VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

Primerjava MongoDB in PostgreSQL za obdelavo in analizo velikih podatkov

Kandidat: Nejc Drobnič

Vrsta študija: študent izrednega študija

Študijski program: Informatika

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: Petra Munda, inž. inf.

Lektor: Marija Filej, univ. dipl. prof. slo. in ang.

Maribor, 2024

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani/a Nejc Drobnič, sem avtor/ica diplomskega dela z naslovom Primerjava MongoDB in PostgreSQL za obdelavo in analizo velikih podatkov, ki sem ga napisal/a pod mentorstvom mag. Ervina Schaffa.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel/a, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, junij 2024 Podpis študenta:

ZAHVALA

Zahvaljujem se mentorju mag. Ervinu Schaffu za pomoč in mentorstvu pri opravljanju diplomskega dela.

POVZETEK

V zadnjem obdobju smo opazili izjemno rast in razvoj podatkovnih baz, ki postajajo ključni gradniki pri obdelavi obsežnih količin podatkov, znanih kot veliki podatki. Veliki podatki ne le vsebujejo ogromne količine informacij, temveč pogosto vključujejo tudi raznolike vrste podatkov, kot so tekst, slike, zvok, video posnetki in številni drugi formati. Zaradi te raznolikosti in obsega je ključno imeti ustrezne podatkovne baze, ki omogočajo učinkovito shranjevanje, upravljanje in analizo teh podatkov. MongoDB in PostgreSQL sta med vodilnimi podatkovnimi bazami, ki se uporabljajo za reševanje izzivov velikih podatkov, vsaka s svojimi edinstvenimi lastnostmi, prednostmi in omejitvami.

To diplomsko delo se osredotoča na primerjalno analizo podatkovnih baz MongoDB in PostgreSQL, ter njihovo uspešnost in učinkovitost pri obdelavi in analizi velikih podatkov. MongoDB, kot predstavnik ne-relacijskih podatkovnih baz, se izkaže kot zelo uporabno orodje v ekosistemu velikih podatkov. Zmožnost shranjevanja nestrukturiranih podatkov in enostavno horizontalno skaliranje postavljata MongoDB med priljubljene izbire organizacij, ki se spoprijemajo z raznolikimi in hitro rastočimi podatki. Na drugi strani je PostgreSQL, klasična relacijska podatkovna baza, ki se je s prilagodljivostjo in močjo SQL jezika izkazala kot zanesljiva rešitev za kompleksne analize podatkov. Njena integracija s sistemom za upravljanje transakcij (ACID) omogoča doslednost in zanesljivost v kritičnih okoljih.

V okviru te raziskave bomo izvedli štiri ključne meritve. Prva meritev nam bo podala rezultate hitrosti vstavljanja dokumentov in poizvedb na sekundo. Druga bo osredotočena na hitrost vstavljanja podatkov v obe bazi. S to meritvijo bomo ugotovili njihovo zmogljivost pri obvladovanju večjih količin podatkov. Tretja meritev se bo posvetila učinkovitosti transakcijskega vstavljanja s poudarkom na zagotavljanju doslednosti in integritete podatkov. Četrta in zadnja meritev bo preučila obseg in razširljivost ter porabo pomnilnika pri vstavljanju v podatkovno bazo. Z našo raziskavo želimo jasno predstaviti, kje so močne točke, omejitve in prednosti MongoDB ter PostgreSQL pri obdelavi in analizi velikih podatkov. Cilj je pridobiti vpogled v njihove zmožnosti, omejitve in kako se obnesejo pri različnih scenarijih obdelave podatkov.

Ključne besede: Big Data, MongoDB, PostgreSQL, podatkovne baze, obdelava velikih podatkov.

ABSTRACT

Comparison of MongoDB and PostgreSQL for processing and analyzing Big Data

In recent times, we have witnessed tremendous growth and development in databases, which have become crucial building blocks in processing vast amounts of data, known as big data. Big data not only encompasses enormous volumes of information but often includes diverse types of data, such as text, images, sound, video recordings, and numerous other formats. Due to this diversity and scale, it is crucial to have appropriate databases that enable efficient storage, management, and analysis of such data. MongoDB and PostgreSQL are among the leading databases used to tackle the challenges of big data, each with its unique features, advantages, and limitations.

This thesis focuses on the comparative analysis of MongoDB and PostgreSQL databases, and their performance and efficiency in processing and analyzing big data. MongoDB, as a representative of non-relational databases, proves to be a highly useful tool in the ecosystem of big data. Its ability to store unstructured data and easily scale horizontally places MongoDB among the popular choices for organizations dealing with diverse and rapidly growing data. On the other hand, PostgreSQL, a classic relational database, has demonstrated itself as a reliable solution for complex data analysis, with its flexibility and the power of the SQL language. Its integration with the transaction management system (ACID) ensures consistency and reliability in critical environments.

Within the framework of this research, we will conduct four key measurements. The first measurement will provide us with results on the speed of document insertion and queries per second. The second will focus on the speed of data insertion in both databases, allowing us to determine their capacity in handling large volumes of data. The third measurement will address the efficiency of transactional insertion, with an emphasis on ensuring data consistency and integrity. The fourth and final measurement will examine the scope, scalability, and memory consumption during insertion into the database. With our research, we aim to clearly present the strengths, limitations, and advantages of MongoDB and PostgreSQL in processing and analyzing big data.

Keywords: Big Data, MongoDB, PostgreSQL, databases, data processing.

Kazalo vsebine

KAZALO SLIK

KAZALO TABEL

Razlaga strokovnih izrazov in kratic

- Veliki podatki (ang. Big Data): se nanaša na velike količine podatkov, ki presegajo zmogljivosti tradicionalnih podatkovnih baz in orodij za obdelavo podatkov.
- Strukturirani podatki (ang. Structured Data): so podatki, ki so organizirani in shranjeni v formatu, kar omogoča enostavno iskanje, razumevanje in analizo.
- Nestrukturirani podatki (ang. Unstructured Data): so podatki, ki nimajo jasne predhodno določene strukture ali oblike. Ti podatki so pogosto v obliki besedil, slik, zvokov ali videoposnetkov.
- Podatkovna baza (ang. Database): gre za organizirano zbirko podatkov, ki jih je mogoče shranjevati, urejati, iskati in upravljati na sistematičen način.
- Poizvedba (ang. Query): je niz ukazov ali poizvedb, ki jih izvajamo nad podatkovno bazo.
- Strukturirani poizvedovalni jezik SQL (ang. Structured Query Language): je jezik za upravljanje podatkovnih baz, ki omogoča izvajanje poizvedb in manipulacijo podatkov.
- NoSQL (ang. Not Only SQL): je pojem, ki označuje skupino podatkovnih baz, ki ne sledijo tradicionalnemu relacijskemu modelu z uporabo jezika SQL.
- Byte (ang. Byte): osnovna enota za digitalne podatke.
- Megabyte – MB (ang. Megabyte): enota za merjenje količine podatkov, ki je približno milijon bajtov.
- Gigabyte – GB (ang. Gigabyte): enota za merjenje količine podatkov, ki je približno milijardo bajtov ali 1000 megabajtov.
- Terabyte – TB (ang. Terabyte): enota za merjenje količine podatkov, ki je približno bilijon bajtov ali 1000 gigabajtov.
- Python (ang. Python): programski jezik, v katerem so bile napisane skripte.

UVOD

1.1 Opis področja in opredelitev problema

Raziskava v diplomskem delu se osredotoča na primerjavo in analizo uporabe MongoDB in PostgreSQL v okviru obdelave velikih podatkov (Big Data). Osredotoča se na učinkovitost shranjevanja, vstavljanja in poizvedb v obeh sistemih ter primerja njune zmogljivosti. Prav tako raziskuje primerljivost pristopa k obdelavi in organizaciji podatkov v kontekstu Big Data ter analizira, katera platforma je bolj primerna za specifične zahteve.

1.2 Namen, cilji in osnovne trditve

Namen raziskave je proučiti glavne razlike med MongoDB in Postgre SQL, ter analizirati njuno uporabo v kontekstu obdelave velikih količin podatkov. Cilj raziskave je identificirati najbolj priljubljene podatkovne baze za obdelavo Big Data.

V raziskovalnem delu smo iskali odgovore na naslednja vprašanja:

- Kakšne so glavne razlike med MongoDB in Postgre SQL?
- Kje in zakaj se pogosto uporablja MongoDB v primerjavi s Postgre SQL?
- Kako se MongoDB odnese pri obdelavi velikih količin podatkov v primerjavi s Postgre SQL?
- Katere baze so najbolj popularne za obdelovanje Big Data?

V raziskovalnem delu smo dokazali naslednje hipoteze:

- **Hipoteza 1:** MongoDB bo ponudil učinkovitejšo obdelavo velikih količin podatkov v primerjavi s PostgreSQL.
- **Hipoteza 2:** MongoDB se pogosto uporablja v primerjavi s Postgre SQL v določenih okoljih, kot so aplikacije, ki zahtevajo visoko skalabilnost in hiter razvoj.
- **Hipoteza 3:** Pri obdelavi velikih količin podatkov bo MongoDB ponudil boljše rezultate v primerjavi s Postgre SQL glede na hitrost in učinkovitost.
- **Hipoteza 4:** Najbolj priljubljene baze za obdelavo Big Data bodo tiste, ki omogočajo visoko zmogljivost, skaliranje in podporo za analizo nestrukturiranih podatkov.

1.3 Predpostavke in omejitve

Raziskava se osredotoča na analizo in primerjavo relacijske in ne relacijske podatkovne baze, MongoDB in PostgreSQL, ter njuno učinkovitost, enostavnost in delovanje pri obdelavi velikih podatkov.

Predpostavke:

- Podatki so s skripto generirani naključno za obe podatkovni bazi, kar nam zagotavlja nepristranski pristop pri ocenjevanju njihovih zmogljivosti. To pomeni, da se zanašamo na to, da bodo rezultati, pridobljeni iz naših meritev, ponazarjali splošno zmogljivost obeh podatkovnih baz v pogostih scenarijih obdelave podatkov.
- Osredotočili se bomo na osnovne lastnosti in učinkovitost MongoDB in PostgreSQL. Predpostavljali bomo, da so meritve, ki jih izvajamo, značilne za običajne primere obdelave podatkov.

Omejtive:

- Koncept velikih podatkov je izjemno širok, zato smo se omejili na tri ključne meritve, saj je koncept velikih podatkov zelo obsežen. Čeprav bi lahko izvajali dodatne meritve za širši vpogled, so te omejene zaradi omejenih virov in časovnih omejitev.
- Meritve bodo izvedene le na dveh podatkovnih bazah: MongoDB in PostgreSQL. Obstajajo še druge podatkovne baze, ki bi lahko bile relevantne v kontekstu obdelave velikih podatkov.
- Omejitev zmogljivosti računalnika je dejavnik, ki lahko vpliva na meritve, saj obdelava velikih podatkov zahteva znatno procesno moč.
- Izvajanje meritve je bilo izvedeno s pomočjo programskega jezika Python in uporabe ustreznih knjižnic, v primeru uporabe bolj ali manj optimiziranega programskega jezika bi lahko bili rezultati drugačni.
- Meritve rezultatov so tudi rezultat prej definiranega števila transakcij in števila poizvedb oziroma dokumentov, kar v pravem življenju ni vedno tako. Zato je priporočljivo, da se meritve izvajajo in primerjajo v kontekstu specifičnega projekta in okolja, ki ga želimo oceniti.

1.4 Uporabljene raziskovalne metode

Za pisanje diplomskega dela in izvedbe meritev ter primerjav sem si pomagal z literaturo strokovnih knjig in internetnih dokumentacij. Področje raziskovanja je primerjava MongoDB in PostgreSQL podatkovnih baz in meritve, kako učinkoviti sta, kadar se uporablja velika količina podatkov.

- **Pregled literature:** Pri izdelavi tega diplomskega dela sem opravil temeljit pregled literature, ki zajema strokovne knjige in dostopno internetno dokumentacijo obeh podatkovnih baz. Literarni pregled mi je omogočil pridobitev globljega razumevanja značilnosti, prednosti in morebitnih omejitev MongoDB in PostgreSQL v kontekstu obdelave velikih podatkov.
- **Izvajanje meritev:** Za pridobitev konkretnih podatkov sem izvedel natančne meritve, ki so se osredotočale na tri ključna področja. Uporabil sem preverjene metode in orodja, ter strogo sledil načrtu, s čimer sem zagotovil zanesljive in pomembne rezultate. Meritve so bile izpeljane z namenom ocenjevanja dejanske učinkovitosti obeh podatkovnih baz pri obvladovanju obsežnih podatkovnih nalog v resničnem okolju.
- **Analiza rezultatov:** V sklopu analize rezultatov sem podrobno preučil pridobljene podatke iz meritev. Uporabil sem prikaz grafov za vizualizacijo rezultatov, da bi identificiral prednosti in omejitve obeh podatkovnih baz. Analiza rezultatov je ključna za razumevanje, kako se MongoDB in PostgreSQL obnašata v specifičnih pogojih obdelave velikih podatkov.

Kombinacija literarnega pregleda, meritev in analize rezultatov omogoča celovito primerjavo med MongoDB in PostgreSQL ter podajanje smernic za njuno optimalno uporabo v praksi. Skozi te raziskovalne metode lahko prispevamo k boljšemu razumevanju in uporabi podatkovnih baz v kontekstu obdelave velikih podatkov, kar pripomore k izboljšanju učinkovitosti in zanesljivosti informacijskih sistemov v organizacijah.

2 PREGLED LITERATURE

2.1 Definicija podatkovnih baz in Big Data

Uporaba podatkovnih baz je razširjena v sodobnih storitvah in aplikacijah, ne glede na to, ali so podatki shranjeni v aplikaciji na mobilnem telefonu, računalniku ali na internetu. Delujoča baza podatkov učinkovito shrani in organizira potrebno količino podatkov za nemoteno delovanje ter omogoča enostaven dostop uporabnikom. V povprečni aplikaciji so shranjeni podatki o uporabnikih, izdelkih, poslovnih podatkih ali storitvah ter odnosih med njimi. V raziskavi bomo spoznali klasično relacijsko podatkovno bazo in NoSQL podatkovne baze, bolj podrobno tudi dokumentno orientirano podatkovno bazo (IBM, brez datuma).

2.1.1 Relacijske podatkovne baze

Relacijske podatkovne baze so sistem, razvit v 1970-ih, namenjen učinkovitemu izvajanju osnovnih operacij nad podatki, znanih kot operacije CRUD (ustvarjanje, branje, posodabljanje in brisanje podatkov). Uporabljajo strukturirani poizvedovalni jezik (ang. Structured Query Language), kateri omogoča interakcijo in izvajanje poizvedb nad podatki v bazi, kot so iskanje, dodajanje, posodabljanje in brisanje podatkov (Oracle, brez datuma).

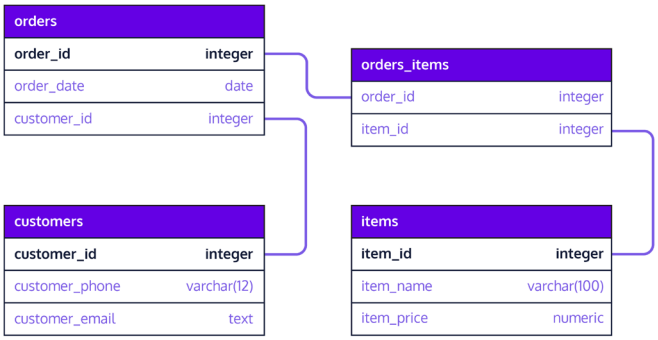
S pomočjo SQL jezika je mogoče tudi oblikovanje kompleksnih poizvedb za pridobivanje specifičnih informacij iz relacijskih baz. Ta jezik omogoča tudi vzpostavljanje povezav med različnimi tabelami preko tujih ključev, kar olajša združevanje podatkov iz več tabel pri večjih programih ali aplikacijah. Omogoča tudi ustvarjanje novih tabel in brisanje starih tabel, kar omogoča prilagajanje strukture baze glede na spremenjene potrebe in zahteve aplikacije.

Relacijske podatkovne baze delujejo v skladu s principi ACID, kar pomeni, da so zasnovane tako, da zagotavljajo zanesljivost in doslednost transakcij. To zagotavlja, da so spremembe podatkov varne in dosledne, kar je ključno za ohranjanje integritete podatkov v teh bazah. Glavni ACID Principi:

- **Atomsko (Atomicity):** Vse ali nič. Ali se izvede celotna transakcija, ali pa sploh ne sme biti izvedena.

- **Konsistenčnost (Consistency):** Po koncu transakcije mora baza podatkov ostati v doslednem stanju brez neveljavnih podatkov. To zagotavlja, da so vsi podatki v skladu z določenimi pravili in omejitvami.
- **Izoliranost (Isolation):** Izvedba ene transakcije ne sme vplivati na izvedbo druge, dokler nista obe končani. To preprečuje, da bi se med transakcijami pojavljali neskladni podatki.
- **Trajnost (Durability):** Po zaključku transakcije morajo biti podatki trajno shranjeni, ne glede na morebitne izpade sistema. To zagotavlja, da se podatki ohranijo tudi v primeru nepričakovanih dogodkov ali izpadov sistema (Chapple, 2023).

Vsaka tabela v relacijskih podatkovnih bazah vsebuje svoj edinstven ključ, ki identificira posamezen zapis. Na primer, v tabeli uporabnik lahko imamo dva uporabnika z enakim imenom, v tem primeru se uporabi edinstven ključ, da se podatke loči in se lahko najde pravi. Tabele se lahko med sabo tudi povezujejo, temu pravimo relacije.



Slika 1: Shema relacijske podatkovne baze

(Vir: https://www.codecademy.com/learn/fscp-designing-relational-databases/modules/fscp-designing-a-database/cheatsheet)

Relacijske baze imajo tudi nekaj omejitev, na te omejitve so kasneje kot rešitev prišle NoSQL podatkovne baze. Nekatere od teh omejitev vključujejo:

- **Vertikalno skaliranje:** Tradicionalne baze podatkov pogosto zahtevajo vertikalno skaliranje, kar pomeni, da je potrebno nadgraditi strojno opremo z višjo zmogljivostjo. To pa lahko hitro postane draga in težko vzdržljiva praksa.
- **Fiksna shema:** Relacijske baze običajno zahtevajo fiksno shemo, kar pomeni, da je potrebno predhodno definirati strukturo tabel in relacij. To postane težavno pri obvladovanju raznolikih in hitrih sprememb v aplikacijah ali procesih.

- **Zapletene JOIN operacije:** Pri uporabi relacijskih baz se kompleksnost poveča pri izvajanju JOIN operacij med več povezanimi tabelami. To lahko vodi do počasnih poizvedb, še posebej, ko imamo veliko število zapletenih povezav.

2.1.2 NoSQL podatkovne baze

Nove generacije podatkovnih baz, imenovane tudi baze NoSQL (ang. Not Only SQL), so visoko razširljive, prilagodljive in z nizko latenco. Podatkovne baze z nizko latenco so ključne za sodobne aplikacije, saj omogočajo hitro obdelavo zahtev in hitro vračanje rezultatov. Ta vrsta baz podatkov nudi rešitev na omejitve, s katerimi se soočajo tradicionalne baze podatkov pri obvladovanju obsežnih, raznolikih in hitro generiranih podatkov.

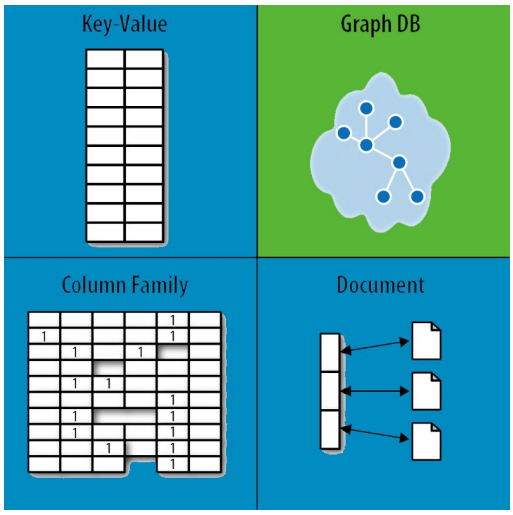
NoSQL podatkovno modeliranje ima bolj preprost pristop, kjer se osredotočamo na to, kako aplikacija potrebuje in uporablja podatke, ne toliko na medsebojne povezave znotraj baze in tabel, kot pri tradicionalnih relacijskih bazah. Glavna načela NoSQL oblikovanja poudarjajo prilagodljivost podatkov za razliko od strogo določenih relacijskih shem. V NoSQL je dovoljeno podvojiti ali razširiti podatke, saj ni potrebe po zapletenih operacijah združevanja (JOIN) med tabelami (MongoDB, brez datuma).

Pri relacijskih podatkovnih bazah smo se seznanili z akronimom ACID. Nasprotno pa imamo pri NoSQL podatkovnih bazah akronim BASE. Ta model omogoča prilagodljivost, ki jo NoSQL in podobni pristopi ponujajo pri upravljanju in urejanju nestrukturiranih podatkov. BASE sestavljajo tri načela:

- **Osnovna razpoložljivost (Basic Availability):** Pristop se osredotoča na razpoložljivost podatkov. Baza podatkov ostane na voljo tudi v primeru napak in sicer s širjenjem podatkov na več mest.
- **Mehko stanje (Soft State):** Osnovna ideja za BASE je, da je doslednost podatkov skrb razvijalca in je ne bi smele upravljati podatkovne baze.
- **Končna doslednost (Eventual Consistency):** Podatki v bazi so vedno v pričakovanem stanju ali obliki. Ko se izvede določena operacija ali transakcija na podatkih, se pričakuje, da bodo vsi uporabniki sistema videli enake, konsistentne podatke.

NoSQL baze zahtevajo drugačen pristop k strukturiranju podatkov kot tisti, ki se uporablja pri tradicionalnih relacijskih sistemih za upravljanje baz podatkov (RDBMS). Spodaj je na kratko opisanih nekaj glavnih načinov, kako se lahko strukturirajo podatki:

- **Ključ-vrednost** (Key-value database): To je najbolj prilagodljiva vrsta NoSQL baze podatkov, razdeljena je na ključ in vrednost. Informacije se pridobivajo na podlagi edinstvenega ključa.
- **Dokumentne podatkovne baze** (Document oriented database): Poznane tudi kot dokumentno orientirane baze podatkov. Podatke shranjujejo v formatih XML, JSON ali BSON.
- **Grafične podatkovne baze** (Graph database): Ta baza podatkov organizira podatke kot vozlišča in odnose, ki prikazujejo povezave med vozlišči. To omogoča bogatejšo in celovitejšo predstavitev podatkov. Grafične baze podatkov se uporabljajo v družbenih omrežjih, rezervacijskih sistemih in za odkrivanje goljufij.
- **Široki-stolpec** (Wide-column database): Podatke shranjujejo in upravljajo v obliki tabel, vrstic in stolpcev podobno kot relacijske baze. Široko so uporabljene v aplikacijah, ki zahtevajo stolpčno obliko za zajemanje sheme neodvisnih podatkov (Scylladb, brez datuma).



Slika 2: Vizualni prikaz tipov NoSQL podatkovnih baz
(Vir: https://cdn-media-1.freecodecamp.org/images/1*k7VI_3bUow1CvXHxSBKaww.jpeg)

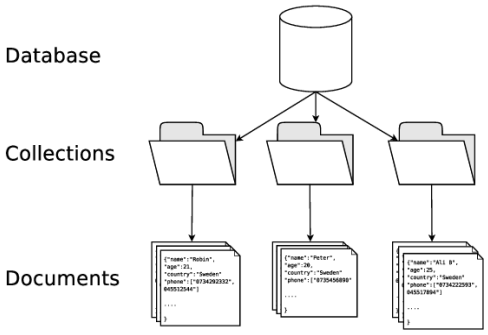
2.1.3 Dokumentno orientirana podatkovna baza

Podrobneje se bomo poglobili v dokumentno orientirane baze, s katerimi bomo upravljali v MongoDB platformi. Dokumentno orientirana podatkovna baza je vrsta NoSQL podatkovne baze, ki podatke shranjuje v obliki dokumentov, namesto v tabelah s stolpci in vrsticami, kot je značilno za tradicionalne relacijske baze podatkov. Ti dokumenti lahko imajo različne oblike, kot so JSON (JavaScript Object Notation), BSON (Binary JSON Object Notation) ali XML (Extensible Markup Language). Velika prednost dokumentno orientiranih podatkovnih baz je enostavnost uporabe. Razvijalci najdejo delo s podatki v dokumentih preprostejše in bolj intuitivno oziroma bolj naravno razumljivo, kot delo s tabelami. Dokumenti se odlično prilagajajo programskim jezikom, kar zmanjšuje potrebo po zapletenih manipulacijah podatkov in izboljšuje produktivnost razvijalcev.

Struktura dokumentno orientiranih baz temelji na zbirkah ali kolekcijah, pri čemer vsaka kolekcija vsebuje različne dokumente, predstavljajoče raznolike informacije ali vsebine. Gibljivost modela dokumenta omogoča shranjevanje raznolikih podatkov brez potrebe po fiksni shemi. V okviru naše raziskave uporabljamo orodje MongoDB, kjer so dokumenti zapisani v formatu JSON (Amazon Web Services, brez datuma).

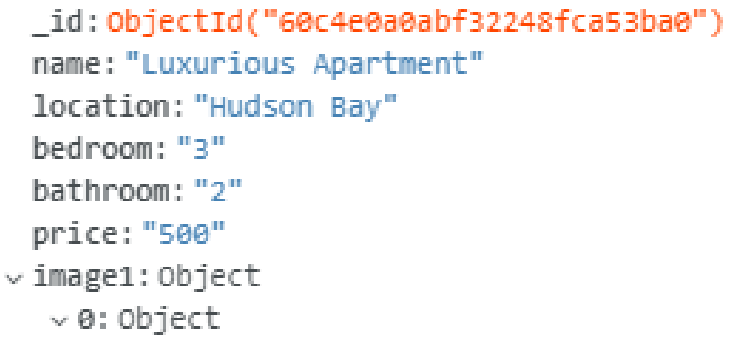
Vsak dokument ima svoj unikaten ključ, običajno označen kot "id". Ta ključ omogoča enolično identifikacijo vsakega dokumenta v zbirki, pri čemer ga lahko dodeli sistem za upravljanje baz podatkov ali aplikacija.
Osnovna zgradba dokumentno orientiranih podatkovnih baz:

- **Dokumenti:** To so zapisi znotraj dokumentne podatkovne baze, ki vsak vsebuje informacije o enem objektu in njegovih povezanih metapodatkih.
- **Zbirke:** Dokumenti so združeni v zbirke, ki lahko shranjujejo različne vsebine. Zbirke izkoriščajo prilagodljivo shemo, kar pomeni, da vsi dokumenti v zbirki niso nujno opremljeni s istimi polji.



Slika 3: Struktura dokumentno orientirane baze

(Vir: https://www.semanticscholar.org/paper/Document-Oriented-NoSQL-Databases-Gustafsson/1ee2fe97c41f5b94620b1e0558aff99629228db3)



Slika 4: JSON zgradba podatkovne baze

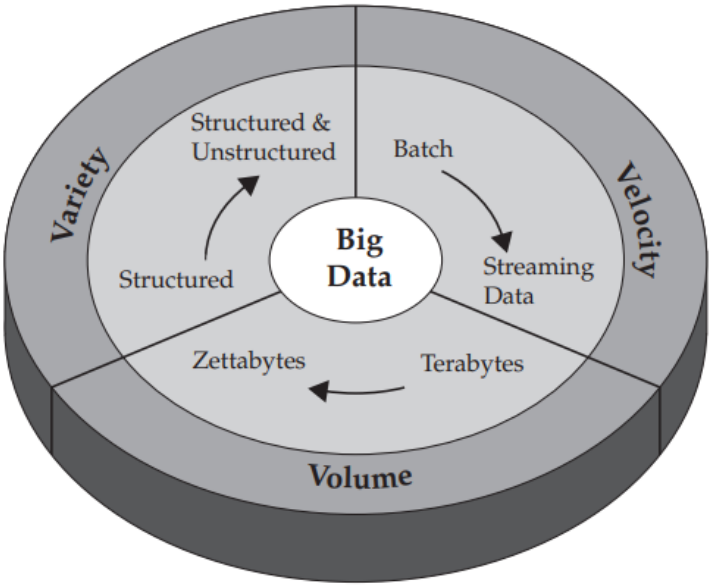
(Vir: https://www.mongodb.com/community/forums/t/example-code-for-mongodb-http-webhook-function-for-uploading-files/111014)

2.1.4 Veliki podatki

Veliki podatki so trenutno najbolj pereča tema v informacijskem svetu, mnogi pa verjamejo, da bodo dejansko preoblikovali svet. Njihov vpliv lahko celo preseže pomen interneta, saj močno vplivajo na vsakodnevno življenje posameznikov. Gre za sposobnost zajemanja in analiziranja ogromnih količin podatkov, kar odpira nove priložnosti in izzive na področju tehnologije, raziskav in družbe kot celote.

Izraz veliki podatki se nanaša na informacije, ki jih ni mogoče obdelati ali analizirati z uporabo tradicionalnih postopkov ali orodij. Vedno več organizacij se danes sooča z vse večjimi izzivi velikih podatkov. Imajo dostop do bogastva informacij, vendar ne vedo, kako bi iz njih pridobili vrednost, ker so v najbolj surovi obliki ali v pol-strukturirani ali nestrukturirani obliki.

Podjetja se soočajo s temi izzivi v okolju, kjer imajo sposobnost in kapacitete shranjevanja česarkoli, ter si ustvarjajo podatke kot nikoli prej v zgodovini. Skupaj to predstavlja resničen izziv informacij. Gre za zagate: današnje podjetje ima več dostopa do potencialnih vpogledov kot kadar koli prej, vendar se ta potencialna zlata jama podatkov kopiči (Chris Eaton, 2012).



Slika 5: Prikaz modela »3 V« kot ga je karakteriziral IBM.
(Vir: https://www.immagic.com/eLibrary/ARCHIVES/EBOOKS/I111025E.pdf)

Velike podatke določajo tri lastnosti: obseg, raznolikost in hitrost (kot je prikazano na Sliki 4). Skupaj te lastnosti določajo tisto, kar imenujemo "Veliki podatki". Ustvarile so potrebo po novem razredu sposobnosti, ki dopolnjujejo način, kako se danes izvajajo stvari, da zagotovijo boljši pregled in nadzor nad našimi področji in informacijami (Chris Eaton, 2012).

- **Volumen (ang. Volume):** Količina podatkov je pomembna. Pri velikih podatkih morate obdelovati visoke količine nizke gostote nestrukturiranih podatkov. To lahko vključuje podatke neznanega pomena, kot so podatki iz virov Twitterja, sledenje, kliki na spletni strani, mobilni aplikaciji ali senzorsko omogočena oprema. Za nekatere organizacije lahko to pomeni desetine terabajtov podatkov. Za druge pa morda stotine petabajtov.
- **Hitrost (ang. Velocity):** Hitrost, s katero se podatki prejemajo in (morda) obdelujejo. Običajno z najvišjo hitrostjo podatki neposredno pritekajo v pomnilnik, namesto da bi se pisali na disk. Nekateri izdelki, povezani z internetom, delujejo v realnem času ali skoraj v realnem času in zahtevajo takojšnje ocenjevanje in ukrepanje.

Raznolikost (ang. Variety): Raznolikost se nanaša na različne vrste podatkov, ki so na voljo. Tradicionalni tipi podatkov so bili strukturirani in so se lepo prilegali v relacijsko bazo podatkov. S pojavom velikih podatkov prihajajo nove nestrukturirane vrste podatkov. Nestrukturirani in delno strukturirani tipi podatkov, kot so besedilo, zvok in video, zahtevajo dodatno predobdelavo za izluščenje pomena in podporo metapodatkom (Taylor, 2023).

Tukaj je le nekaj primerov, kako so industrije uporabile velepodatke, da so presegale zgolj merjenje in štetje ter se sposobno posluževale napovedovanja in razumevanja. V današnjem času je vredno omeniti tudi ChatGPT, kateri je mogoč zaradi uporabe velikih podatkov.

V svojem bistvu ChatGPT temelji na besedilih, da gradi jezikovne modele za pisanje odgovorov. Z analizo velike količine besedil jezikovni model ugotovi, katere so pogoste fraze in katera besedila najbolje sledijo posameznim frazam. Nato uporabi to znanje, da generira najboljšo naslednjo besedo glede na vse besede v dialogu. Ko je beseda generirana, se postopek ponovi ob upoštevanju novega generiranega besedila. Tako ChatGPT piše odgovore, besedo za besedo (Lee, 2023).

Primeri:

- **Poslovna analitika:** Podjetjem omogočajo, da izboljšajo analitiko, razumejo vzorce v potrošniškem vedenju, napovedujejo trende in optimizirajo poslovne procese.
- **Zdravstvena industrija:** Analitika velikih podatkov nudi prednosti pri izboljšanju diagnoze izidov pacientov, zmanjšanju stroškov in povečanju operativne učinkovitosti. Uporablja se lahko za različne primere, kot so nadzor nad boleznimi, podpora kliničnim odločitvam, personalizirana medicina in odkrivanje zdravil.
- **Uporabniška izkušnja in marketing:** Izboljšujejo uporabniško izkušnjo in personalizirani marketing z zagotavljanjem vpogledov v vedenje, preference in potrebe strank. To omogoča podjetjem prilagajanje marketinških strategij in ponudb izdelkov, kar vodi v večje zadovoljstvo in zvestobo strank.
- **Kibernetska varnost:** Veliki podatki igrajo ključno vlogo pri kibernetski varnosti z omogočanjem zaznavanja in preprečevanja kibernetskih groženj v realnem času. Uporablja se za analizo prometa v omrežju, zaznavanje anomalij in obveščanje o grožnjah, kar organizacijam omogoča hitro in učinkovito odzivanje na grožnje.
- **Prepoznavanje besedila in slik:** Veliki podatki se lahko uporabijo pri prepoznavanju, kjer napredni algoritmi strojnega učenja samodejno izvlečejo besedilo ali prepoznajo slike v fotografijah. To ima več potencialnih uporabnih primerov v različnih industrijah, na primer v trgovinski industriji, kjer se lahko uporablja za identifikacijo izdelkov, ali v zdravstveni industriji, kjer se lahko uporablja za prepoznavanje bolezni ali nepravilnosti na medicinskih slikah.

  Pametni klepetalni roboti: Revolucionirajo storitve za stranke v sodobnih podjetjih. Podjetjem ponujajo stroškovno učinkovit način izboljšanja izkušenj in zadovoljstva strank (Data Forest, 2023).

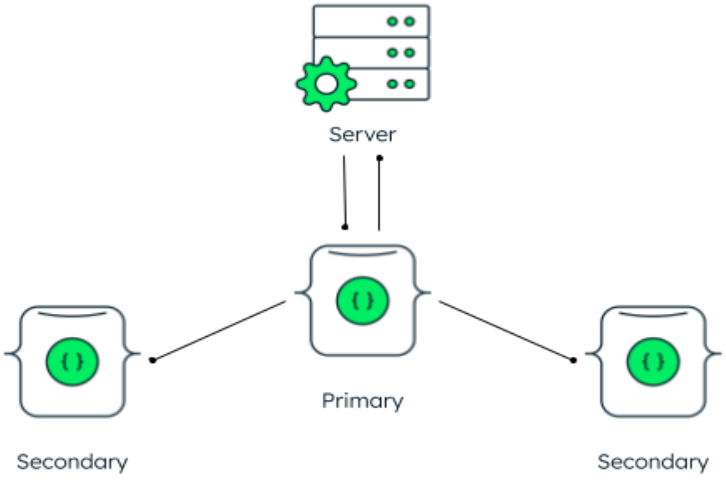
Pri velepodatkih gre manj za individualne kose podatkov in bolj za izluščevanje vrednosti in pomena iz njih. Danes podatki prihajajo iz več virov, v več formatih in hitreje kot kadarkoli prej. S pravilno vzpostavljeno platformo za podatke lahko organizacije izluščijo vrednost iz svojih podatkov, kar jim omogoča hitrejše odločanje na podlagi podatkov.

2.2 MongoDB

MongoDB je odprtokodni sistem za upravljanje z NoSQL podatkovnimi bazami. Zasnovan je za shranjevanje in upravljanje velikih količin podatkov na prilagodljiv in razširljiv način. Znan je po svoji zmožnosti obdelave polstrukturiranih in nestrukturiranih podatkov, zaradi česa je primeren za večino sodobnih aplikacij (Gillis, brez datuma).

Je dokumentno usmerjen sistem, ki ne zahteva predhodno definiranih shem, kot relacijske podatkovne baze. To pomeni, da se lahko prilagodi spremembam v podatkovni strukturi, zaradi tega je eden najbolj priljubljenih za »real-time« aplikacije. Podatki so shranjeni kot dokumenti, vsak dokument ima svoj edinstven ključ. Dokumenti so shranjeni v zbirkah, katere delujejo kot tabele v relacijskih bazah. Uporabniki lahko ustvarijo več podatkovnih baz z več zbirkami, da organizirajo svoje podatke.

Nudijo horizontalno skalabilnost, kar omogoča aplikacijam, da se razširijo na več vozlišč, ko naraščajo njihove potrebe. Gre za način obvladovanja naraščajočih podatkov z dodajanjem več strežnikov, namesto da bi v obstoječ strežnik dodali več podatkov. Primer lahko ponazorimo na trgovini s sladkarijami, katere prodajamo strankam. Sladkarije postajajo priljubljene in število strank narašča, čez čas ena trgovina postane premajhna samo za naraščajoče število strank, zato se odpre še ena trgovina na drugi lokaciji.



Slika 6: Vizualizacija delitve pri horizontalni skalabilnosti
(Vir: https://www.mongodb.com/basics/horizontal-vs-vertical-scaling)

2.2.1 Prednosti in slabosti MongoDB

Prednosti:

- **Preprostost:** Ponuja preprost poizvedbeni jezik v primeru s SQL jezikom pri relacijskih podatkovnih bazah.
- **Hitri zapis in branje:** MongoDB je zasnovan za hitre operacije zapisovanja in branja, kar ga naredi za odlično izbiro za aplikacije v realnem času, kot so družabna omrežja in mobilne aplikacije.
- **Razširljivost:** Dejstvo, da je MongoDB zbirka podatkov s horizontalno razširljivostjo, je zelo ugodno. Med ravnanjem z njimi lahko razpršite ogromno količino podatkov med številne stroje.
- **Zunanja podpora:** MongoDB podpira več različnih mehanizmov za shranjevanje podatkov preko zunanjih API vmesnikov.

Slabosti:

- **JOIN tabel:** MongoDB ne podpira povezav (JOIN) tako kot relacijske podatkovne baze. Za uporabo funkcionalnosti povezav (JOIN) mora programer to ročno vključiti.
- **Velikost**: Velikost dokumenta je omejena na 16 MB in gnezdenje dokumentov na globini več kot 100 slojev ni dovoljeno.
- **Podvojevanje**: Podvojevanje podatkov, omejitev otežuje obvladovanje naborov podatkov, saj relacije niso dobro določene. Sčasoma lahko podvojevanje podatkov privede do korupcije, saj ne ustreza ACID standardom.

Varnost: Uporabniška avtentikacija ni privzeto omogočena v podatkovnih bazah. Posledica tega je, da so nezaščitene baze velikokrat tarča zlonamernih napadov (KnowledgeNile, brez datuma).

SQL je že mnogo let uspešen, vendar s prihodom Big Data se zdi, da ima MongoDB obetavno prihodnost. Vendar to ne pomeni, da bo SQL Server v celoti odpravljen. Katera podatkovna baza se bo uporabila za določen scenarij, je še vedno odvisno od uporabnikove zahteve.

2.2.2 Orodja MongoDB

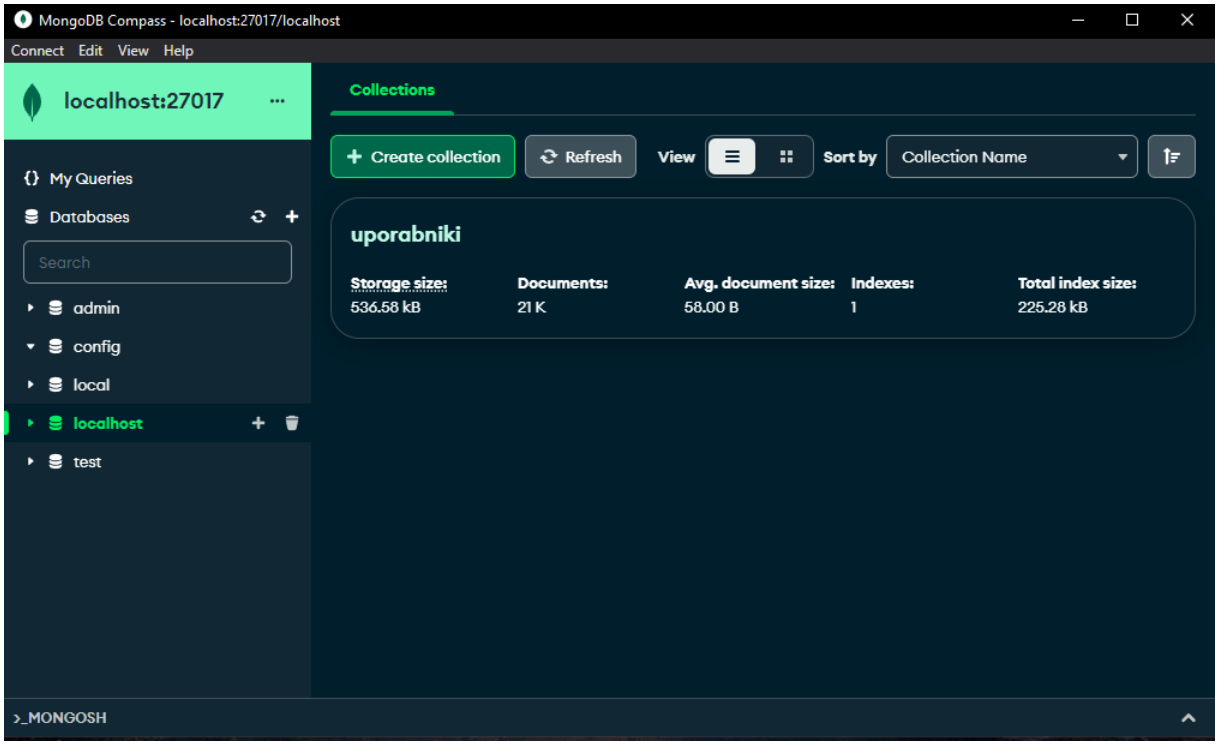
Na voljo so nam številna orodja za učinkovito delo s podatkovno zbirko, vključno z MongoDB Compassom, za delo preko grafičnega vmesnika za vizualno delo in MongoDB Shellom za delo preko terminala. Vsako od teh orodij ima svoje značilnosti, ki omogočajo upravljanje podatkovne zbirke, administracijo, poizvedovanje, analitiko in organizacijo podatkovne strukture.

MongoDB Kompas (ang. Compass) je zmogljiv GUI (ang. Graphical User Interface) za vizualno poizvedovanje, združevanje in analiziranje podatkov v MongoDB podatkovni bazi. Je brezplačen za uporabo, trenutna podpora je na operacijskih sistemih MacOS, Windows in Linux (Pedamkar, 2023).

Omogoča kar nekaj koristnih funkcij, kot so:

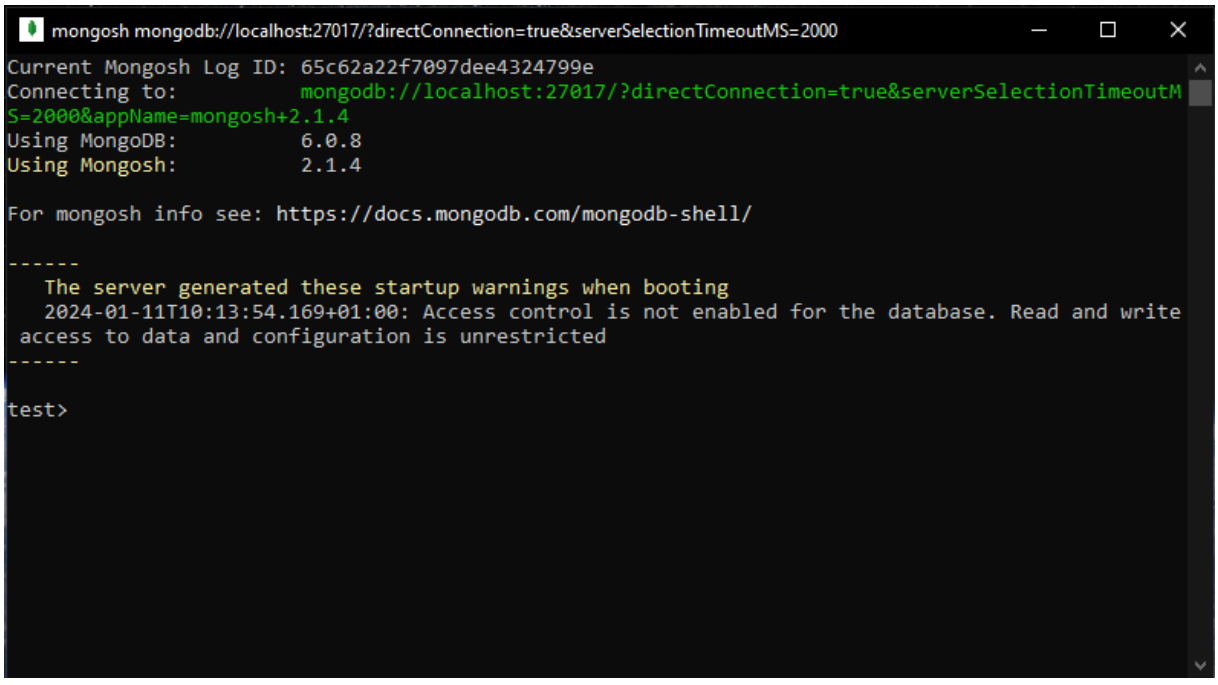
- **Uvoz in izvoz podatkov:** Lahko se uporabi za uvoz velikih datotek, podpira datoteke JSON (Javascript Object Notation) ali CSV (Comma-Seperated Values), datoteke vrednosti, ki so ločene z vejico.
- **Poizvedovanje:** Omogoča proces iskanja, filtriranja in pridobivanja specifičnih informacij iz zbirke podatkov. V nasprotju z relacijskimi bazami se tukaj uporablja MQL (MongoDB Query Language), kar je v bistvu enako SQL jeziku.

MongoDB lupina (ang. Shell) je interaktivna ukazna vrstica, katera se lahko uporabi za povezovanje s strežnikom podatkovne baze in za neposredno izvajanje ukazov preko terminalnega okna. Ukaze, katere lahko izvajamo, nam omogočajo branje, pisanje, urejanje ali manipuliranje podatkov. Prav tako omogoča tudi izvajanje zunanjih skriptov za izvajanje ponavljajočih procesov (Papiernik, 2021)**.**



Slika 7: Grafični vmesnik MongoDB Compass

(Vir: lastni vir)



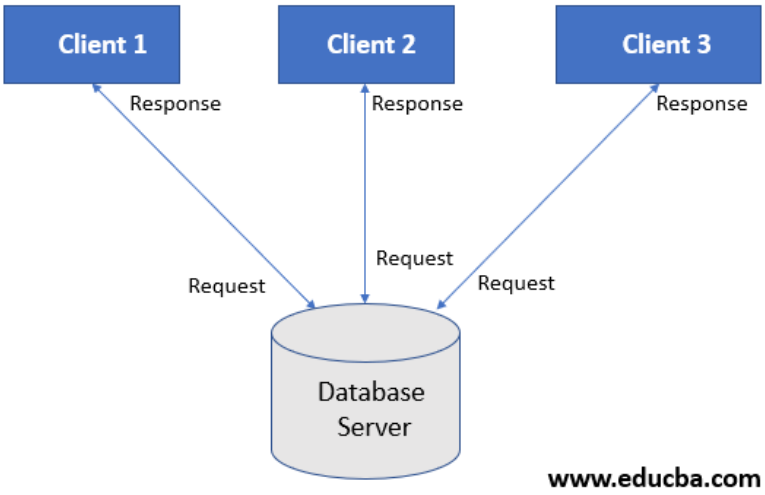
Slika 8: MongoDB Shell orodje

(Vir: lastni vir)

2.3 PostgreSQL

PostgreSQL, pogosto izgovarjan kot "Post-GRES," je odprtokodna podatkovna baza, ki slovi po svoji zanesljivosti, prilagodljivosti in podpori odprtih tehničnih standardov. Za razliko od drugih relacijskih sistemov za upravljanje podatkov (RDMBS) PostgreSQL podpira tako ne logične kot relacijske podatkovne tipe. To ga postavlja med bolj skladne, stabilne in zrele relacijske podatkovne baze, ki so na voljo danes. Sprva razvit leta 1986 kot nadaljevanje projekta »INGRES«, kateri je deloval kot odprtokodna SQL relacijska podatkovna baza, ki se je začela razvijati v zgodnjih sedemdesetih letih. »POSTGRES«, zdaj znan kot PostgreSQL, je nastal po zamisli Michaela Stonebreakerja, profesorja računalniške znanosti na univerzi Berkeley (IBM, brez datuma).

Arhitektura PostgreSQL sledi modelu »odjemalec-strežnik«. Primarni del deluje kot storitev, ki je odgovorna za določanje struktur podatkov, shranjevanje podatkov in obdelavo poizvedb. Ta arhitektura omogoča PostgreSQL sistemu, da zadovolji več odjemalcev, ki se povezujejo lokalno ali preko omrežja. Ko glavni proces prejme povezavo od odjemalca, ustvari nov samostojen proces, ki je posvečen tej določeni povezavi. Če se povezuje več odjemalcev, potem vsak dobi svoj posebej ustvarjen proces (Punreddy, 2023).



Slika 9: Arhitektura »Client-Server« PostgreSQL

(Vir: https://www.educba.com/postgresql-client/)

2.3.1 Prednosti in slabosti PostgreSQL

PostgreSQL je eden najstarejših, a hkrati najbolj naprednih sistemov za upravljanje odprtokodnih podatkovnih baz (Dhruv, 2024).

Prednosti:

- **Odprtokodnost:** PostgreSQL je popolnoma odprtokoden, kar pomeni, da je njegova izvorna koda brezplačno na voljo pod licenco odprte kode. Imate prilagodljivost, da ga spreminjate, uporabljate in implementirate glede na svoje potrebe.
- **Stroškovno učinkovit:** PostgreSQL je popolnoma odprtokoden, kar pomeni, da ni treba plačevati ničesar za uporabo PostgreSQL.
- **Odpornost na napake:** Značilnosti, kot je vodenje dnevnikov zapisovanja pred dejanskim zapisovanjem, omogočajo, da je PostgreSQL visoko odporen na napake.
- **Podpora za geografske objekte:** PostgreSQL podpira geografske objekte, tako da lahko uporabniki enostavno shranjujejo in uporabljajo geoprostorske podatke glede na svoje potrebe.

Slabosti:

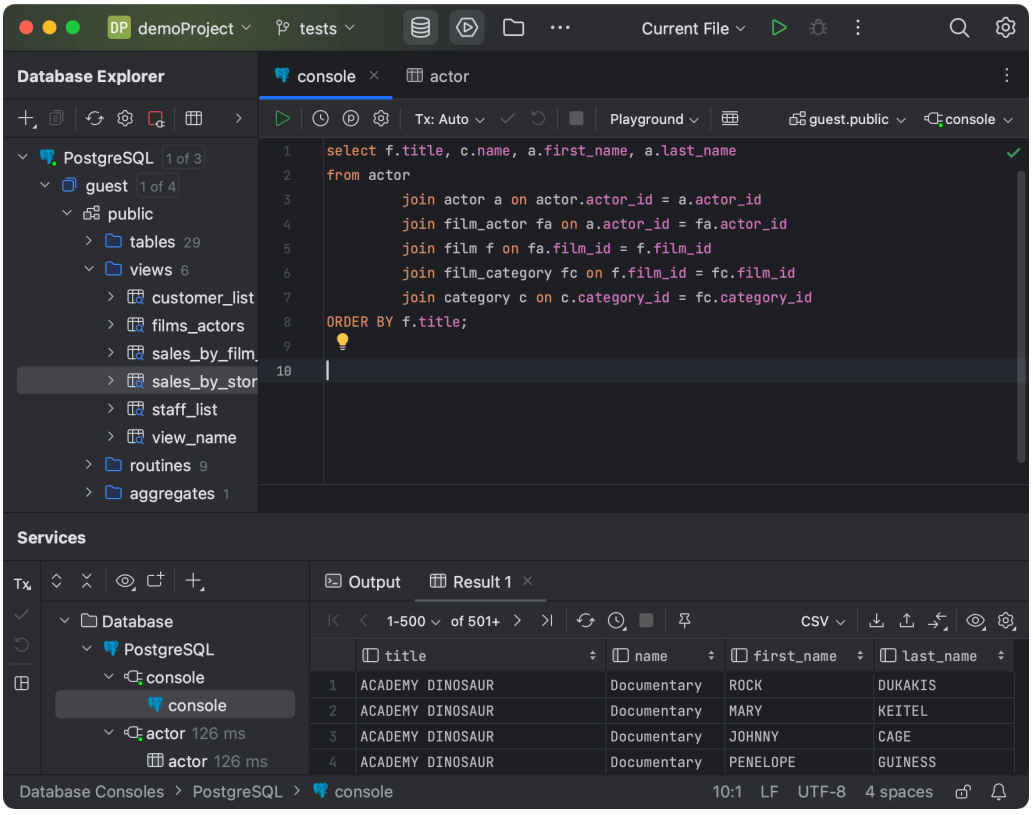
- Obdelava poizvedb postane počasna, če obstaja veliko število zapisov.
- Če je učinkovitost za manjšo bazo in preprostost prednost, bi morali izbrati MySQL, saj je PostgreSQL počasnejši od MySQL v takih situacijah.
- Horizontalno skaliranje PostgreSQL-a ni enostavna naloga.
- PostgreSQL deluje počasi, ko se v SQL tabelah shranjuje veliko video/avdio podatkov.
- Zapleteni uvoz: V PostgreSQL-u je težko doseči uvoz zapletenih JSON podatkov v podatkovno bazo.

2.3.2 Orodje DataGrip

Sistem za upravljanje baz podatkov (DBMS) je bistveni računalniški sistem, ki omogoča uporabnikom izvajanje različnih operacij za manipulacijo in upravljanje podatkov v bazi. V svoji raziskavi sem uporabil DataGrip.

DataGrip je integrirano razvojno okolje (IDE), ki je namenjeno za upravljanje in razvoj podatkovnih baz. Razvilo ga je podjetje JetBrains, znano tudi po razvoju drugih IDE-jev, kot sta IntelliJ IDEA in PyCharm. Omogoča izvajanje poizvedb v različnih načinih, pregledno navigacijo in vizualno reprezentacijo po shemi podatkovne baze (JetBrains, brez datuma).

Za lažji razvoj in načrtovanje podatkovne baze je na voljo tudi podrobna razlaga načrta, poizvedb ter pametno dokončanje kode za hitrejše pisanje SQL poizvedb. Vključuje tudi razhroščevalnik (ang. Compiler), kateri sproti preverja pravilnost kode.



Slika 10: Slika uporabniškega vmesnika DataGrip

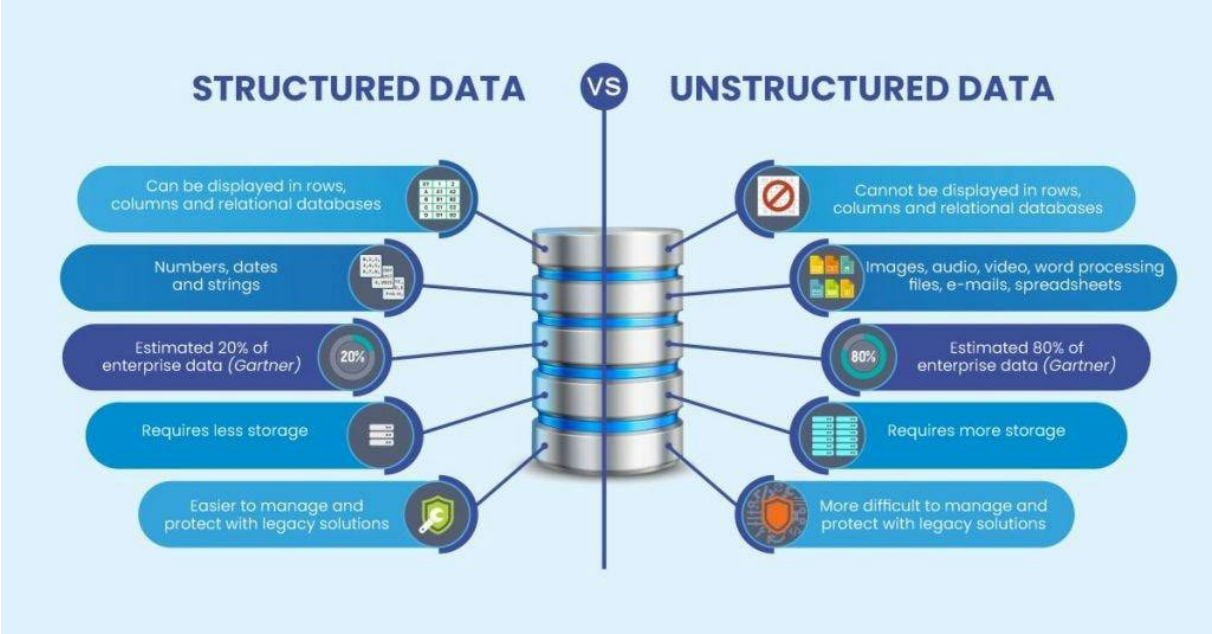
(Vir: https://www.jetbrains.com/help/datagrip/new-ui.html)

2.4 Primerjava med MongoDB in PostgreSQL glede na obdelavo BigData

V razvoju informacijske tehnologije in digitalnega poslovanja so podatki postali eno ključnih sredstev za pridobivanje vpogledov, optimizacijo procesov in sprejemanje informiranih odločitev. S hitrim naraščanjem obsega in kompleksnosti podatkovnih zbirk se pojavlja potreba po učinkovitih in prilagodljivih rešitvah za obdelavo velikih podatkov.

V preteklosti je bila večina podatkov shranjenih v obliki strukturiranih podatkov. Vendar pa se je zaradi naprednih tehnologij večina premaknila na nestrukturirane. Od leta 2023 je 80 do 90 % velikih podatkov v obliki nestrukturiranih, sem sodijo slike, videi, dokumenti, naslovi elektronske pošte. To je rezultat naprednejših aplikacij, kot so Google Mail, YouTube in podobne strani (Dawar, 2020).

Večina podjetij hrani varnostno kopijo svojih podatkov. Vendar pa trenutne ocene kažejo, da se podatki, povezani s poslovanjem, vsako leto povečujejo, kar predstavlja izziv za shranjevanje podatkov. Večina poslovnih podatkov je "hladnih" podatkov (podatki, ki niso bili dostopani 30 dni), kar zamaši drage trde diske in povečuje stroške shranjevanja (Smallcombe, 2024).



Slika 11: Primerjava med strukturiranimi in nestrukturiranimi podatki

(Vir: Medium 2020)

2.4.1 MongoDB in Big Data

MongoDB je priljubljena izbira za obvladovanje velikih podatkov zaradi svoje sposobnosti enostavnega obvladovanja različnih formatov podatkov, podpore za analizo v realnem času, hitrega sprejemanja podatkov, nizke zakasnitve pri delovanju, prilagodljivega modela podatkov, enostavnega vodoravnega razširjanja ter močnega poizvedovalnega jezika (MongoDB, brez datuma).

Dobra izbira za MongoDB je:

- Če imamo shemo podatkovne baze dinamično in se lahko pogosto spreminja, kar zahteva zmožnost hitrega preoblikovanja podatkov.
- V prvi vrsti potrebujemo dobro razširljivost, ki se lahko enostavno širi vodoravno, ne da bi žrtvovali zmogljivost strojne opreme in delovanja same aplikacije.
- Imamo raje dokumentni model, kateri je glede na vrsto aplikacije bolj primeren kot relacijska podatkovna baza.
- Združljivost velikih podatkov in platforme, kot je MongoDB Atlas, katera nudi možnosti oblaka.

Kot vsaka podatkovna baza ima tudi MongoDB svoje omejitve glede velikosti in procesiranja velike količine podatkov:

- **Omejitev velikosti baze:** Velikost podatkovne baze MongoDB ne sme presegati 64 terabajtov (TB).
- **Omejitve velikosti posameznega dokumenta:** MongoDB ima največjo velikost dokumenta 16 megabajtov (MB). Če dokument presega to omejitev, ga ni mogoče shraniti v bazo.
- **Omejitev povezav:** MongoDB postavlja omejitev na število sočasnih povezav, ki jih je mogoče vzpostaviti s posameznim primerkom MongoDB. Ta omejitev je odvisna od razpoložljivega pomnilnika in procesorskih virov strežnika.
- **Finančna omejitev:** Odvisno je tudi, katero verzijo MongoDB uporabljamo. V moji raziskavi sem uporabil brezplačno verzijo, katera je primerna za manjše aplikacije in

študentske projekte. Večje aplikacije pa zahtevajo nadgradnjo storitev, katere so plačljive.

2.4.2 PostgreSQL in Big Data

PostgreSQL bolj primeren za tradicionalne aplikacije, ki temeljijo na SQL, in manj za okolja z ogromnimi količinami podatkov. Za takšne scenarije je potrebna dodatna strežniška oprema. Kadar govorimo o relacijskih podatkovnih bazah, kot je PostgreSQL, je ključnega pomena tudi upoštevati koncept normalizacije. Normalizacija v SQL-u je postopek, kjer gre preprosto za organiziranje podatkov v tabelah na način, ki zmanjšuje ponavljanje informacij in zagotavlja, da so podatki dosledni. To dosežemo z razdelitvijo podatkov na manjše dele in povezovanjem teh delov med seboj.

Igra ključno vlogo pri učinkovitem upravljanju s podatki, še posebej v okoljih z velikimi količinami podatkov, kot so veliki podatki. Kljub temu lahko v določenih primerih relacijske podatkovne baze obravnavajo tudi veliko količino podatkov, vendar se moramo zavedati nekaj omejitev:

- **Omejena horizontalna razširljivost:** PostgreSQL ima omejeno podporo za horizontalno razširljivost, kar lahko otežuje obvladovanje velikih količin podatkov, ki presegajo zmogljivosti enega samega strežnika.
- **Zagotavljanje doslednosti**: Medtem ko PostgreSQL zagotavlja doslednost podatkov, se lahko to pokaže kot omejitev pri obdelavi velikih količin podatkov v realnem času ali pri potrebi po hitrem skaliranju.
- **Kompleksna uprava**: Upravljanje zmogljivosti in skaliranje PostgreSQL sistema za obvladovanje big data zahteva kompleksno upravljanje in nadzor, kar lahko predstavlja izziv za organizacije.
- **Optimizacija poizvedb:** Kljub temu, da PostgreSQL ponuja mehanizme za optimizacijo poizvedb, je treba pri velikih količinah podatkov posebej paziti na učinkovitost poizvedb in prilagoditi sheme ter indekse glede na potrebe.

ACID lastnosti: Kadar relacijska podatkovna baza deluje z veliko količino podatkov, potem ACID lastnosti, kot sta konsistenčnost in izolacija, nista več zagotovljeni. To je še posebej

težje danes, kjer večina aplikacij obravnava podatke v realnem času »real-time« (Dawar, 2020).

3 METODE RAZISKAVE

3.1 Izbor meril za primerjavo med MongoDB in Postgre SQL

V tej raziskavi se osredotočamo na primerjavo med dvema priljubljenima podatkovnima bazama, MongoDB in PostgreSQL, s poudarkom na njuni zmogljivosti glede hitrosti vstavljanja podatkov, obdelave in izvajanja poizvedb. Cilj je pridobiti celoten vpogled v njuno učinkovitost pri obdelavi velikih količin podatkov.

3.1.1 Meritev hitrosti časa vstavljanja podatkov v podatkovno bazo

Za hipotezo 1: MongoDB bo ponudil učinkovitejšo obdelavo velikih količin podatkov v primerjavi s PostgreSQL.

S to meritvijo se osredotočimo na merjenje časa, ki ga zahteva vstavljanje določene količine podatkov v podatkovno bazo oziroma število vnešenih dokumentov na sekundo/minuto. S pomočjo ustvarjene skripte lahko natančno določimo, koliko časa potrebuje baza za sprejem in shranjevanje različnih obsežnosti podatkov.

Meritev hitrosti časa vstavljanja podatkov se osredotoča na:

- generiranje testnih podatkov, kateri se bodo vstavili v obe podatkovni bazi;
- implementiranje optimizirane Python skripte, ki bo vstavljala testne podatke v podatkovno bazo,
- določitev meril, na podlagi katerih bomo ocenili hitrost vstavljanja podatkov. To vključuje čas, ki ga baza potrebuje za vstavljanje določene količine podatkov in število vnesenih dokumentov na sekundo,
- izvedba meritve, meritev se lahko tudi večkrat ponovi, da se pridobi zanesljive povprečne rezultate.

Ta metodologija nam omogoča sistematičen pristop k merjenju hitrosti vstavljanja podatkov v obe bazi podatkov ter objektivno oceno njune učinkovitosti v tem kontekstu.

3.1.2 Meritev učinkovitosti poizvedb nad podatki iz podatkovne baze

Za hipotezo 1: MongoDB bo ponudil učinkovitejšo obdelavo velikih količin podatkov v primerjavi s PostgreSQL.

S to meritvijo se osredotočamo na oceno učinkovitosti poizvedb za vstavljanje in branje podatkov iz podatkovne baze MongoDB in PostgreSQL. Cilj merjenja je primerjati čase izvajanja poizvedb med obema bazama podatkov, da bi ugotovili, katera baza ponuja hitrejše branje ali vstavljanje podatkov.

Meritev učinkovitosti poizvedb za branje podatkov iz baze se osredotoča na:

- implementacijo enakovrednih poizvedb za branje ali vstavljanje podatkov v MongoDB in PostgreSQL;
- izvedbo meritev časa, ki ga vsaka baza potrebuje za izvajanje poizvedb za branje ali vstavljanje podatkov;
- analizo rezultatov in primerjavo časov izvajanja poizvedb med MongoDB in PostgreSQL.

Ta meritev bo dodatno prispevala k razumevanju učinkovitosti in zmogljivosti obeh baz podatkov, ter bo pomagala pri sklepanju o njihovi primerljivosti glede na hipotezo o obdelavi velikih količin podatkov.

3.1.3 Primerjava učinkovitosti transakcijskega vstavljanja

Za hipotezo 3: Pri obdelavi velikih količin podatkov bo MongoDB ponudil boljše rezultate v primerjavi s Postgre SQL glede na hitrost in učinkovitost.

Meritev vstavi večje število poizvedb v transakciji tako v MongoDB kot PostgreSQL, primerjajoč čas izvedbe in doslednost. Zabeleženi so posamezni in skupni časi vstavljanja,

ponujajoč vpogled v učinkovitost obdelave velikega števila vstavljanj v vsaki od baz. Meritev omogoča oceno, katera baza bolje izpolnjuje zahteve po hitrosti in doslednosti transakcij.

Meritev se osredotoča na:

- vstavljanje večjega števila poizvedb v transakciji tako v MongoDB kot PostgreSQL;
- merjenje časa izvajanja posameznih in skupnih transakcij za vstavljanje podatkov v vsako od baz;
- zabeleženje časov izvajanja ter analiza posameznih in skupnih časov vstavljanja;
- ocenjevanje doslednosti rezultatov v vsaki bazi.

Ta meritev bo omogočila pridobitev vpogleda v učinkovitost obdelave velikega števila transakcijskega vstavljanja v vsaki od baz podatkov, kar bo pomagalo pri oceni, ali MongoDB res ponuja boljše rezultate v primerjavi s PostgreSQL glede na hitrost in učinkovitost.

3.1.4 Meritev obsega in zmogljivosti skaliranja ter poraba pomnilnika

Za hipotezo 3: Pri obdelavi velikih količin podatkov bo MongoDB ponudil boljše rezultate v primerjavi s Postgre SQL glede na hitrost in učinkovitost.

S to meritvijo se osredotočamo na oceno obsega in zmogljivosti skaliranja MongoDB in PostgreSQL. Cilj je primerjati sposobnost obeh baz podatkov za obvladovanje velikega obsega podatkov ter njihovo zmogljivost pri skaliranju infrastrukture za podporo večjim obremenitvam.

Meritev obsega in zmogljivosti skaliranja se osredotoča na:

- izvedbo testov obsega z uvajanjem vedno večje količine podatkov v obe bazi podatkov;
- merjenje odzivnega časa in zmogljivosti obeh baz podatkov pri različnih obremenitvah;
- analizo rezultatov in primerjavo sposobnosti skaliranja MongoDB in PostgreSQL;
- ocenjevanje, kako dobro se vsaka baza prilagaja povečanju obsega podatkov in obremenitvam glede na porabo pomnilnika.

Ta meritev bo omogočila pridobitev vpogleda v to, kako se MongoDB in PostgreSQL obnašata pri obdelavi velikih količin podatkov in kako se obnašata pri skaliranju infrastrukture, kar bo

pomagalo pri oceni, ali MongoDB res ponuja boljše rezultate glede na hitrost in učinkovitost v primerjavi s PostgreSQL. Statistiko bomo pridobili iz povprečne porabe pomnilnika računalnika, kateri bo zraven izvajal druge procese.

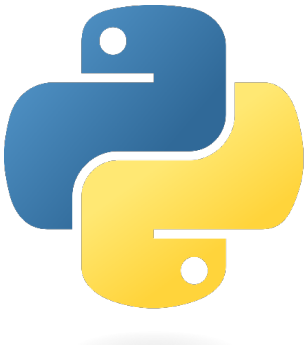
3.2 Zbiranje podatkov o MongoDB in PostgreSQL

3.2.1 Programski jezik Python

Python je široko uporabljen, razlagalni, objektno usmerjen in visokonivojski programski jezik z dinamično semantiko, ki se uporablja za programiranje splošne rabe. Je povsod prisoten, ljudje vsakodnevno uporabljajo številne naprave, ki delujejo na Pythonu, pa če se tega zavedajo ali ne. Ustvarjen je bil s strani Guida van Rossuma in prvič izdan 20. februarja 1991 (Python, brez datuma).

Je priljubljen jezik za izvajanje skriptnih nalog, še posebej tistih, povezanih z uporabo podatkovnih baz. V tem jeziku je na voljo obsežna knjižnica orodij in modulov, kar olajša izvajanje raznovrstnih nalog. Poleg tega je Python neodvisen od platforme, zato se skripti, napisani v tem jeziku, brez težav izvajajo na različnih operacijskih sistemih. Ta značilnost omogoča, da se skripte enostavno prilagajajo različnim okoljem, kar je še posebej koristno pri implementaciji v raznolikih sistemskih okoljih (Coursera, 2023).

Ima tudi orodje za upravljanje paketov (pip). S pomočjo pip-a lahko enostavno namestimo, upravljamo Python pakete ter njihove odvisnosti. To orodje nam omogoča, da učinkovito upravljamo z zunanjimi knjižnicami in moduli, ki jih potrebujemo za svoje projekte. Knjižnice se namestijo z ukazom **pip install ime_knjižnice**. Vključijo se z ukazom se z vrstico **import ime_knjižnice**.



Slika 10: Logotip Pythona
(Vir: https://1000logos.net/python-logo/)

3.3 Razlaga skript za izvajanje meritev

Za izvajanje meritev zmogljivosti in učinkovitosti MongoDB in PostgreSQL baz podatkov smo uporabili več skript, katere so bile ključne za zajem podatkov, izvajanje testov ter analizo rezultatov. Uporabili smo različne knjižnice v jeziku Python, ki so omogočile povezavo s podatkovnimi bazami, generiranje testnih podatkov, merjenje časa izvajanja operacij ter analizo in oblikovanje rezultatov. V tem poglavju bomo podrobno razložili delovanje vsake skripte, njene ključne funkcionalnosti in kako smo uporabili knjižnice za izvedbo meritev.

3.3.1 Uporabljene knjižnice

Pri skriptah za meritve smo uporabili več knjižnic za delo s podatki, meritvami in analizo rezultatov:

Pymongo: Knjižnica za delo z MongoDB bazami podatkov, ki omogoča povezavo, upravljanje zbirke podatkov ter izvajanje poizvedb.

Psycopg2: Knjižnica za delo s PostgreSQL bazami podatkov, ki omogoča povezavo, upravljanje s shemami ter izvajanje SQL poizvedb.

Faker: Knjižnica za generiranje lažnih oziroma naključnih podatkov, kot so imena, naslovi, e-poštni naslovi itd., kar je koristno pri ustvarjanju testnih podatkov.

Time: Vgrajena knjižnica za merjenje časa izvajanja operacij in ustvarjanje časovnih oznak.

Psutil: Knjižnica za pridobivanje informacij o sistemu, kot je poraba pomnilnika, CPU obremenitev.

Pandas: Knjižnica za analizo in manipulacijo podatkov, ki omogoča ustvarjanje in oblikovanje tabel ter statistično analizo rezultatov.

Te knjižnice so bile ključne pri izvajanju meritev, zajemu rezultatov ter analizi podatkov za oceno zmogljivosti in učinkovitosti MongoDB in PostgreSQL baz podatkov. Uporabijo se lahko samo v programskem jeziku Python.

3.3.2 Razlaga skript

Pristop programiranja je bil za vsako skripto približno enak, sledil pa je naslednji strukturi:

- **Uvoz knjižnic:** Najprej je bilo potrebno vključiti vse potrebne knjižnice, katere so omenjene v poglavju 3.3.1. To vključuje knjižnice za delo s podatkovno bazo, generiranje podatkov, merjenje časa, vizualizacijo rezultatov in druge potrebne funkcionalnosti.

```
from pymongo import MongoClient, InsertOne
from faker import Faker
import random
import time
import matplotlib.pyplot as plt
```

Slika 12: Uvoz knjižnic (Primer za meritev 1)
(Vir: Lastni vir)

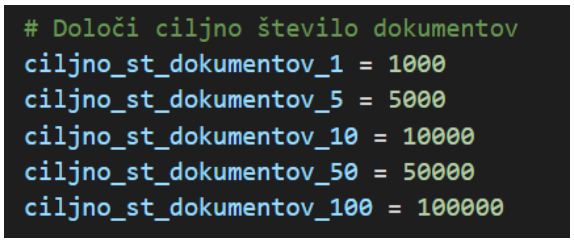
- **Povezava na bazo podatkov:** Nato se vzpostavi povezava s podatkovno bazo, s katero se bo izvajala meritev. To vključuje določitev parametrov povezave, kot so ime baze podatkov, uporabniško ime, geslo, gostitelj in vrata.

```
# Povezava na MongoDB strežnik
klient = MongoClient("mongodb://localhost:27017")
zbirka = klient["diplomskodelo"]
kolekcija = zbirka["uporabniki"]
```

Slika 13: Povezava na strežnik MongoDB (Primer za meritev 1)
(Vir: Lastni vir)

Slika 14: Povezava na strežnik PostgreSQL (Primer za meritev 1)
(Vir: Lastni vir)

- **Določitev ciljnega števila vstavljenih dokumentov**: Potrebno je definirati ciljno število dokumentov ali poizvedb, dokumente/poizvedbe sem vstavljal po 1000, 5000, 10000, 50000, 100000.



Slika 15: Primer izseka kode za vstavljanje podatkov (Primer za meritev 1)
(Vir: Lastni vir)

- **Izvedba meritev**: Sledi izvedba dejanskih meritev, ki vključujejo manipulacijo podatkov, izvajanje poizvedb ali drugih operacij na podatkovni bazi. To lahko vključuje generiranje testnih podatkov, vstavljanje podatkov v bazo, izvajanje poizvedb, merjenje časa izvajanja in zbiranje rezultatov. Spodnja slika prikazuje primer za meritev 1.

Slika 16: Primer izseka kode za generiranje testnih podatkov (Primer za meritev 1)
(Vir: Lastni vir)

- **Analiza in vizualizacija rezultatov:** Končno se pridobljeni rezultati analizirajo in vizualizirajo, da bi lahko iz njih izluščili pomembne ugotovitve. To lahko vključuje ustvarjanje grafov, tabel ali poročil, ki ponazarjajo rezultate meritev in omogočajo lažje razumevanje zbranih podatkov.

```
# Ustvarjanje tabele
print("Število dokumentov  |  Hitrost vstavljanja (dokumenti/sekundo)")
print("-------------------------------------------------------------")
for st_dokumentov, hitrost_vstavljanja in tabela_rezultatov:
    print(f"{st_dokumentov: <18} |  {hitrost_vstavljanja:.2f}")
```

Slika 17: Primer izseka kode za generiranje tabele (Primer za meritev 1)

(Vir: Lastni vir)

- **Zaključek in zaprtje povezav:** Po vsaki izvedbi meritve je bilo potrebno v obeh podatkovnih bazah izbrisati vse podatke, da se je lahko izvedla naslednja meritev. Za to sem pripravil dve skripti še za praznjenje MongoDB in PostgreSQL podatkov. Uporabljeni sta bili funkciji DELETE in delete_many.

```
import psycopg2
# Povezava na PostgreSQL strežnik
connection = psycopg2.connect(
    dbname="diploma",
    user="postgres",
    password="test",
    host="localhost",
    port="5432"
)
cursor = connection.cursor()
# Izvedba ukaza za brisanje vseh podatkov iz tabele
cursor.execute("DELETE FROM uporabniki")
# Potrditev sprememb
connection.commit()
# Zaprtje povezave
cursor.close()
connection.close()
```

Slika 18: Skript za izbris podatkov v PostgreSQL

(Vir: Lastni vir)

```python
from pymongo import MongoClient
# Povezava na MongoDB strežnik
klient = MongoClient("mongodb://localhost:27017")
# Dostop do zbirke, iz katere želite izbrisati dokumente
zbirka = klient["diplomskodelo"]["uporabniki"]
# Izbris vseh dokumentov iz zbirke
zbirka.delete_many({})
print("Vsi dokumenti so bili izbrisani iz zbirke.")
```

Slika 19: Skript za izbris podatkov v MongoDB

(Vir: Lastni vir)

4 REZULTATI IN ANALIZA

4.1 Primerjalna analiza zmogljivosti MongoDB in PostgreSQL

V tem poglavju bomo analizirali rezultate in prikaz v tabelah za MongoDB in PostgreSQL, s poudarkom na več dejavnikih, kot so strojna oprema, konfiguracija sistema in obremenitev med izvajanjem meritev. Cilj je pridobiti celovit vpogled v zmogljivost obeh sistemov. Rezultate bomo predstavili v tabelah, ki bodo zajemale ključne metrike, kot so čas izvajanja operacij in poraba pomnilnika. Te tabele bodo omogočale jasno primerjavo med zmogljivostjo MongoDB in PostgreSQL pri različnih obremenitvah. Zraven tega bomo obravnavali tudi podobne meritve za PostgreSQL, kar nam bo omogočilo primerjalno analizo med obema sistemoma ter oceno njune učinkovitosti in skalabilnosti.

4.1.1 Omejitev strojne opreme

Točno je težko definirati, kakšna velikost podatkov spada pod izraz »veliki podatki«, za primer lahko vzamemo podjetja ali večje korporacije, kjer imajo podatke o strankah ali izdelkih v terabytih. Za lažjo predstavo 1 TB je enak 1000 GB. **S**eveda takšno količino težko dosežem z osebnim računalnikom, zato sem v podatkovno bazo za meritev vnesel podatke v velikosti več tisoč poizvedb.

Nekaj glavnih faktorjev pri omejitvi strojne opreme:

- **Procesorska moč (CPU):** Hitrost procesorja je neposredno povezana s hitrostjo procesiranja podatkov. Če sistem deluje na zmogljivem procesorju, se bodo tudi podatki hitreje vstavili v bazo.
- **Pomnilnik (RAM):** Enako kot za procesor več RAM-a omogoča boljše upravljanje velike količine podatkov v pomnilniku. Rezultate obremenitve bomo videli v meritvi 4.
- **Hitrost omrežne povezave:** Če se podatki prenašajo med oddaljenimi strežniki in klienti, je hitrost internetne povezave zelo pomembna. Slaba povezava lahko povzroči dolge čase prenosa podatkov.

4.1.2 Analiza zmogljivosti MongoDB

Meritev 1:

Rezultati meritev kažejo, kako se je hitrost vstavljanja dokumentov v bazo spreminjala med izvajanjem programa. Program je vstavljal po 1000 dokumentov naenkrat, nato pa je zapisal hitrost vstavljanja za vsakih 1000 dokumentov.

Definicije tabele:

- **Število dokumentov:** Vsebuje število dokumentov, ki so bili vstavljeni v vsaki iteraciji zanke. Vsako vstavljanje dodaja novo vrednost v seznam, ki predstavlja skupno število dokumentov.
- **Čas vstavljanja (sekunde):** Vsebuje rezultate časa vstavljanja v sekundah.
- **Hitrost vstavljanja:** Vsebuje hitrost vstavljanja dokumentov (v dokumentih na sekundo) v vsaki iteraciji zanke. Za vsako iteracijo se izračuna hitrost vstavljanja na podlagi pretečenega časa od začetka izvajanja programa.

Tabela 1: Rezultat meritve 1 za MongoDB

Število dokumentov	Čas vstavljanja (sekunde)	Hitrost vstavljanja (doc/sekundo)
1000	0.970	892.24
5000	1.421	3127.44
10000	2.545	4422.29
50000	8.210	6965.62
100000	15.240	6426.80

(Vir: lastni vir)

Naša merjenja hitrosti vstavljanja dokumentov v bazo MongoDB so pokazala, da se hitrost vstavljanja povečuje s povečanjem števila vstavljenih dokumentov. To kaže na dobro skalabilnost baze MongoDB, saj je sposobna učinkovito obdelovati večje količine podatkov. Ta ugotovitev nam pripomore k dokazovanju, da je MongoDB primerna izbira za aplikacije, ki zahtevajo visoko zmogljivost in skalabilnost pri delu s podatkovno bazo.

Meritev 2:

Tabela prikazuje rezultate meritev učinkovitosti vstavljanja dokumentov in branja v MongoDB ter izvajanja poizvedb za različno število dokumentov.

Vstavljanje v podatkovno bazo:

- **Število dokumentov:** Ta stolpec prikazuje število dokumentov, za katere je bila izvedena meritev.
- **Čas vstavljanja (sekunde):** V tem stolpcu so prikazani časi, ki so bili potrebni za vstavljanje določenega števila dokumentov v bazo MongoDB. Meritve so izražene v sekundah.
- **Čas izvajanja poizvedbe (sekunde):** Ta stolpec prikazuje čase, potrebne za izvedbo poizvedb nad bazo MongoDB, pri čemer je bilo uporabljeno določeno število dokumentov. Meritve so tudi izražene v sekundah.

Tabela 2: Rezultat meritve 2 za MongoDB

Število dokumentov	Čas vstavljanja (sekunde)	Čas izvajanja poizvedbe (sekunde)
1000	0.303	0.028
5000	1.125	0.009
10000	2.789	0.015
50000	9.542	0.107
100000	12.154	0.249

(Vir: lastni vir)

Na podlagi teh rezultatov lahko sklepamo o učinkovitosti delovanja baze podatkov MongoDB pri obdelavi različnih količin podatkov. Na primer, lahko opazimo, da se časi izvajanja poizvedb povečujejo s povečanjem števila dokumentov, kar lahko nakazuje na to, da se čas izvajanja poizvedb povečuje s povečanjem obsega podatkovne baze.

Branje podatkov iz podatkovne baze:

- **Število dokumentov:** To stolpec prikazuje število dokumentov, za katere je bila izvedena meritev.
- **Čas izvajanja poizvedbe (sekunde):** V tem stolpcu so prikazani časi, ki so bili potrebni za branje določenega števila dokumentov iz podatkovne baze.

Tabela 3: Rezultat meritve branja 1 za MongoDB

Število dokumentov	Čas izvajanja poizvedbe
1000	0.009
5000	0.028
10000	0.042
50000	0.107
100000	0.249

(Vir: lastni vir)

Tabela 4: Rezultat meritve branja 2 za MongoDB

Število dokumentov	Čas izvajanja poizvedbe s parametrom
1000	0.150
5000	0.088
10000	0.119
50000	0.085
100000	0.086

(Vir: lastni vir)

Na podlagi teh rezultatov lahko sklepamo o učinkovitosti pridobivanja podatkov MongoDB pri obdelavi različnih količin podatkov. Izvedemo osnovno poizvedbo za branje in poizvedbo, kjer nastavimo parameter starosti, ki samo išče uporabnike starejše od 18. leta starosti. V obeh primerih vidimo, enako kot pri prvi meritvi, da čas raste eksponentno. Vidimo tudi, da je druga poizvedba hitrejša, ker izloči veliko podatkov.

Meritev 3:

Ta tabela prikazuje rezultate izvedbe meritev za različna ciljna števila transakcij v MongoDB. Vsaka vrstica tabele predstavlja eno transakcijo, pri čemer je v prvem stolpcu navedeno število dokumentov, ki so bili vstavljeni v okviru te transakcije in v drugem stolpcu čas izvajanja transakcije v sekundah. Razlaga rezultatov:

- **Število dokumentov:** To je število dokumentov, ki so bili vstavljeni v okviru posamezne transakcije. Z vsako novo transakcijo se to število povečuje.
- **Čas izvajanja (sekunde):** Ta stolpec prikazuje čas, ki je bil potreben za izvedbo posamezne transakcije v sekundah. Z večanjem števila dokumentov se običajno povečuje tudi čas izvajanja, kar kaže na to, da je potrebno več časa za obdelavo večjih količin podatkov.

Tabela 5: Rezultat meritve 3 za MongoDB

Število dokumentov posamezne transakcije	Čas izvajanja (sekunde)
1000	1.150
5000	2.456
10000	6.451
50000	10.451
100000	14.458

(Vir: lastni vir)

Na podlagi teh rezultatov lahko ocenimo, kako se spreminja čas izvajanja transakcij v odvisnosti od števila vstavljenih dokumentov. Opazimo, da se čas izvajanja povečuje z večanjem števila dokumentov, kar kaže na to, da se obdelava večjih količin podatkov običajno izvaja počasneje. Vedeti moramo tudi, da transakcijsko vstavljane ni enako kot navadno. Transakcijsko vstavljanje zagotavlja, da se vsi vstavki izvedejo kot ena celota, pri čemer se celotna transakcija bodisi uspešno zaključi, bodisi prekliče v primeru napake. To zagotavlja doslednost podatkovne baze. Nasprotno pa pri navadnem vstavljanju vsak vstavek poteka ločeno in neodvisno, kar pomeni, da lahko napaka pri enem vstavku vpliva samo na ta vstavek, na ostale pa ne.

Meritev 4:

Rezultati tabele predstavljajo rezultate meritev obsega in zmogljivosti skaliranj. Podatki o porabi pomnilnika in času izvajanja nam pomagajo razumeti, kako se MongoDB obnaša pri različnih obremenitvah in nam omogočajo primerjavo s pričakovanji in morebitnimi zahtevami sistema. Tukaj je razlaga posameznih stolpcev in njihovih vrednosti:

- **Število vstavkov:** To je število dokumentov, ki so bili vstavljeni v zbirko podatkov MongoDB v vsaki posamezni merilni iteraciji.
- **Poraba pomnilnika (MB):** Ta stolpec prikazuje spremembo v porabi pomnilnika med izvajanjem vstavkov. Vrednosti so izražene v megabajtih. Pozitivne vrednosti kažejo na povečanje porabe pomnilnika, medtem ko negativne vrednosti kažejo na zmanjšanje porabe pomnilnika.
- **Čas izvajanja (sekunde):** Ta stolpec prikazuje skupni čas, potreben za izvajanje vstavkov v zbirko podatkov MongoDB v vsaki posamezni merilni iteraciji.

Tabela 6: Rezultat meritve 4 za MongoDB

Število vstavljanj	Poraba pomnilnika (MB)	Čas izvajanja (sekunde)
1000	1.128	0.418
5000	-6.593	1.660
10000	-7.488	3.412
50000	5.128	17.684
100000	7.390	28.987

(Vir: lastni vir)

Pri analizi zmogljivosti in skaliranja MongoDB je pomembno upoštevati več dejavnikov, vključno s strojno opremo, konfiguracijo sistema in drugimi dejavniki. Med izvajanjem meritev je bil računalnik uporabljen za druge naloge, kar lahko vpliva na razpoložljive sistemske vire za izvajanje meritev. To lahko povzroči manj natančne rezultate zaradi dodatne obremenitve sistema.

4.1.3 Analiza zmogljivosti PostgreSQL

Meritev 1:

Rezultati meritev kažejo, kako se je hitrost vstavljanja poizvedb v podatkovno bazo PostgreSQL spreminjala med izvajanjem programa. Program je vstavljal po 1000 poizvedb naenkrat, nato pa je zapisal hitrost vstavljanja za vsakih 1000 poizvedb.

- **Število poizvedb**: Je ekvivalentno številu dokumentov pri vstavljanju v MongoDB.
- **Čas vstavljanja (sekunde):** To je čas, ki ga vsaka baza potrebuje za vstavljanje določenega števila poizvedb. Opazimo, da se čas vstavljanja povečuje s povečanjem števila vstavljenih poizvedb.
- **Hitrost vstavljanja (insert/sekundo):** To je število insert poizvedb, ki jih baza vstavi v eni sekundi. Kljub povečanju števila vstavljenih insertov opažamo, da se hitrost vstavljanja ohranja relativno stabilna.

Tabela 7: Rezultat meritve 1 za PostgreSQL

Število poizvedb	Čas vstavljanja (sekunde)	Hitrost vstavljanja (insert/sekundo)
1000	0.214	9318.28
5000	1.639	9423.90
10000	5.564	9621.20
50000	9.184	9117.57
100000	17.427	7225.21

(Vir: lastni vir)

Na podlagi rezultatov tabele za vstavljanje podatkov pri MongoDB vidimo, da je PostgreSQL rahlo počasnejši kot pa MongoDB. Čas vstavljanja je za večjo količino večji kot pri MongoDB. Enako velja za hitrost vstavljanja, na začetku je hitrost pri mali količini podatkov relativno enaka MongoDB, ko pa pride do večje količine dokumentov, pa tudi hitrost bistveno pade.

Meritev 2:

Tabela prikazuje rezultate meritev učinkovitosti vstavljanja in branja podatkov v PostgreSQL ter izvajanja poizvedb.

Vstavljanje v podatkovno bazo:

- **Število poizvedb:** To stolpec prikazuje število podatkov, za katere je bila izvedena meritev.
- **Čas vstavljanja (sekunde):** V tem stolpcu so prikazani časi, ki so bili potrebni za vstavljanje določenega števila podatkov v bazo PostgreSQL. Meritve so izražene v sekundah.
- **Čas izvajanja poizvedbe (sekunde):** Ta stolpec prikazuje čase, potrebne za izvedbo poizvedb nad bazo PostgreSQL, pri čemer je bilo uporabljeno določeno število podatkov. Meritve so tudi izražene v sekundah.

Tabela 8: Rezultat meritve 2 za PostgreSQL

Število poizvedb	Čas vstavljanja (sekunde)	Čas izvajanja poizvedbe (sekunde)
1000	0.121	0.021
5000	1.254	0.056
10000	3.125	0.090
50000	10.120	0.154
100000	16.245	0.335

(Vir: lastni vir)

Iz tabele lahko razberemo, da se čas vstavljanja poizvedb povečuje s povečanjem števila dokumentov enako kot pri vstavljanju v MongoDB. Prav tako se čas izvajanja poizvedbe povečuje s povečanjem števila podatkov. Če rezultate primerjamo s tabelo vstavljanja v MongoDB, vidimo da je PostgreSQL počasnejši pri vstavljanju, čas izvajanja poizvedbe pa je rahlo hitrejši, kar je možno zaradi različnih arhitektur podatkovnih baz in indeksiranja pri vstavljanju podatkov. PostgreSQL ima tudi bolj optimizirano izvajanje poizvedb, kot MongoDB.

Branje podatkov iz podatkovne baze:

- **Število poizvedb:** To stolpec prikazuje število dokumentov, za katere je bila izvedena meritev.
- **Čas izvajanja poizvedbe (sekunde):** V tem stolpcu so prikazani časi, ki so bili potrebni za branje določenega števila dokumentov iz podatkovne baze.

Tabela 9: Rezultat meritve branja 1 za PostgreSQL

Število poizvedb	Čas izvajanja poizvedbe
1000	0.021
5000	0.025
10000	0.050
50000	0.129
100000	0.285

(Vir: lastni vir)

Tabela 10: Rezultat meritve branja 2 za PostgreSQL

Število poizvedb	Čas izvajanja poizvedbe s parametrom
1000	0.013
5000	0.001
10000	0.002
50000	0.020
100000	0.070

(Vir: lastni vir)

V primerjavi z MongoDB vidimo, da je branje in vstavljanje podatkov skoraj enako, čeprav se čas izvajanja poizvedbe s parametrom odzove hitreje pri PostgreSQL kot MongoDB za nekaj milisekund. Poudariti je potrebno, da so podatki za obe bazi generirani naključno, kar pomeni

da je v PostgreSQL lahko manj uporabnikov starih nad 18 let in se zato tudi poizvedba izvede hitreje.

Meritev 3:

Ta tabela prikazuje rezultate izvedbe meritev za različna ciljna števila transakcij v PostgreSQL. Vsaka vrstica tabele predstavlja eno meritev, pri čemer je v prvem stolpcu navedeno število poizvedb, ki so bile vstavljene v okviru te transakcije, in v drugem stolpcu čas izvajanja. Razlaga rezultatov:

- **Število poizvedb:** To je število poizvedb, ki so bile vstavljene v okviru posamezne transakcije. Z vsako novo transakcijo se to število povečuje.
- **Čas izvajanja (sekunde):** Ta stolpec prikazuje čas, ki je bil potreben za izvedbo posamezne transakcije v sekundah. Z večanjem števila poizvedb se običajno povečuje tudi čas izvajanja, kar kaže na to, da je potrebno več časa za obdelavo večjih količin podatkov.

Tabela 11: Rezultat meritve 3 za PostgreSQL

Število poizvedb	Čas izvajanja (sekunde)
1000	0.114
5000	0.568
10000	1.119
50000	5.445
100000	11.136

(Vir: lastni vir)

Glede na tabelo rezultatov lahko razberemo, da se PostgreSQL boljše odnese pri transakcijskem vstavljanju kot MongoDB. Vidi se bistveno hitrejša razlika - ne samo v milisekundah, ampak celo v sekundah. Posledica tega so ACID lastnosti pri PostgreSQL prav tako dobra optimizacija za transakcije.

Meritev 4:

Rezultati tabele predstavljajo rezultate meritev obsega in zmogljivosti skaliranj. Podatki o porabi pomnilnika in času izvajanja nam pomagajo razumeti, kako se PostgreSQL obnaša pri različnih obremenitvah v nasprotju z MongoDB. Razlaga tabele:

- **Število vstavkov:** To je število poizvedb, ki so bile vstavljene v zbirko podatkov PostgreSQL v vsaki posamezni merilni iteraciji.
- **Poraba pomnilnika (MB):** Ta stolpec prikazuje spremembo v porabi pomnilnika med izvajanjem vstavkov. Vrednosti so izražene v megabajtih. Pozitivne vrednosti kažejo na povečanje porabe pomnilnika, medtem ko negativne vrednosti kažejo na zmanjšanje porabe pomnilnika.
- **Čas izvajanja (sekunde):** Ta stolpec prikazuje skupni čas, potreben za izvajanje vstavkov v zbirko podatkov PostgerSQL v vsaki posamezni merilni iteraciji.

Tabela 12: Rezultat meritve 4 za PostgreSQL

Število vstavljanj	Poraba pomnilnika (MB)	Čas izvajanja (sekunde)
1000	2.123	0.245
5000	6.133	2.451
10000	-1.345	4.122
50000	5.164	20.450
100000	16.123	35.815

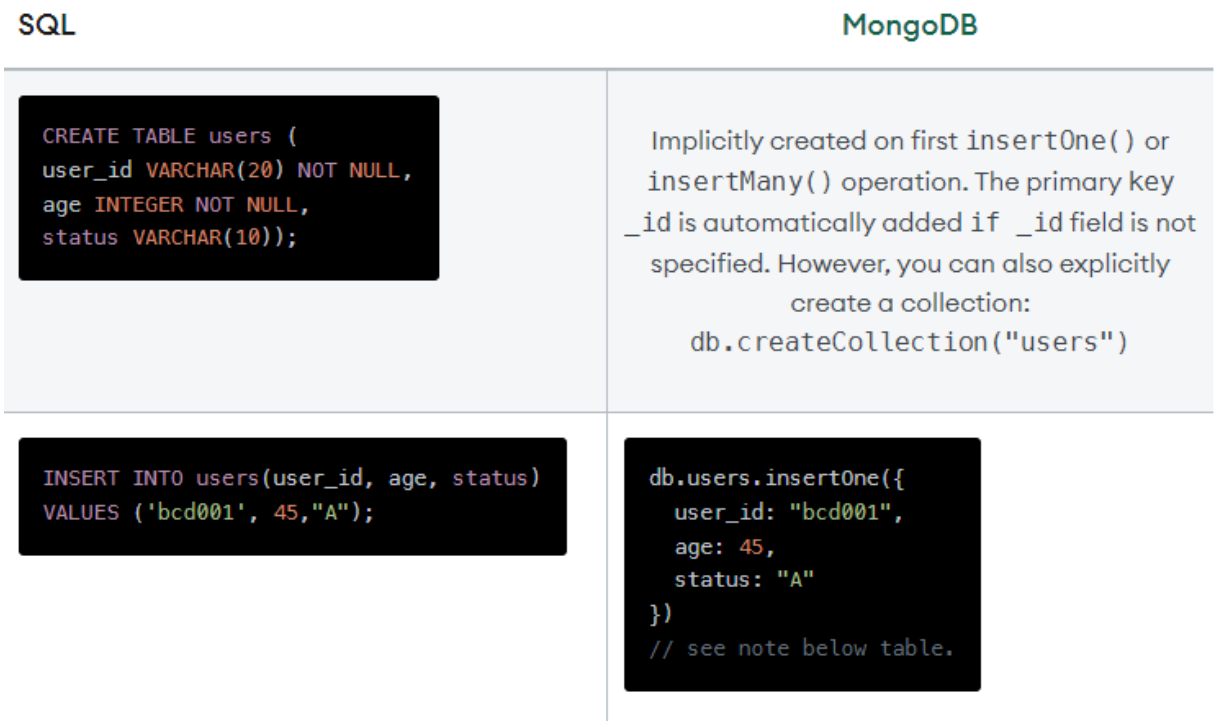
(Vir: lastni vir)

Pri rezultatih tabele lahko iz stolpca porabe pomnilnika razberemo, da PostgreSQL porabi več pomnilnika pri vstavljanju kot MongoDB. Vedeti moramo, da sem pri vstavljanju uporabil tudi program DataGrip, že to vpliva na porabo pomnilnika. Poudariti je treba, da imata oba tudi različno alokacijo spomina in postopek upravljanja pomnilnika. Rezultati so odvisni tudi od konfiguracije podatkovnih baz in predvsem strojne in programske opreme.

4.2 *Primerjava uporabe MongoDB in PostgreSQL*

Izbira med MongoDB in PostgreSQL v moderni aplikaciji je odvisna od specifičnih potreb projekta. MongoDB ponuja hitrost razvoja in prilagodljivost sheme, medtem ko PostgreSQL zagotavlja doslednost, podporo za kompleksne transakcije in bogat SQL jezični vmesnik. Razvijalci bi morali upoštevati zahteve aplikacije, pričakovano obremenitev, potrebe po transakcijah in druge dejavnike, ki vplivajo na izbiro podatkovne baze. Prav tako uporaba relacijskih podatkovnih baz zahteva programsko opremo, za katero se po navadi potrebuje licenca, kot v mojem primeru uporaba programa DataGrip, za katerega imam študentsko začasno licenco.

Tako PostgreSQL kot MongoDB imata bogat jezik poizvedb. Spodaj so nekateri primeri SQL stavkov in kakšna je njihova ekvivalenca v MongoDB. Bolj obsežen seznam izjav je na voljo v dokumentaciji MongoDB. V MongoDB imamo tudi prednost vizualnega vmesnikov, kar pomeni, da dejansko ne potrebujemo query jezika kot pri relacijskih podatkovnih bazah (MongoDB, brez datuma).



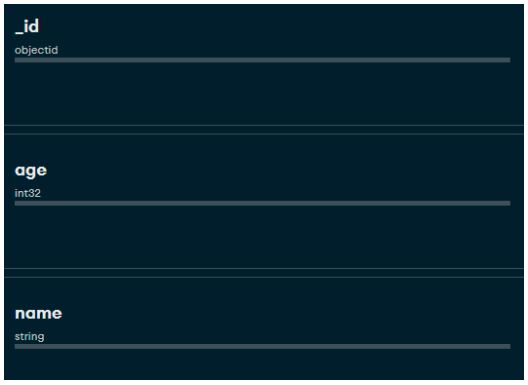
Slika 10: Primerjava poizvedovalnega jezika
(Vir: https://www.mongodb.com/compare/mongodb-postgresql)

4.2.1 Struktura in shema podatkovne baze

Podatki morajo biti v podatkovni bazi dosledni in berljivi, zato je potrebno definirati shemo. Shema podatkovne baze vključuje podatkovni tip, kateri določa, kako se različni podatki shranjujejo, obdelujejo in omejujejo. Uporabil sem dva podatkovna tipa, spremenljiv značaj (ang. Variable Character), kateri se uporablja za besede, besedila ali opise v mojem primeru imena. Drugi podatkovni tip je celo število (ang. Integer), ki je uporabljen za starost uporabnika. V podatkovni bazi se za tipe uporabijo kratice, v tem primeru VARCHAR in INT.

Struktura tabele bo vsebovala:

- Edinstveni ključ je atribut, ki zagotavlja edinstveno identifikacijo podatka.
- Ime (VARCHAR) - definirano je ime uporabnika.
- Starost (INT) - definirana je starost uporabnika.



Slika 21: Struktura baze v MongoDB
(Vir: lastni vir)

V MongoDB sheme ni potrebno definirati, preden vstavimo podatke, saj se shema lahko analizira, glede na strukturo JSON dokumenta posameznega dokumenta. Kadar pa delamo z relacijskimi podatkovnimi bazami, pa se struktura tabele definira s poizvedbo CREATE TABLE.

```
CREATE TABLE uporabniki (
    id SERIAL PRIMARY KEY,
    ime VARCHAR(25),
    starost INT
);
```

Slika 22: Poizvedba za kreiranje tabele

(Vir: lastni vir)

4.3 Identifikacija priljubljenih podatkovnih baz za obdelavo Big Data

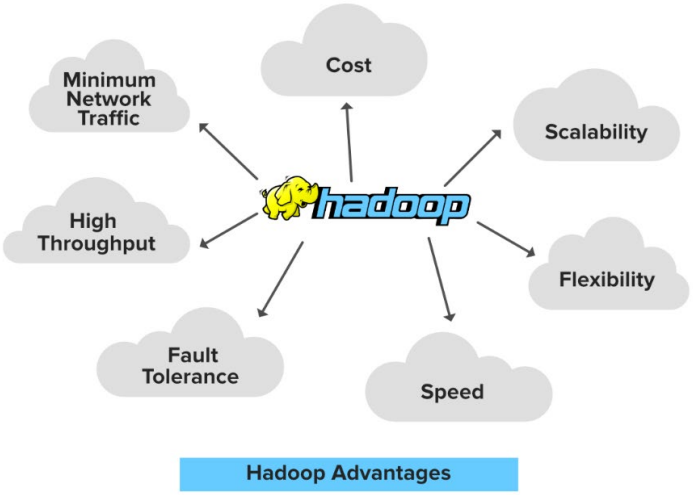
Pri obdelavi velikih količin podatkov se pojavlja potreba po različnih prilagodljivih in zmogljivih podatkovnih bazah, ki lahko učinkovito obvladujejo zahtevne obdelovalne procese. Danes zraven MongoDB in PostgreSQL obstaja več drugih priljubljenih podatkovnih baz, ki so se izkazale kot uporabne pri obdelavi Big Data. Pogledali si bomo dve priljubljeni, to sta Hadoop Hbase in Apache Cassandra.

4.3.1 Hadoop HBase

Hadoop HBase je še ena možnost za obdelavo velikih količin podatkov, še posebej za hranjenje časovnih vrst in redko posodobljenih podatkov. To je primerno za scenarije, kjer je potrebno ohraniti zgodovino sprememb podatkov in izvajati časovno občutljive poizvedbe. Vendar pa je HBase lahko manj primeren za kompleksne poizvedbe, ki jih omogočata MongoDB in PostgreSQL. Poleg tega zahteva nekoliko več vzdrževanja in konfiguracije kot nekatere druge možnosti. Ima odprtokodni programski okvir, ki se uporablja za shranjevanje in obdelavo velikih količin podatkov v porazdeljenem računalniškem okolju. Namenjen je obvladovanju velikih podatkov in temelji na programskem modelu MapReduce, ki omogoča vzporedno obdelavo obsežnih naborov podatkov (GeeksForGeeks, brez datuma).

Ima dve glavni komponenti, kateri pripomoreta k sposobnosti strojne opreme:

- **HDFS (Hadoop Distributed File System):** To je shranjevalna komponenta Hadoopa, ki omogoča shranjevanje velikih količin podatkov preko več računalnikov. Zasnovana je za delovanje s cenovno ugodno strojno opremo, kar jo naredi stroškovno učinkovito.
- **YARN (Yet Another Resource Negotiator):** To je komponenta upravljanja virov v sistemu Hadoop, ki upravlja z dodeljevanjem virov (kot so CPU in pomnilnik) za obdelavo podatkov, shranjenih v HDFS.



Slika 23: Hadoop Advantages

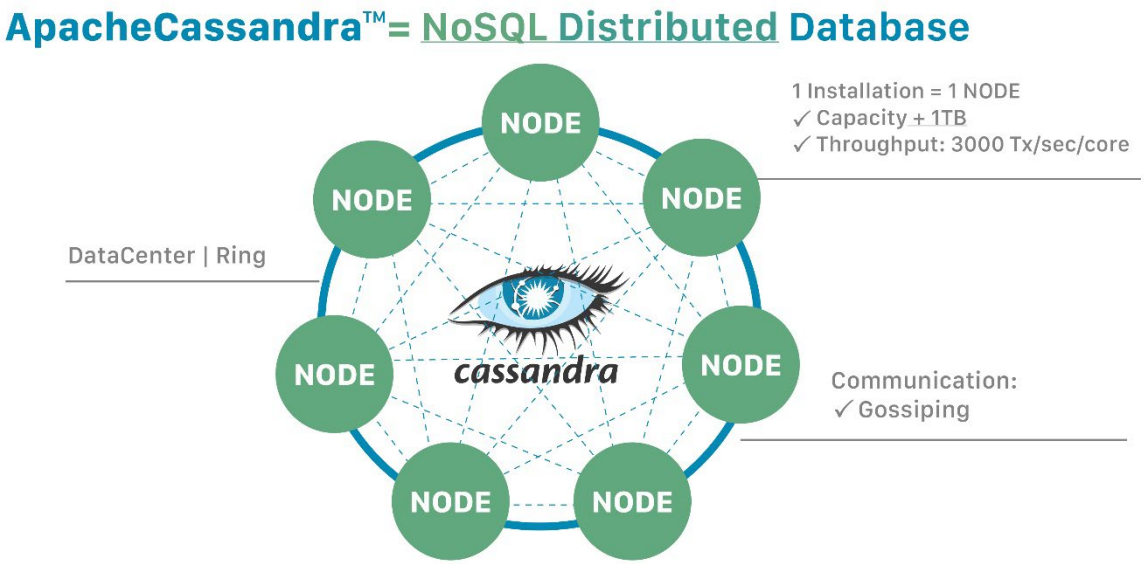
(Vir: https://www.geeksforgeeks.org/hadoop-pros-and-cons/)

Hadoop ima več prednosti, ki ga naredijo kot priljubljeno izbiro za obdelavo velikih podatkov:

- **Razširljivost:** Hadoop se lahko enostavno razširi za obdelavo velikih količin podatkov z dodajanjem več vozlišč v grozd.
- **Cenovno učinkovit:** Hadoop je zasnovan za delo z običajno strojno opremo, kar ga naredi za cenovno učinkovito možnost za shranjevanje in obdelavo velikih količin podatkov, kar je v prvi vrsti primarno za manjša podjetja.
- **Odpornost na napake:** Porazdeljena arhitektura Hadoopa zagotavlja vgrajeno odpornost na napake, kar pomeni, da se lahko podatki še vedno obdelujejo, če eno vozlišče v grozdu odpove, saj se obdelava prenese na druga vozlišča.
- **Prilagodljivost:** Hadoop lahko obdeluje strukturirane, delno strukturirane in nestrukturirane podatke, kar ga naredi za vsestransko izbiro za širok spekter scenarijev velikih podatkov.
- **Odprtokodnost**: Hadoop je odprtokodna programska oprema, kar pomeni, da je brezplačen za uporabo in spreminjanje.
- **Integracija:** Hadoop je zasnovan za delo z drugimi tehnologijami za obdelavo velikih podatkov, kot so Spark, Storm in Flink, kar omogoča integracijo z raznovrstnimi orodji za obdelavo in analizo podatkov.

4.3.2 Apache Cassandra

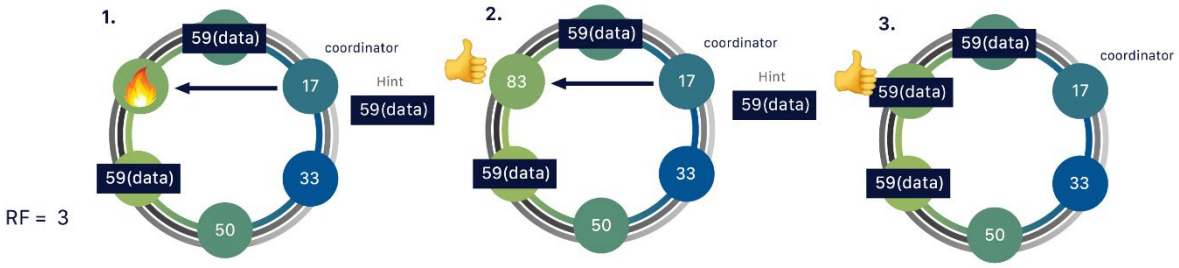
Cassandra je porazdeljena podatkovna baza tipa NoSQL, ki omogoča dinamično razširjanje podatkovnih baz. Razvita je bila pri podjetju Facebook, najprej za poganjanje funkcij iskanja v nabiralniku, kasneje izdana kot odprtokodni projekt na platformi Google Code. Temelji na vozliščih, kar omogoča horizontalno razširljivost z uporabo cenene strojne opreme. Za povečanje zmogljivosti ali pretoka podatkov preprosto podvojite število vozlišč. To omogoča dinamično prilagajanje zmogljivosti brez nedelovanja sistema. Vozlišča se med seboj pogovarjajo s pomočjo protokola gossip, kar omogoča izolirano prilagajanje prepustnosti branja ali pisanja (Apache Cassandra, brez datuma).



Slika 24: Delovanje vozlišč

(Vir: https://cassandra.apache.org/)

Podatek lahko podvojimo na več vozlišč, kar zagotavlja zanesljivost in toleranco napak. V primeru, da pride do napake na enem od vozlišč, ne pride do izgube podatkov. Cassandra podpira koncept faktorja replikacije (RF), ki opisuje, koliko kopij podatkov naj obstaja v podatkovni bazi. Če povečamo faktor replikacije na tri (RF = 3), morajo biti podatki shranjeni tudi na drugi repliki in tretji.



Slika 25: Replikacija podatkov na vozliščih

(Vir: https://cassandra.apache.org/_/cassandra-basics.html)

Eden od razlogov za priljubljenost Cassandre je, da omogoča razvijalcem dinamično prilagajanje njihovih podatkovnih baz z uporabo standardne strojne opreme brez ustavljanja sistema. Lahko povečamo zmogljivost, ko je to potrebno, in jo zmanjšamo, če zahtevajo aplikacijske potrebe takšen pristop. Pri podatkovnih bazah, kot so Oracle in MySQL, je potrebna za razširjevanje baze in omogočanje shranjevanja večje kapacitete podatkov zahtevna dodaja procesorske moči, RAM-a ali hitrejših diskov.

Ima vodoravno arhitekturo, katera se imenuje tudi širitev. Zato s cenejšo strojno opremo omogoča uporabnikom in podjetjem, da podvojijo zmogljivost, pretok podatkov ali število vozlišč. Če se potrebuje več moči, se doda več vozlišč, to pride prav pri večjih podjetjih, ki uporabljajo velike podatke v svojih aplikacijah ali storitvah.

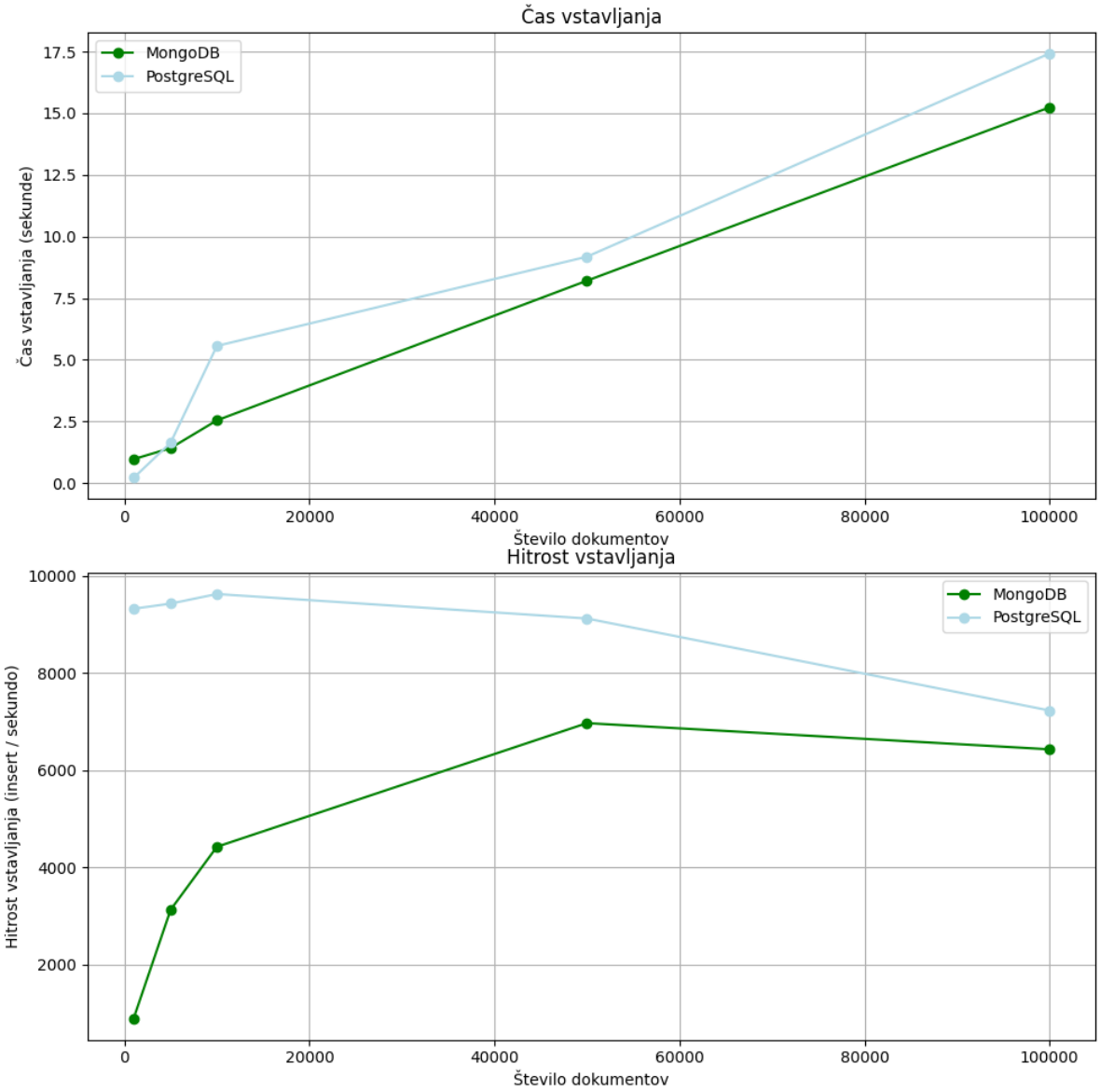
Nekaj večjih podjetij, ki danes uporabljajo Cassandro:

- **Uber:** Cassandro je ključen del infrastrukture Uberja, saj omogoča delovanje kritičnih komponent, kot so analitika v realnem času, spremljanje in upravljanje podatkov uporabnikov, kar omogoča nemoteno delovanje na globalni ravni.
- **eBay:** Cassandro ima ključno vlogo v podatkovni infrastrukturi eBaya, saj napaja sisteme za upravljanje kataloga, uporabniške preference in analitiko v realnem času za izboljšanje izkušnje e-trgovine.
- **Netflix:** Priljubljena storitev pretakanja vsebin se zanaša na Cassandro za upravljanje in obdelavo obsežnih količin podatkov, omogočajoč personalizirane priporočitve, dostavo vsebin in analitiko (Rahman, 2023).

5 RAZPRAVA

5.1 Interpretacija rezultatov

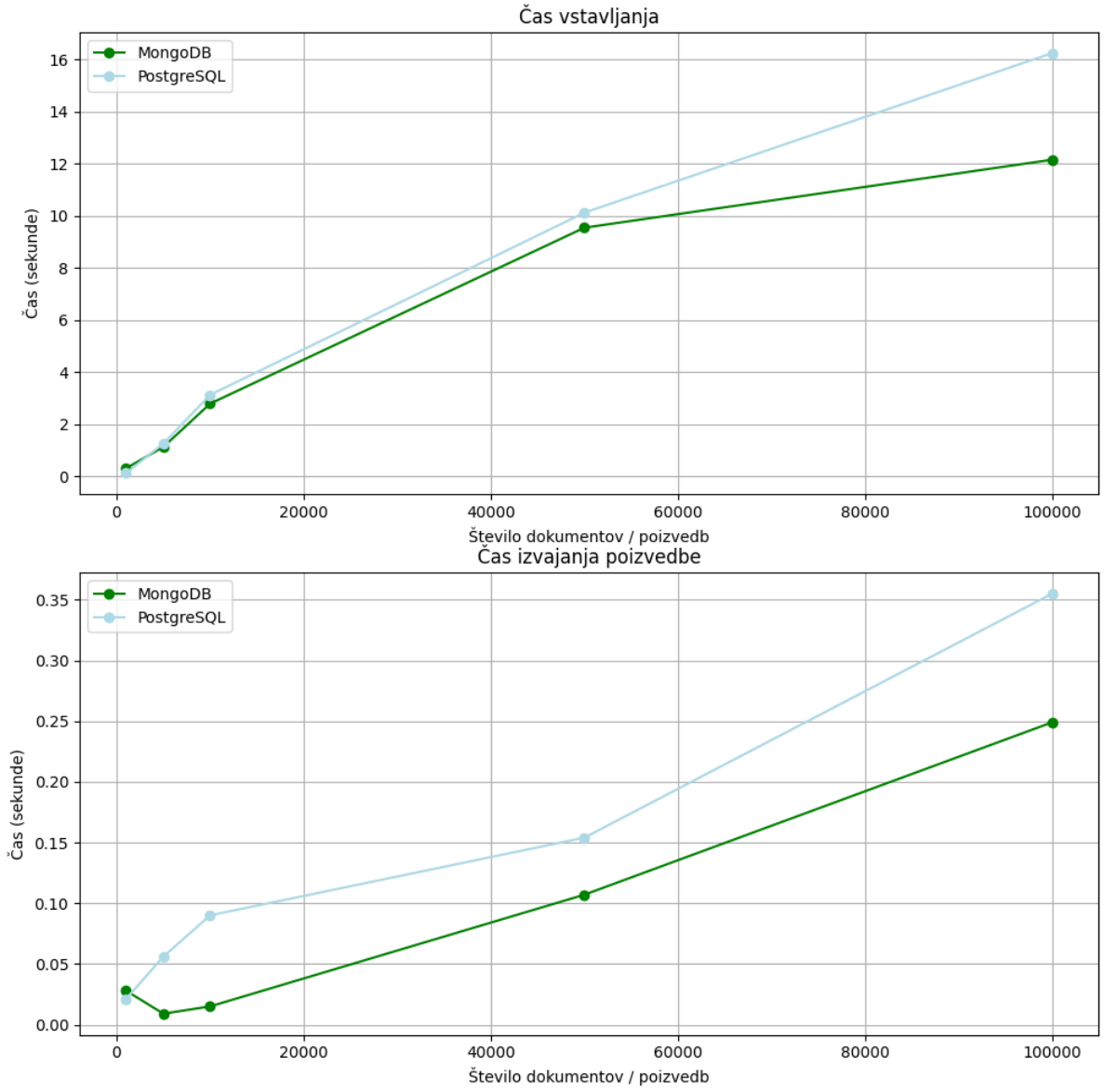
Grafični prikaz rezultatov meritve 1, »Meritev hitrosti časa vstavljanja podatkov v podatkovno bazo «, kjer smo merili hitrost vstavljanja podatkov na sekundo. Zelena črta prikazuje rezultate MongoDB, svetlo morda pa rezultate PostgreSQL. Rezultati so razloženi v poglavju 4.1.2 in 4.1.3.



Slika 26: Grafični prikaz rezultatov meritve 1

(Vir: Lastni vir)

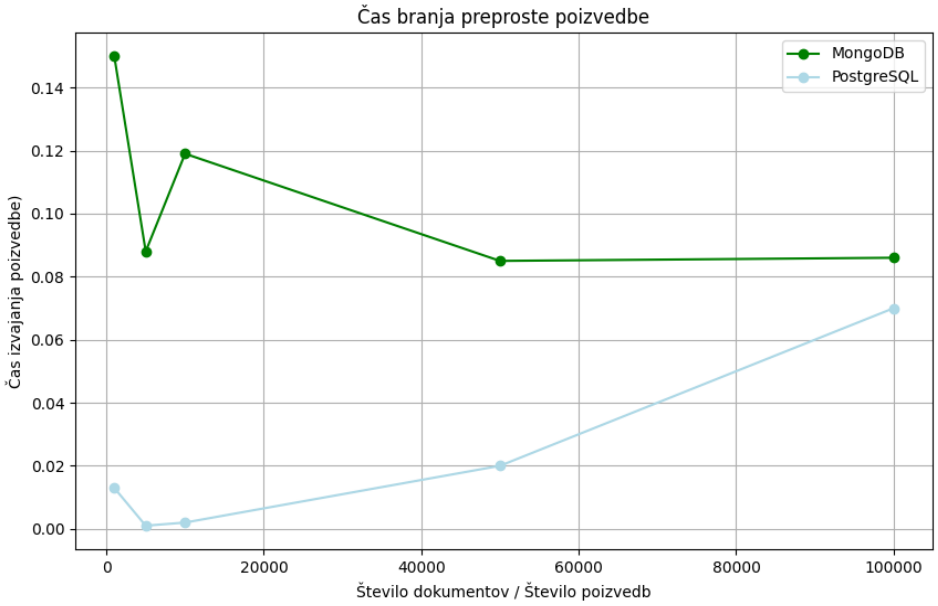
Grafični prikaz rezultatov meritve 2, »Meritev učinkovitosti poizvedb nad podatki iz podatkovne baze«, kjer smo merili hitrost vstavljanja podatkov. Zelena črta prikazuje rezultate MongoDB, svetlo morda pa rezultate PostgreSQL. Rezultati so razloženi v poglavju 4.1.2 in 4.1.3.



Slika 27: Grafični prikaz rezultatov meritve 2

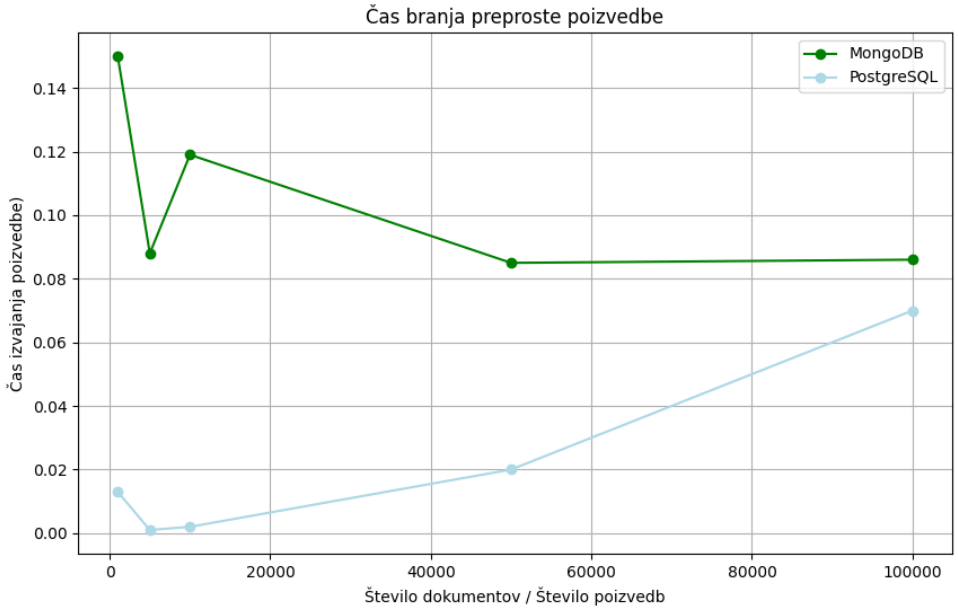
(Vir: Lastni vir)

Grafični prikaz rezultatov meritve 2, »Meritev učinkovitosti poizvedb nad podatki iz podatkovne baze«, kjer smo merili hitrost branja podatkov. Najprej smo izvedli preprosto branje podatkov, nato pa v poizvedbo dodali parameter, kjer poišče uporabnike stare nad 18 let. Rezultati so razloženi v poglavju 4.1.2 in 4.1.3.



Slika 28: Grafični prikaz rezultatov meritve 2 poizvedba_1

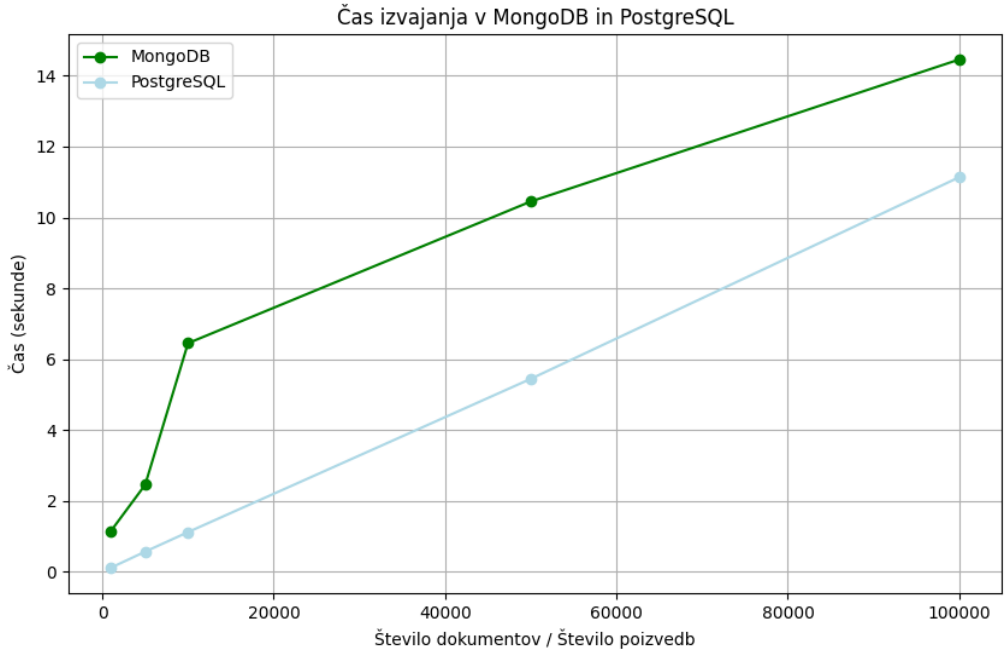
(Vir: Lastni vir)



Slika 29: Grafični prikaz rezultatov meritve 2 poizvedba_2

(Vir: Lastni vir)

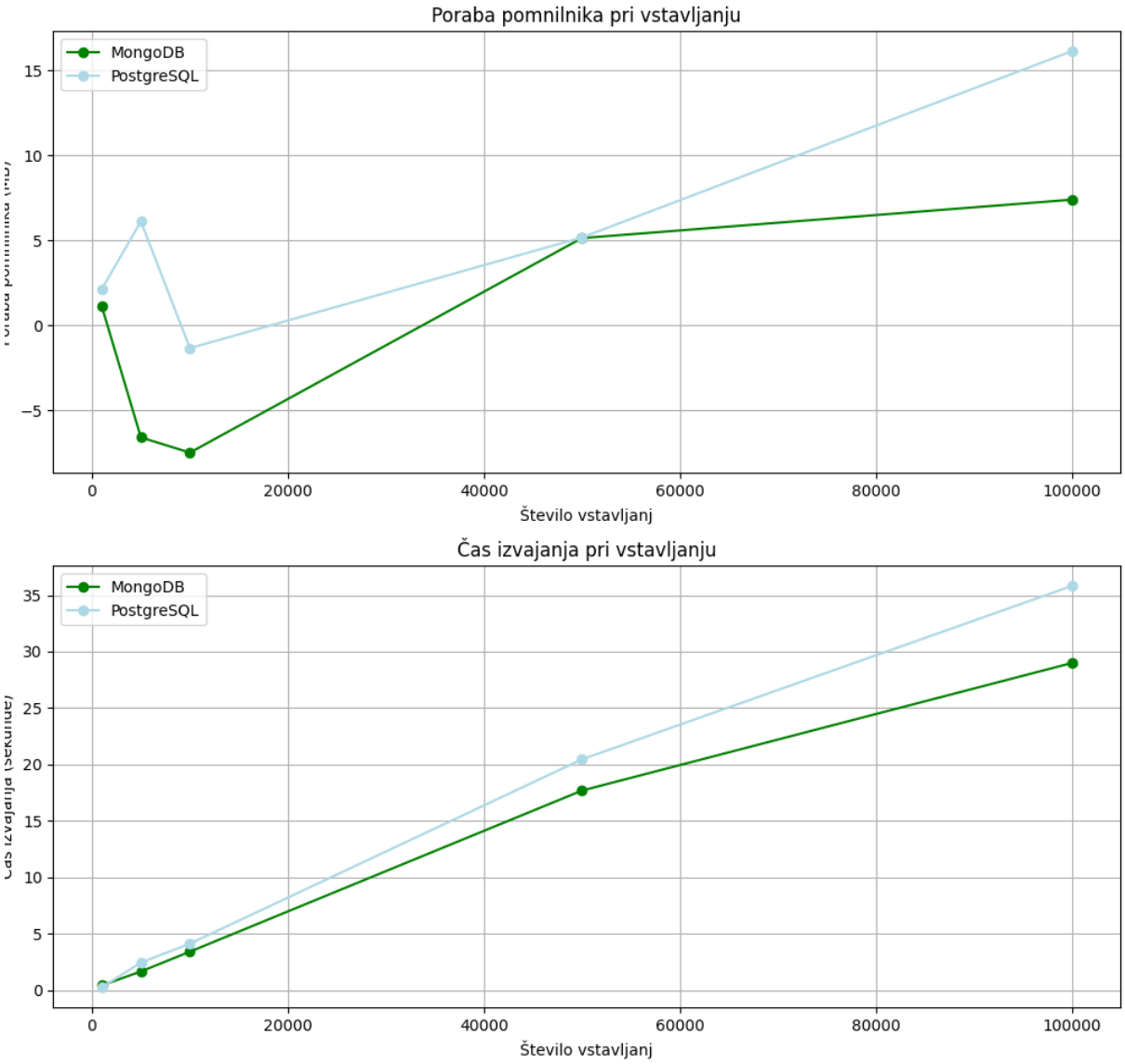
Grafični prikaz rezultatov meritve 3 »Primerjava učinkovitosti transakcijskega vstavljanja«, kjer smo merili hitrost, učinkovitost in doslednost transakcij. Zelena črta prikazuje rezultate MongoDB, svetlo morda pa rezultate PostgreSQL. Rezultati so razloženi v poglavju 4.1.2 in 4.1.3.



Slika 30: Grafični prikaz rezultatov meritve 3

(Vir: Lastni vir)

Grafični prikaz rezultatov meritve 4 »Meritev obsega in zmogljivosti skaliranja ter poraba pomnilnika«, kjer smo merili obseg in porabo pomnilnika. Zelena črta prikazuje rezultate MongoDB, svetlo morda pa rezultate PostgreSQL. Rezultati so razloženi v poglavju 4.1.2 in 4.1.3.



Slika 31: Grafični prikaz rezultatov meritve 4

(Vir: Lastni vir)

5.2 Razprava o ugotovitvah in njihovem pomenu

V poglavju bomo analizirali rezultate meritev, opravljenih med primerjavo zmogljivosti med sistemoma za upravljanje zbirke podatkov MongoDB in PostgreSQL. Združene ugotovitve nam bodo pomagale bolje razumeti zmogljivost obeh sistemov in njihovo primerljivost v različnih pogojih uporabe.

Meritev 1:

- **Ni bistvene razlike pri vstavljanju:** Pri prvi meritvi nismo opazili bistvene razlike med MongoDB in PostgreSQL v času vstavljanja dokumentov. Oba sistema sta se obnesla podobno do neke točke, razlika se začne pojavljati pri večjem številu, kar bi se za količino podatkov, kot je terabajt, z bolj kompleksno shemo in več tabelami še bolj razlikoval.
- **Padec hitrosti vstavljanja dokumentov:** Opazili smo, da se hitrost vstavljanja dokumentov na sekundo zmanjšuje, ko povečujemo število vstavljenih dokumentov. To lahko kaže na omejitve sistema ali pa način, kako se sistem obnaša pri večjih obremenitvah.
- **Povečanje časa poizvedbe z večjim številom dokumentov:** Pri obeh sistemih smo opazili, da se čas izvajanja poizvedb povečuje z večjim številom dokumentov. To je pričakovano, saj več dokumentov zahteva več časa za obdelavo.
- **MongoDB je hitrejši pri večjem številu vstavljanj:** Pri večjem številu vstavljanj dokumentov, na primer okoli 10000, je vidna razlika, da je MongoDB hitrejši od PostgreSQL. To lahko pomeni, da je MongoDB bolje optimiziran za takšne obremenitve ali pa ima boljše lastnosti za delo s tovrstnimi podatki.

Meritev 2:

V drugi meritvi smo se osredotočili na učinkovitost poizvedb nad podatki v primerjavi z meritvijo vstavljanja dokumentov. Rezultati so pokazali naslednje:

- **Čas vstavljanja dokumentov:** Podobno kot pri prvi meritvi smo tudi tu opazili, da je MongoDB hitrejši pri vstavljanju dokumentov v primerjavi s PostgreSQL. Ta razlika v hitrosti se je povečala s povečanjem števila dokumentov, kar kaže na učinkovitost MongoDB-ja pri obvladovanju večjih obremenitev.

- **Hitrost branja podatkov:** Pri branju podatkov iz podatkovne baze za enako število dokumentov smo opazili, da se MongoDB in PostgreSQL odzoveta skoraj enako hitro. Razlika med njima je bila le v milisekundah, kar kaže na podobno učinkovitost obeh sistemov pri branju podatkov.
- **Poizvedba s pogojem:** Pri izvedbi poizvedbe, ki je vključevala pogoj za iskanje uporabnikov nad 18 let, smo opazili, da se je PostgreSQL odzval rahlo hitreje kot MongoDB. Ta razlika v hitrosti izvajanja poizvedb s pogojem lahko nakazuje na razlike v optimizaciji in izvajanju poizvedb med obema sistemoma. Kot smo že prej omenili, je lahko razlika tudi v podatkih, kateri so bili naključno generirani.

Skupaj ti rezultati ponujajo vpogled v učinkovitost in zmogljivost MongoDB in PostgreSQL pri različnih vrstah poizvedb in obremenitvah podatkovne baze. Medtem ko je MongoDB izkazal prednost pri hitrosti vstavljanja dokumentov, je bila hitrost branja podatkov med obema sistemoma podobna, medtem ko je PostgreSQL pokazal nekoliko boljšo učinkovitost pri izvajanju poizvedb s pogojem.

Meritev 3:

V tretji meritvi smo preučevali transakcijsko vstavljanje dokumentov v podatkovno bazo, pri čemer smo izvedli pet različnih vstavljanj za vsako število dokumentov: 1000, 5000, 10000, 50000 in 100000. Rezultati so pokazali naslednje, da se je PostgreSQL izkazal za boljšega od MongoDB pri transakcijskem vstavljanju dokumentov.

To pomeni, da je PostgreSQL bolje obvladoval večje obremenitve transakcijskega vstavljanja v primerjavi z MongoDB. Razlika v učinkovitosti med obema sistemoma se je še posebej pokazala pri večjih številih vstavljanj, kjer je bil PostgreSQL hitrejši in bolj zanesljiv pri izvajanju transakcij.

Ti rezultati kažejo na prednost PostgreSQL-a pri transakcijskih operacijah, kar je lahko ključnega pomena za aplikacije, ki zahtevajo visoko stopnjo doslednosti in zanesljivosti pri manipulaciji s podatki.

Meritev 4:

V zadnji meritvi, ki je vključevala merjenje porabe pomnilnika, smo opazili naslednje ugotovitve:

- **Poraba pomnilnika:** PostgreSQL je pokazal večjo porabo pomnilnika v primerjavi z MongoDB-jem. To kaže na to, da relacijski model podatkov in zapletene notranje strukture PostgreSQL-a zahtevajo več pomnilniških virov v primerjavi z dokumentnim modelom MongoDB-ja.
- **Naraščanje porabe pomnilnika:** Ugotovili smo, da se poraba pomnilnika povečuje s povečevanjem števila dokumentov v obeh podatkovnih bazah. To je pričakovano, saj večji obseg podatkov zahteva več pomnilniških virov za shranjevanje, obdelavo in izvajanje poizvedb.
- **Stabilnost časa izvajanja:** Kljub povečevanju porabe pomnilnika smo opazili, da se čas izvajanja poizvedb še vedno stabilno povečuje s povečevanjem števila dokumentov v obeh podatkovnih bazah. To kaže na dosledno delovanje obeh sistemov, kljub povečanju obsega podatkov.
- **Skupna učinkovitost:** Medtem ko PostgreSQL porabi več pomnilnika, je ključno upoštevati, da so ostale ključne značilnosti, kot je čas izvajanja poizvedb, še vedno primerljive med obema podatkovnima bazama. To poudarja pomembnost upoštevanja različnih dejavnikov, ko se odločamo med različnimi podatkovnimi bazami, in potrebo po prilagajanju konfiguracij za dosego optimalne učinkovitosti v določenem scenariju. PostgreSQL uporablja relacijski model podatkov, ki zahteva večjo strukturo in organizacijo podatkov v primerjavi z MongoDB-jem, ki uporablja dokumentni model. Zaradi tega lahko PostgreSQL zahteva več pomnilniških virov za shranjevanje in obdelavo podatkov. Enako velja glede arhitekture - PostgreSQL ima dosti starejšo arhitekturo od MongoDB-ja.

6 ZAKLJUČEK

Hipoteza 1: MongoDB bo ponudil učinkovitejšo obdelavo velikih količin podatkov v primerjavi s PostgreSQL.

Na podlagi ugotovitev lahko delno potrdimo hipotezo. Med vstavljanjem podatkov je MongoDB ponudil učinkovitejšo obdelavo velikih količin podatkov v primerjavi s PostgreSQL. Vendar pa glede na čas izvajanja poizvedb ni bilo znatnih razlik med obema podatkovnima bazama, razen manjših variacij, ki niso bile dovolj velike, da bi enoznačno potrdile hipotezo. Poleg tega je bila poraba pomnilnika večja v PostgreSQL v primerjavi z MongoDB, kar lahko vpliva na celotno učinkovitost sistema. Pri transakcijskem vstavljanju se MongoDB ni izkazal za hitrejšega od PostgreSQL, kar je pomembno upoštevati pri analizi učinkovitosti obeh sistemov. Zato bi lahko rekli, da se hipoteza delno drži.

Hipoteza 2: MongoDB se pogosto uporablja v primerjavi s Postgre SQL v določenih okoljih, kot so aplikacije, ki zahtevajo visoko skalabilnost in hiter razvoj.

Glede na ugotovitve iz naše analize lahko trdimo, da je MongoDB v določenih okoljih, kot so aplikacije, ki zahtevajo visoko skalabilnost in hiter razvoj, lahko bolj privlačna izbira kot PostgreSQL. Razlogi za to vključujejo hitrejše vstavljanje podatkov v MongoDB, njegovo učinkovito obdelavo velikih količin podatkov ter zmožnost prilagajanja spremembam v zahtevah aplikacije brez potrebe po predhodni shemi. Zraven tega je MongoDB znan po svoji sposobnosti skaliranja vodoravno, kar pomeni, da lahko enostavno dodajamo več strežnikov za povečanje zmogljivosti, kar je ključnega pomena za aplikacije z visokim obsegom podatkov. Glede na to, da je danes večina aplikacij takšnih, da potrebujejo hitro in denarnici prijazno rešitev za širjenje in obdelovanje podatkov, se mi zdi, da je MongoDB boljša izbira. Zato bi lahko rekel, da hipoteza drži.

Hipoteza 3: Pri obdelavi velikih količin podatkov bo MongoDB ponudil boljše rezultate v primerjave s Postgre SQL glede na hitrost in učinkovitost.

Ugotovili smo, da je MongoDB v nekaterih vidikih učinkovitejši pri obdelavi velikih količin podatkov v primerjavi s PostgreSQL. Na primer, hitrost vstavljanja dokumentov v MongoDB je pogosto hitrejša kot v PostgreSQL, še posebej pri večjih količinah podatkov. Prav tako se je MongoDB odzval rahlo hitreje pri branju podatkov v primerjavi s PostgreSQL, zanemarljiva razlika pa je bila opažena pri poizvedbah nad velikimi količinami dokumentov. Kljub temu pa je treba opozoriti, da se PostgreSQL v nekaterih primerih izkaže za primerljivo ali celo bolj učinkovito izbiro pri obdelavi velikih količin podatkov. Na primer pri transakcijskem vstavljanju večjega števila dokumentov se je PostgreSQL izkazal za bolj učinkovitega od MongoDB, ampak če pogledamo večjo sliko, je bolj pomembna hitrost in obnašanje pri navadnem vstavljanju podatkov z manjšo porabo pomnilnika, tako da hipoteza drži.

Hipoteza 4: Najbolj priljubljene baze za obdelavo Big Data bodo tiste, ki omogočajo visoko zmogljivost, skaliranje in podporo za analizo nestrukturiranih podatkov.

Najbolj priljubljene baze za obdelavo velikih podatkov so tiste, ki omogočajo visoko zmogljivost, skaliranje in podporo za analizo nestrukturiranih podatkov. MongoDB je ena od takih baz, saj ponuja visoko skalabilnost, hitrost ter podporo za shranjevanje in obdelavo nestrukturiranih podatkov, kar je v skladu z zahtevami za obdelavo Big Data. Vendar pa ni edina. V raziskavi smo pogledali tudi druge baze, kot je na primer Apache Cassandra ali Hbase Hadoop. Obe sta med zelo priljubljenimi za obdelavo velikih podatkov.

Tako lahko sklepamo, da so najbolj priljubljene baze za obdelavo velikih podatkov tiste, ki izpolnjujejo zahteve po zmogljivosti, skaliranju in podpori za analizo strukturiranih in nestrukturiranih podatkov, pri čemer je MongoDB ena od možnih izbir, vendar ne edina.

Meritve, predstavljene v tem poročilu, so bile izvedene na mojem računalniku z lastnimi specifikacijami, kar pomeni, da bi na računalniku z boljšo strojno opremo ali slabšo rezultati lahko bili drugačni.

Izvajanje meritve je bilo izvedeno s pomočjo programskega jezika Python in uporabe ustreznih knjižnic za povezavo in delo s podatkovnimi bazami MongoDB ter PostgreSQL. Pomembno je poudariti, da lahko rezultati meritev variirajo glede na različna okolja ter konfiguracijo podatkovnih baz. Kot sem omenil pri razpravi rezultatov meritve 2, shema podatkovnih baz uporabljenih za meritev ni kompleksna, kar pomeni, da bi lahko bili rezultati vstavljanja v drugem scenariju drugačni.

Meritve rezultatov so tudi rezultat prej definiranega števila transakcij in števila poizvedb oziroma dokumentov, kar v pravem življenju ni vedno tako. Zato je priporočljivo, da se meritve izvajajo in primerjajo v kontekstu specifičnega projekta in okolja, ki ga želimo oceniti.

7 VIRI

7.1 Seznam uporabljene literature

Amazon Web Services. (brez datuma). *What is a document database*. Pridobljeno iz AWS: https://aws.amazon.com/nosql/document/

Apache Cassandra. (brez datuma). *Cassandra Basics*. Pridobljeno iz Apache Cassandra: https://cassandra.apache.org/_/cassandra-basics.html

Bučar, F. (2012). *Rojstvo države.* Ljubljana: Didakta.

Chapple, M. (24. Julij 2023). *LifeWire*. Pridobljeno iz LifeWire: https://www.lifewire.com/abandoning-acid-in-favor-of-base-1019674

Chris Eaton, D. D. (2012). *Understading Big Data.*

Coursera. (20. November 2023). *What Is Python Used For? A Beginner's Guide*. Pridobljeno iz Coursera: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

Data Forest. (16. Marec 2023). *Top 20 Highly Effective Use Cases of Big Data Analytics for Businesses in 2024*. Pridobljeno iz Data Forest: https://dataforest.ai/blog/big-data-analytics-use-cases

Dawar, H. (24. Maj 2020). *Can RDBMS handle Big Data?* Pridobljeno iz Medium: https://medium.com/analytics-vidhya/can-rdbms-handle-big-data-b583c1584d5f

Dhruv, S. (1. Januar 2024). *PostgreSQL Advantages and Disadvantages*. Pridobljeno iz AALPHA: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/

GeeksForGeeks. (brez datuma). *Introduction to Hadoop*. Pridobljeno iz GeeksForGeeks: https://www.geeksforgeeks.org/hadoop-an-introduction/

Gillis, A. S. (brez datuma). *MongoDB*. Pridobljeno iz TechTarget: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

IBM. (brez datuma). *Relational Databases*. Pridobljeno iz IBM: https://www.ibm.com/topics/relational-databases

IBM. (brez datuma). *What is postgreSQL*. Pridobljeno iz IBM: https://www.ibm.com/topics/postgresql

JetBrains. (brez datuma). *What is DataGrip*. Pridobljeno iz Jetbrains: https://www.jetbrains.com/datagrip/

KnowledgeNile. (brez datuma). *Understanding the Pros and Cons of MongoDB*. Pridobljeno iz KnowledgeNile: https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb

Lee, Y. K. (Avgust 2023). *How big data made ChatGPT possible*. Pridobljeno iz Linkedin: https://www.linkedin.com/pulse/how-big-data-made-chatgpt-possible-non-technical-audience-lee/

MongoDB. (brez datuma). *Big Data Explained*. Pridobljeno iz MongoDB: https://www.mongodb.com/basics/big-data-explained

MongoDB. (brez datuma). *Comparing MongoDB vs PostgreSQL*. Pridobljeno iz MongoDB: https://www.mongodb.com/compare/mongodb-postgresql

MongoDB. (brez datuma). *Relational vs non-relational databases*. Pridobljeno iz MongoDB: https://www.mongodb.com/compare/relational-vs-non-relational-databases

Oracle. (brez datuma). *What is a relational database*. Pridobljeno iz Oracle: https://www.oracle.com/database/what-is-a-relational-database/

Papiernik, M. (29. Julij 2021). *How to use MongoDB Shell*. Pridobljeno iz DigitalOcean: https://www.digitalocean.com/community/tutorials/how-to-use-the-mongodb-shell

Pedamkar, P. (15. Marec 2023). *MongoDB Tools*. Pridobljeno iz EDUCBA: https://www.educba.com/mongodb-tools/

Punreddy, S. (19. Julij 2023). *Understanding the Fundamentals of PostgreSQL Architecture*. Pridobljeno iz InstaClustr: https://www.instaclustr.com/blog/postgresql-architecture/

Python. (brez datuma). *Python documentation*. Pridobljeno iz Python: https://docs.python.org/3/

Rahman, W. (4. Julij 2023). *Top 8 reasons to use Apache Cassandra Database in 2023*. Pridobljeno iz Bootcam UXDesign: https://bootcamp.uxdesign.cc/top-8-reasons-to-use-apache-cassandra-database-in-2023-e02b9d9bdb62

Scylladb. (brez datuma). *NoSQL Design Principles*. Pridobljeno iz Scylladb: https://www.scylladb.com/glossary/nosql-design-principles/

Smallcombe, M. (22. Februar 2024). *tructured vs Unstructured Data:*. Pridobljeno iz Integrate: https://www.integrate.io/blog/structured-vs-unstructured-data-key-differences/

Taylor, E. (28. September 2023). *A comprehensive Guide on Big Data 3v*. Pridobljeno iz TheKnowledgeAcademy: https://www.theknowledgeacademy.com/blog/big-data-3v/

7.2 Viri podatkov in orodij, uporabljenih v raziskavi

- **Visual Studio Code (VS Code):** To je integrirano razvojno okolje (IDE), ki sem ga uporabil za razvoj programske kode v jeziku Python. VS Code ponuja široko paleto funkcij, ki olajšujejo pisanje, urejanje in odpravljanje napak v programski kodi.
- **Python:** Programski jezik za pisanje skript (več v poglavju 3.2.1).
- **DataGrip:** Gre za orodje za upravljanje in raziskovanje relacijskih podatkovnih baz (več v poglavju 2.3.2).
- **MongoDB:** Za shranjevanje, upravljanje in analizo podatkov (več v poglavju 2.2).
- **PostgreSQL:** Za shranjevanje, upravljanje in analizo relacijskih podatkov (več v poglavju 2.3).

Z uporabo teh orodij sem uspešno izvedel raziskavo in analizo podatkov ter implementiral potrebne algoritme za dosego ciljev raziskave.

8 PRILOGE

Vsa potrebna koda skript, katere sem kreiral za izvedbo meritev in analize, je priložena v datotekah.