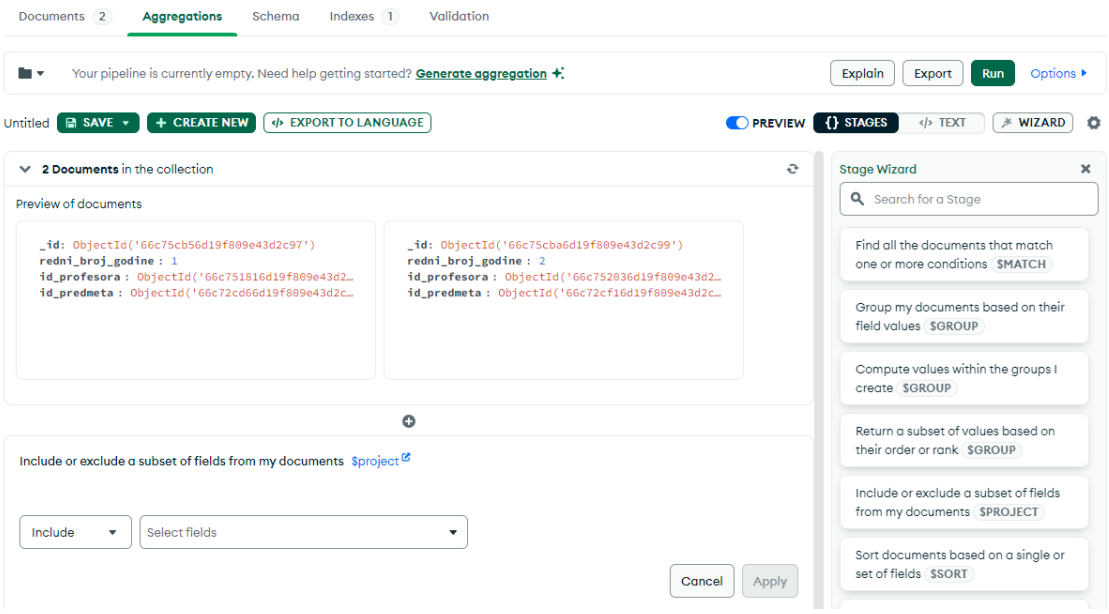
```
      }
    },
    {
      "$lookup": {
        "from": "Predmet",
        "localField": "id_predmeta",
        "foreignField": "_id",
        "as": "predmet"
      }
    },
    {
      "$match": {
        "predmet.naziv": "Baze_podataka_2",
        "ocjena": { "$gte": 4 }
      }
    },
    {
      "$project": {
        "_id": 0,
        "student.ime": 1,
        "student.prezime": 1,
        "predmet.naziv": 1,
        "ocjena": 1
      }
    }
  ]
```

Upit započinje s dvije `$lookup` faze. Prvi `$lookup` pridružuje trenutnu zbirku zbirci `'Student'`, povezujući polje `'id_studenta'` trenutne zbirke s poljem `'_id'` u zbirci Student. Slijedi ugrađivanje studentskih dokumenata kao niz u polje trenutnih dokumenata. Druga faza `$lookupa` je spajanje sa zbirkom `'Predmet'`. Ovaj se cjevovod spaja na polje `'id_predmeta'` u trenutnoj zbirci s poljem `'_id'` u zbirci `'Predmet'`.

Nakon izvršenja pretraživanja, cjevovodna obrada isključuje sve dokumente osim onih u kojima je `predmet_info.naziv` jednak „Baze_podataka_2“, a ocjena je veća ili jednaka 4. Ovo filtriranje osigurava da će se dalje obrađivati samo relevantni zapisi.

Posljednja faza cjevovoda je operacija `$project`. Postavlja polja za ispis, izuzima polje `'_id'` iz rezultata i uključuje `student_info.ime`, `student_info.prezime`, `predmet_info.naziv` i `ocjena`.

Ovaj se složeni upit mogao također mogao realizirati kroz takozvani čarobnjak. Čarobnjak za agregaciju izvorni je alat jednostavan za korištenje u MongoDB Compass-u za pojednostavljenje procesa stvaranja složenih upita za agregaciju. Ima grafičko sučelje putem kojeg se mogu graditi agregacijski cjevovodi u koracima bez potrebe za ručnim pisanjem koda. Lako se mogu dodati različiti stupnjevi, poput `$match`, `$group` i `$sort`, te primijeniti filtre i transformacije na svoje podatke.



Slika 19. Ekran agregacijskog okvira (Izvor: [24])

7. Testiranje i evaluacija

Ingo, Daly [22] i Matallah, Belalem, Bouamrane [23] navode da su testiranje i evaluacija u MongoDB-u važni kako bi se osiguralo da baza podataka služi u svrhu izvedbe, pouzdanosti i skalabilnosti. Testiranje se provodi kako bi se određene situacije iz stvarnog života usporedile s funkcionalnošću i ponašanjem baze podataka pod takvim različitim uvjetima. Testiranje performansi pomoći će odrediti uska grla i optimizirati izvršenje upita, indeksiranje i korištenje resursa. Faza evaluacije prati metrike i analizira te rezultate kako bi se osiguralo da baza podataka radi optimalno i da je spremna podnijeti očekivano opterećenje. Testiranje i evaluacija, kao redovita praksa, sredstva su za zdravu i učinkovitu implementaciju MongoDB-a.

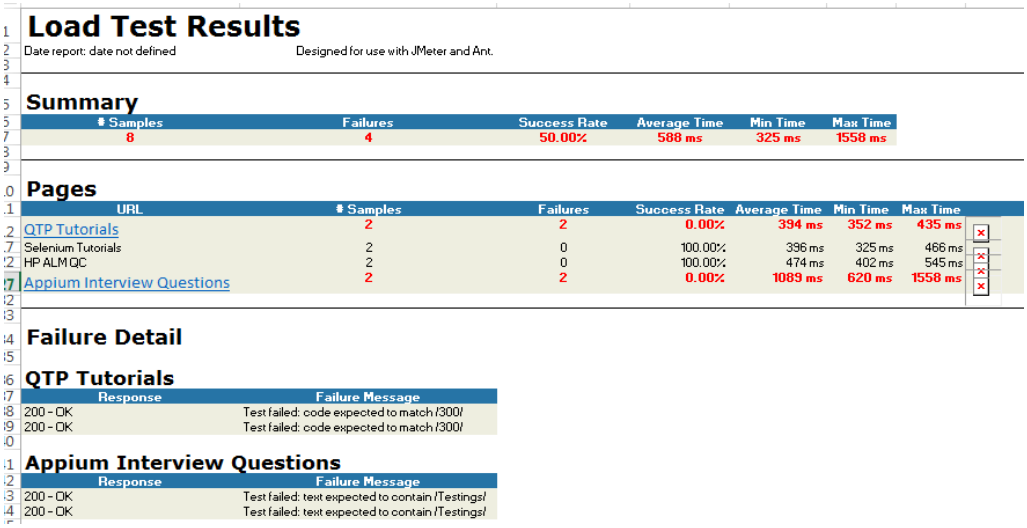
7.1. Testiranje performansi

Testiranje performansi osmišljeno je za provjeru brzine, odziva i stabilnosti sustava u različitim uvjetima i opterećenjima. Redovito testiranje performansi osigurava da aplikacije i baze podataka mogu raditi optimalno i skalirati se s povećanjem potražnje.

7.1.1. Metode testiranje performansi

Jedna od metoda koja se koristi u testiranju baza podataka u ekstremnim uvjetima je stres testiranje. Testiranje uključuje opterećenje sustava izvan njegovih normalnih radnih granica kako bi se otkrile točke prekida ili maksimalni kapacitet. Testiranje otpornosti na stres ima za cilj utvrditi kako baza podataka reagira u takvim uvjetima velike potražnje, kao što su velike količine transakcija ili skokovi u broju korisnika.

Ingo, Daly [22] navode da testiranje opterećenja procjenjuje performanse baze podataka u normalnim očekivanim radnim uvjetima. Time će se testirati izvođenje baze podataka sa simuliranim opterećenjem stvarnih korisnika, čime će se dokazati da može ispravno raditi pod prosječnim razinama prometa i transakcija. Svrha testiranja opterećenja je osigurati da se sustav nosi s očekivanim opterećenjem bez primjetne degradacije performansi. Takvi testovi također olakšavaju identifikaciju uskih grla koja mogu utjecati na korisničko iskustvo i pružaju put za optimizaciju na vrijeme.



Load Test Results

Date report: date not defined — Designed for use with JMeter and Ant.

Summary

Samples	Failures	Success Rate	Average Time	Min Time	Max Time
8	4	50.00%	588 ms	325 ms	1558 ms

Pages

URL	Samples	Failures	Success Rate	Average Time	Min Time	Max Time
QTP Tutorials	2	2	0.00%	394 ms	352 ms	435 ms
Selenium Tutorials	2	0	100.00%	396 ms	325 ms	466 ms
HP ALM QC	2	0	100.00%	474 ms	402 ms	545 ms
Appium Interview Questions	2	2	0.00%	1089 ms	620 ms	1558 ms

Failure Detail

QTP Tutorials

Response	Failure Message
200 - OK	Test failed: code expected to match /300/
200 - OK	Test failed: code expected to match /300/

Appium Interview Questions

Response	Failure Message
200 - OK	Test failed: text expected to contain /Testings/
200 - OK	Test failed: text expected to contain /Testings/

Slika 20. Test opterećenja u alatu Jmeter (Izvor: [21])

Ingo, Daly [22] također navode da se testiranjem propusnosti mjeri količina podataka obrađenih u bazi podataka unutar određenog vremenskog ograničenja. Tehnika mjeri sposobnost sustava s obzirom na transakcije ili operacije koje se mogu izvršiti po jedinici vremena. Testovi propusnosti određuju punu učinkovitost baze podataka i pokazuju gdje se mogu poboljšati kako bi se povećala brzina obrade podataka.

7.1.2. Alati za testiranje performansi

Apache JMeter je open-source, širok, fleksibilan i sveobuhvatan alat za testiranje performansi. Može se koristiti za stvaranje i izvršavanje testova opterećenja kako bi se vidio broj korisnika i transakcija obrađenih u MongoDB bazi podataka. Izvješća o izvedbi generirana iz JMeter-a vrlo su detaljna, mogu točno odrediti vrijeme odziva, propusnost i sva moguća uska grla. Podržava skriptiranje i prilagodbe, tako da je pogodan za složene testne scenarije.

MongoDB Atlas je potpuno upravljana usluga u oblaku koja uključuje praćenje performansi i optimizaciju performansi. Atlas pruža uvid u rad baze podataka u stvarnom vremenu, uključujući performanse upita, korištenje resursa i zdravlje sustava. Nadalje ima automatizirana upozorenja o izvedbi, prijedloge indeksa i preporuke o tome kako optimizirati upite za poboljšanu izvedbu.

7.2. Evaluacija

Matallah, Belalem, Bouamrane [23] navode da evaluacija izvedbe MongoDB-a znači analiziranje kako se baza podataka ponaša pod različitim opterećenjima i traženje uskih grla

kako bi se napravila poboljšanja. To je među kritičnim procesima koji osiguravaju učinkovit rad baze podataka, postizanje svih očekivanih performansi i skaliranje pod rastućim opterećenjima. Procjena performansi MongoDB-a uključuje različite metode i alate koji analiziraju, između ostalog, učinkovitost upita, korištenje resursa i opće stanje sustava.

Matallah, Belalem, Bouamrane [23] također navode da je evaluacije izvedbe upita vrlo kritična za razumijevanje učinkovitosti kojom se podaci dohvaćaju ili manipuliraju. MongoDB pruža neke ugrađene alate, uključujući metodu objašnjenja, za objašnjenje kako se upiti izvršavaju. Metoda objašnjenja objašnjava plan izvršenja upita, uključujući pojedinosti o upotrebi indeksa, broju dokumenata koji su skenirani i vremenu izvršenja. Programeru stoga postaje moguće pratiti upite koji su potencijalno spori ili neučinkoviti i optimizirati ih dodavanjem indeksa ili prepisivanjem upita za bolju izvedbu.

Mjerenje (eng. Benchmarking) je proces za izvođenje standardiziranih testova koji mjere i uspoređuju izvedbu MongoDB-a u odnosu na definirane metrike ili druge sustave baza podataka. Alati i okviri za usporedbu, kao što je Yahoo! Cloud Serving Benchmark, pružaju jednoobrazan način mjerenja propusnosti, latencije i skalabilnosti MongoDB-a. Usporedna analiza izvedbe pomaže u postavljanju osnovnih vrijednosti, ciljeva izvedbe i grafikona napretka tijekom vremena.

7.3. Najbolje prakse

Chodorow [1] navodi da je učinkovit dizajn sheme jedan od najjednostavnijih, ali i najboljih postupaka u MongoDB-u. Za razliku od tradicionalnog dizajna relacijske baze podataka, MongoDB implementira fleksibilne sheme putem kojih se povezani podaci mogu ugraditi u jedan dokument. Time se može značajno poboljšati izvedba čitanja budući da složeni spojevi nisu potrebni. Na primjer, niz komentara može se ugraditi izravno u dokument posta bloga kako bi se ubrzali procesi dohvaćanja podataka. Iako je ugrađivanje podataka korisno na svom mjestu, također treba biti svjestan izbjegavanja pretjeranog ugnježđivanja, jer duboko ugniježđeni dokumenti mogu pogoršati izvedbu. Uravnotežen pristup bio bi ugraditi podatke tamo gdje to ima smisla i koristiti reference za vrlo velike poddokumente kojima se često ne pristupa s glavnim dokumentom.

Još jedno područje gdje najbolje prakse mogu duboko utjecati na performanse MongoDB-a je indeksiranje. Pravilno indeksiranje polja, koja se redovito pretražuju, osigurava brzo i učinkovito pretraživanje. Pažljiva analiza uzoraka upita pomoći će odrediti koje indekse izgraditi. Međutim, ne radi se samo o izgradnji indeksa, već ulaz/izlaz pravilnom upravljanju

njima, budući da ih previše može početi štetiti performansama pisanja. Indekse treba redovito pregledavati i njihovu izvedbu optimizirati kako bi baza podataka bila učinkovita i brza.

Ovo se odnosi na najbolju praksu u upravljanju podacima u MongoDB-u kako bi se osiguralo da će ostati visokoučinkovit i pouzdan. Također uključuje praćenje i upravljanje korištenjem resursa: CPU, memorija i disk ulaz/izlaz. MongoDB Atlas ima ugrađene nadzorne ploče koje pružaju uvid u korištenje resursa u stvarnom vremenu, omogućujući administratorima da proaktivno identificiraju uska grla i skaliraju resurse prema potrebi. Zbog učinkovitog upravljanja resursima, baza podataka može podnijeti dodatno radno opterećenje bez degradacije performansi.

8. Zaključak

Ovaj završni rad daje osvrt o tome kako je implementirati bazu podataka u sustavu MongoDB-u. Prvo je razmotrena identifikacija zahtjeva aplikacije, gdje se ostvaruje dubinsko razumijevanje korisničkih potreba i ciljeva vezanih uz upravljanje podacima. Drugo, raspravljalo se o dizajnu shema podataka, s fokusom na to kako je fleksibilna struktura MongoDB-a korisna za performanse i skalabilnost.

U fazi implementacije praktično je pokazano kako postaviti MongoDB, kreirati baze podataka i raditi CRUD u MongoDB Compass-u. Ovaj praktičan pristup je pokazao koliko je jednostavan za korištenje i koliko MongoDB može biti moćan s različitim tipovima podataka i složenim upitima. Štoviše, pokrivene su neke od najboljih praksi u upravljanju podacima, a to su indeksiranje, za bolju izvedbu, sigurnost, za sprječavanje napada te na posljetku praćenje performansi, kako bi baza podataka bila učinkovita i sigurna kada je pod različitim opterećenjima.

Testiranje i procjena performansi MongoDB baze podataka bio je još jedan bitan dio ove teme. Tehnike testiranja opterećenja i stresa, s alatima za ocjenu performansi, omogućile su otkrivanje mogućih uskih grla, a time i optimizaciju sustava za bolju učinkovitost. Putem takvih procjena moguće je održavati robustan i pouzdan sustav baze podataka koji se prilagođava zahtjevima današnjih aplikacija.

U zaključku ove teme naglašena je važnost MongoDB-a kao fleksibilnog i snažnog NoSQL alata za izradu baze podataka. Sa svojom sposobnošću rukovanja ogromnim količinama nestrukturiranih podataka. Riječ je o fleksibilnom alatu koji se temelji na performansama u ovo doba velikih podataka i analitike u stvarnom vremenu. Dodatno se može raditi na podešavanju implementacija MongoDB-a i proučavanju naprednih značajki kako bi se poboljšale njegove mogućnosti. Ovaj rad doprinosi boljem razumijevanju MongoDB-a u modernom upravljanju podacima i nudi vrijedne uvide programerima i administratorima baza podataka o strukturi, funkcionalnostima te dizajnu baza podataka.

Popis literature

[1] Kristina Chodorow, *SECOND EDITION MongoDB: The Definitive Guide*., O'Reilly Media, Inc., 2013.

[2] Arvind Padmanabhan, "MongoDB" (bez dat.) [Na internetu]. Dostupno: https://devopedia.org/mongodb [Pristupljeno: 21.8.2024.]

[3] Walker Rowe, "*MongoDB Overview: Getting Started with MongoDB*" [Blog], 2017. [Na internetu]. Dostupno: https://www.bmc.com/blogs/mongodb-overview-getting-started-with-mongodb/ [Pristupljeno: 29.7.2024.]

[4] "MongoDB Shell.", (bez dat.) [Na internetu]. Dostupno: https://www.mongodb.com/products/tools/shell [Pristupljeno: 22.7.2024.]

[5] Anjali Chauhan, *A Review on Various Aspects of MongoDB Databases*., International Journal of Engineering Research & Technology (IJERT), 2019.

[6] "Databases & Collections - Documents.", (bez dat.) [Na internetu]. Dostupno: https://www.mongodb.com/docs/manual/core/document/ [Pristupljeno: 22.7.2024.]

[7] "MongoDB Advantages & Disadvantages.", 2021. [Na internetu]. Dostupno: https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/ [Pristupljeno: 22.7.2024.]

[8] Putcha Vaishnavi, "What is MongoDB?" [Blog], 2023. [Na internetu]. Dostupno: https://mindmajix.com/what-is-mongodb [Pristupljeno: 21.8.2024.]

[9] Javier Espinazo Pagán, Jesús Sánchez Cuadrado i Jesús García Molina, *NoSQL Databases*, Stuttgart Media University, 2015.

[10] "Types of NoSQL Databases.", 2023 [Na internetu]. Dostupno: https://www.geeksforgeeks.org/types-of-nosql-databases/ [Pristupljeno: 23.7.2024.]

[11] Craig S. Mullins, "NoSQL (Not Only SQL database).", (bez dat.) [Na internetu]. Dostupno:https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL [Pristupljeno: 23.7.2024.]

[12] Huawei Gaussdb, "DATABASE PRINCIPLES AND TECHNOLOGIES-BASED ON HUAWEI GAUSSDB", Huawei Technologies Co., Ltd., 2021.

[13] Paresh Saraf i Pascal Desmarets, “Optimize Data Modeling and Schema Design with Hackolade and MongoDB.” [Blog], 2021. [Na internetu]. Dostupno: https://www.mongodb.com/blog/post/optimize-data-modeling-and-schema-design-with-hackolade-and-mongodb [Pristupljeno: 24.7.2024.]

[14] “10 Best Database Design Tools to Visualize and Build Data Models in 2024.”, 2024. [Na internetu]. Dostupno: https://clickup.com/blog/database-design-tools/ [Pristupljeno: 24.7.2024.]

[15] S Yadav Ravi, Rishabh Jindal i Santwana Anand, *A Study on Comparison of UML and ER Diagram*, International Research Journal of Engineering and Technology (IRJET), 2020.

[16] “Install MongoDB.”, (bez dat.) [Na internetu]. Dostupno: https://www.mongodb.com/docs/manual/installation/ [Pristupljeno: 26.7.2024.]

[17] “How to Create a Database in MongoDB.”, (bez dat.) [Na internetu]. Dostupno: https://www.mongodb.com/resources/products/fundamentals/create-database [Pristupljeno: 26.7.2024.]

[18] Ciprian-Octavian Truica, Alexandru Boicea i Ionut Trifan, *CRUD Operations in MongoDB*., Atlantis Press, 2013.

[19] Rupali Chopade and Vinod Pachghare, *MongoDB indexing for performance improvement*., Springer Nature, 2020.

[20] Sahib Singh, *Security analysis of mongodb*, Heinz College, Carnegie Mellon University USA, 2019.

[21] Jose Benymol i Abraham Sajimon, “Performance analysis of NoSQL and relational databases with MongoDB and MySQL, 2020. [Na internetu]. Dostupno: https://www.sciencedirect.com/science/article/pii/S2214785320324159 [Pristupljeno: 29.7.2024.]

[22] Henrik Ingo i David Daly, “Automated system performance testing at MongoDB.“, 2020. [Na internetu]. Dostupno: https://arxiv.org/pdf/2004.08425 [Pristupljeno: 29.7.2024.]

[23] Houcine Matallah, Ghalem Belalem i Karim Bouamrane, *Evaluation of nosql databases: Mongodb, cassandra, hbase, redis, couchbase, orientdb*, International Journal of Software Science and Computational Intelligence (IJSSCI), 2020.

[24] MongoDB, Inc., *MongoDB Compass (GUI)* (Verzija: 1.43.4) (2024), Dostupno: `https://www.mongodb.com/try/download/compass` [Preuzeto: 21.7.2024.]

Popis slika