# Usporedba učinkovitosti sustava za upravljanjem relacijskih i NoSQL baza podataka

**Gizdulić, Mauro**

SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Preddiplomski sveučilišni studij računarstva

Završni rad

# USPOREDBA UČINKOVITOSTI SUSTAVA ZA UPRAVLJANJEM RELACIJSKIH I NOSQL BAZA PODATAKA

Rijeka, srpanj 2022.

Mauro Gizdulić

0069087889

SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Preddiplomski sveučilišni studij računarstva

Završni rad

# USPOREDBA UČINKOVITOSTI SUSTAVA ZA UPRAVLJANJEM RELACIJSKIH I NOSQL BAZA PODATAKA

Mentor: Doc. dr. sc. Sandi Ljubić

Mauro Gizdulić

0069087889

Rijeka, srpanj 2022.

**SVEUČILIŠTE U RIJECI**
**TEHNIČKI FAKULTET**
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 3. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Baze podataka**
Grana: **2.09.02 informacijski sustavi**

# ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Mauro Gizdulić (0069087889)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Usporedba učinkovitosti sustava za upravljanjem relacijskih i NoSQL baza podataka / Efficiency Comparison of Relational and NoSQL Database Management Systems**

Opis zadatka:

Potrebno je implementirati ogledni primjer informacijskog sustava (može biti desktop aplikacija ili web aplikacija) sa sučeljem prema bazi podataka koja može biti ili relacijska ili tipa NoSQL. Unutar sustava potrebno je omogućiti testiranje potpornih baza podataka na način da se automatski može zapisivati, ažurirati, brisati i čitati različit broj zapisa, pri čemu podaci u pojedinom zapisu mogu biti različite veličine i pri čemu se koristi različita frekvencija dotične operacije. Kao izlazne metrike primarno treba pratiti vrijeme odziva baze podataka (tj. vrijeme izvršavanja operacije), a, ako je to moguće, dodatno i potrošnju računalnih resursa - opterećenje procesora i utrošak memorije. Na temelju izlaznih pokazatelja za pojedinačne konfiguracije ulaza, komparativno analizirati učinkovitost sustava za upravljanjem relacijskih i NoSQL baza podataka.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Mauro Gizdulić

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Doc. dr. sc. Sandi Ljubić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristijan Lenac

# IZJAVA O SAMOSTALNOJ IZRADBI RADA

Pri punoj svijesti i čistoj savjesti izjavljujem da sam samostalno izradio ovaj završni rad, što potvrđujem i vlastoručnim potpisom ove izjave.

Rijeka, 13.07.2022.

Mauro Gizdulić

Mauro Gizdulić

# Sadržaj

# 1 Uvod

Razvoj tehnologije u području baza podataka omogućuje više opcija za pohranu podataka, koja je od izuzetne važnosti zbog činjenice kako su podaci temelj svakog poslovanja, istraživanja, odnosno rada u cjelini. S obzirom na način pohrane podataka razlikujemo relacijske od NoSQL baza podataka. Relacijske baze podataka zasnovane su na relacijama između entiteta, odnosno tablica [1]. Svaki entitet definiran je atributima, a pristup podacima u relacijskim bazama se izvodi pomoću standardiziranog SQL jezika [2] kojim se kreiraju upiti. Alternativa relacijskim bazama podataka jesu NoSQL baze podataka [3]. Takve se baze podataka temelje na dokumetima, grafovima, stupcima ili *key/value* vrijednostima. Za pristup podacima ne koristi se standardizirani jezik upita, već se podacima pohranjenim u takvim bazama podataka pristupa izravno.

Ovaj rad bavi se usporedbom i analizom dva sustava za upravljanje bazama podataka (skraćeno SUBP), pri čemu svaki od njih adresira dotičnog predstavnika navedenih vrsta baza podataka. Upravitelj relacijske baze podataka jest MySQL SUBP, dok se kao predmet istraživanja NoSQL baze podataka uzeo MongoDB SUBP. Glavne operacije nad podacima u bazama jesu prijenos, odnosno unos podataka, ažuriranje podataka, pretraživanje podataka te brisanje podataka. Osmišljen je informacijski sustav, izveden kao web aplikacija, koji služi kao posrednik između korisnika i baze podataka. U obzir je uzeta frekvencija izvršavanja operacija, broj ponavljanja operacije te dodatni parametri koji se mogu predstaviti kroz domenu filtera podataka.

Glavni parametri na temelju kojih je moguće obaviti usporedbu učinkovitosti sustava za upravljanje bazama podataka jesu vrijeme izvršavanja svake pojedine operacije, opterećenje procesora te korištenje resursa RAM memorije. Na temelju analize rezultata istih, moguće je steći realan temelj za usporedbu učinkovitosti navedenih sustava za upravljanje bazama podataka. Deskriptivna statistika [4] prikazuje rezultate usporedbi.

Valja obratiti pozornost kako bitna činjenica na odabir sustava za upravljanje bazama podataka koji će biti temelj programskog proizvoda jest kompatibilnost odabranih tehnologija. Iako određeni sustav za upravljanje bazama podataka prema navedenim parametrima može biti učinkovitiji od onog s kojim se uspoređivao, prilikom odabira istog valja poznavati tehnologiju izrade pozadinskog dijela programskog proizvoda te i navedeni parametar uzeti u obzir. Time se minimizira mogućnost pojave kolizija između odabranih tehnologija. Konačno, iako će rezultati pokazati razinu učinkovitosti sustava za upravljanje bazama podataka, prilikom odabira valja uzeti u obzir i kontekst korištenja istih.

# 2 SQL I NoSQL SUSTAVI ZA UPRAVLJANJE BAZAMA PODATAKA

Informacije se temelje na podacima, stoga su podaci u današnjem svijetu najbitniji segment svakog radnog kolektiva. Bez postojanja podataka, cjelokupna programska podrška postaje bezvrijedna. Stoga je vrlo važno pohraniti podatke na ispravan i siguran način. Baze podataka služe za pohranu podataka, dok sustavi za upravljanjem baza podataka služe za manipulaciju istim. Aktualne tehnologije omogućuju pohranjivanje podataka u dva različita sustava; temeljenima na relacijama te na dokumentima, grafovima, indeksima stupaca te ključ/vrijednost (engl. *key/value*) vrijednostima. Prvi su poznati pod nazivom SQL sustavi za upravljanje bazama podataka, dok su drugi, negirani SQL, odnosno NoSQL sustavi za upravljanje bazama podataka. U sljedećim potpoglavljima pružen je pregled navedenih sustava.

## 2.1 RELACIJSKE I NoSQL BAZE PODATAKA

Relacijske baze podataka dobile su naziv prema relacijama na kojima se temelje. Teorijski su izuzetno razrađene, što rezultira širokom rasprostranjenošću te čestim korištenjem u mnogim programskim rješenjima. Za pristup i izvođenje operacija koristi se standardizirani SQL jezik, koji je podržan od brojnih razvojnih zajednica diljem svijeta, što ga čini jednim od najpodržanijih i najkorištenijih jezika u svijetu tehnologije.

Relacijske baze podataka temelje se na relacijama, odnosno vezama koje povezuju entitete stvorene unutar baze podataka. Realizirani entitet jest tablica, koja se sastoji od redaka i stupaca, što je vrlo intuitivno za shvaćanje i korištenje široj zajednici ljudi, stoga je ovaj tip sustava za upravljanje bazom podataka opće prihvaćen. Tablica je sadržana od različitih atributa, od kojih primarni ključ ima ulogu identifikatora, dok strani ključ služi za povezivanje s ostalim tablicama.

NoSQL baze podataka lišene su relacija između entiteta, odnosno tablica. Navedene baze podataka temelje se na dokumentima [5], koji su najmamja jedinica za pohranu podataka unutar baze. U usporedbi s relacijskim bazama podataka, složenije su za korištenje te je teorija ovakvih baza teže shvatljiva. Naime, u NoSQL bazama podataka moguće je pohraniti podatke različitog tipa, što može postati problem kada je riječ o velikim skupovima podataka. To je ujedno i glavni nedostatak iste. Lako i brzo dohvaćanje dokumenata jest jedno od glavnih prednosti ovog tipa baza podataka. Dokumenti se povezuju u kolekcije te se time stvara privid entiteta relacijske baze podataka. Svaki zasebni dokument definiran je s identifikacijskim brojem, a svi su podaci pohranjeni u .json, odnosno .bson formatu,

koji je jednostavan za iščitavanje i manipulaciju.

Osim relacijskih i NoSQL baza podataka temeljenim na dokumentima, još valja spomenuti nekoliko modela potonjih baza. To su prije svega NoSQL baze podataka temeljene na grafovima, *Columnar* te *In-Memory Data Grids* baze podataka. *Graph* NoSQL baze podataka temelje se na grafovima, a u pozadini su satkani od čvorova i svojstava. Takva vrsta baza podataka izuzetno je pogodna za ostvarivanje složenih veza između pohranjenih podataka. NoSQL *Columnar* baza podataka najsličinija je relacijskim bazama podataka. Ista se temelji na stupcima čiji se sadržaj u obliku indeksa u slijedu zapisuje na disk. Za manipulaciju takvim podacima moguće je koristiti SQL jezik upita, što predstavlja olakotnu okolnost prilikom korištenja navedene baze podataka. Posljednje spomenuta NoSQL *In-Memory Data Grids* baza podataka temeljena je na podacima koji su u nju zapisani u obliku ključ/vrijednost te se stoga može pohvaliti dobrim performansama pretraživanja i dohvaćanja podataka.

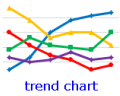
## 2.2 Pregled i karakteristike popularnih SQL i NoSQL sustava za upravljanje bazama podataka

Sve spomenute vrste baza podataka čine bazu iz koje su se razvila programska rješenja za pohranu i manipulaciju podataka. Prema [6], što je i prikazano na slici 2.1, najpopularniji sustavi za upravljanje bazama podataka su: Oracle, MySQL te Microsoft SQL server. Svi navedeni sustavi za upravljanje bazama podataka pripadaju relacijskom modelu, a rade na istom principu. Njihova glavna razlika jest proizvođač. Peti najpopularniji sustav za upravljanje bazama podataka jest MongoDB, koji je predstavnik Dokument (engl. *Document*) NoSQL modela. Iza njega slijedi Redis SUBP koji je predstavnik ključ/vrijednost NoSQL modela.

**DB-Engines Ranking**

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.



trend chart

398 systems in ranking, June 2022

| Rank Jun 2022 | Rank May 2022 | Rank Jun 2021 | DBMS | Database Model | Score Jun 2022 | Score May 2022 | Score Jun 2021 |
|---|---|---|---|---|---|---|---|
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1287.74 | +24.92 | +16.80 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1189.21 | -12.89 | -38.65 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 933.83 | -7.37 | -57.25 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 620.84 | +5.55 | +52.32 |
| 5. | 5. | 5. | MongoDB | Document, Multi-model | 480.73 | +2.49 | -7.49 |
| 6. | 6. | ↑ 7. | Redis | Key-value, Multi-model | 175.31 | -3.71 | +10.06 |
| 7. | 7. | ↓ 6. | IBM Db2 | Relational, Multi-model | 159.19 | -1.14 | -7.85 |
| 8. | 8. | 8. | Elasticsearch | Search engine, Multi-model | 156.00 | -1.70 | +1.29 |
| 9. | 9. | ↑ 10. | Microsoft Access | Relational | 141.82 | -1.62 | +26.88 |
| 10. | 10. | ↓ 9. | SQLite | Relational | 135.44 | +0.70 | +4.90 |
| 11. | 11. | 11. | Cassandra | Wide column | 115.45 | -2.56 | +1.34 |
| 12. | 12. | 12. | MariaDB | Relational, Multi-model | 111.58 | +0.45 | +14.79 |
| 13. | ↑ 14. | ↑ 26. | Snowflake | Relational | 96.42 | +2.91 | +61.67 |
| 14. | ↓ 13. | ↓ 13. | Splunk | Search engine | 95.56 | -0.79 | +5.30 |
| 15. | 15. | 15. | Microsoft Azure SQL Database | Relational, Multi-model | 86.01 | +0.68 | +11.22 |
| 16. | 16. | 16. | Amazon DynamoDB | Multi-model | 83.88 | -0.58 | +10.12 |
| 17. | 17. | ↓ 14. | Hive | Relational | 81.58 | -0.03 | +1.89 |
| 18. | 18. | ↓ 17. | Teradata | Relational, Multi-model | 70.41 | +2.02 | +1.07 |
| 19. | 19. | ↓ 18. | Neo4j | Graph | 59.53 | -0.61 | +3.78 |

*Slika 2.1:* ***Popis najpopularnijih sustava za upravljanje bazama podataka s pripadnim modelima baza***

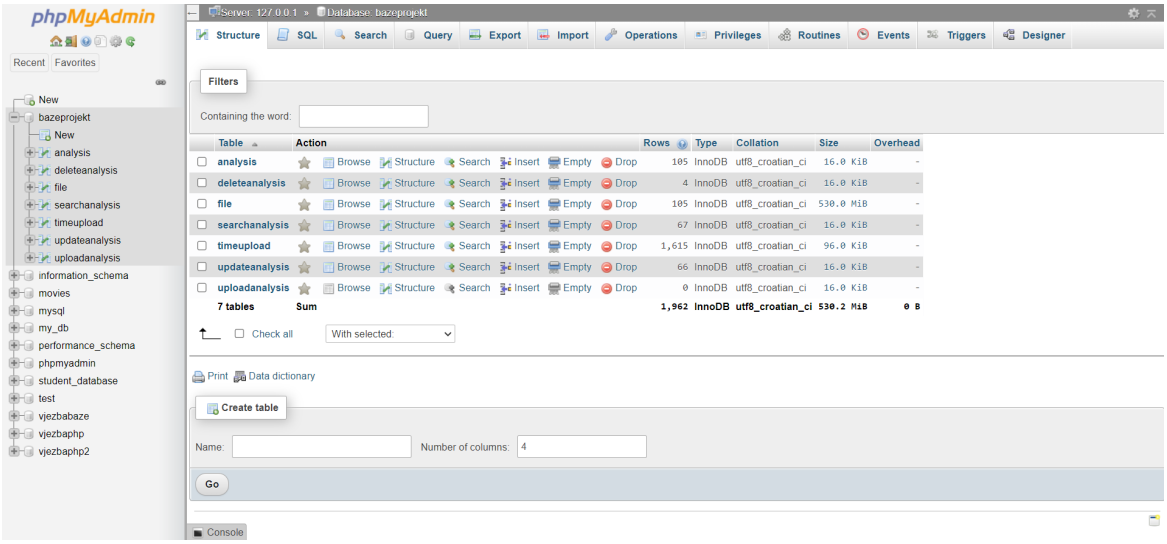
Valja naglasiti kako je od tri najpopularnija sustava za upravljanje bazama podataka, jedino MySQL SUBP otvoren za korištenje. Ostala dva sustava za upravljanje bazama podataka su komercijalna. Sva tri navedena SUBP-a imaju jednake karakteristike u suštini. Njihova se arhitektura temelji na tablicama koje su karakterizirane atributima, a međusobno povezane relacijama. MongoDB SUBP te Redis SUBP sa svojim karakteristikama znatno se razlikuju u odnosu na tri najpopularnija sustava za upravljanje bazama podataka.

## 2.3 MySQL SUBP

MySQL SUBP [7] jest jedan od najpopularnijih sustava za upravljanje bazama podataka današnjice. Otvorenog je koda, što omogućuje njegovo preuzimanje, instalaciju i korištenje bez naknade. Osnivač ovog SUBP-a jest Michael Widenius, a danas je u vlasništvu Oracle korporacije. MySQL SUBP, uz Linux, Apache, Perl/PHP/Python, dio je LAMP programskog proizvoda koji služi za razvoj web aplikacija.

MySQL SUBP vrlo je jednostavan za korištenje, a najčešće se upotrebljava u kombinaciji s PHP programskim jezikom, zbog kompatibilnosti koje te tehnologije omogućuju. Široko je rasprostranjen. Jedan od glavnih nedostataka MySQL SUBP-a jest niska razina skalabilnosti. MySQL SUBP svojim karakteristikama najbolje će zadovoljiti potrebe prilikom razvoja web aplikacija manjeg opsega te manjeg skupa podataka.

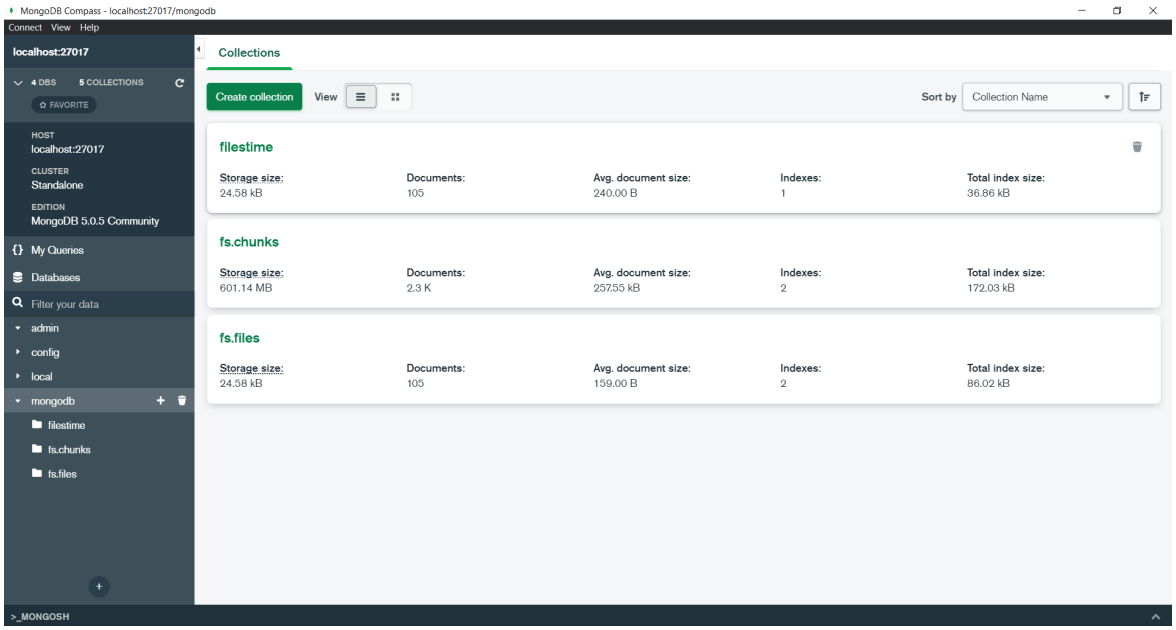
MySQL sustav za upravljanje bazom podataka koristi se za manipulaciju u relacijskoj bazi zasnovanoj na entitetima, odnosno tablicama, koje su pak karakterizirane atributima. Svaka tablica sadrži atribut primarnog te stranog ključa. Prvo navedeni jest identifikator tablice, dok se posljednje spomenuti koristi za ostvarivanje relacija. MySQL SUBP koristi se stvaranjem upita u standardiziranom SQL jeziku. Može se koristiti kroz mnoštvo aplikacija ili putem naredbenog retka. Na slici 2.2 prikazan je početni zaslon aplikacije PHPmyadmin, pomoću koje je korišten MySQL SUBP.




*Slika 2.2:* ***Prikaz sučelja zaslona PHPmyadmin aplikacije pomoću koje se koristi MySQL SUBP***

## 2.4 MongoDB SUBP

MongoDB jest sustav za upravljanje bazama podataka koji se temelji na NoSQL dokument orijentiranom modelu. Besplatan je za preuzimanje, instalaciju te korištenje. Nakon određenog vremena provedenog u istraživanju tehnologije MongoDB SUBP [8], primjećeno je kako je istu moguće koristiti na dva različita načina. Prvi način odnosi se na stvaranje korisničkog računa u sklopu MongoDB Atlas platforme te zakupljivanje dijela baze podataka, odnosno korištenje 500MB slobodnog prostora baze u svrhu edukacije. Alternativa tome odnosi se na preuzimanje lokalnog sustava pod nazivom MongoDB Compass [9], što je i korišteno u sklopu ovog rada. Zaslon navedene aplikacije vidljiv je na slici 2.3.




*Slika 2.3:* ***Prikaz sučelja zaslona MongoDB Compass aplikacije***

Prethodno navedenu aplikaciju potrebno je instalirati na računalo te pokrenuti lokalnu vezu. U slučaju da nije moguće pokrenuti vezu, potrebno je ručno pokrenuti SUBP pomoću servisa Windows operacijskog sustava. Nakon pokrenute veze prikazane su postojeće baze podataka. Svaka baza podataka sadrži kolekcije unutar sebe. Navedene kolekcije u MongoDB-u slične su tablicama u MySQL-u. Svaka kolekcija sadrži podatke koje su prikazane u redovima, a svaki red predstavlja objekt s jedinstvenim ključem.

Za potrebe ovoga rada, unutar dokumenta „konekcijaBaze.php“, stvorene su kolekcije unutar određene baze podataka. Svrha stvaranja ove baze podataka jest prijenos, ažuriranje, brisanje i pretraživanje podataka te mjerenje vremena navedenih operacija, usporedba opterećenja procesora i količina korištene RAM memorije. Veličina podataka za prijenos u MongoDB SUBP jest ograničena na 15 MB. Sve datoteke veće od navedene vrijednosti nije moguće prenijeti.

Navedeni se problem može zaobići gotovom funkcijom koja je kreirana od strane programera MongoDB-a. Funkcija *selectGridFSBucket()* [10] ima zadaću kreirati dvije kolekcije (ako već nisu kreirane) koje se nazivaju *fs.chunks* i *fs.files*. Svrha same funkcije je izbjeći prijenos podataka veći od 15MB, pa radi toga funkcija određeni podatak dekomponira na više manjih podataka koji su svi veličine 255KB. Zbog toga kolekcija *fs.files* pohranjuje osnovne podatke (ime, veličina datoteke, datum i vrijeme, md5 te *chunkSize*) o samoj datoteci koja se prenosi, dok kolekcija *fs.chunks* prenosi sadržaj određene datoteke na

način kao što je navedeno, odnosno sadržaj jedne datoteke dekomponira na više manjih datoteka veličine 255KB te sve numerira kako bi bio poznat redoslijed same datoteke.

# 3 SUSTAV ZA USPOREDBU UČINKOVITOSTI UPRAVITELJA SQL I NOSQL BAZA PODATAKA

Ogledni sustav jest web aplikacija implementirana u sljedećim tehnologijama: HTML, CSS, Javascript i PHP. Dizajn web aplikacije, odnosno njezin prezentacijski segment, stvoren je pomoću HTML, CSS i Javascript tehnologija, a isti se predstavlja kombinacijom četiri ključne boje. Sučelje je moderno i intuitivno za korištenje. Pozadinski dio aplikacije implementiran je pomoću PHP programskog jezika.

## 3.1 ARHITEKTURA SUSTAVA

Sustav za usporedbu učinkovitosti upravitelja SQL i NoSQL baza podataka zasniva se na platformi XAMPP. Cjelokupna platforma definira *cross-platform* sustav zasnovan na sljedećim tehnologijama: *Apache, MySQL, PHP i Perl*. Unutar platforme XAMPP, pri dijelu implementacije glavnog čimbenika ovog rada korišten je MySQL SUBP. Pomoću SQL naredbi u MySQL SUBP-u kreirano je sedam tablica: *file*, *analysis*, *timeupload*, *searchanalysis*, *deleteanalysis*, *updateanalysis* i *uploadanalysis*. Unutar MongoDB SUBP-a stvorene su kolekcije *filestime*, *fs.chunks* te *fs.files*. Sve su tablice u MySQL SUBP-u i kolekcije u MongoDB SUBP-u preduvjet provedbe usporedbe učinkovitosti upravitelja SQL i NoSQL baza podataka.

Sustav za usporedbu učinkovitosti upravitelja SQL i NoSQL baza podataka nastoji na temelju pretraživanja, brisanja, ažuriranja te prijenosa podataka usporediti vremena izvršavanja, opterećenje procesora te količinu korištenja RAM memorije za SQL i NoSQL sustave upravljanja bazama podataka. Zbog toga je sustav podijeljen na četiri glavne cjeline: prijenos podataka, pretraživanje podataka, ažuriranje podataka i brisanje podataka. Svaka navedena cjelina u nastavku će biti detaljno objašnjena.

*Prijenos podataka SQL i NoSQL baza podataka* prvi je dio razvijenog sustava kojemu je svrha prenijeti datoteke u MySQL i MongoDB sustave za upravljanje bazama podataka. Cjelina se sastoji od dva dijela.

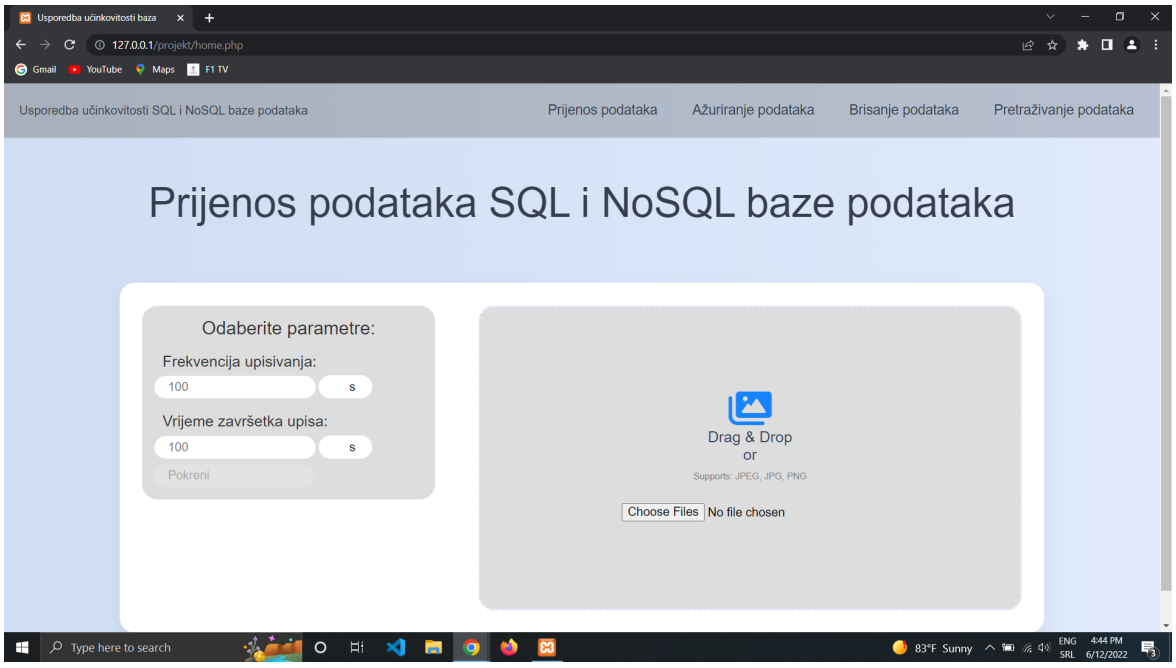
Lijevi dio sučelja sadrži konfiguracijski obrazac u kojemu je moguće definirati ulazne parametre eksperimenta. Polja za upis podataka i padajuće liste ispod tekstualne odrednice „Frekvencija upisivanja“ služe za definiranje frekvencije kojom se podaci zapisuju u bazu podataka, dok elementi u podnožju tekstualne odrednice „Vrijeme završetka upisa“ označavaju upravo vrijeme završetka zapisivanja podataka u bazu. Navedene padajuće

liste sadrže četiri vrijednosti: min (minuta), s (sekunda), ms (mili sekunda) i µs (mikro sekunda). Prethodno navedene vrijednosti uparuju se sa vrijednošću iz polja za unos podataka.

Desni dio sučelja smješten je u povećem okviru, a odnosi se na odabir podataka koji će biti prenesen i zapisan u odgovarajuće baze podataka. Cjelokupna površina okvira omogućuje korisniku prijenos podataka pomoću metode povuci-i-otpusti (engl. *Drag & Drop*). Ista podrazumijeva povlačenje i ispuštanje odabranih podataka unutar navedenog područja. U slučaju da korisnik želi odabrati podatke pretragom datotečnog sustava računala, omogućen je gumb „Pregledaj". Odabirom istog prikazuje se standardno sučelje za manualno pretraživanje datotečnog sustava.

Prijenos podataka u MySQL i MongoDB sustave za upravljanje bazama podataka izgleda kao na slici 3.1, a radi na sljedeći način:

1. Definira se frekvencija upisivanja
2. Odabiru se podaci ili se prenose pomoću metode povuci-i-otpusti
3. Pritišće se gumb „Pokreni"




*Slika 3.1:* ***Prijenos podataka SQL i NoSQL baza podataka***

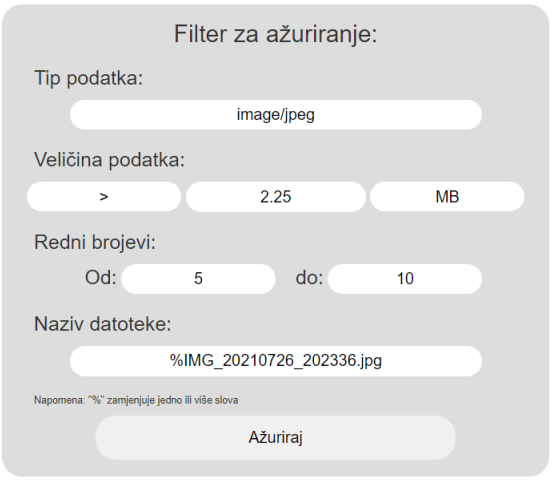
*Ažuriranje podataka SQL i NoSQL baza podataka* dio je sustava kojemu je svrha ažurirati imena pohranjenih datoteka unutar MySQL i MongoDB SUBP-ova. Cjelina se sastoji od tri dijela: filtera za ažuriranje, odabira frekvencije ažuriranja i prikaza postojećih datoteka.

Filter za ažuriranje, prikazan na slici 3.2, smješten je u lijevom dijelu sučelja te sadrži četiri glavna parametra: „Tip podatka“, „Veličina podatka“, „Redni brojevi“ i „Naziv datoteke“. Padajuća lista ispod odrednice „Tip podatka“ sadrži tipove podataka koji su pohranjeni u MySQL i MongoDB SUBP-ovima te služi za odabir tipa podatka čije ime želimo ažurirati.

Parametar „Veličina podatka“ sastoji se od jednog polja za unos brojeva te dvije padajuće liste. U prvoj padajućoj listi odabire se jedan od pet mogućih operatora. Zatim se u polje za unos brojeva unosi broj koji može imati vrijednost od 0 do 999999. Za kraj, u padajućoj listi odabire se jedna veličina od sljedećih: B, KB, MB i GB.

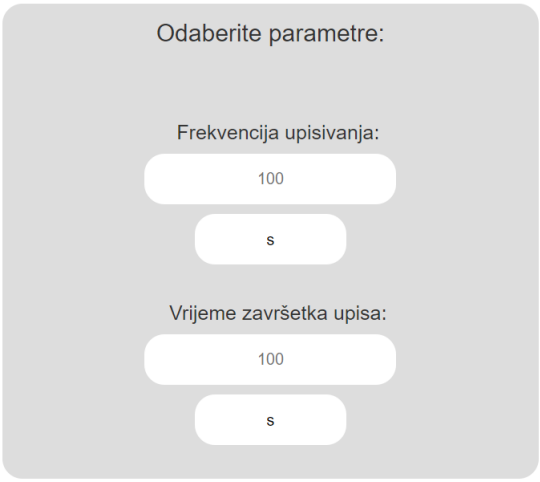
Parametar „Redni brojevi“ sadrži dvije padajuće liste. Svaka padajuća lista sadrži redne brojeve dokumenata koji su prikazani u tablici ispod filtera. U prvoj padajućoj listi odabire se broj od kuda će započeti ažuriranje naziva datoteka, dok se u drugoj padajućoj listi odabire broj s kojim bi ažuriranje naziva datoteka trebalo završiti. Odabirom oba parametara definira se format pretrage, čiji sadržaj može izgledati poput sljedećeg primjera: „od 12 do 20“. U navedenom primjeru bit će ažurirani nazivi datoteka od 12. do 20. datoteke.

Zadnji parametar „Naziv datoteke“ sadrži jedno polje za unos naziva datoteke. Unutar navedenog polja postoje dvije mogućnosti upisivanja datoteka. Prva mogućnost je da se napiše cijeli naziv datoteke, dok je druga mogućnost da se u naziv datoteke unese znak „%“ koji omogućuje zamjenu jednog ili više znakova u samome nazivu datoteke. Na primjeru „%Ime datoteke“, sustav će ažurirati nazive datoteka koji opcionalno sadrže jedan ili više znakova ispred sintagme „Ime datoteke“.




*Slika 3.2:* ***Filter za ažuriranje podataka***

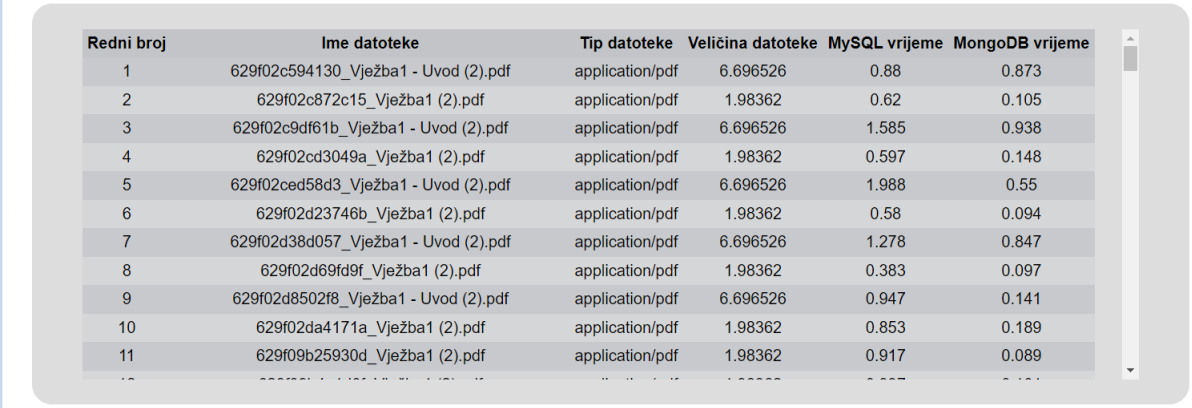
U desnom dijelu sučelja smještena je frekvencija ažuriranja. Iste je funkcionalnosti kao i kod dijela *Prijenos podataka SQL i NoSQL baza podataka* s ponešto drugačijom vizualnom prezentacijom. Na slici 3.3 vidljiv je izgled frekvencije ažuriranja.




*Slika 3.3:* ***Frekvencija ažuriranja***

U donjem dijelu sučelja smještena je tablica vizualizirana slikom 3.4, a koja prikazuje sve podatke pohranjene u MySQL i MongoDB sustavima za upravljanje bazama podataka. Za svaki od podataka, tablica prikazuje: redni broj, ime podatka, tip podatka, veličinu podatka, vrijeme prijenosa podatka u MySQL SUBP i vrijeme prijenosa podataka u Mon-

goDB SUBP. Vrlo je korisna u slučaju postavljanja filtera za pretragu, brisanje ili ažuriranje podataka.

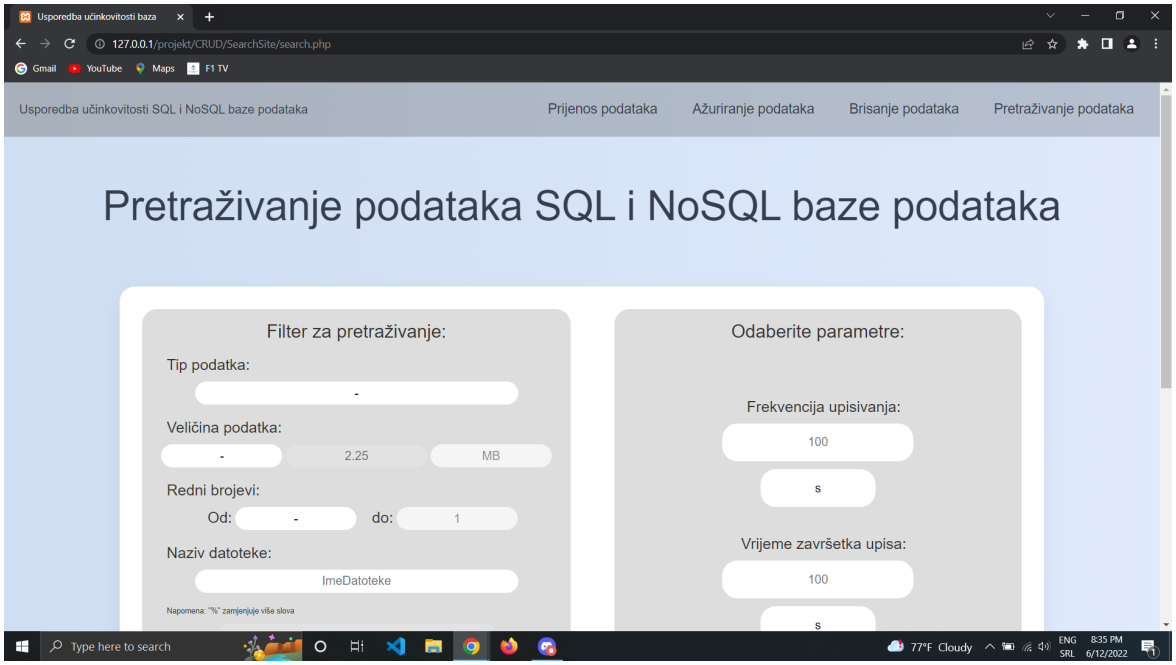


| Redni broj | Ime datoteke | Tip datoteke | Veličina datoteke | MySQL vrijeme | MongoDB vrijeme |
|---|---|---|---|---|---|
| 1 | 629f02c594130_Vježba1 - Uvod (2).pdf | application/pdf | 6.696526 | 0.88 | 0.873 |
| 2 | 629f02c872c15_Vježba1 (2).pdf | application/pdf | 1.98362 | 0.62 | 0.105 |
| 3 | 629f02c9df61b_Vježba1 - Uvod (2).pdf | application/pdf | 6.696526 | 1.585 | 0.938 |
| 4 | 629f02cd3049a_Vježba1 (2).pdf | application/pdf | 1.98362 | 0.597 | 0.148 |
| 5 | 629f02ced58d3_Vježba1 - Uvod (2).pdf | application/pdf | 6.696526 | 1.988 | 0.55 |
| 6 | 629f02d23746b_Vježba1 (2).pdf | application/pdf | 1.98362 | 0.58 | 0.094 |
| 7 | 629f02d38d057_Vježba1 - Uvod (2).pdf | application/pdf | 6.696526 | 1.278 | 0.847 |
| 8 | 629f02d69fd9f_Vježba1 (2).pdf | application/pdf | 1.98362 | 0.383 | 0.097 |
| 9 | 629f02d8502f8_Vježba1 - Uvod (2).pdf | application/pdf | 6.696526 | 0.947 | 0.141 |
| 10 | 629f02da4171a_Vježba1 (2).pdf | application/pdf | 1.98362 | 0.853 | 0.189 |
| 11 | 629f09b25930d_Vježba1 (2).pdf | application/pdf | 1.98362 | 0.917 | 0.089 |

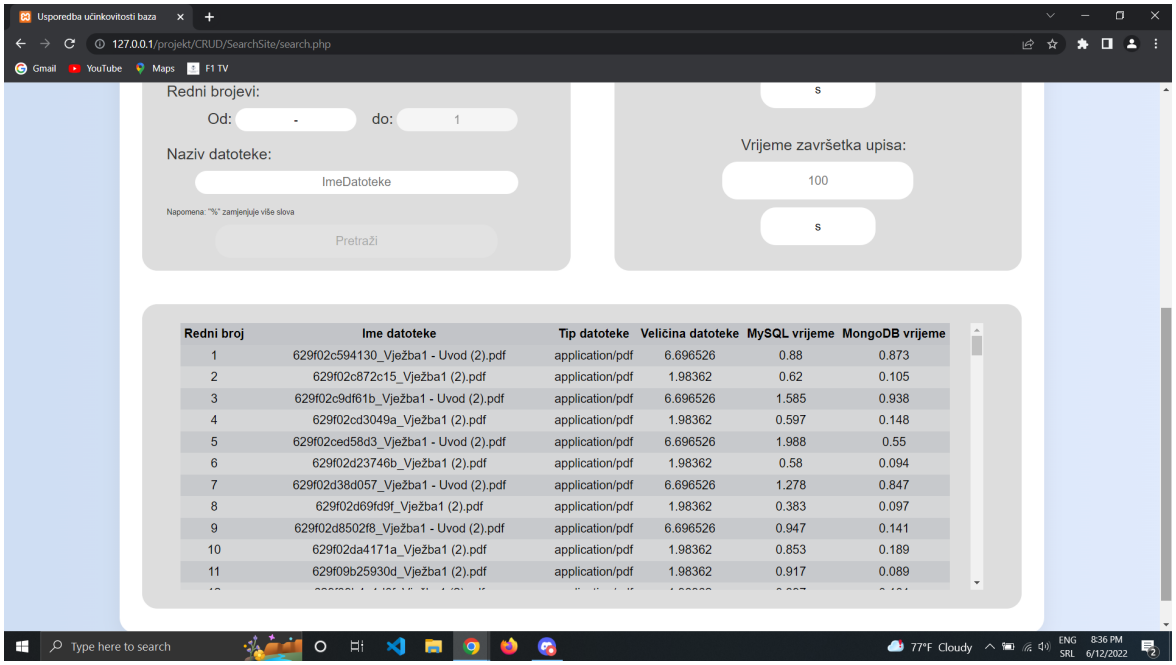
*Slika 3.4:* ***Tablica s pohranjenim podacima***

*Pretraživanje podataka SQL i NoSQL baza podataka* dio je sustava čiji je cilj pretražiti podatke te zabilježiti vrijeme pretraživanja, iskorištene resurse procesora i RAM memorije za MySQL i MongoDB SUBP-ove. Ovaj podsustav analognog je izgleda kao i podsustav *Ažuriranje podataka SQL i NoSQL baza podataka* te ima iste funkcionalnosti. Jedina razlika između navedenih podsustava jest orijentacija. Prvo spomenuti podsustav orijentiran je na ažuriranje, dok je sljedeći orijentiran na pretraživanje. Način rada podsustava ažuriranja ili pretraživanja SQL i NoSQL baza podataka izgleda kao na slikama 3.5 i 3.6, a isčitava se kroz sljedećih nekoliko koraka:

1. Pregledavaju se podaci koji se nalaze u tablici
2. Nakon pregleda podataka pohranjenih u SQL i NoSQL bazama podataka, definiraju se parametri filtera za pretraživanje/ažuriranje
3. Definira se frekvencija pretraživanja/ažuriranja
4. Pritišće se gumb „Pretraži“/„Ažuriraj“




*Slika 3.5:* ***Prvi dio podsustava Pretraživanje podataka SQL i NoSQL baze podataka***




*Slika 3.6:* ***Drugi dio podsustava Pretraživanje podataka SQL i NoSQL baze podataka***

*Brisanje podataka SQL i NoSQL baza podataka* predstavlja zadnju cjelinu ovoga sustava, a svrha jest brisanje podataka iz MySQL i MongoDB SUBP-ova te praćenje performansi. Sastoji se od dva dijela: filtera za brisanje podataka i tablice koja prikazuje sve

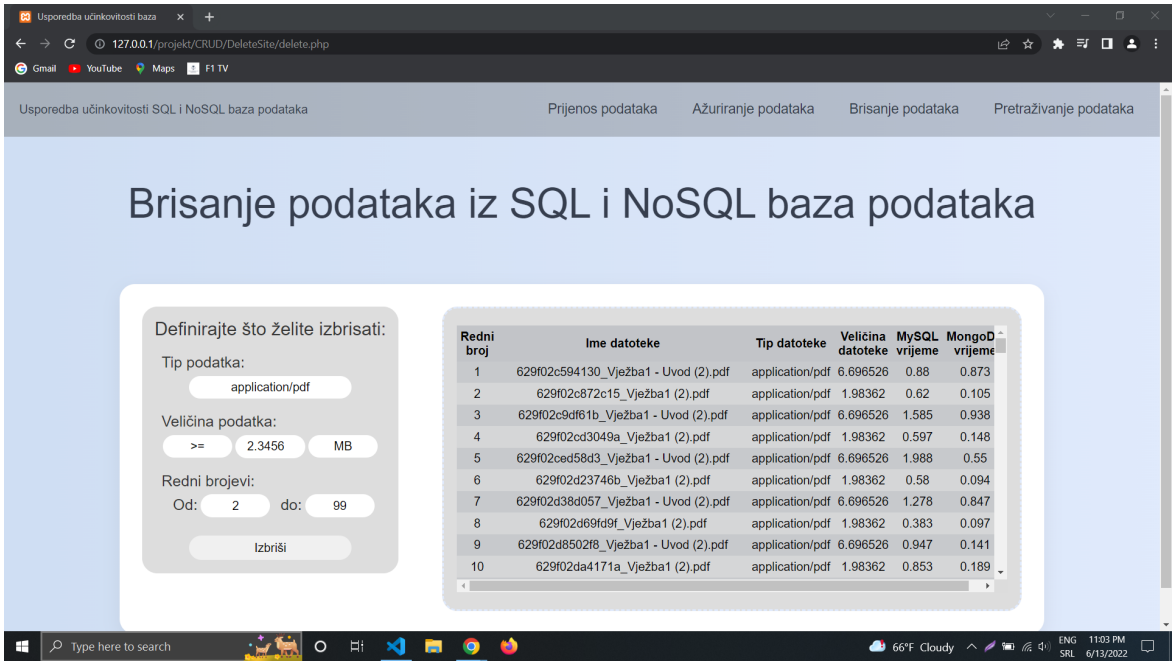
pohranjene podatke u navedenim sustavima za upravljanje bazama podataka. Slika 3.7 prikazuje vizualni identitet navedenog podsustava.

Filter za brisanje podataka smješten je u lijevom dijelu sučelja te sadrži tri parametra: „Tip podatka“, „Veličina podatka“ i „Redni brojevi“. Parametar „Tip podatka“ sadrži padajuću listu u kojoj se nalaze tipovi podataka pohranjeni u MySQL i MongoDB SUBP-ovima. Parametri „Veličina podatka“ i „Redni brojevi“ konfiguriraju se istim postupkom kao kod podsustava *Pretraživanje podataka SQL i NoSQL baze podataka*.

U desnom dijelu sučelja nalazi se tablica koja prikazuje sve pohranjene podatke u bazama podataka. Kako bi se lakše odabrali svi parametri za brisanje datoteka, u tablici se za svaku datoteku prikazuje: redni broj, ime podatka, tip podatka, veličina podatka, vrijeme prijenosa podatka u MySQL SUBP i vrijeme prijenosa podatka u MongoDB SUBP. Primjer tablice vidljiv je na desnoj strani slike 3.7.

Način rada podsustava vezanog za brisanje podataka SQL i NoSQL baza podataka je sljedeći:

1. Pregledavaju se podaci koji se nalaze u tablici
2. Nakon pregleda podataka pohranjenih u SQL i NoSQL bazama podataka, definiraju se parametri filtera za brisanje
3. Pritišće se gumb „Izbriši“




*Slika 3.7:* ***Brisanje podataka SQL i NoSQL baza podataka***

Također, sustav za usporedbu učinkovitosti SQL i NoSQL baza podataka sadrži još cjeline vezane za sam prikaz učinkovitosti SUBP-ova. Cjeline se dijele na: *Usporedba učinkovitosti prijenosa podataka u SQL i NoSQL baze podataka*, *Analiza ažuriranja podataka MySQL i MongoDB SUBP-ova*, *Analiza pretraživanja podataka MySQL i MongoDB SUBP-ova* i *Analiza brisanja podataka MySQL i MongoDB SUBP-ova.*

Svaka cjelina sadrži grafove: usporedbe vremena izvršavanja operacija MySQL i MongoDB sustava za upravljanje bazama podataka, usporedbe opterećenja prilikom korištenja procesora za MySQL SUBP i MongoDB SUBP, usporedbe korištenja RAM memorije, prikaz svih arhiviranih vremenskih usporedbi, sortirani prikaz svih arhiviranih vremenskih usporedbi i omjer usporedbe vremena izvođenja operacija MySQL i MongoDB SUBP-ova prikazan u postocima.

*Usporedba učinkovitosti prijenosa podataka u SQL i NoSQL baze podataka* jedina je cjelina koja uz prethodno navedene grafove sadrži još dva dodatna grafa: „Usporedba prosječnih vremena za datoteke“ i „Usporedba prosječnog korištenja RAM memorije za datoteke“. Sve prethodno navedene cjeline s podržanim grafovima su prikazane i detaljno objašnjene u poglavlju 4.

## 3.2 KARAKTERISTIČNI SLUČAJEVI KORIŠTENJA

U ovom potpoglavlju objašnjene su izvedbe i funkcionalnosti frekvencijskih parametara, metode povuci-i-otpusti, učitavanja stranice, filtera te operacija vezane za prijenos, ažuriranje, brisanje i pretraživanje podataka unutar MySQL i MongoDB SUBP-ova.

Frekvencijski parametri koriste se u podsustavima *Prijenos podataka SQL i NoSQL baze podataka*, *Ažuriranje podataka SQL i NoSQL baza podataka* i *Pretraživanje podataka SQL i NoSQL baza podataka.* U svakom prethodnom podsustavu frekvencijski parametri imaju istu funkcionalnost te izgledaju kao na slici 3.3.

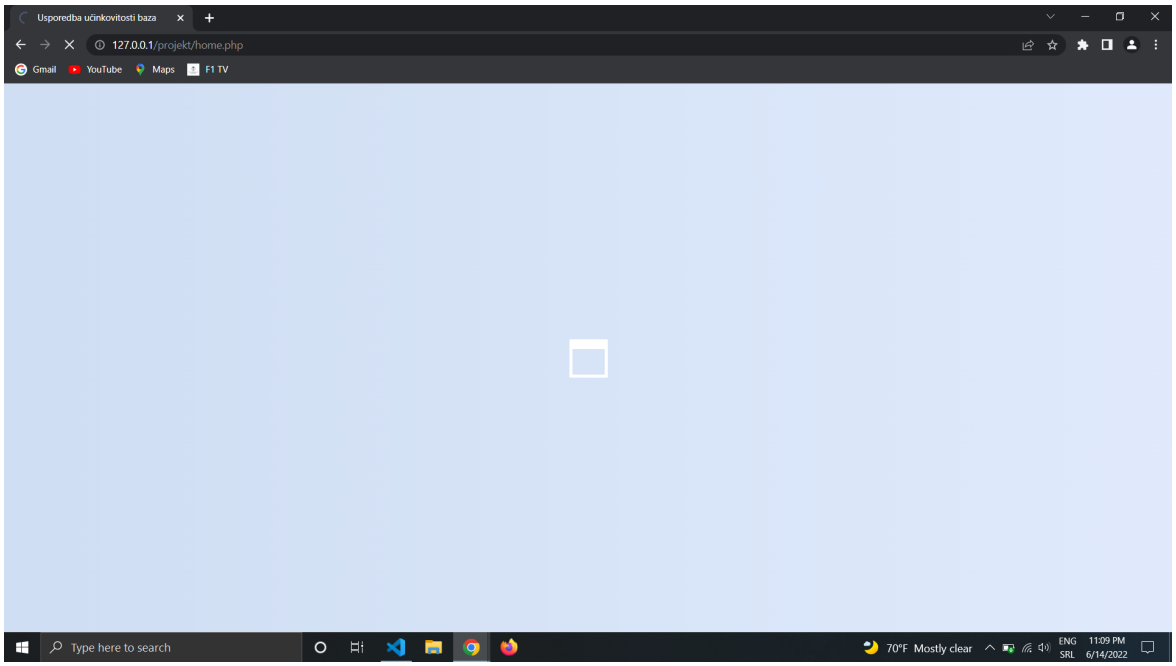
Upisivanjem podataka u polje ispod tekstualne odrednice „Frekvencija upisivanja“ i odabirom jedne od vrijednosti iz padajuće liste, određuje se frekvencija kojom se podaci žele prenijeti u baze podataka. Upisivanjem podataka u polje i odabirom vrijednosti iz padajuće liste ispod tekstualne odrednice „Vrijeme završetka upisa“, određuje se vrijeme završetka prijenosa podataka u baze podataka. Naime, zapis podataka započinje od nule te se vrijednost frekvencije upisivanja inkrementira pri svakom prijenosu podataka u određenu bazu podataka. Iteracija traje sve do zadane vrijednosti vremena završetka upisa. U svrhu boljeg shvaćanja procesa rada web aplikacije, isti je objašnjen na primjeru u nastavku.

Primjer je opisan za situaciju u kojoj je postavljena vrijednost frekvencije upisivanja jednaka 500ms te u slučaju kada je vrijednost završetka upisa jednaka 1s. Prijenos podataka će se odviti u tri koraka. Vrijednost vremenske odrednice od 0ms uzima se kao prvi korak prijenosa podataka, odnosno začetak prijenosa podataka. Drugim korakom se smatra trenutak inkrementacije vrijednosti s 0 na 500ms. Inkrement vrijednosti s 500ms na 1s ujedno je i treći korak prijenosa podataka. Svaki od koraka ima zadaću prenijeti podatke u baze. U trenutku kada vremenska odrednica poprimi krajnju vrijednost procesa prijenosa podataka, isti se završava. Tada su svi podaci preneseni u odgovarajuće baze.

Metoda povuci-i-otpusti služi za odabir podataka u odgovarajuće baze podataka. Po odabiru podataka za prijenos, potrebno je odabrati opciju „Pokreni“. Zatim se određeno vrijeme obavlja prijenos podataka na način opisan u prethodnom odlomku. Po završetku prijenosa podataka prikazat će se detalji istog. Sučelje web aplikacije s unesenim parametrima te pokrenutim prijenosom podataka prikazano je na slici 3.1.

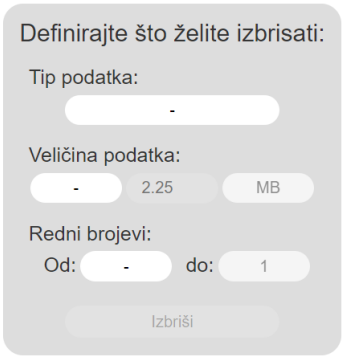
Funkcionalnost učitavanja stranice prikazuje se tijekom izvođenja operacija prijenosa,

ažuriranja, brisanja i pretraživanja. Stranica aktivno prikazuje grafički indikator i njezina svrha je prikazivanje korisniku da je operacija izvođenja u tijeku. Navedeno učitavanje stranice moguće je vidjeti na slici 3.8.




*Slika 3.8:* ***Učitavanje stranice***

Filteri korišteni u podsustavima *Ažuriranje podataka SQL i NoSQL baza podataka* i *Pretraživanje podataka SQL i NoSQL baza podataka* istih su funkcionalnosti te izgledaju kao na slici 3.2, dok filter korišten unutar podsustava *Brisanje podataka SQL i NoSQL baza podataka* ne sadrži samo zadnji parametar „Naziv datoteke“. Na slici 3.9 moguće je vidjeti filter podsustava *Brisanje podataka SQL i NoSQL baza podataka.*




*Slika 3.9:* ***Filter podsustava Brisanje podataka SQL i NoSQL baza podataka***

Filteri se koriste na način da se odabire kombinacija ponuđenih parametara: „Tip podatka“, „Veličina podatka“, „Redni brojevi“ i „Naziv datoteke“. Kako bi se dobio cjelovit upit za svaku moguću kombinaciju unutar MySQL SUBP-a koriste se četiri *if* uvjeta. Izvedba filtra vidljiva je na ispisu 3.1.

```
$searchMySQL = "SELECT * FROM analysis WHERE";

    if ($type != '-') {
        $searchMySQL .= " Type = '" . $type . "' AND ";
    }

    if ($operator != '-') {
        $searchMySQL .= " SizeMB " . $operator ." ". $finalSize ." AND ";
    }

    if ($from != '-') {
        $searchMySQL .= " ID BETWEEN " . $from . " AND " . $to . " AND ";
    }

    if (!empty($ime)) {
        $searchMySQL .= " FileName LIKE ('" . $ime . "') AND ";
    }

    $searchMySQL = substr_replace($searchMySQL, "", -4);
    $mySQLstart = microtime(true);
```

*Ispis 3.1:* ***Programski kod za izvedbu višeparametarskog filtra***

Kombinaciju filterskih parametara za MongoDB SUBP nije bilo moguće napraviti na način prikazan u programskom kodu MySQL upita prema MySQL SUBP-u. Zbog toga je za kreiranje filtera bilo potrebno pozvati funkciju ažuriranja ili pretraživanja. U programskom kodu, koji je prikazan u ispisu 3.2, vidljivo je da se poziva funkcija pretraživanja nad kolekcijom „filestime“ te da se filter ostvaruje u obliku polja (engl. *array*) unutar kojih su prethodno navedeni parametri.

```
require __DIR__ . '/vendor/autoload.php';
$conn2 = new MongoDB\Client("mongodb://localhost:27017");

$db = $conn2->mongodb;
$mongoFiles = $db->filestime;

```

```
$cursor = $mongoFiles->find(array("Type" => $type, "Name" => array('$in'
    => $fileNames)));

$cursor = $mongoFiles->find(array("Name" => array('$in' => $fileNames), "
    Name" => array('$regex' => $ime)));
```

*Ispis 3.2:* ***Primjer programskog koda za pretraživanje i ažuriranje podataka u MongoDB SUBP-u***

Prijenos podataka unutar MySQL SUBP-a izveden je na način da se spremaju svi važni čimbenici (ime podatka, tip podatka, veličina podatka...) zajedno sa samim sadržajem podatka u tablicu koja se nalazi unutar MySQL SUBP-a. Izvedba prijenosa podataka vidljiva je u ispisu 3.3.

```
$filename = uniqid() . "_" . $_FILES['fileToUpload']['name'][$key];
$type = $_FILES['fileToUpload']['type'][$key];
$size = $_FILES['fileToUpload']['size'][$key] / 1000000;
$tmp_name = $_FILES['fileToUpload']['tmp_name'][$key];
$realTime = date('Y-m-d H:i:s', $toTime);
$idTime = uniqid();
$image = $_FILES['fileToUpload']['tmp_name'][$key];
ini_set('memory_limit', '-1');
$content = addslashes(file_get_contents($image));

$sql_query = $conn->prepare("INSERT INTO file ( Name, Type, SizeMB,
    Tmp_name, Upload_time, ID_time, Content) VALUES ('" . $filename . "'
    , '" . $type . "'
    , '" . $size . "'
    , '" . $tmp_name . "'
    , '" . $realTime . "'
    , '" . $idTime . "'
    , '" . $content . "')");

$command = 'start /B php E:\Programi\XAMPP\htdocs\projekt\CPU_RAM\
    usingRAMmySQL.php  > NUL';
pclose(popen($command, 'r'));
$sql_query->execute();
$sql_query->close();
```

*Ispis 3.3:* ***Programski kod za prijenos podataka u MySQL SUBP***

Za iniciranje konekcije na bazu, kao i u slučaju prijenosa i zapisa podataka u istu nije potrebno koristiti upite, već je sve izvedeno pomoću već gotovih funkcija. Navedeno ko-

rištenje funkcija za MongoDB tehnologiju moguće je vidjeti u progamskom kodu prikazanom na ispisu 3.4.

```
$bucket = $db->selectGridFSBucket();
$resource = fopen($tmp_name, "a+");
$bucket->uploadFromStream($filename, $resource);
$command = 'start /B php E:\Programi\XAMPP\htdocs\projekt\CPU_RAM\
    usingRAMmongo.php  > NUL';
pclose(popen($command, 'r'));
```

*Ispis 3.4:* ***Programski kod za prijenos podataka u MongoDB SUBP***

Ažuriranje podataka u MySQL i MongoDB SUBP-ovima izvodi se slanjem upita prema MySQL SUBP-u koji ima cilj pronaći imena svih podataka. Nakon pronalska svih imena unutar *while* petlje, za svaki podatak posebno, šalje se MySQL upit te posebno se poziva MongoDB funkcija ažuriranja. Naime, cilj ove operacije je promijeniti odnosno ažurirati naziv podatka u novi nasumično odabrani naziv. Operacije su vidljive na ispisu 3.5.

```
while (($row = $query->fetch_assoc()) !== null) {
    $getName = substr($row["FileName"], 13);
    $newName = uniqid() . $getName;
    $updateMySQL = "UPDATE analysis SET Filename = '" . $newName . "'
    WHERE FileName = '" . $row["FileName"] . "'";
    $mySQLstart = microtime(true);
    $query2 = $conn->query($updateMySQL);
    $mySQLstop = microtime(true);

    $updateMongo = $mongoFiles->updateMany(array('Name' => $row["FileName
    "]), array('$set' => array('Name' => $newName)));
}
```

*Ispis 3.5:* ***Programski kod za ažuriranje podataka u MySQL i MongoDB SUBP-ovima***

Brisanje podataka u MySQL i MongoDB SUBP-ovima, čija je izvedba prikazana ispisom 3.6, ponešto je drugačiji. Brisanje podataka unutar MySQL SUBP izvodi se upitom s ključnom riječju *DELETE* te s spomenutim filterom. Koristeći upit s filterom, brišu se svi željeni podaci, dok u MongoDB SUBP-u to nije moguće. Kao što je objašnjeno u potpoglavlju 2.4, MongoDB tehnologija razlaže podatke koristeći knjižnicu *GridFS*. Podaci unutar *fs.files* sadrže md5 kriptografsku funkciju, naziv podatka i identifikacijski broj. S posljednje navedenim su povezani svi podaci koji sadrže isti identifikacijski broj unutar *fs.chunks*. Zbog toga

je brisanje unutar MongoDB SUBP-a moguće jedino ako se identificira identifikacijski broj određenog podatka. Ovaj slučaj je izveden na sljedeći način:

1. Pronalaze se nazivi svih podataka koje je potrebno izbrisati
2. Na temelju naziva, za svaki podatak dobije se identifikacijski broj
3. S obzirom na broj podataka, unutar petlje, za svaki podatak koristi se gotova MongoDB funkcija *delete()* te se u argument zapisuje identifikacijski broj
4. Brišu se svi podaci s navedenim identifikacijskim brojem unutar *fs.chunks* i *fs.files*

```
/* Dio za MySQL */
$selectDeletedFiles = "SELECT Name, file.Type, file.SizeMB, FileName FROM
    file INNER JOIN analysis ON file.ID = analysis.ID WHERE";

$deleteMySQL = "DELETE analysis, file FROM analysis INNER JOIN file
       ON analysis.ID_time = file.ID_time WHERE";

if ($type != '-') {
    $deleteMySQL .=  " file.Type = '" . $type . "' AND ";
    $selectDeletedFiles .= " file.Type = '" . $type . "' AND ";
}

if ($operator != '-') {
    $deleteMySQL .= " file.SizeMB " . $operator . " " . $finalSize . "
   AND ";
    $selectDeletedFiles .= " file.SizeMB " . $operator . " " . $finalSize
    . " AND ";
}

if ($from != '-') {
    $deleteMySQL .= " analysis.ID  BETWEEN " . $from . " AND " . $to . "
   AND ";
    $selectDeletedFiles .= " file.ID  BETWEEN " . $from . " AND " . $to .
    " AND ";
}

$selectDeletedFiles = substr_replace($selectDeletedFiles, "", -4);
$result = $conn->query($selectDeletedFiles);

/* Dio za MongoDB */
```

```
while ($row = $result->fetch_assoc()) {
    $idsToDelete[$i] = $row["Name"];
}

/* Koristi se GridFS za pretrazivanje */
$bucket = $db->selectGridFSBucket();
$ispis = $bucket->find(array("filename"  => array('$in' => $idsToDelete))
    );

/* Brisu se datoteke iz fs.files i fs.chunks */
foreach ($ispis as $red) {
    $bucket->delete($red['_id']);
}
```

*Ispis 3.6:* ***Programski kod za brisanje podataka u MySQL i MongoDB SUBP-ovima***

Pretraživanje podataka u MySQL i MongoDB SUBP-ovima je zadnja i ujedno najlakša operacija, kako za implementaciju, tako i za izvođenje. Za MySQL SUBP definiran je upit koji sadrži ključnu riječ *SELECT* te filter pretraživanja, dok se za MongoDB SUBP koristi *find()* funkcija. Način izvedbe moguće je vidjeti u programskom kodu na ispisu 3.7.

```
/* Dio za MySQL */
$searchMySQL = "SELECT * FROM analysis WHERE";

if ($type != '-') {
    $searchMySQL .= " Type = '" . $type . "' AND ";
}

if ($operator != '-') {
    $searchMySQL .= " SizeMB " . $operator . " " . $finalSize . " AND ";
}

if ($from != '-') {
    $searchMySQL .= " ID  BETWEEN " . $from . " AND " . $to . " AND ";
}

if (!empty($ime)) {
    $searchMySQL .= " FileName LIKE ('" . $ime . "') AND ";
}

$searchMySQL = substr_replace($searchMySQL, "", -4);
$mySQLstart = microtime(true);
```

```
$query = $conn->query($searchMySQL);
$mySQLstop = microtime(true);

/* Primjer dijela za MongoDB */
$cursor = $mongoFiles->find(array("Name" => array('$in' => $fileNames), "
    Name" => array('$regex' => $ime)));

$cursor = $mongoFiles->find(array("Size(MB)" => array($operatorQuery =>
    $finalSize), "Name" => array('$regex' => $ime)));
```

*Ispis 3.7:* ***Programski kod za pretraživanje podataka u MySQL i MongoDB SUBP-ovima***

# 4 USPOREDBA UČINKOVITOSTI SUSTAVA MYSQL I MONGODB

U ovome poglavlju detaljno su opisani dijelovi sustava koji uspoređuju učinkovitost i performanse MySQL i MongoDB sustava za upravljanje bazama podataka. Dijelovi sustava koji su odgovorni za prikaz učinkovitosti su: *Analiza pretraživanja podataka MySQL i MongoDB SUBP-ova*, *Analiza brisanja podataka MySQL i MongoDB SUBP-ova*, *Analiza ažuriranja podataka MySQL i MongoDB SUBP-ova* i *Usporedba učinkovitosti prijenosa podataka u SQL i NoSQL baze podataka.*

## 4.1 DIZAJN EKSPERIMENTA

Eksperiment se sastoji od četiri glavna dijela. Svaki dio se odnosi na jednu operaciju, pa tako razlikujemo četiri temeljne operacije: prijenos, ažuriranje, brisanje i pretraživanje podataka.

U eksperimentu prijenosa podataka prenosi se pet različitih podataka od kojih su tri slikovne datoteke tipa jpeg i dvije tekstualne datoteke tipa pdf. Svaka datoteka jest različite veličine. Sljedeće slikovne datoteke korištene su u eksperimentu:

- IMG_20210726_203114 veličine 1.86 MB
- IMG_20210726_210822 veličine 6.19 MB
- IMG_20210726_205710 veličine 10.1 MB

Tekstualne datoteke korištene u eksperimentu jesu sljedeće:

- Dokument1 veličine 2.67 MB
- Dokument2 veličine 6.38 MB

Sve prethodno navedene datoteke prenose se unaprijed definiranom frekvencijom. Frekvencija upisivanja iznosi 200 ms, dok je vrijeme završetka upisa jednako 4 s. Ovako definirani parametri eksperimenta uzrokuju prijenos navedenih datoteka u dvadeset jednom ponavljanju, odnosno sveukupno će biti preneseno 105 datoteka u MySQL i MongoDB sustave za upravljanje bazama podataka. Rezultate ovog eksperimenta moguće je vidjeti u potpoglavlju 4.2.1.

Drugi dio provedbe eksperimenta podrazumijeva ažuriranje podataka. Ažurirat će se naziv odabranih datoteka te zbog toga veličina ili tip datoteke nema nikakvog utjecaja na performasne ažuriranja unutar MySQL SUBP-a i MongoDB SUBP-a. Važni parametri u provedbi eksperimenta operacije ažuriranja jesu frekvencija i broj odabranih datoteka kojima se želi ažurirati naziv.

U postavu eksperimenta ažuriranja koristi se frekvencija koja ima cilj opteretiti sustave baza podataka. Frekvencija ažuriranja iznosi 200 ms, dok vrijeme završetka ažuriranja iznosi dvije sekunde. Eksperiment je proveden nad prvih trideset datoteka unutar baza podataka. Definirani parametri određuju kako se eksperiment ažuriranja nad datotekama izvodi u jedanaest iteracija. Naziv svake zasebne datoteke ažurirat će se jedanaest puta. Rezultati ovog eksperimenta prikazani su u potpoglavlju 4.2.2.

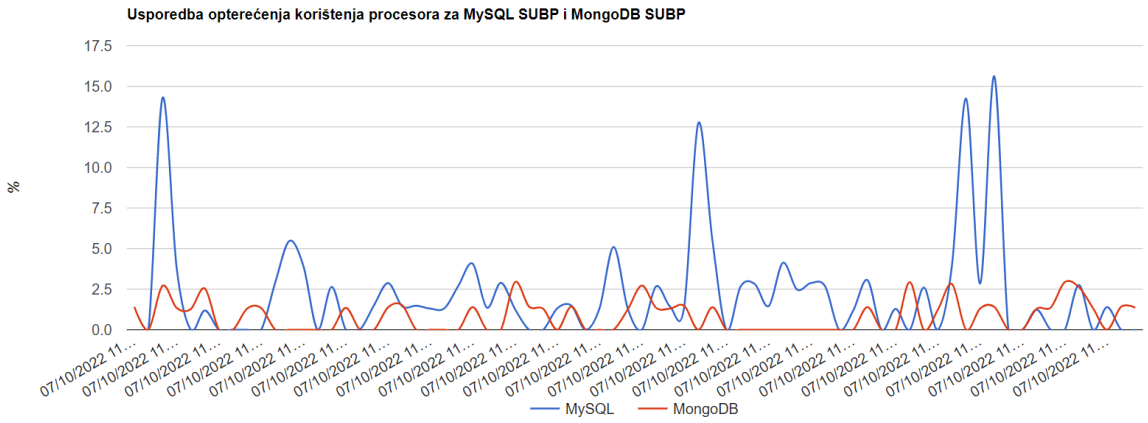
Treći dio provedbe eksperimenta jest ispitivanje funkcionalnosti pretraživanja datoteka. Kao i kod izvedbe eksperimenta operacije ažuriranja, veličina ili tip datoteke nema utjecaja na performanse pretraživanja te su zbog toga važni parametri u ovom eksperimentu frekvencija i broj odabranih datoteka za pretraživanje. Izvode se dva slučaja pretraživanja. U prvom slučaju frekvencija pretraživanja jest jednaka 200 ms, dok je vrijeme završetka pretraživanja definirano na 5 sekundi. U jednom ciklusu izvršava se pretraživanje dvadeset datoteka. Zbog ovako definirane frekvencije, ovaj dio eksperimenta jest proveden u dvadeset i šest ciklusa, što bi značilo kako je pretraživanje izvršeno na sveukupno petsto dvadeset datoteka.

U drugom slučaju frekvencija pretraživanja jest jednaka 500 ms, dok je vrijeme završetka pretraživanja definirano na šest sekundi. U jednom ciklusu ovog dijela provedbe eksperimenta pretražuje se 40 datoteka, no s obzirom na definiranu frekvenciju, ovaj slučaj će biti proveden u 13 ciklusa. Na taj je način izjednačen broj pretraženih datoteka u oba slučaja ovog dijela eksperimenta. Time je ispunjen cilj prikazivanja rezultata pretraživanja s manjom frekvencijom i većim brojem pretraživanja po ciklusu te rezultata pretraživanja s većom frekvencijom i manjim brojem pretraživanja po ciklusu provedenog eksperimenta.

Posljednji dio provedbe eksperimenta jest ispitivanje funkcionalnosti brisanja datoteka. Zadatak ovog dijela eksperimenta jest pobrisati sve podatke iz MySQL i MongoDB sustava za upravljanje bazama podataka. Time jest zaokružena provedba cjelokupnog eksperimenta jer su sve prenesene, ažurirane i pretražene datoteke iz prethodnih dijelova eksperimenta pobrisane. Ovaj dio eksperimenta dijeli se u nekoliko faza. U prvoj fazi, brisanje datoteka provodi se definiranjem dva parametra: tip datoteke jednak „image/jpeg“

i kapacitet datoteke mora biti veći od 7MB. U drugoj fazi provedbe ovog dijela eksperimenta definirana su sljedeća dva parametra: tip jednak „image/jpeg“ i kapacitet datoteke mora biti manji ili jednak 7MB. U posljednjoj fazi izbrisane su datoteke čiji je tip jednak „application/pdf“. Izvršavanjem sve tri faze posljednjeg dijela eksperimenta, sve datoteke u bazama podataka jesu izbrisane. Rezultati ovog dijela provedbe eksperimenta vidljivi su u potpoglavlju 4.2.4.

Ovom postavom provedbe eksperimenta obuhvaćeni su svi navedeni podsustavi. Rezultati eksperimenata svih navedenih podsustava prikazani su kroz šest glavna grafa, od kojih je primjer istog moguće vidjeti na slici 4.1. Rezultati provedbe eksperimenta u podsustavu *Usporedba učinkovitosti prijenosa podataka u SQL i NoSQL baze podataka* uz glavne grafove sadrži i dva ključna grafa koji prikazuju prosječno vrijeme prijenosa istog podatka te prosječnu količinu korištenja RAM-a za isti preneseni podatak. Svi rezultati provedenih eksperimenata prikazani su grafovima u potpoglavlju 4.2.




*Slika 4.1:* ***Primjer grafa „Usporedba opterećenja korištenja procesora za MySQL SUBP i MongoDB SUBP“***

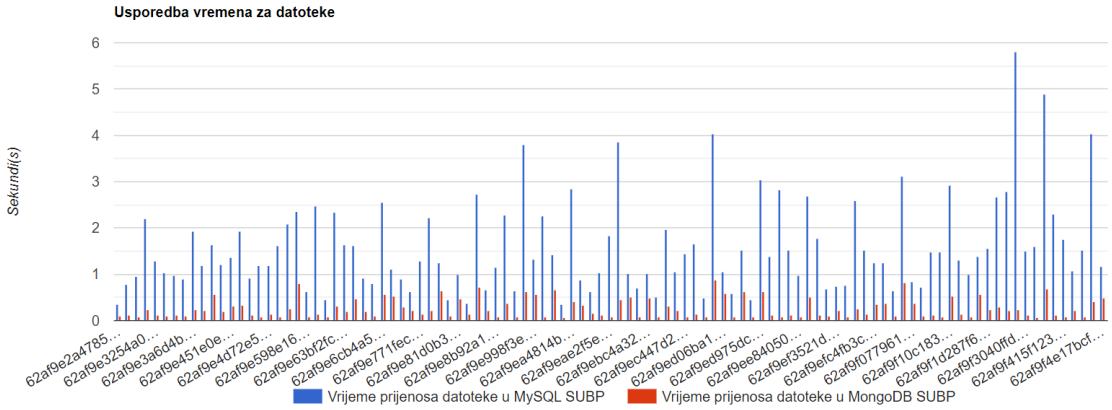
## 4.2 REZULTATI

### 4.2.1 PRIJENOS PODATAKA

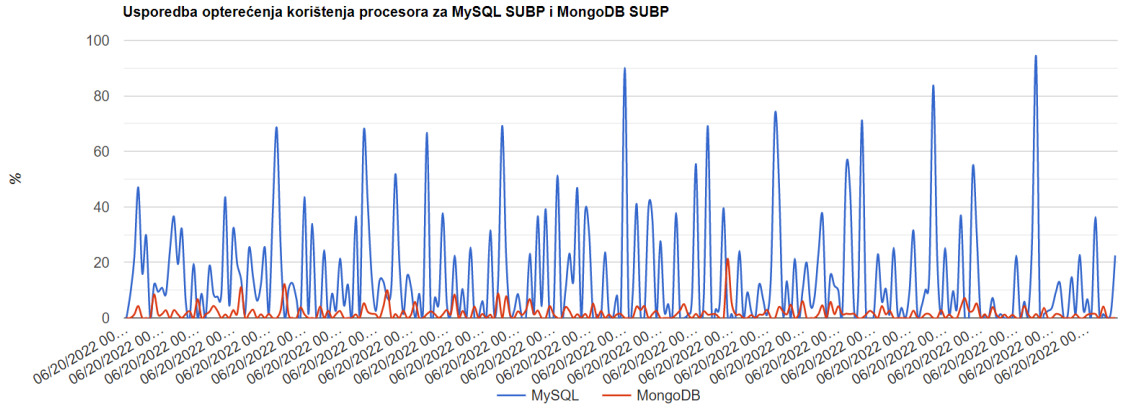
U nastavku su prikazani rezultati provedenog eksperimenta prijenosa podataka u MySQL SUBP i MongoDB SUBP. Postava eksperimenta jest definirana u prethodnom poglavlju, a rezultati istog prikazani su grafovima u nastavku. Eksperiment je dizajniran s ciljem jednakog opterećenja oba sustava za upravljanje bazama podataka.

Graf 4.2 prikazuje vrijeme potrebno za prijenos svake datoteke u MySQL i MongoDB

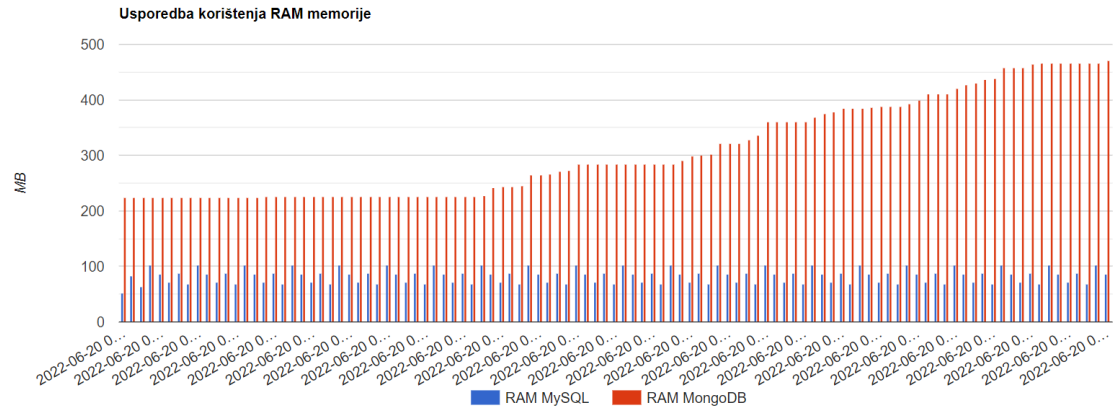
sustave za upravljanje bazama podataka. Plavom bojom označeno je vrijeme prijenosa datoteka u MySQL SUBP, dok je crvenom bojom označeno vrijeme prijenosa datoteka u MongoDB SUBP. Jednaka obilježja i nomenklature korištene su na svim prikazanim grafovima. Na navedenom grafu vidljiva je drastična razlika između vremena prijenosa datoteka navedenih sustava za upravljanje bazama podataka. U početku prijenosa podataka zamjećuje se kako je potrebno manje vremena za prijenos iste datoteke nego pred kraj prijenosa. Značajna je razlika između prijenosa podataka u MySQL i MongoDB sustave za upravljanje bazama podataka, no detaljnija usporedba vidljiva je na grafu 4.5.




*Slika 4.2:* ***Prijenos podataka - usporedba vremena prijenosa datoteka u MySQL SUBP i MongoDB SUBP***

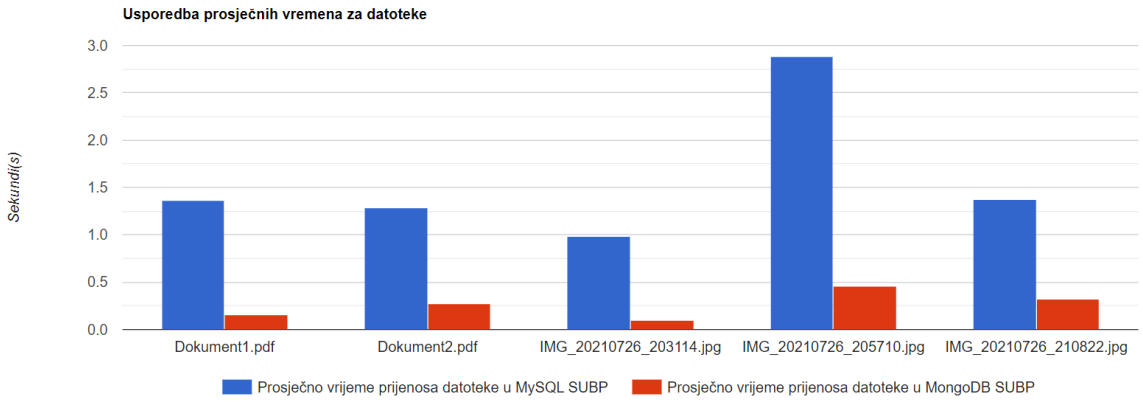



*Slika 4.3:* ***Prijenos podataka - usporedba opterećenja korištenja procesora***




*Slika 4.4:* ***Prijenos podataka - korištenje RAM memorije za svaku datoteku***

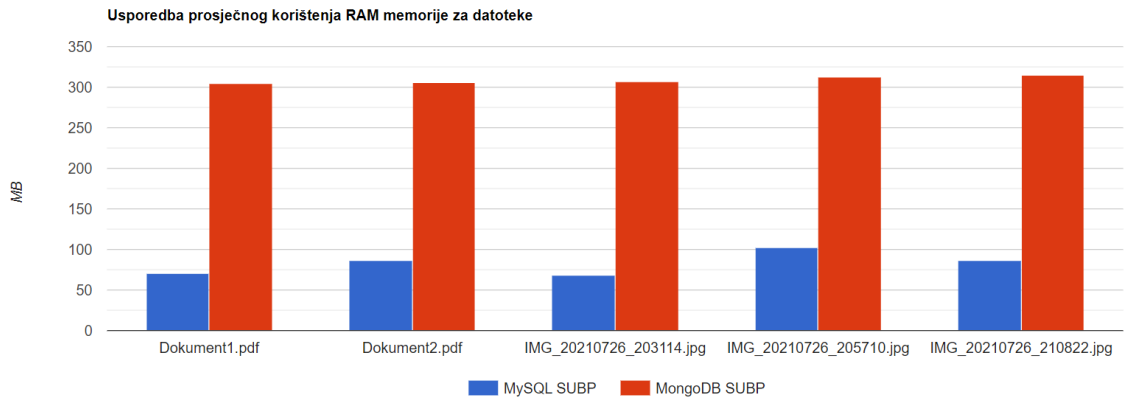
Grafovi 4.3 i 4.4 prikazuju korištenje procesora i RAM memorije pri prijenosu svake datoteke. Iz prvog grafa uočava se značajna razlika između korištenja resursa procesora MySQL i MongoDB SUBP-a. Naime, MySQL SUBP-u potrebno je mnogo više resursa procesora. Na grafu 4.3 se može zamijetiti pojavu vrlo čestog postizanja ekstrema; konkretno, korišteno je 80% ili 90% resursa procesora. Za razliku od MySQL-a, MongoDB SUBP koristi manje od 20% resursa procesora čak i za najveću datoteku u prijenosu. Usporedbom korištenja RAM memorije, zaključuje se kako je MySQL SUBP-u potrebno otprilike 100 MB, dok je MongoDB SUBP-u potrebno oko 220 MB za prijenos datoteke. Isto tako, vidljiv je drastični porast korištenja RAM memorije pri završetku prijenosa podataka. Navedeni porast je značajan. U početku je MongoDB SUBP-u bilo potrebno oko 220 MB RAM memorije, dok je pred kraj prijenosa datoteka korištenje RAM memorije naraslo do 460 MB, što predstavlja povećanje korištenja navedenog resursa za više od 100%.




*Slika 4.5:* ***Prijenos podataka - prosječno vrijeme prijenosa za iste datoteke***

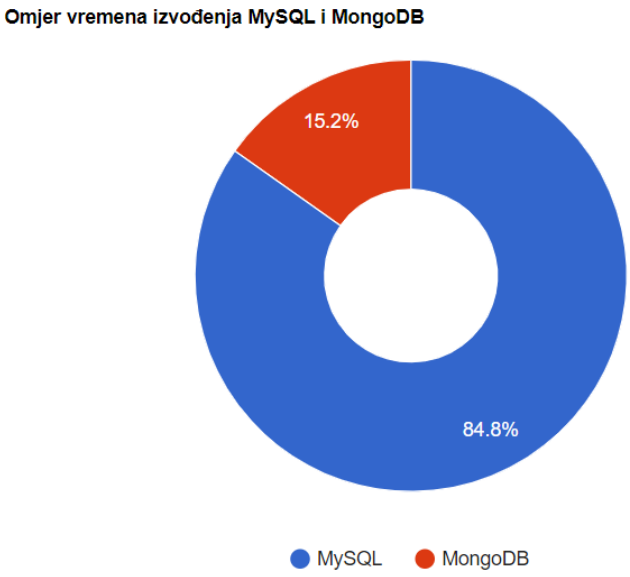
Graf 4.5 prikazuje rezultate usporedbe prosječnih vremena prijenosa za pet prenese-

nih datoteka. S obzirom na visinu stupaca, odnosno vremena prijenosa podataka, moguće je zaključiti kako je datoteka IMG_20210726_203114 najmanje veličine te datoteka IMG_20210726_205710 najveće veličine. Isto tako, vidljiva je značajna razlika između prijenosa svih datoteka u MySQL i MongoDB sustavima za upravljanje bazama podataka. Ovim grafom uočljiva je razlika između prijenosa podataka u SQL i NoSQL bazama podataka.




*Slika 4.6:* ***Prijenos podataka - prosječno korištenje RAM memorije za iste datoteke***

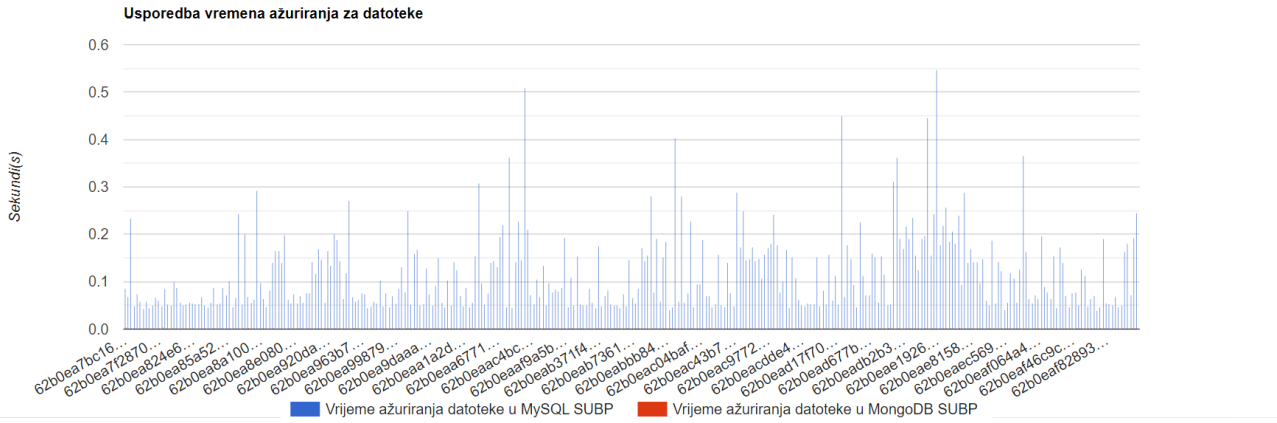
Na grafu 4.6 vidljivo je prosječno korištenje RAM memorije za iste prenesene datoteke. Može se iščitati značajna razlika između korištenja RAM memorije prilikom prijenosa podataka u MongoDB ili MySQL SUBP-u. Iz navedenog prikaza vidljivo je kako je prosječno korištenje RAM memorije u MongoDB SUBP-u otprilike 300 MB, zbog drastičnog porasta korištenja RAM memorije pri kraju prijenosa datoteka. Za razliku od MongoDB SUBP-a, MySQL SUBP otprilike koristi 70 MB, što je četiri puta manje korištene RAM memorije u usporedbi s MongoDB SUBP-om.




*Slika 4.7:* ***Prijenos podataka - omjer vremena prijenosa podataka***

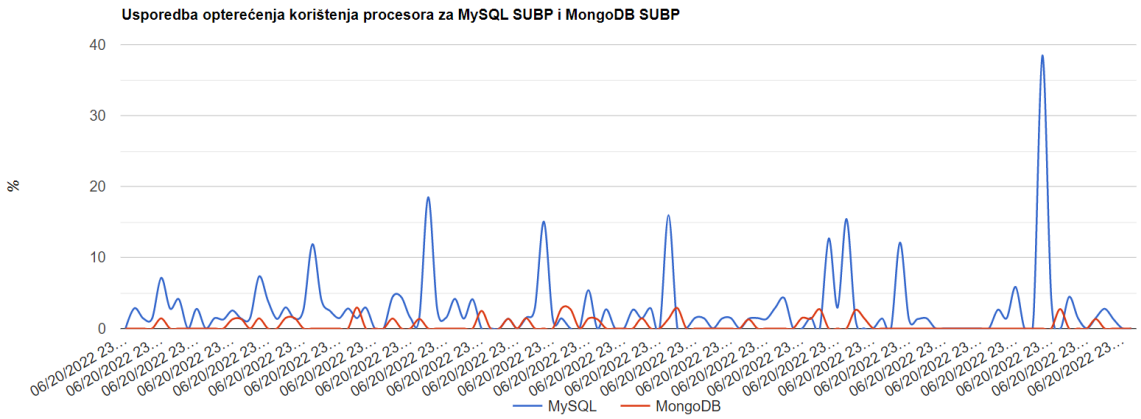
### 4.2.2 AŽURIRANJE PODATAKA

U ovom su potpoglavlju prikazani rezultati provedenog eksperimenta ažuriranja podataka, što je drugi dio cjelokupno provedenog eksperimenta. Odabrani su frekvencijski parametri kao što je navedeno u potpoglavlju 4.1 s ciljem da se opterete sustavi za upravljanje bazama podataka te da se prikažu realni rezultati.

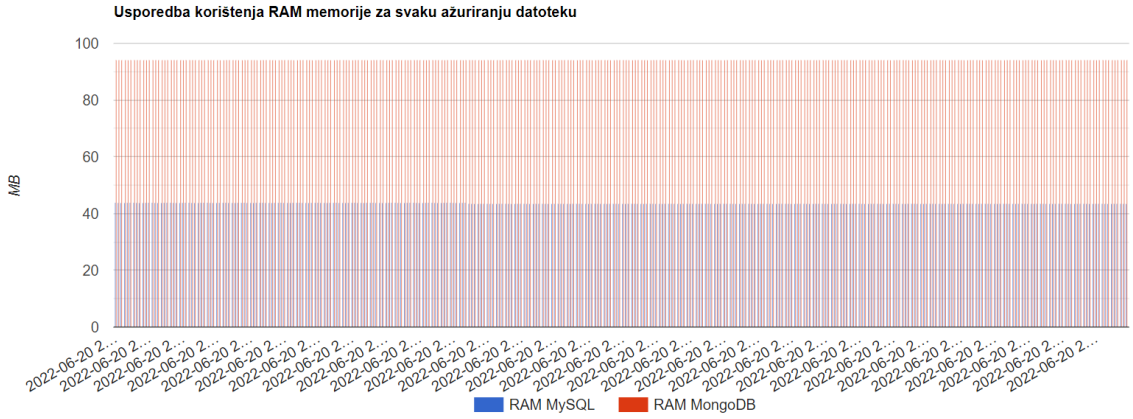



*Slika 4.8:* ***Ažuriranje podataka - prikaz vremena***

Kao što je moguće vidjeti na grafu 4.8 vrijeme potrebno za ažuriranje imena datoteka jest manje od pola sekunde za slučaj MySQL SUBP-a, dok je kod MongoDB SUBP-a vrijeme ažuriranja podataka manje od jedne desetinke sekunde. Vidljiva je značajna razlika u vremenu ažuriranja podataka između MySQL i MongoDB sustava za upravljanje bazama podataka. Detaljniji prikaz stvarne razlike između ažuriranja dviju navedenih sustava za upravljanje bazama podataka moguće je vidjeti na grafu 4.12 koji prikazuje omjere vremena ažuriranja u navedenim SUBP-ovima.



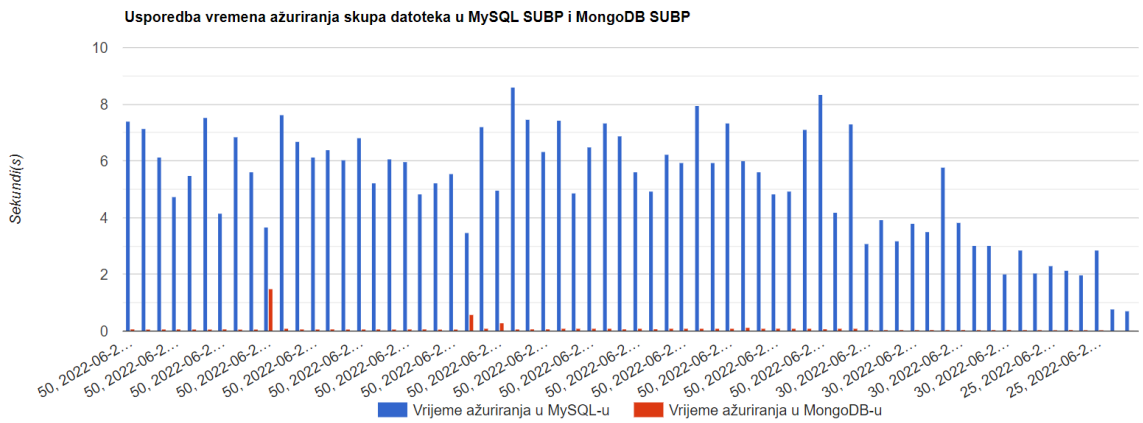

*Slika 4.9:* ***Ažuriranje podataka - usporedba opterećenja korištenja procesora***




*Slika 4.10:* ***Ažuriranje podataka - korištenje RAM memorije za svaku datoteku***

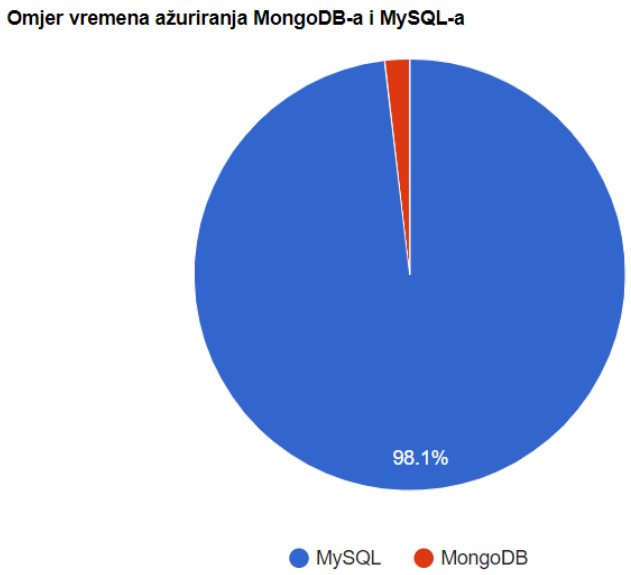
Grafovi 4.9 i 4.10 prikazuju slične rezultate kao i u dijelu eksperimenta prijenosa datoteka. MySQL SUBP koristi puno više resursa procesora u odnosu na MongoDB SUBP. S prvo navedenog grafa vidljivo je kako upotreba procesora kod MySQL SUBP-a iznosi otprilike 12%, dok kod MongoDB SUBP-a upotreba procesora iznosi 3%. U slučaju korištenja resursa RAM memorije i dalje je vidljiva značajna razlika između MySQL SUBP-a i Mon-

goDB SUBP-a. Naime, MongoDB SUBP koristi prosječno 90 MB za ažuriranje podataka, dok MySQL SUBP koristi duplo manju količinu RAM-a.




*Slika 4.11:* ***Ažuriranje podataka - arhivirana ažuriranja***

Na grafu 4.11 prikazuje se ukupno vrijeme potrebno za ažuriranje datoteka u ovom dijelu eksperimenta. Na ordinati se prikazuje vrijeme u sekundama, dok je na apscisi definiran prikaz svih arhiviranih ažuriranja. Počevši od ishodišta, naveden je zbroj vremena provedbe svakog ciklusa za sve datoteke uključene u ovaj dio eksperimenta. Rezultati su poredani silazno; prvo je prikazan najveći zbroj vremena ažuriranja datoteka u ciklusu, dok je posljednji u nizu najmanji zbroj vremena ažuriranja datoteka u ciklusu provedbe ovog dijela eksperimenta.



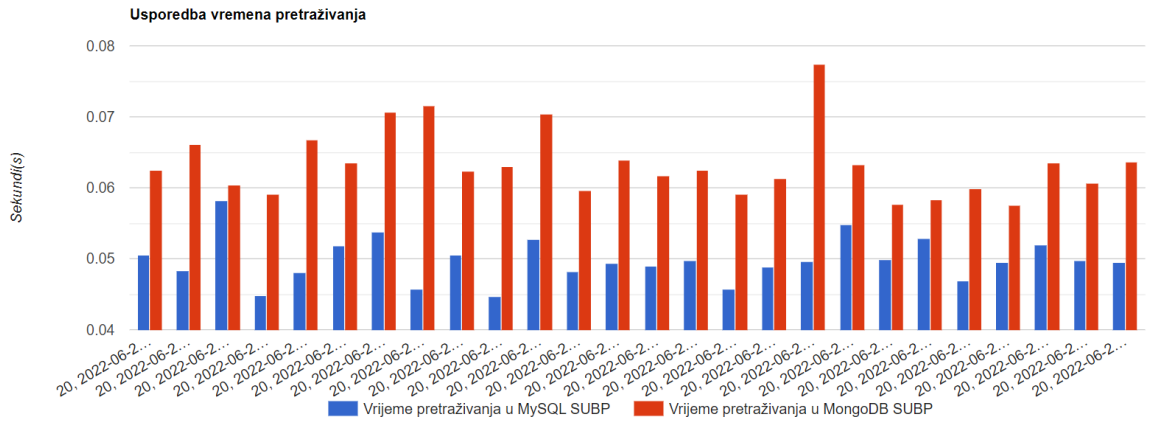

*Slika 4.12:* ***Ažuriranje podataka - omjer vremena ažuriranja***

Na posljedna dva grafa vidljiva je velika razlika između vremena potrebnog za ažuriranje datoteka kod MySQL i MongoDB SUBP-a. Slika 4.12 prikazuje omjer na temelju svih vremena arhiviranih ažuriranja u MySQL i MongoDB SUBP-u.

### 4.2.3 PRETRAŽIVANJE PODATAKA

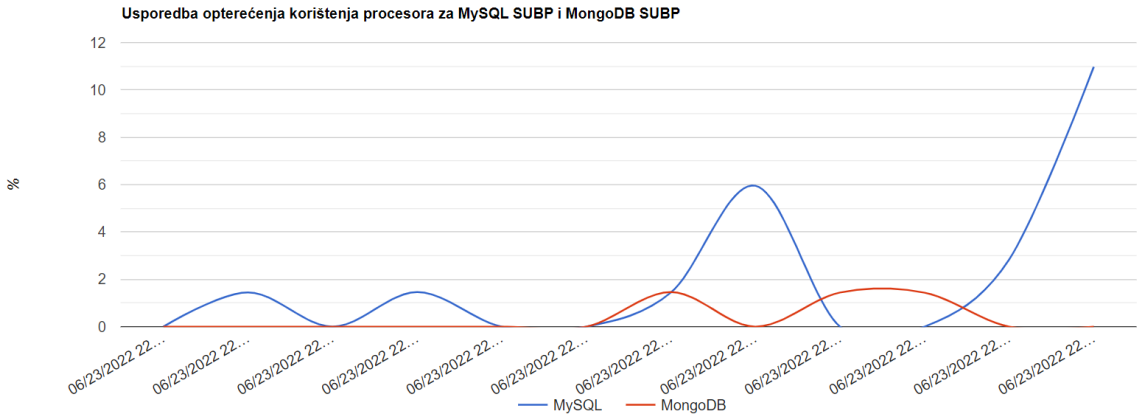
Treći dio eksperimenta odnosi se na pretraživanje podatka, što je i obrađeno u ovom potpoglavlju. Kao što je objašnjeno u poglavlju 4.1, ovaj dio eksperimenta izrađen je u dva slučaja koja su detaljno prikazana i objašnjena u nastavku.

**1. slučaj**

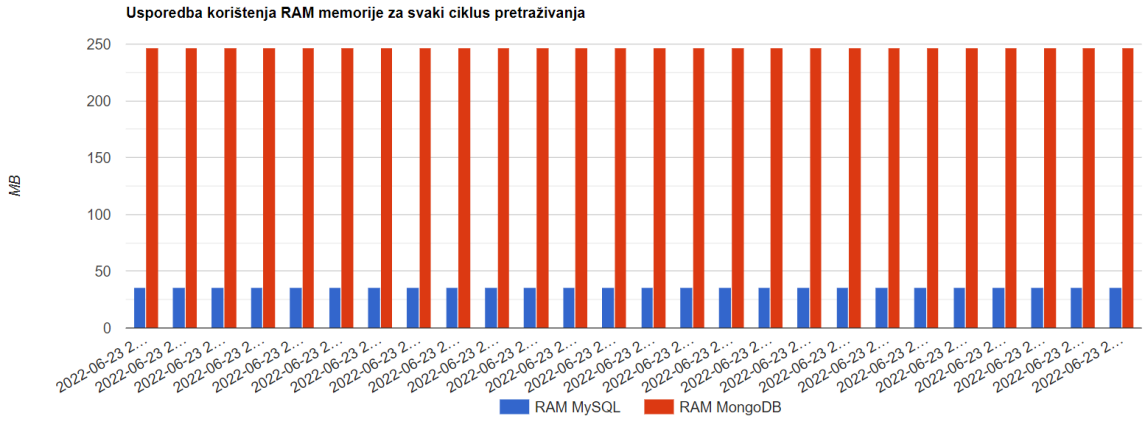



*Slika 4.13:* ***Pretraživanje podataka 1. slučaj - usporedba vremena pretraživanja za svaki ciklus***

U prvom slučaju, na grafu 4.13 prikazuju se vremena potrebna za pretraživanje. Iz grafa je moguće vidjeti kako je pretraživanje izvršeno u 26 iteracija, a pretraživano je 20 dokumenata. Pretraživanja u MySQL SUBP-u u većini slučajeva su izvršena za jednu stotinku brže u odnosu od pretraživanja unutar MongoDB SUBP-a. Isto tako, moguće je zaključiti kako pretraživanja traju jako kratko, svega šest, sedam stotinki.



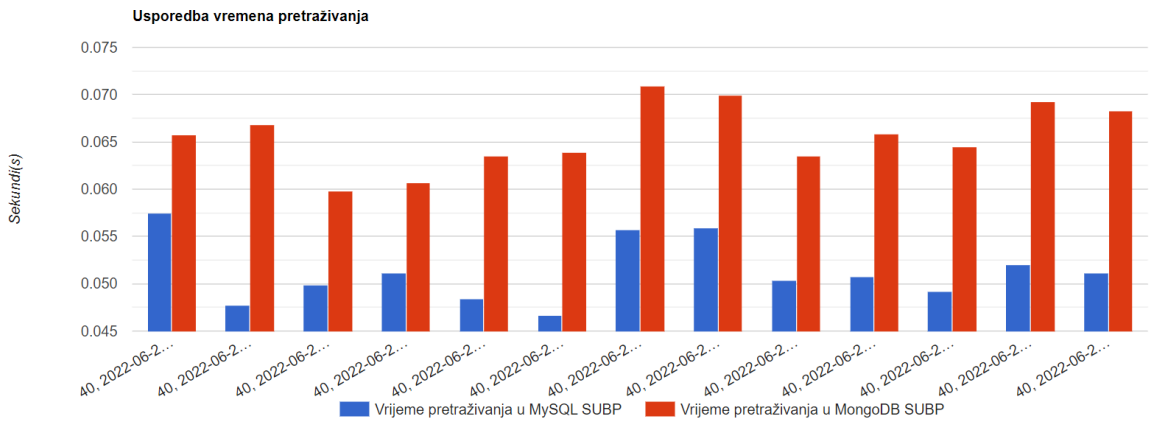

*Slika 4.14:* ***Pretraživanje podataka 1. slučaj - usporedba opterećenja korištenja procesora***



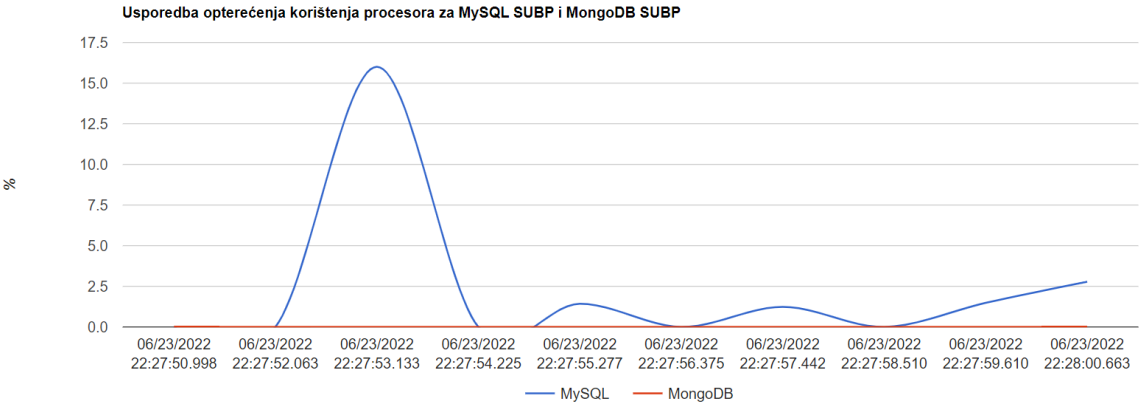

*Slika 4.15:* ***Pretraživanje podataka 1. slučaj - korištenje RAM memorije za svaki ciklus***

Upotreba resursa procesora vidljivog na grafu 4.14 prikazuje kako MongoDB SUBP pri pretraživanju koristi 2% resursa procesora, dok MySQL SUBP u nekim trenucima radi veliku razliku te dostiže do 12% resursa procesora. Razlika u upotrebi procesora između dvaju sustava za upravljanje bazama podataka vidljiva je još od operacije prijenosa i ažuriranja datoteka. Također, na grafu 4.15 je moguće zamijetiti značajnu razliku u korištenju RAM memorije. Naime, značajna razlika pri korištenju RAM memorije vidljiva je i u rezultatima provedbe prethodnih dijelova eksperimenta.

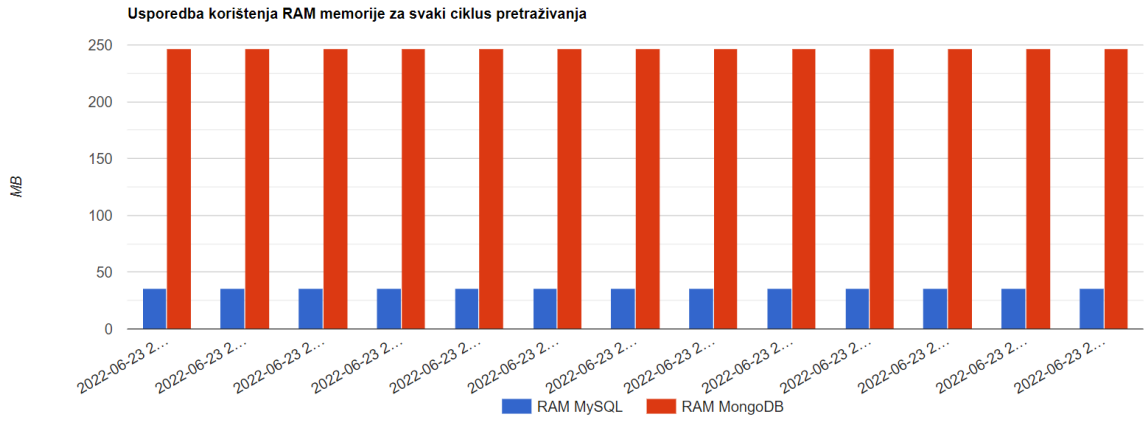
## 2. slučaj




*Slika 4.16:* ***Pretraživanje podataka 2. slučaj - usporedba vremena pretraživanja za svaki ciklus***

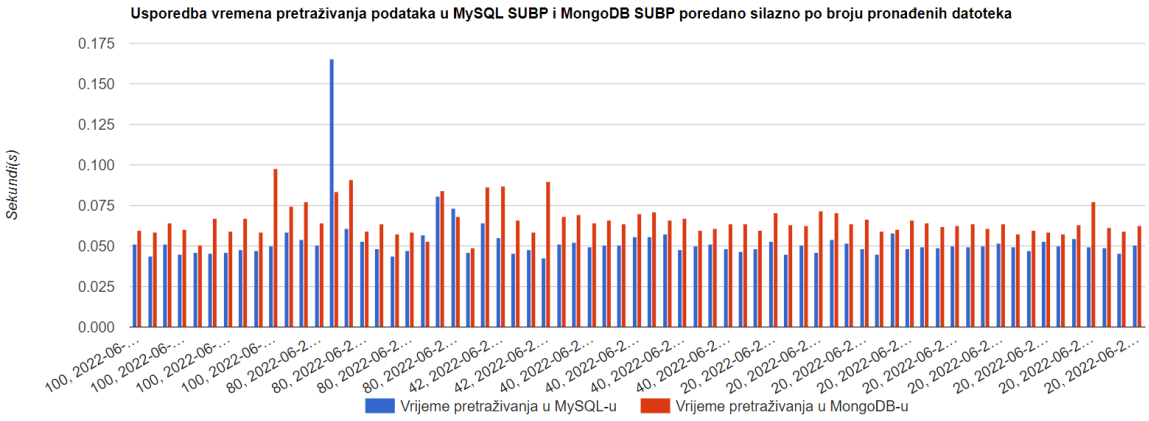



*Slika 4.17:* ***Pretraživanje podataka 2. slučaj - usporedba opterećenja korištenja procesora***

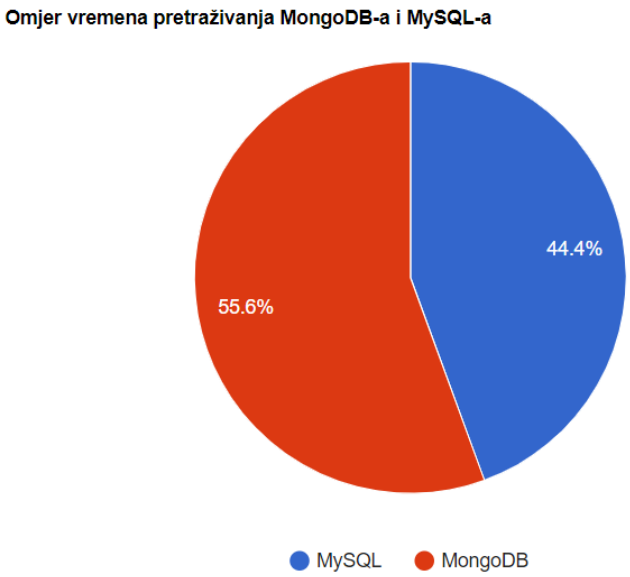



*Slika 4.18:* ***Pretraživanje podataka 2. slučaj - korištenje RAM memorije za svaki ciklus***

Cilj drugog slučaja bio je pretražiti duplo više datoteka, ali s manjom frekvencijom. Na grafovima 4.16, 4.17 i 4.18 nije vidljiva skoro pa nikakva razlika u odnosu na grafove prvoga slučaja. Iako se u drugom slučaju pretraživalo dvostruko više datoteka, vrijeme pretraživanja ostalo je isto. Nije uočena nikakva velika razlika. Korištenje RAM memorije ostalo je u potpunosti isto.




*Slika 4.19:* ***Usporedba vremena pretraživanje svih arhiviranih datoteka***



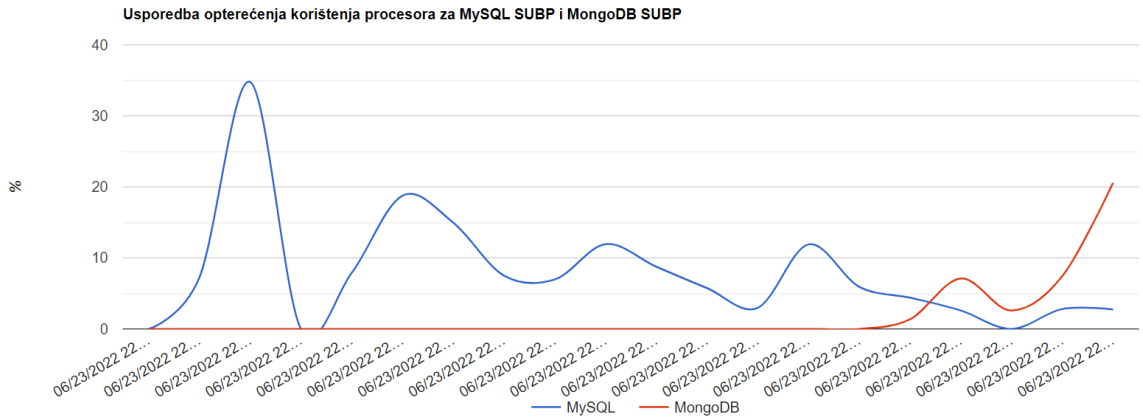

*Slika 4.20:* ***Omjer utrošenog (ukupnog) vremena za pretraživanje podataka u MySQL i MongoDB SUBP-ovima***

Zaključak operacije pretraživanja provedenog eksperimenta nameće se iščitavanjem grafa 4.19. Navedeni graf prikazuje sva arhivirana pretraživanja. S obzirom na prikazano, moguće je zaključiti kako je za pretraživanje većeg ili manjeg broja datoteka u ciklusu, potrebno slično vrijeme. U konačnici, više vremena za pretraživanje potrebno je MongoDB SUBP-u, stoga je omjer pretraživanja s obzirom na sve arhivirane podatke vidljiv na slici 4.20.

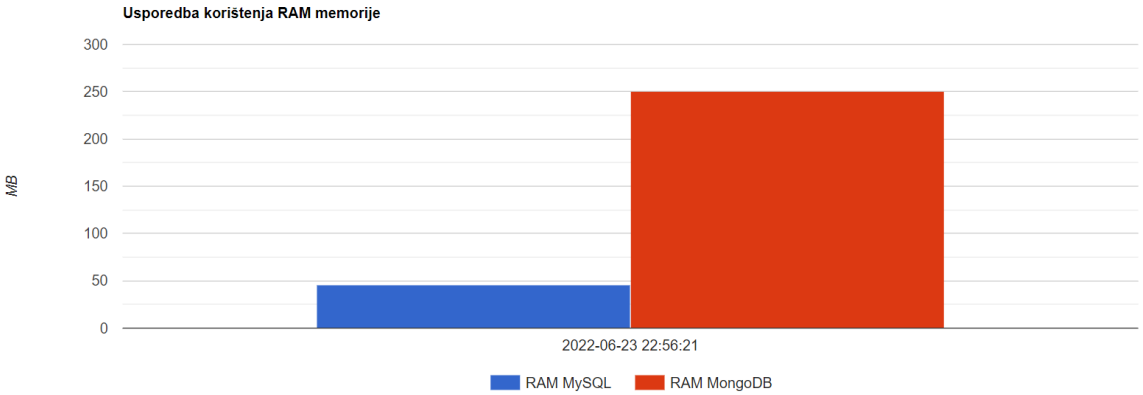
### 4.2.4 Brisanje podataka

Posljednji dio rezultata provedbe eksperimenta predstavljaju oni brisanja podataka. Ovaj dio eksperimenta sastoji se od tri faze brisanja datoteka s ciljem boljeg prikaza analize i donošenja realnih zaključaka. Faze su detaljno objašnjene i prikazane u nastavku.

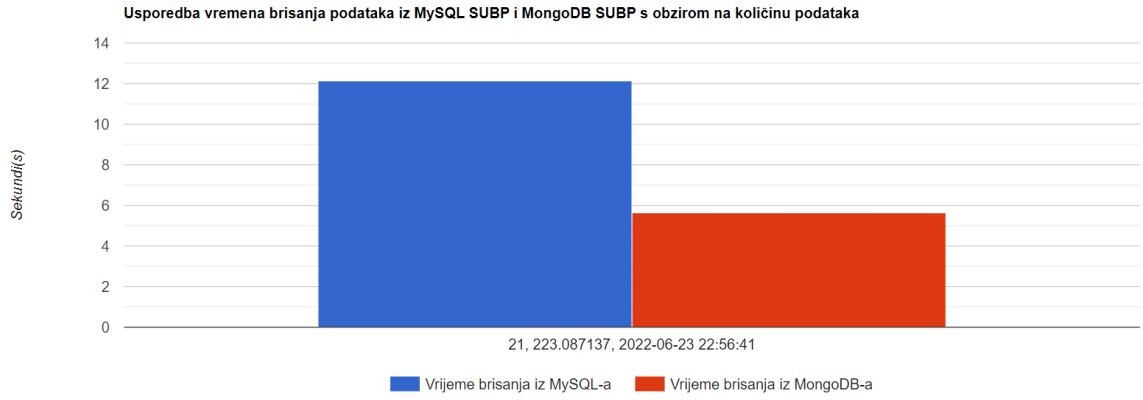
U prvoj fazi provedbe posljednjeg dijela eksperimenta pobrisane su datoteke prema parametrima navedenim u potpoglavlju 4.1. Na taj način izbrisana je 21 datoteka čiji volumni udio iznosi ukupno 223 MB.




*Slika 4.21:* ***Brisanje 1. faza - usporedba opterećenja korištenja procesora prilikom brisanja datoteka***




*Slika 4.22:* ***Brisanje 1. faza - korištenje RAM memorije***

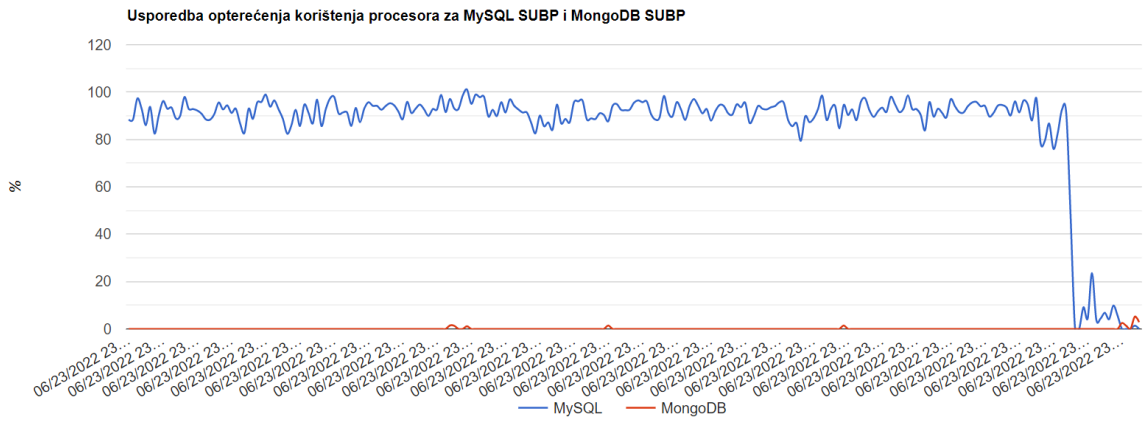



*Slika 4.23:* ***Brisanje 1. faza - vrijeme potrebno za brisanje datoteka***

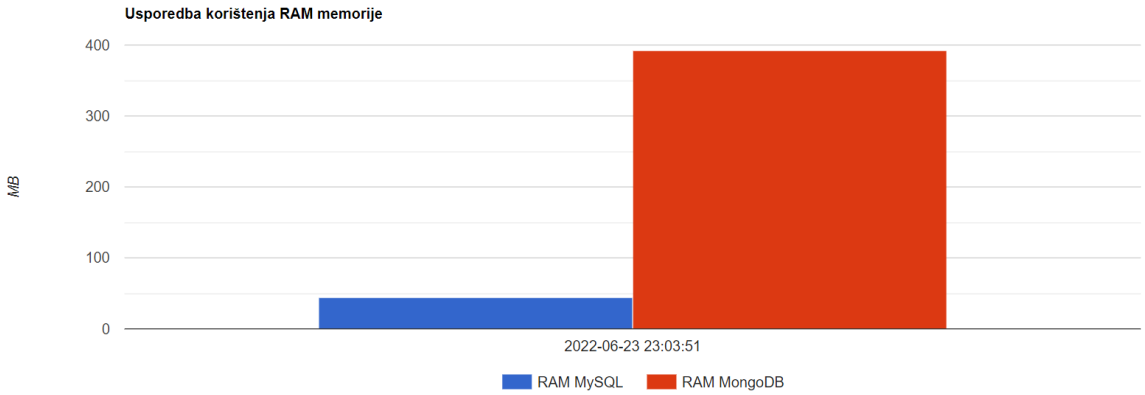
Analiza ove faze vidljiva je na grafovima 4.21, 4.22 i 4.23. Na prvo navedenom grafu prikazuje se upotreba resursa procesora potrebnog za brisanje podataka. Kao i u već provedenim dijelovima ovog eksperimenta pokazalo se kako MySQL SUBP upotrebljava puno više resursa procesora za razliku od MongoDB SUBP-a. Vidljivo je kako MySQL SUBP u nekim trenucima koristi i do 35% resursa procesora, dok kod MongoDB SUBP-a jest za istu akciju potrebno 5% - 10%. Upotreba resursa procesora ove faze kod MySQL SUBP-a je veća u odnosu na druge dvije faze koje su obrazložene u nastavku.

Na drugome grafu prikazana je upotreba korištenja RAM memorije prilikom brisanja podataka. Vidljivo je kako MongoDB SUBP-u treba 250 MB, dok MySQL SUBP-u pet puta manje navedenog resursa.

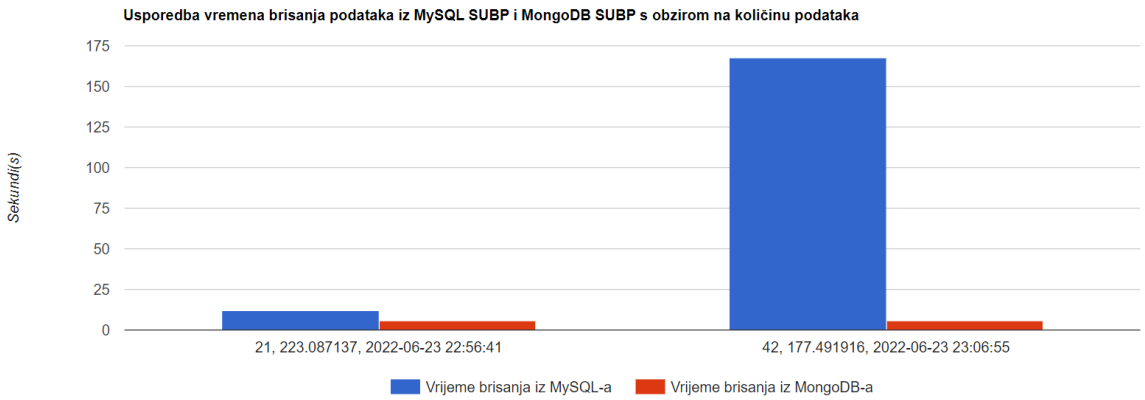
Vrijeme brisanja datoteka prikazano je na grafu 4.23. Za izvršvanje ovakve operacije MySQL SUBP-u bilo je potrebno 12 sekundi, dok su naredbe unutar MongoDB SUBP-a obavile istu akciju u polovicu toga vremena.




*Slika 4.24:* ***Brisanje 2. faza - usporedba opterećenja korištenja procesora prilikom brisanja datoteka***




*Slika 4.25:* ***Brisanje 2. faza - korištenje RAM memorije***



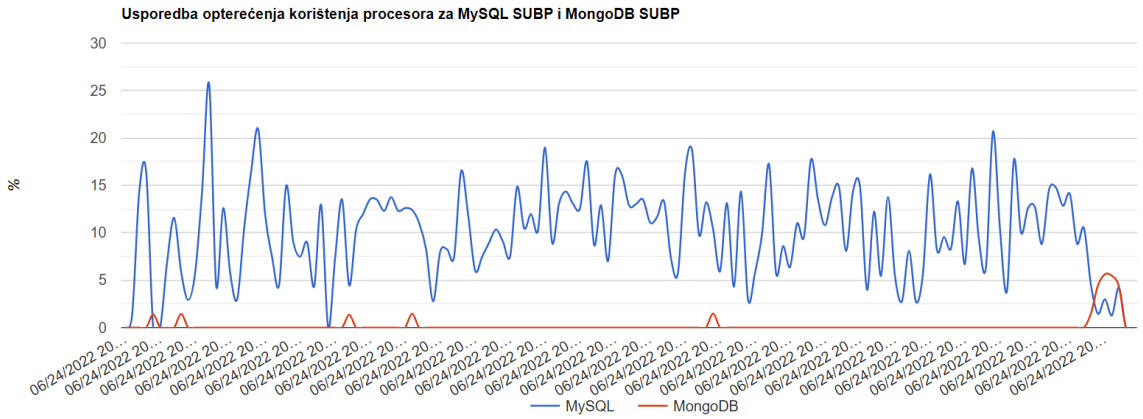

*Slika 4.26:* ***Brisanje 2. faza - vrijeme potrebno za brisanje datoteka***

Na grafovima 4.24, 4.25 i 4.26 prikazuju se rezultati druge faze provedbe eksperimenta u dijelu brisanja datoteka. U navedenoj fazi obrisane su 42 datoteke koje veličinom sveukupno iznose 177 MB. U ovom slučaju obrisano je duplo više datoteka u odnosu na prvi slučaj. Na grafu 4.24 je uočljiva značajna razlika u upotrebi resursa procesora MySQL SUBP-a i MongoDB SUBP-a. Naime, prosjek opterećenja procesora kod MySQL SUBP-a bio je oko 90%, dok je kod MongoDB SUBP-a jedva uočljivo korištenje procesora.

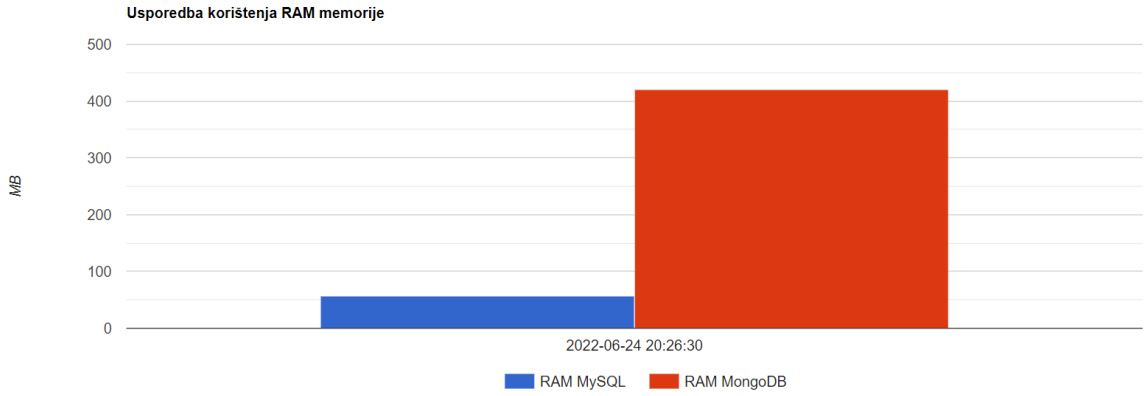
Korištenje resursa RAM memorije u drugoj fazi prikazano je grafom 4.25. MySQL SUBP pri brisanju koristi istu količinu RAM memorije kao i u prethodnoj fazi, dok MongoDB SUBP koristi 150 MB više nego li je to slučaj u prethodnoj fazi.

Posljednji rezultati ove faze prikazani su grafom 4.26, koji opisuje usporedbu vremena utrošenog prilikom brisanja svih datoteka. S obzirom da je vrijeme brisanja datoteka u

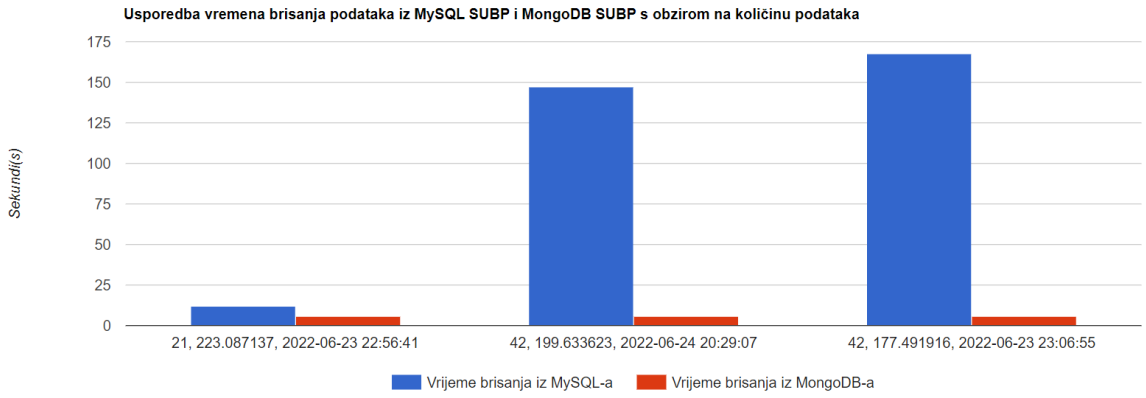
MySQL SUBP-u bilo jednako 162 sekunde te kako je opterećenje procesora u prosjeku iznosilo 90%, može se zaključiti da je prilikom izvršavanja operacije brisanja MySQL SUBP bio pod velikim opterećenjem. Na grafu je također moguće iščitati kako je MongoDB sustavu, za obavljanje operacije brisanja, bilo potrebno otprilike isto vremena kao i u prethodnom slučaju te kako postoji razlika u korištenju resursa RAM memorije.




*Slika 4.27:* ***Brisanje 3. faza - usporedba opterećenja korištenja procesora prilikom brisanja datoteka***




*Slika 4.28:* ***Brisanje 3. faza - korištenje RAM memorije***



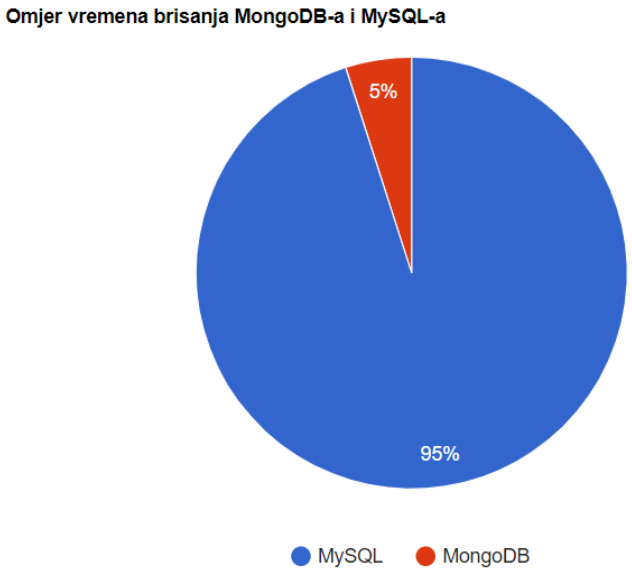

*Slika 4.29:* ***Brisanje 3. faza - vrijeme potrebno za brisanje datoteka***

Rezultati zadnje faze ovog dijela eksperimenta prikazane su na grafovima 4.27, 4.28 i 4.29. U ovoj fazi izbrisan je isti broj datoteka kao i u prethodnoj fazi, s razlikom što sveukupna količina izbrisanih datoteka ovdje iznosi 200 MB. Na grafu 4.27 je prikazana upotreba procesora u najboljem slučaju. MySQL SUBP-u u prosjeku je bilo potrebno 15% resursa procesora za izvršvanje navedene zadaće, što je puno ekonomičnija upotreba procesora u odnosu na prethodnu fazu. Kod MongoDB SUBP-a ne postoji razlika u obavljanju zadaće brisanja u obje faze.

Na grafu 4.28 prikazano je korištenje RAM memorije. Kod MySQL SUBP nema nikakve razlike pri korištenju resursa RAM-a između faza. MongoDB SUBP postiže malu razliku u odnosu na prethodnu fazu. U obje faze izbrisane su 42 datoteke. U posljednjoj fazi, ukupan zbroj veličine datoteka iznosi 200 MB. U drugoj fazi ukupan zbroj veličine datoteka iznosi 177 MB. Jasno je kako se povećanjem zbroja veličine datoteka mijenja vrijeme potrebno za brisanje istih, odnosno što je zbroj veličine datoteka veći, vrijeme potrebno za brisanje istih proporcionalno se povećava. Navedeno ne vrijedi u slučaju kada broj datoteka korištenih u eksperimentu brisanja nije jednak.

Na grafu 4.29, koji je ujedno i posljednji graf ove faze, vidljive su razlike vremena brisanja u MySQL i MongoDB SUBP-u. Vrijeme brisanja datoteka kod MongoDB SUPB-a u sve tri faze iznosi približno isto te nema velikih razlika. Kod MySQL-a to nije slučaj. Uočljiva je značajna razlika između prve i druge te prve i treće faze. U drugoj i trećoj fazi eksperimenta brisanja datoteka, korištene su 42 datoteke čiji zbroj veličine iznosi manje od zbroja veličina datoteka korištenih u prvoj fazi ovog dijela eksperimenta. Stoga razlika između prve faze i preostale dvije jest u broju zasebno izbrisanih datoteka. Iako je izbrisano duplo više datoteka, vrijeme izvršavanja operacije u druge dvije faze eksponencijalo

je veće u odnosu na prvu. Na slici 4.30 moguće je vidjeti konačni omjer izbrisanih datoteka s obzirom na sve provedene faze brisanja.




*Slika 4.30:* ***Omjer ukupnog vremena brisanja datoteka***

## 4.3 DISKUSIJA

Rezultati provedenih eksperimenata pokazali su kako postoje određene prednosti te nedostaci oba sustava za upravljanje bazama podataka. Važno je naglasiti kako je upotreba procesora kod MySQL SUBP-a puno veća u odnosu na MongoDB SUBP. Isto se odnosi za sve dijelove provedenih eksperimenata, odnosno za sve operacije nad podacima. Korištenje resursa RAM memorije veće je prilikom provedbe eksperimenata u MongoDB SUBP-u u odnosu na MySQL SUBP. Rezultati pokazuju kako MongoDB SUBP prilikom manipulacije nad podacima koristi duplo više resursa RAM memorije.

Prva operacija korištena u eksperimentu je prijenos datoteka. Obavljen je prijenos pet datoteka različitih veličina s određenom, točno definiranom frekvencijom. U najmanjoj datoteci veličine 1.86 MB, za prijenos podataka u MySQL SUBP bila je potrebna jedna sekunda, dok je za prijenos podataka u MongoDB SUBP to izvedeno za 0.1 sekundu. Najvećoj prenesenoj datoteci veličine 10.1 MB, bilo je potrebno 2.75 sekundi za prijenos u MySQL SUBP, odnosno 0.45 sekundi za prijenos putem MongoDB SUBP-a. Navedena razlika u vremenu prijenosa iskazuje kako je MySQL SUBP potrebno 84.8%, a MongoDB SUBP-u 15.2% resursa procesora. Korištenje resursa RAM memorije duplo je veće u slu-

čaju prijenosa podataka u MongoDB SUBP i iznosi u prosjeku 200 MB. Problem se pritom javljao kod dužine prijenosa podataka. Što je duži prijenos podataka, to je korištenje resursa RAM memorije eksponencijalno raslo. Zbog toga je prilikom prijenosa podataka, u nekim trenucima MongoDB SUBP dostigao vrijednost od 460 MB iskorištenih resursa RAM memorije. Na temelju svih ovih podataka moguće je zaključiti kako je MongoDB SUBP puno efikasniji pri prijenosu datoteka, uz otegotnu okolnost povećeg trošenja resursa RAM memorije. Kod MySQL SUBP-a slučaj je obrnut.

Drugi dio provedenog eksperimenta jest ažuriranje naziva datoteka pohranjenih u bazama podataka. Ažuriranje pojedinačnih naziva datoteka vremenski je iznosilo između 0.05 i 0.4 sekunde za MySQL SUBP te manje od 0.05 sekundi za MongoDB SUBP. Zbog toga moguće je zaključiti kako samo mijenjanje naziva datoteka ne predstavlja posebno opterećenje za sustave upravljanja bazama podataka te kako obje baze podataka izvršavaju tu operaciju za manje od pola sekunde. U ciklusu u kojem se ažuriralo 50 datoteka u MySQL SUBP-u, bilo je potrebno šest sekundi za izvršavanje, dok je ciklusu u kojem se ažuriralo duplo manje datoteka bilo potrebno u prosjeku 2 sekunde. Na temelju toga se dolazi do zaključka kako ciklusu sa više datoteka koje je potrebno ažurirati, treba i više vremena za navedeno, u slučaju korištenja MySQL SUBP. Kod MongoDB SUBP to nije slučaj jer se tada svaki ciklus, neovisno o broju datoteka, obavlja za manje od 0.2 sekunde. Stoga se može zaključiti kako MongoDB SUBP značajno brže ažurira nazive datoteka te kako je izrazito pogodan kod ažuriranja velikog broja naziva datoteka. Sve navedeno potvrđuje graf 4.12 koji prikazuje kako vrijeme ažuriranja datoteka unutar MySQL SUBP-a iznosi 98.1% dok je kod MongoDB SUBP-a ta vrijednost 1.9%. Korištenje resursa RAM memorije u prosjeku kod ažuriranja datoteka u MongoDB SUBP-u iznosi 90 MB, što je puno manje u odnosu na korištenje resursa RAM memorije prilikom operacije prijenosa podataka. U slučaju korištenja MySQL SUBP-a nije izražena tolika razlika. U operaciji ažuriranja iznosi 45 MB u prosjeku, dok u operaciji prijenosa datoteka iznosi u prosjeku 80 MB.

Treći dio provedenog eksperimenta jest pretraživanje datoteka pohranjenih u bazama podataka. Operacija pretraživanja datoteka prikazuje kako je za pretraživanje u ciklusima koji sadrže 20 ili 100 datoteka potrebno u prosjeku 0.05 sekundi prilikom korištenja MySQL SUBP-a te 0.06 sekundi prilikom korištenja MongoDB SUBP-a. Vidljivo je kako je ovo najbrža operacija od svih provedenih u eksperimentu. Primjerice, za ciklus pretraživanja 100 datoteka potrebno je 0.05 sekundi u slučaju korištenja MySQL SUBP-a, dok je za ciklus ažuriranja 20 datoteka potrebno dvije sekunde. U slučaju korištenja MongoDB SUBP-a, pretraživanje i ažuriranje podataka u bazi sadrži otprilike jednake vrijednosti. Na temelju

podataka moguće je zaključiti kako je potrebno više vremena za pretraživanje podataka u MongoDB SUBP-u iz razloga jer isti ne sadrži relacije. Uočeno je kako se prilikom pretraživanja podataka pomoću MongoDB SUBP-a znaju dogoditi ekstremi. U jednom slučaju za pretraživanje podataka je potrebno 0.06 sekundi, dok je u drugom slučaju potrebno 0.5 sekundi. Ulazni parametri su jednaki.

Posljednji dio provedenog eksperimenta je brisanje datoteka koje su bile pohranjenje tokom provedbe prvog dijela eksperimenta. Ovaj dio eksperimenta podijeljen je u tri faze. Ovakva podjela eksperimenta u dijelu brisanja datoteka osmišljena je zbog prikaza različitih opterećenja sustava uzrokovanih definiranjem različitih vrijednosti ulaznih parametara prilikom izvršavanja eksperimenta. U prvoj fazi izbrisana je 21 datoteka. Vrijeme potrebno za brisanje datoteka u ovoj fazi eksperimenta iznosilo je 12 sekundi prilikom korištenja MySQL SUBP-a te 5.6 sekundi prilikom korištenja MongoDB SUBP-a. U provedbi druge faze izbrisano je duplo više datoteka u odnosu na prethodnu fazu. Za provedbu druge faze ovog dijela eksperimenta vrijeme potrebno za izvršavanje operacije brisanja iznosilo je 167 sekundi prilikom korištenja MySQL SUBP-a, dok je za izvršavanje iste operacije u ovog fazi eksperimenta prilikom korištenja MongoDB SUBP-a iznosilo 5.9 sekundi. Iz navedenog je moguće zaključiti kako je razliku u vremenu izvršavanja ove operacije između testiranih SUBP-a značajna. Također je uočljivo kako je mnogo više vremena za obavljanje operacije brisanja MySQL SUBP-u bilo potrebno prilikom druge faze provedenog eksperimenta u odnosu na prvu fazu ovog dijela provedenog eksperimenta. U trećoj fazi izbrisano je isti broj datoteka kao i u prethodnoj fazi. Rezultati prikazuju kako je razlika u vremenu izvršavanja operacija brisanja prilikom korištenja spomenutih SUBP-a značajna. Razlika u vremenu provedbe operacije brisanja između faza provedenog eksperimeta je vidljiva zbog zbroja veličine ulaznih datoteka korištenih u eksperimentu.

# 5 ZAKLJUČAK

Rezultati ovog rada pokazali su kako je razlika u prijenosu, ažuriranju, pretraživanju i brisanju podataka između SQL i NoSQL baza, upravljanima promatranim SUBP-ovima, jasno vidljiva. Višestruko brže se odvijaju operacije nad podacima u MongoDB SUBP-u, dok standardna provedba operacija nad podacima u MySQL SUBP-u pomoću upita zahtjeva mnogo više vremena i resursa. Sve navedeno može se isčitati iz provedene analize manipulacije nad podacima.

Uz brzinu izvršavanja operacija nad podacima u bazi, istražena je razina opterećenja procesora te trošenje resursa RAM memorije. U slučaju korištenja procesora pokazalo se kako je NoSQL baza podataka učinkovitija u korištenju navedenih resursa prilikom manipulacije nad podacima, dok je MySQL SUBP ekonomičniji prilikom trošenja resursa RAM memorije u odnosu na MongoDB SUBP. MySQL SUBP, uz sva pozitivna svojstva koja se vežu za isti, svojim relacijskim modelom, u provedenim eksperimentima, ne može konkurirati NoSQL bazi podataka. Arhitektura sustava zasnovana na dokumentima iznjedrila je optimalniju potrošnju resursa testiranih u pomno osmišljenim eksperimentima.

Osim veličine podataka s kojima se manipulira u bazi, na konačan ishod utječu i ulazni parametri, odnosno frekvencija upisivanja i zadano vrijeme završetka upisa provedenog eksperimenta. Što je frekvencija upisivanja veća, ukupno je potrebno više vremena za izvršiti neku od operacija nad podacima u bazi podataka.

U konačnici se može zaključiti kako je tehnologija NoSQL mnogo pogodnija za korištenje zbog izvrsnih vremenskih performansi, pogotovo u slučaju kada je potrebno izvršiti operacije nad većim skupom podataka istovremeno. SQL baza podataka daje veću kontrolu prilikom manipulacije podataka, no evidentno je sporija, a zahtjeva i poznavanje SQL jezika te stvaranja upita prema bazi.

# 6 LITERATURA


[1] Ljubić S.: „Razvojni ciklus i relacijski model“, prezentacija kolegija Baze podataka, ak.god. 2021./2022.

[2] Ljubić S.: „SQL jezik“, prezentacija kolegija Baze podataka, ak.god. 2021./2022.

[3] Sullivan D.: „NoSQL for Mere Mortals“, Addison-Wesley Professional, 1st edition, USA, 2015.

[4] Koceić Bilan N.: „Primijenjena statistika“, Prirodoslovno-matematički fakultet u Splitu, 2011.

[5] Document Database - NoSQL,
https://www.mongodb.com/document-databases

[6] DB-Engines Ranking,
https://db-engines.com/en/ranking

[7] Comparing Database Management Systems,
https://www.altexsoft.com/

[8] MongoDB - The application data platform,
https://www.mongodb.com/

[9] MongoDB Compass,
https://www.mongodb.com/products/compass

[10] Funkcija selectGridFSBucket(),
https://github.com/mongodb/MongoDB-selectGridFSBucket.txt

# Sažetak

Relacijske baze podataka zasnovane su na semantičkim relacijama između entiteta koji su određeni svojim skupom atributa. Za pristup i manipulaciju podacima koristi se standardizirani jezik SQL. S druge strane, NoSQL baze podataka ne zasnivaju se na relacijskom modelu, već koriste pohranu u modalitetima dokumenata, uređenih parova ključ-vrijednost, širokih stupaca te grafova. U ovome radu implementirana je web aplikacija kojom se testirala učinkovitost dvaju sustava za upravljanje bazama podataka. Kao predstavnik upravitelja relacijskom bazom podataka odabran je sustav MySQL, a za NoSQL baze sustav MongoDB. Ispitna aplikacija omogućava testiranje potpornih baza podataka na način da se automatski može zapisivati, ažurirati, brisati i pretraživati zapise, pri čemu podaci u pojedinom zapisu mogu biti različite veličine i pri čemu se koristi različita frekvencija dotične operacije. Izlazne metrike obuhvaćaju vrijeme odziva baze podataka (tj. vrijeme izvršavanja operacije) i potrošnju računalnih resursa – opterećenje procesora i utrošak memorije. Provedeni su eksperimenti za različite konfiguracije ulaznih varijabli, a rezultati, koji ukazuju na različitu učinkovitost sustava za upravljanjem relacijskih i NoSQL baza podataka, prezentirani su na razini deskriptivne statistike.

***Ključne riječi* — relacijske baze podataka, NoSQL baze podataka, MySQL, MongoDB, analiza učinkovitosti**

# Abstract

Relational databases are based on semantic relations between entities determined by a set of attributes. The standardized SQL language is used to access and manipulate the data. NoSQL databases, on the other hand, are not based on a relational model, but use documents, key-value pairs, wide-column stores, and graphs as storage modalities. In this thesis, a web application was implemented to test the effectiveness of two database management systems. The MySQL system was chosen to represent relational database management and the MongoDB system was chosen to represent NoSQL database management. The provided application allows testing the supporting databases in such a way that it is possible to automatically insert, update, delete and search records, where the data in a single record can be of different sizes and a different frequency of each operation is used. Output metrics include database response time (i.e., operation execution time) and computational resource consumption - CPU and memory load. Experiments were conducted for different configurations of input variables, and the results, indicating different efficiency of relational and NoSQL database management systems, were presented at the descriptive statistics level.

***Keywords* — relational databases, NoSQL databases, MySQL, MongoDB, performance analysis**