# Prednosti i nedostaci dokumentno orijentiranih baza podataka

**Barić, Matija**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni preddiplomski studij računarstva**

# Prednosti i nedostaci dokumentno orijentiranih baza podataka

**Završni rad**

**Matija Barić**

**Osijek, 2021**

# SADRŽAJ

# 1. UVOD

Razvoj tehnologije svakodnevno sve više omogućava visoku razinu povezanosti i dostupnosti informacijama na globalnoj razini te istovremeno predstavlja problem pohranjivanja i pristupanja istim informacijama. Zbog toga se pojavljuje potreba za implementacijom baza podataka, koje su kolekcije informacija i atributa koji ih opisuju. Baze podataka su sveprisutna tehnologija koja se koristi za organizaciju podataka u velikim i malim institucijama, obradama kreditnih kartica, pohranu elektroničke pošte i na kraju krajeva, u čuvanju više-manje svih podataka na Internetu. Najčešća podjela baza podataka se svodi na dva glavna tipa; relacijske baze podataka i ne-relacijske ili baze podataka bez slijeda.

U ovom radu će se govoriti o prednostima i nedostacima jedne on ne-relacijskih baza podataka, to jest dokumentno orijentiranih baza podataka, naspram relacijskih baza podataka. Rad je podijeljen na pet poglavlja: uvod, baze podataka, alati, praktični dio rada i zaključak.

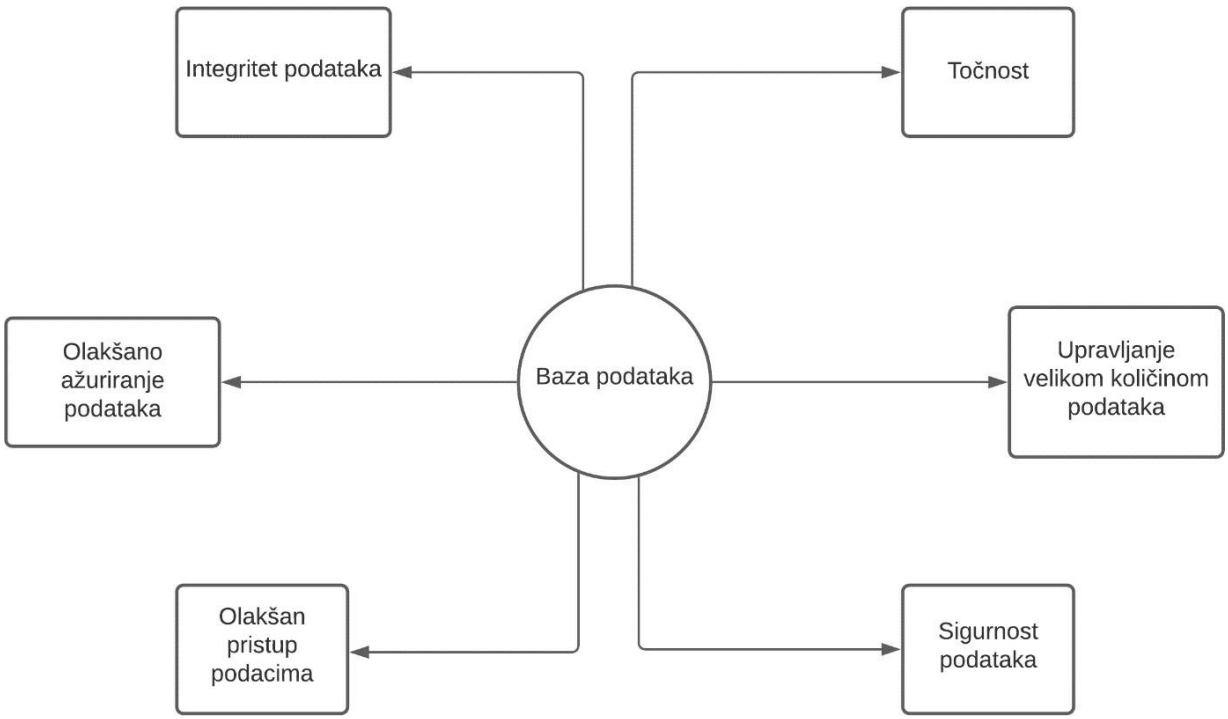
U drugom poglavlju opisati će se koncept baze podataka i čitatelju približiti dva tipa baza koje uspoređujemo; relacijske baze podataka i dokumente baze podataka. U trećem poglavlju se daje teorijska podloga za alate koji se koriste prilikom kreiranja i rada sa opisanim bazama podataka. Četvrto poglavlje uspoređuje performanse različitih tipova baza te ističe prednosti i mane istih. U ovom poglavlju će se promatrati brzine potrebne da se obavi određena operacija nad bazom, brzinu dohvaćanja i grupiranja podataka te količina memorije koju će ta baza potom zauzeti.

## 1.1. Zadatak završnog rada

U ovom radu prikazat će se prednosti i nedostaci dokumentno orijentiranih baza podataka, naspram relacijskih baza podataka. Pri tome, cilj je usporediti brzinu upisa u bazu za iste podatke, brzinu dohvaćanja i grupiranja podataka te veličinu baza kada se u njih upišu isti podaci. Dokumentno orijentirana baza podataka izgrađena je korištenjem platforme *MongoDB*, dok se relacijska baza podataka realizira pomoću sotvera *Microsoft SQL*.

# 1. BAZE PODATAKA

Baza podataka je kolekcija međusobno povezanih podataka pri čemu se treba osigurati dijeljenje istih među različitim aplikacijama i korisnicima uz uvjet da postoji jedinstveni i kontrolirani pristup unosu, brisanju, izmjeni i dohvaćanju podataka. Obično se dizajnira na način koji omogućava jednostavno pohranjivanje i pristup informacijama koje sadrži. Glavni razlog kreiranja baze je česta potreba za postavljanjem upita nad podacima, njihovo sortiranje i manipulacija istim prema potrebama korisnika. Grafički prikaz baze podataka i bitnih faktora prikazan je ne Slici 2.1.




*Slika 2.1. Baza podataka*

Pravilno osmišljena baza podataka je ključni faktor u bilo kojoj instituciji ili organizaciji. To proizlazi iz činjenice da baze sadrže sve ključne podatke nekog entiteta, kao što su evidencija zaposlenika, zapisi o provedenim transakcijama, detalji o plaćama, itd. Svaka baza podataka ima osobine koje su zajedničke za sve sustave koji ih implementiraju. Apstraktni model podataka[5] organizira elemente i standardizira njihov međusobni odnos i svojstva entiteta iz

stvarnog svijeta. Baza podataka može imati definiranu hijerarhiju i prava pristupa koja su dana ovlaštenim korisnicima kako bi se olakšao i osigurao pristup podacima koji se nalaze u bazi. Jedan od najbitnijih upita ove značajke je takozvani *MapReduce* o kojem će se detaljniji govoriti u trećem poglavlju. Sustav također treba pružati mogućnost upravljanja transakcijama u okruženju sa više korisnika baze te kontrolu pristupa i vlasništvo nad podacima. Podatci koji se nalaze unutar baze trebaju biti konzistentni te se stoga ugrađuju rutine koje osiguravaju da su isti točni i korisni i takve rutine često nazivamo „pravila provjere valjanosti“, „ograničenja provjere valjanosti“ ili „provjere rutine“. Podatci također trebaju biti osigurani u slučaju ispada sustava i/ili strojne opreme, pa su poželjne sigurnosne kopije podataka u sustavu koji ih sadržava.

## 2.1. **Relacijske baze podataka**

Relacijska baza podataka je tip baze podataka koja pohranjuje i pruža pristup podatkovnim točkama koje su međusobno u relaciji. Baziraju se na relacijskom modelu, intuitivnom i izravnom načinu prikazivanja podataka u tablicama. U relacijskim bazama podataka, svaki red u tablici je bilješka sa jedinstvenim ID-om koji se naziva ključ. Stupci tablice sadrže atribute podataka, a svaka bilješka obično ima vrijednost za svaki atribut, što olakšava uspostavu veza između podatkovnih točaka.

Strukturirani podaci se također mogu prikazivati u polustrukturiranom modelu. Shema je u relacijskim bazama podataka matematički opisana kao *r1(A, B, C), r2(C, D),* gdje su *r1* i *r2* imena relacija, te *A, B, C* I *C, D* nazivi stupaca u te dvije relacije. Vrijednosti relacija prikazani su kao niz podataka unutar objekta. Uz ovo, trebalo bi definirati i domenu svih stupaca. Struktura i vrijednosti u relacijama mogu se vidjeti na slici 2.1.

| r1 | A | B | C |
|---|---|---|---|
| | a1 | b1 | c1 |
| | a2 | b2 | c2 |

| r2 | C | D |
|---|---|---|
| | c2 | d2 |
| | c3 | d3 |
| | c4 | d4 |

*Slika 2.1. Prikaz relacija i njihovih vrijednosti*

Podaci iz relacija se također mogu prikazati kao polustrukturirani tip podatka, kao što je prikazanu u kodu na slici 2.2. Kroz polustrukturirani model moguće je prikazati strukturirane podatke. Glavna prednost polustruktirianog tipa podatka se očituje u činjenici da objekti nisu obvezani implementirati iste atribute te atributima nije potrebno da tip podatka bude isti.

```
{
r1: {
        redak: {
            { a: a1, b: b1, c: c1 }
        },
        redak: {
            { a: a2, b: b2, c: c2 }
        }
    },
 r2:  {
        redak: {
            { c: c2, d: d2 }
        },
        redak: {
            { c: c3, d: d3 }
        },
        redak: {
            { c: c4, d: d4 }
        }
    }
}
```

*Slika 2.2. Prikaz podataka iz relacija u polustrukturiranom obliku*

Može se uočiti da su vrijednosti redova iz relacija *r1* i *r2* prikazane kao niz podataka unutar objekata za pojedinu relaciju.

## 2.2. **Dokumente baze podataka**

Dokumentne baze podataka pohranjuju podatke u obliku dokumenata, za razliku od tradicionalnih relacijskih baza podataka koje podatke pohranjuju u redove i stupce unutar tablica. Dokumenti pohranjeni u bazu podataka najčešće su strukturirani koristeći JSON format. Takav oblik dokumenata je intuitivan i prirodan način modeliranja podataka, te odgovara paradigmi dokumentno orijentiranog programiranja, gdje svaki dokument predstavlja jedan objekt. Svaki dokument se može sastojati od jednog ili više atributa, koji mogu imati vrijednosti kao što su broj, *string*, *boolean* ili niz. Ovaj način pohrane podataka obično rezultira u bržem dohvaćanju i ažuriranju podataka, no to i dalje ovisi o načinu na koji se baza koristi. Primjer dokumenta u bazi podataka dan je na slici 2.3.


*Slika 2.3. Prikaz dokumenta i vrijednosti koje sadrži*

Jedna od glavnih prednosti dokumentnih baza podataka je smanjeno vrijeme pristupa podacima koji se nalaze unutar baze iz razloga što se isti vežu na dokument u koji se pohranjuju. Time se uklanja potreba za kompleksnih spajanjem tablica. To također znači da je pretraživanje vrlo jednostavno, jer se podatci najčešće mogu pretraživati prema bilo kojem atributu unutar dokumenta.

Shema u ovakvom tipu baze podataka zapravo ne postoji, iz razloga što svaki dokument može sadržavati različite atribute. Time se postiže fleksibilnost te olakšava modeliranje polustrukturiranih ili polimorfnih tipova podataka, odnosno podataka čiji tip nije strogo definiran. Ova sposobnost baze također omogućava lagan način ažuriranja podataka jer se novi atribut jednostavno samo doda u dokument.

# 2. KORIŠTENI ALATI

## 3.1.SQL

Strukturirani upitni jezik [3] (engl. Structured Query Language – SQL) je strukturni jezik za pretraživanje koji se koristi u programiranju i dizajniranju za upravljanje podatcima i obradi tokova koji se nalaze u relacijskom sustavu upravljanja bazama podataka. Posebno je koristan u rukovanju strukturiranim podacima, to jest, obuhvaćanju veza između entiteta i varijabli. Naspram starijih sučelja za programiranje aplikacija kao što su indeksirana metoda sekvencijalnog pristupa i način pristupa virtualnoj pohrani, SQL donosi dvije važne inovacije. Predstavlja koncept pristupa velikoj količini zapisa jednom naredbom i eliminira potrebu za određivanjem načina dolaska do zapisa, to jest omogućava taj postupak sa ili bez indeksa.

## 3.1.SQL Server

Microsoft SQL Server - *MsSQL*[1] je upravljački sustav razvijen od strane Microsofta. Prva verzija Microsoft SQL Servera izašla je 1989. godine pod imenom SQL Server 1.0 i napisan korištenjem programskih jezike C i C++. Koristi se najviše za upravljanje relacijskim bazama podataka tako da sprema i dohvaća podatke na zahtjev ostalih softverskih aplikacija – koje mogu biti na istom računalu ili na drugom umreženom računalu.

Tipovi podataka koji se mogu koristiti u *MsSQL*-u prikazani su u sljedećim tablicama.

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| CHAR(*N*) | Fiksna duljina zapisa | 8000 karaktera | Definirana širina |
| VARCHAR(*N*) | Varijabilna duljina zapisa | 8000 karaktera | 2 + broj znakova |
| TEXT | Varijabilna duljina zapisa | $2^{31}$-1 karaktera | 4 + broj znakova |

*Tablica 3.1. Znakovni tipovi podataka*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| INT | Cijeli brojevi | Od $-2^{31}$ do $2^{31}-1$ | 4 |
| BIGINT | Cijeli brojevi | Od $-2^{63}$ do $2^{63}-1$ | 8 |
| SMALLINT | Cijeli brojevi | Od $-2^{15}$ do $2^{15}-1$ | 2 |
| TINYINT | Cijeli brojevi | Od 0 do 255 | 1 |
| DECIMAL | Decimalni brojevi | Od $-2^{38}$ do $2^{38}-1$ | 5-17 |

*Tablica 3.2. Numerički tipovi podataka*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| NCHAR(*N*) | Fiksna duljina zapisa | 4000 karaktera | / |
| NVARCHAR(*N*) | Varijabilna duljina zapisa | 4000 karaktera | / |
| NTEXT | Varijabilna duljina zapisa | $2^{30}-1$ karaktera | / |

*Tablica 3.3. Unicode tipovi podataka*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| DATETIME | Zaokruženo na 0.003 sekunde | Od 01.01.1753. do 31.12.9999. | 8 |
| SMALLDATETIME | Zaokruženo na minutu | Od 01.01.1900. do 06.06.2079. | 4 |
| TIMESTAMP | Specijalna namjena | / | / |

*Tablica 3.4. Datum i vrijeme*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| BINARY(*N*) | Fiksna duljina binarnog zapisa | 4000 karaktera | / |
| VARBINARY(*N*) | Varijabilna duljina binarnog zapisa | 4000 karaktera | / |
| IMAGE | Varijabilna duljina zapisa | $2^{31}$-1 karaktera | / |

*Tablica 3.5. Binarni tipovi podataka*

## 3.2. **MongoDB**

*MongoDB*[2] je dokumentno orijentirana baza podataka te program otvorenog koda razvijen od strane kompanije 10gen, napisan u jeziku C++ te služi za upravljanje bazama podataka. Koristi *JSON* (podatci su pohranjeni i transformirani u binarni, kompaktniji format imenom *BSON*[4]), što dopušta postojanje modela podataka bez sheme gdje je jedini zahtjev postojanje identifikacijske oznake.

Tipovi podataka koji se najčešće koriste u *MongoDB*-u, te koji se obično pohranjuju u *BSON* formatu, prikazani su u sljedećim tablicama:

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| STRING | Pohrana tekstualne vrijednosti | Ograničena sučeljem | 4 + (1 × broj_bajta_u_utf_8) + 1 |
| SYMBOL | Niz za jezike koji imaju određenu vrstu simbola | Ograničena sučeljem | Max 1.6e+7 |

*Tablica 3.1. Znakovni tipovi podataka*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| INTEGER | Pohrana cjelobrojne vrijednosti | Od $-2^{31}$ do $2^{31}-1$<br>Ili<br>Od $-2^{63}$ do $2^{63}-1$ | 4 ili 8 |
| DOUBLE | Pohrana podatka s pomičnim zarezom | True ili False | 8 |
| DECIMAL128 | Decimalni brojevi | Od $-10^{6145}$ do $10^{6145}$ | 16 |
| OBJECTID | Pohrana ID-a dokumenta | 24 | 12 |
| MIN/MAX KEYS | Usporedba susjednih BSON elemenata | -1 do 127 | / |

*Tablica 3.2. Numerički tipovi podataka*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| DATE | Pohrana datuma u UNIX formatu | -290 do 290 milijuna godina | 8 |
| TIMESTAMP | Specijalna namjena | / | 8 |

*Tablica 3.3. Datum i vrijeme*

| Tip podatka | Opis | Maksimalna veličina | Memorija (bajt) |
|---|---|---|---|
| BOOLEAN | True/false vrijednost | / | 2 |

| | | | |
|---|---|---|---|
| ARRAYS | Varijabilna duljina zapisa | / | Byte array -> 1<br>Short array -> 2<br>Integer array -> 4 |
| OBJECT | Implementacija ugrađenih dokumenata | / | Max 1.6e+7 |
| NULL | Pohrana Null vrijednosti | / | / |

*Tablica 3.4. Razni tipovi podataka*

```
> db.DataTypes.insert({"Integer example": 62})
WriteResult({ "nInserted" : 1 })
> db.DataTypes.insert({"Shipping status": true})
WriteResult({ "nInserted" : 1 })
> db.DataTypes.insert({"double data type": 3.1415})
WriteResult({ "nInserted" : 1 })
> db.DataTypes.insert({"string data type" : "Dokumentno orijentirane baze podataka."})
WriteResult({ "nInserted" : 1 })
> var leagues = ["La Liga", "Premier League", "Seria A"]
> db.DataTypes.insert({"Array Example" : "Here is an example of array",
... "Football leagues" : leagues})
WriteResult({ "nInserted" : 1 })
> db.DataTypes.insert({"UserID": null})
WriteResult({ "nInserted" : 1 })
> var date=new Date()
> db.DataTypes.insert({"Date":date})
WriteResult({ "nInserted" : 1 })
>
> db.DataTypes.find().pretty()
{ "_id" : ObjectId("60ebbbce6fda967eeced3456"), "Integer example" : 62 }
{ "_id" : ObjectId("60ebbbce6fda967eeced3457"), "Shipping status" : true }
{ "_id" : ObjectId("60ebbbce6fda967eeced3458"), "double data type" : 3.1415 }
{
        "_id" : ObjectId("60ebbbce6fda967eeced3459"),
        "string data type" : "Dokumentno orijentirane baze podataka."
}
{
        "_id" : ObjectId("60ebbbce6fda967eeced345a"),
        "Array Example" : "Here is an example of array",
        "Football leagues" : [
                "La Liga",
                "Premier League",
                "Seria A"
        ]
}
{ "_id" : ObjectId("60ebbbce6fda967eeced345b"), "UserID" : null }
{
        "_id" : ObjectId("60ebbbcf6fda967eeced345c"),
        "Date" : ISODate("2021-07-12T03:49:34.287Z")
}
>
```

*Slika 3.1 Osnovne naredbe tipova podataka*

Primjer korištenja tipova podataka može se vidjeti na slici 3.1. Prikazan je način unošenja i definiranja različitih tipova podataka, kao što su cjelobrojne vrijednosti, tekstualni podaci, nizovi, datumi i ostali često korišteni tipovi. Podatke je potom moguće vidjeti pomoću naredbe *find*[8].

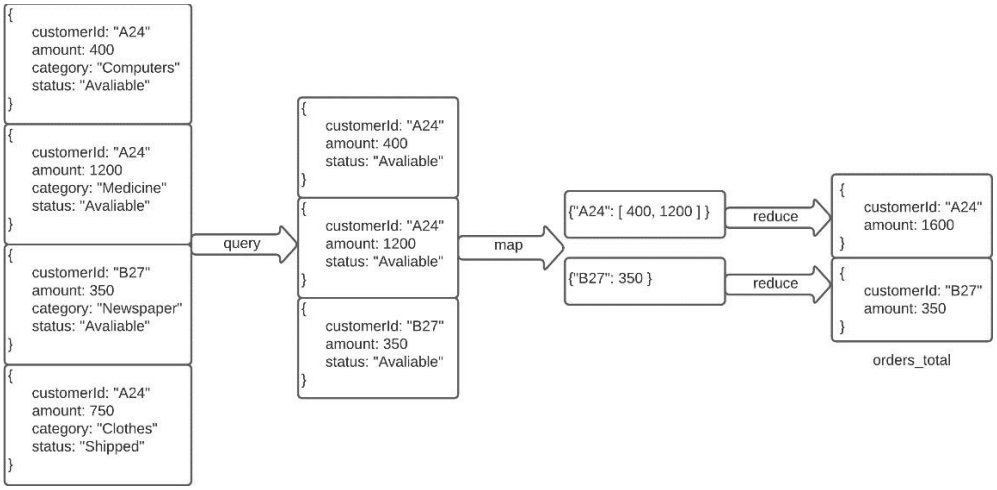
*MongoDB* pruža takozvanu horizontalnu skalabilnost kroz proces koji se naziva *sharding.* Taj proces pruža mogućnost distribuiranja opterećenja upisivanja podataka preko više servera (*shardova*). Ovdje svaki server predstavlja jednu nezavisnu bazu podataka te se kolekcija svih servera može promatrati kao jedna velika logička cjelina koja predstavlja bazu podataka. Jedna od glavnih točaka fokusa *MongoDB*-a je manipulacija nad dokumentima koristeći različite ugradbene okvire, od kojih su posebno korisni *MapReduce* i *Aggregation* Framework.

*MapReduce*[9] je paradigma obrade podataka za sažimanje velikih količina podataka u korisne agregirane rezultate. Kako bi se izvela operacija *map-reduce*, *MongoDB* pruža naredbu *mapReduce* čije djelovanje ćemo pokazati u idućem primjeru na slici 3.2.

```
> var mapFunction1 = function() {
... emit(this.customerId, this.amount);
... };
>
>
> var reduceFunction1 = function(keyCustomerId, valuesAmount){
... return Array.sum(valuesAmount);
... };
>
>
> db.orders.mapReduce(
... mapFunction1,
... reduceFunction1,
... {
... query: {status : "Avaliable" },
... out: "orders_total"
... })
{ "result" : "orders_total", "ok" : 1 }
>
```

*Slika 3.2 mapReduce naredba*

Ova operacija primjenjuje *map* naredbu na svaki dokument u kolekciji koji odgovara uvjetu upita. Funkcija locira i emitira parove ključ/vrijednost. Za ključeve koji imaju više vrijednosti, *MongoDB* prikuplja i sažima agregirane podatke, te ih potom pohranjuje u novu kolekciju. Također, izlaz funkcije smanjenje može proći kroz funkciju finaliziranja kako bi se dodatno zgusnuli ili obradili rezultati agregacije. Grafički prikaz operacije *mapReduce* može se vidjeti na slici 3.3.



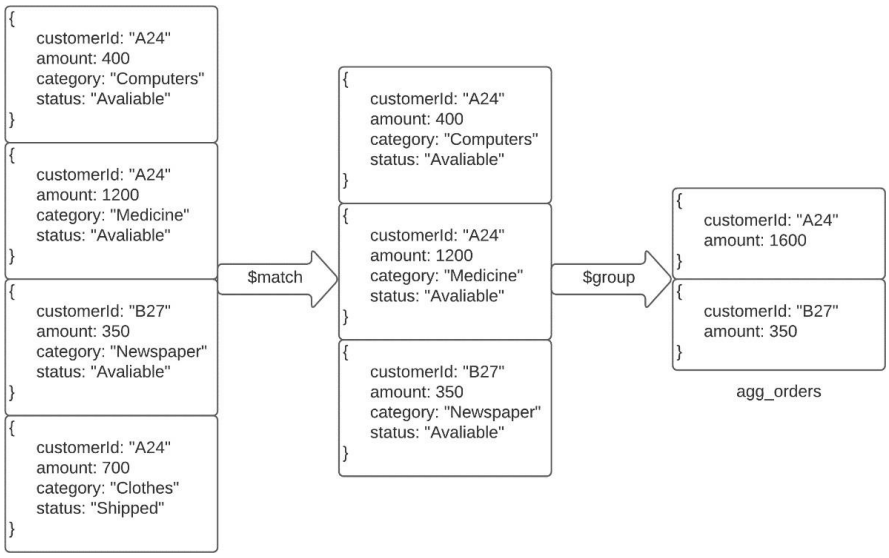

*Slika 3.3 grafički prikaz mapReduce naredbe*

Operacije agregiranja obrađuju zapise podataka i vraćaju izračunate rezultate te grupiraju vrijednosti iz više dokumenata jedno. Također imaju sposobnost izvođenja raznih operacija nad grupiranim podacima za vraćanje jednog rezultata. *MongoDB* pruža naredbu *mapReduce* čije djelovanje će biti prikazano u idućem primjeru na slici 3.4.

```
> db.orders.aggregate([
...     { $match: { status: "Avaliable" } },
...     { $group: { _id: "$customerId", value: { $sum: "$amount" } } },
...     { $out: "agg_orders" }
... ])
>
```

*Slika 3.4. aggregate naredba*

*MongoDB*-ov agregacijski okvir zasnovan je na konceptu cjevovoda za obradu podataka. Dokumenti ulaze u višestupanjski cjevovod koji ih potom pretvara u zbirni rezultat. U prvoj fazi *$match* filter dokumentira *status* i šalje dokumente sa statusom *Avaliable* u iduću fazu. Potom *$group* grupira dokumenta prema parametru *customerId* i kalkulira sumu za svaki jedinstveni *customerID.* Grafički prikaz operacije *aggregate* može se vidjeti na slici 3.5.



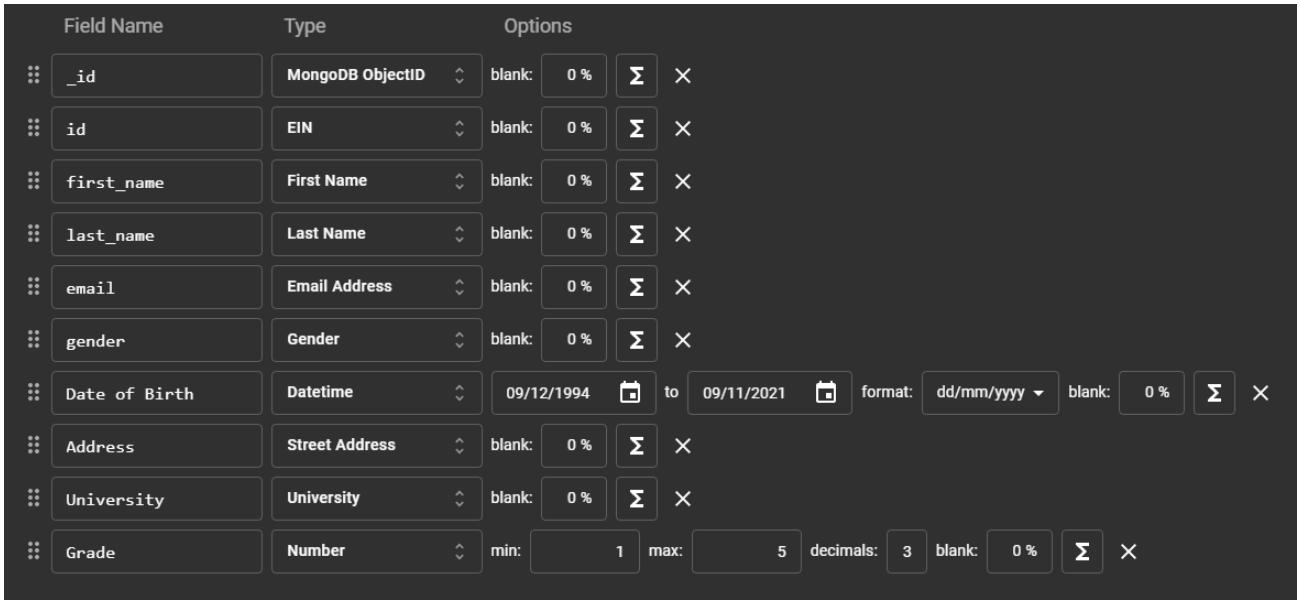

*Slika 3.5 grafički prikaz aggregate naredbe*

# 3. Praktični dio rada

U sljedećih nekoliko poglavlja će se objasniti kako izraditi bazu podataka korištenjem platforme *MongoDB* i *MsSQL* te će se mjeriti i uspoređivati različiti parametri performansi navedenih baza podataka.

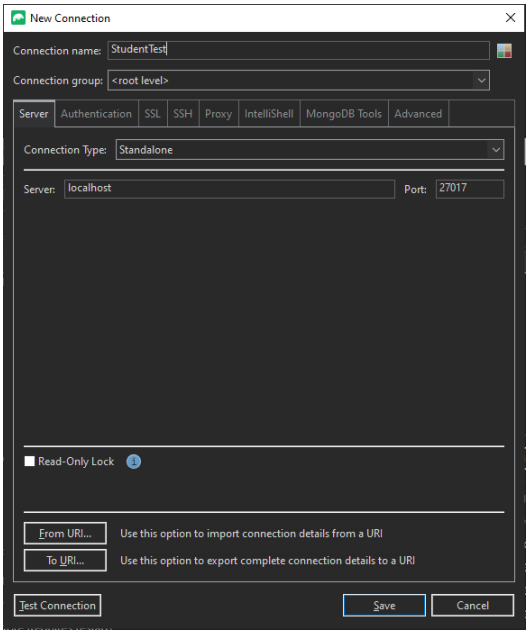
## 4.1. Konfiguriranje baze u MongoDBu

U ovom poglavlju bit će objašnjeno kako se izrađuje baza podataka u okružju *MongoDB*a. Kako bi olakšali pregled i rad nad bazom unutar *MongoDBa*, koristit će se grafičko korisničko sučelje *3T Studio*. Zbog potrebe za većom količinom podataka nad kojima će se provoditi testovi performansi, koristit će se web-aplikacija *Mockaroo* koja nudi stvaranje naizgled stvarnih podataka u *JSON* i *SQL* formatu. *JSON* format biti će potreban prilikom unosa podataka i rada sa *NoSQL* bazama podataka, dok će se u idućem poglavlju za relacijske *SQL* podatke koristiti format *SQL*.



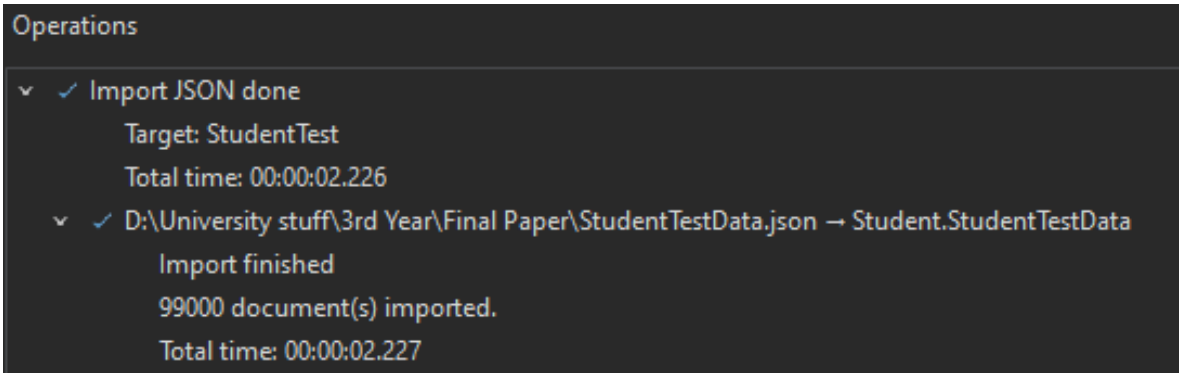

*Slika 4.1. Stvaranje podataka korištenjem web-aplikacije Mockaroo*

Nakon što je generirano 99 000 podataka i spremljeno ih u *JSON* datoteku, koristit će se *3T Studio* kako bi se učitali podaci u bazu, odnosno kolekciju. Prvo se spaja na lokalnog poslužitelja koji će biti korišten kao server, a nalazi se na portu 27017. Kada se prvi put uspostavlja nova konekcija, potrebno je postaviti ime servera, odnosno adresu i port na koji će se spajati i preko kojeg ćemo se komunicirati sa serverom.




*Slika 4.2. Postavljanje konekcije putem 3T Studija sa serverom*

Kada je veza uspješno postavljena sa serverom, potrebno je napraviti bazu podataka i kolekciju koja će se koristiti za pohranu podataka. Kako je već napravljena *JSON* datoteka sa potrebnim podacima, može se ugrađenom naredbom *Import Collection* populirati bazu podataka sa objektima.




*Slika 4.3. Popunjavanje kolekcije StudentTestData potrebnim podacima*

Kada je kolekcija uspješno popunjena sa 99 000 podataka gdje svaki dokument sadrži podatke o pojedinom studentu s odgovarajućim atributima koji ga opisuju, mogu se postavljati upiti i testirati performanse njihovog izvršavanja. Na sljedećoj slici može se vidjeti kako izgleda dokument koji sadrži podatke jednog studenta:


```
{
    "_id" : ObjectId("613d0ba7fc13ae37e100fa67"),
    "id" : "66-6967067",
    "first_name" : "Philis",
    "last_name" : "Habershon",
    "email" : "phabershon0@loc.gov",
    "gender" : "Male",
    "Date of Birth" : "06/01/2010",
    "Address" : "056 Northview Street",
    "University" : "Harare Institute of Technology",
    "Grade" : 1.78
}
```


*Slika 4.4. Primjer dokumenta u kolekcije StudentTestData*

Kao što je ranije opisano, *MongoDB* sprema podatke u dokumente koji sadrže vrijednosti u obliku objekata i polja te se kao takvi mogu usporediti s *JSON* objektima. Vrijednosti pojedinih polja također mogu biti u obliku niza ili nizova dokumenata, pa čak i sam dokument.

## 4.2. **Testiranje baze podataka u MongoDBu**

U ovom poglavlju će se testirati neki osnovne i često korišteni upiti nad bazom podataka u novo-napravljenoj kolekciji te će se mjeriti potrebno vrijeme da se isti izvrše. Upiti će se postaviti deset puta kako bi se dobilo prosječno vrijeme njihovog izvršavanja te će ih se kasnije usporediti sa istim operacijama u relacijskim bazama podataka.

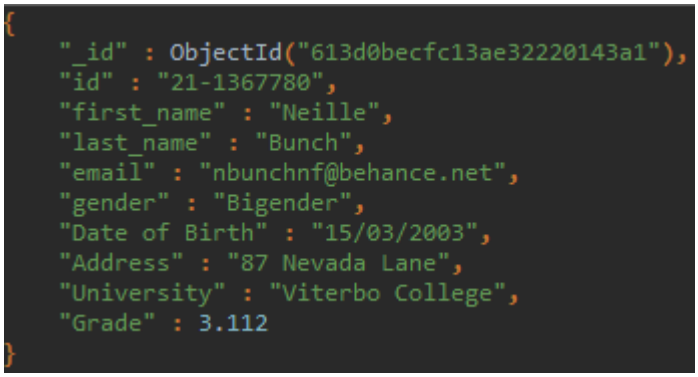
- **Unošenje podataka u kolekcije**

  Unos podataka iz generirane *JSON* datoteke se odrađuje korištenjem naredbe *Import Dana*[7]. Odabere se dokument koji sadrži željene podatke, potom pokreće naredba *Start Import* i podatke se učitava u kolekcije. Prosječno vrijeme trajanja učitavanja 99 000[Prilog 1.1] pojedinih objekata je 2.34 sekunde.

- **Dohvaćanje zapisa iz baze**

```
db.StudentTestData.find({„_id“ : ObjectId(„613d0becfc13ae32220143a1“)})
```

Funckija *find* će pronaći dokumente pomoću parametara koji se koriste za njegovo pretraživanje. Zbog jedinstvenosti parametra, to je najćešće *ObjectId*. Prosječno trajanje ovog upita je 3 milisekunde. Rezultat pretraživanja je sljedeći:




*Slika 4.5. Dohvaćanje zapisa prema jedinstvenom parametru*

- **Dohvaćanje svih zapisa iz kolekcije**

```
db.StudentTestData.find({})
```

Rezultat izvođenja ovog upita je dohvaćanje svih dokumenata koji se nalaze u kolekcije te je prosječno trajanje njegovo izvođenja 51 milisekunda.

- **Dohvaćanje i računanje prosječnih vrijednosti atributa**


*Slika 4.6. Funkcija aggregate za dohvaćanje određenih atributa*

Pomoću naredbe prikazane na slici računa se prosječna ocjena u odnosu na spol studenta, te se također broji ukupna populacija određenog spola. Rezultat izvršene naredbe može se vidjeti na slici 4.7. Prosječno vrijeme izvođenja ove naredbe je 127 milisekundi.

```
{
    "_id" : "Genderqueer",
    "count" : 12480.0,
    "agvGrade" : 3.0044713141025645
}
{
    "_id" : "Polygender",
    "count" : 12448.0,
    "agvGrade" : 3.0011542416452444
}
{
    "_id" : "Female",
    "count" : 12298.0,
    "agvGrade" : 3.010437713449341
}
{
    "_id" : "Bigender",
    "count" : 12305.0,
    "agvGrade" : 2.997156603006908
}
{
    "_id" : "Male",
    "count" : 12398.0,
    "agvGrade" : 3.017760283916761
}
{
    "_id" : "Genderfluid",
    "count" : 12234.0,
    "agvGrade" : 2.987440657184895
}
{
    "_id" : "Agender",
    "count" : 12398.0,
    "agvGrade" : 2.9854295854170028
}
{
    "_id" : "Non-binary",
    "count" : 12439.0,
    "agvGrade" : 2.9986248894605674
}
```


*Slika 4.7. Rezultat računanja prosječne ocijene po spolovima*

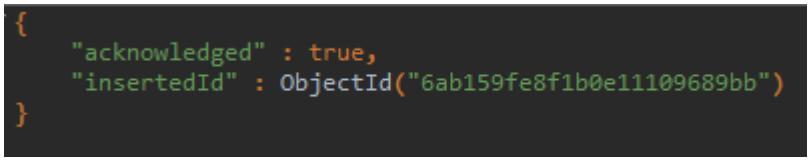
- **Dodavanje novog unosa odnosno zapisa u kolekciju**

```
db.StudentTestData.insertOne(
{
    "_id" : ObjectId("6ab159fe8f1b0e11109689bb"),
    "id" : NumberInt(2) +  NumberInt(7),
    "first_name" : "Matija",
    "last_name" : "Barić",
    "email" : "mbaric@etfos.com",
    "gender" : "Male",
    "Date of birth" : "16.05.1999",
    "Address" : "12 Adresa Adresica",
    "University" : "FERIT",
    "Grade" : 4.20
})
```

*Slika 4.8. Dodavanje novog studenta u kolekciju*

Naredbom prikazanoj na prethodnoj slici dodaje se novi zapis u bazu te prosječno vrijeme potrebno za izvođenje ovog upita iznosi 12 milisekundi.

Ako je naredba obavljena uspješno trebalo bi dobiti sljedeći rezultat:

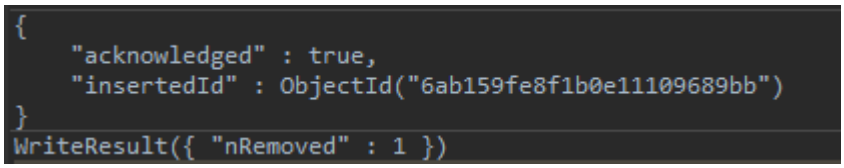



*Slika 4.9. Rezultat unosa novog zapisa u kolekciju*

- **Uklanjanje zapisa iz baze**

```
db.StudentTestData.remove({„_id“ : ObjectId(„6ab159fe8f1b0e11109689bb“)})
```

Ako se prikazan upit pravilno izveo, korisnik prezimena Barić se briše iz kolekcije, a njemu se pristupa putem jedinstvene *_id* oznake. Prosječno vrijeme za izvođenje ove naredbe je 9 milisekundi. Rezultat uspješnog izvođenja naredbe je sljedeći:




*Slika 4.10. Uklanjanje objekta iz kolekcije*

- **Sortiranje podataka**

```
db.StudentTestData.find({},{„first_name“:1,„last_name“:1, „_id“:0}).sort({„first_name“:1,
„last_name“:1,}).limit(100)
```

Prikazanom operacijom sortiraju se podaci prema imenu i prezimenu uzlaznim redoslijedom, to jest abecednim redoslijedom. Prosječno trajanje izvođenja operacije je 117 milisekundi.


*Slika 4.11. Podaci sortirani abecednim redoslijedom prema parametrima imena i prezimena*

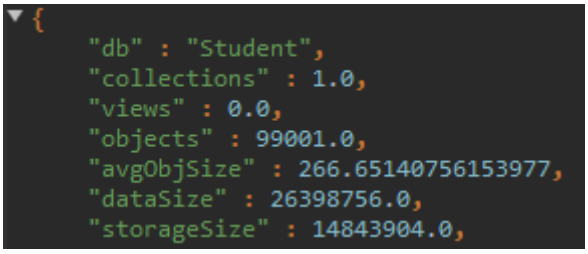
- **Brisanje svih zapisa**

```
db.StudentTestData.deleteMany({})
```

Prikazana naredba uklanja sve zapise iz kolekcije. Potrebno prosječno vrijeme za izvođenje ove operacije nad 99 000 podataka je 1.277 sekundi.

- **Veličina baze podataka**

Veličinu baze podataka može se dobiti korištenjem naredbe *„db.stats()“* te nakon izvođenja dobije se sljedeći rezultat:

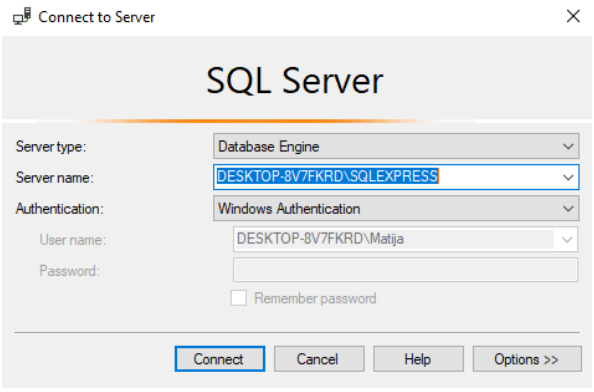



*Slika 4.12. Veličina baze podataka*

*DataSize* predstavlja zbroj veličina u bajtovima svih dokumenata i obloga pohranjenih u bazi podataka, dok je je *storageSize* veličina svih podataka u bazi zajedno sa neiskorištenim alociranim prostorom. Može se vidjeti da je veličina podataka u bazi 14.84MB.

## 4.3. **Konfiguriranje baze podataka u MsSQLu**

U ovom poglavlju će se obraditi konfiguriranje relacijske baze podataka korištenjem softvera *Microsoft SQL Management Studio*[10]. Prvi korak je spajanje na server, pri čemu se određuje tip servera koji će se koristiti za pohranu podataka, ime servera na koji će se spajati te tip autentifikacije.




*Slika 4.13. Spajanje na MsSQL server*

Nakon uspješno postavljene konekcije sa SQL serverom, potrebno je kreirati tablice koje će služiti kao spremnici podataka. To će se napraviti na način prikazan na sljedećoj slici:

```
create table Students (
    id VARCHAR(50),
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    DateOfBirth DATE,
    email VARCHAR(50),
    gender VARCHAR(50),
    Address VARCHAR(50),
    University VARCHAR(200),
    Grade DECIMAL(4,3)
);
```

*Slika 4.14. Prikaz tablice Students za spremanje podataka*

## 4.4. Testiranje baze u MsSQLu

Nakon što su tablice kreirane može ih se popuniti sa podacima koji sa kreirani putem web-aplikacije *Mockaroo*. Podaci su generirani u *SQL* formatu i spremni za korištenje u relacijskim bazama podataka.

- **Unošenje podataka u tablice**

  Potrebni podaci se učitavaju iz *SQL* datoteke korištenjem naredbe *Import dana*[6]. Nakon toga odabire se željeni način učitavanja podatka i pronalazi datoteka koja ih sadrži. Opcionalno mogu se odabrati stupci koje se želi ubaciti u tablicu te potom naredbom *Finish* pokreće se učitavanje podataka. Prosječno vrijeme za učitavanje 99 000[Prilog 2.1] podataka iz datoteke je 19.3 sekunde.

- **Dohvaćanje zapisa iz baze**

  Naredba iz baze dohvaća studenta sa zadanim parametrom te se izvodi putem upita koji ima sljedeći oblik:

```
SELECT * FROM Students WHERE id = ’77-8241361';
```

  Prosječno vrijeme trajanje ove operacije je 9 milisekundi te njen rezultat možemo vidjeti na sljedećoj slici:

| | id | first_name | last_name | DateOfBirth | email | gender | Address | University | Grade |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 77-8241361 | Jewelle | Bartalot | 2005-04-17 | jbartalot9r@omniture.com | Genderqueer | 93 Derek Trail | Baitulmal Management Institute (IPB) | 4.304 |

*Slika 4.15. Rezultat dohvaćanja stupca čiji je id ’77-8241361'*

- **Dohvaćanje svih zapisa**

  Sve zapise iz tablice možemo dohvatiti korištenjem upita koji ima sljedeći oblik:

```
SELECT * FROM Students;
```

Prikazana operacija dobavlja sve stupce, odnosno sve korisnike koji se nalaze u tablici te je prosječno vrijeme izvođenja operacije 932 milisekunde.

- **Dohvaćanje i računanje prosječnih vrijednosti atributa**

```
SELECT gender, COUNT(gender), ROUND(AVG(Grade),2) AS AVGGrade
FROM Students
GROUP BY GENDER;
```

*Slika 4.16. Naredba za računanje prosječne ocjene*

Pomoću prikazane naredbe računaju se prosječne ocjene studenata koji se nalaze u tablici te ih se grupira po spolovima. Prosječno trajanje izvođenja operacije je 44 milisekunde i njen rezultat može se vidjeti na sljedećoj slici:

| | gender | (No column name) | AVGGrade |
|---|---|---|---|
| 1 | Bigender | 12416 | 3.000000 |
| 2 | Genderfluid | 12460 | 2.980000 |
| 3 | Genderqueer | 12605 | 3.010000 |
| 4 | Non-binary | 12537 | 3.000000 |
| 5 | Polygender | 12553 | 3.010000 |
| 6 | Male | 12453 | 2.990000 |
| 7 | Agender | 12413 | 2.990000 |
| 8 | Female | 12563 | 2.990000 |

*Slika 4.17. Prosječna ocjena po spolovima*

- **Dodavanje novog zapisa**

```
INSERT INTO Students VALUES ('42-0133769', 'Matija', 'Barić', '1999/04/20',
'mbaric@etfos.com', 'Male', '77 Adresa Adresica', 'FERIT', '4.200');
```

Prikazanom naredbom dodaje se zapis u tablicu, odnosno bazu podataka. Prosječno vrijeme izvođenja ove operacije je 22 milisekundi.

- **Uklanjanje zapisa iz baze**

```
DELETE FROM Students WHERE id = '42-0133769';
```

Prikazanom naredbom uklanja se jedan korisnik iz baze podataka. Korisnika se locira putem parametra *id* te vrijeme potrebno da se cjelokupna operacije izvede je 28 milisekundi.

- **Sortiranje podataka**

```
SELECT * FROM Students ORDER BY first_name, last_name ASC;
```

Prikazanom naredbom sortiraju se podaci uzlaznim abecednim redoslijedom koristeći parametar imena i prezimena. Prosječno vrijeme izvođenja ove operacije je 487 milisekundi i njen rezultat vidljiv je na sljedećoj slici:

| | id | first_name | last_name | DateOfBirth | email | gender | Address | University | Grade |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 51-3242323 | Aaren | Chrichton | 2001-07-09 | achrichtoncq@mediafire.com | Female | 3378 Elka Road | Fundação Educacional de Ituverava | 4.880 |
| 2 | 70-0053760 | Aaren | Denis | 2012-03-21 | adenisgt@ocn.ne.jp | Female | 63 Hoffman Point | University of Engineering and Technology Peshawar | 4.631 |
| 3 | 11-6794130 | Aaren | Devons | 2019-07-08 | adevonsox@wsj.com | Non-binary | 752 Oriole Center | Vologda State Pedagogical University | 3.418 |
| 4 | 31-6294939 | Aaren | Flexman | 1996-01-23 | aflexmaniw@economist.com | Agender | 522 Harbort Lane | University of Liverpool | 3.531 |
| 5 | 22-8712162 | Aaren | Hatherill | 2017-04-19 | ahatherillm3@issuu.com | Bigender | 58 Ohio Way | Universidade dos Acores | 3.765 |
| 6 | 69-1011941 | Aaren | Knewstubb | 1996-08-17 | aknewstubbrr@blogger.com | Polygender | 916 Bunker Hill Street | University of El Imam El Mahdi University | 1.360 |
| 7 | 59-3094602 | Aaren | Mapston | 2009-07-13 | amapstonia@ft.com | Bigender | 41181 Valley Edge Circle | Southern New Hampshire University | 3.444 |

*Slika 4.18. Uzlazno abecedno sortiranje prema parametru imena i prezimena*

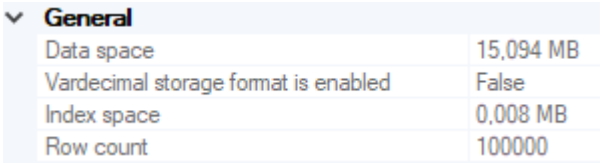
- **Brisanje svih zapisa**

```
DELETE FROM Students;
```

Prikazana naredba briše sve podatke iz tablice. Prosječno vrijeme izvođenja ove operacije je 883 milisekunde.

- **Veličina baze podataka**

  Veličinu baze može se dobiti desnim klikom na bazu podataka, te pod svojstvima provjeriti prostor koji zauzima. Veličina podataka iznosi 15.094MB te se može vidjeti na sljedećoj slici:




*Slika 4.19. Veličina baze podataka*

## 4.5. Usporedba performansi relacijskih i dokumentno orijentiranih baza

Nakon što je odrađeno konfiguriranje i testiranje relacijske i dokumentno orijentirane baze podataka, u ovom poglavlju će se izvršiti usporedba dobivenih rezultata[11]. Uspoređivanje će se praviti na temelju rezultata dobivenih korištenjem okružja *MongoDB* kao *NoSQL* bazu, te *MsSQL* kao *SQL* bazu podataka. Testovi performansi će se određivati prema tablici, odnosno kolekciji od 99 tisuća, 500 tisuća i 1 milijun podataka.

- **Testovi performansi pri radu sa 99,000[Prilog 1.1 i 2.1] zapisa**

Tablica 4.1. Uspoređivanje prosječnih trajanja osnovnih operacija nad relacijskim i dokumentnim bazama podataka te njihovih svojstava na 99 tisuća podataka

| Test | *MongoDB* | *MsSQL* |
|---|---|---|
| Unošenje podataka | 2.34 s | 19.3 s |
| Dohvaćanje jednog zapisa | 3 ms | 9 ms |
| Dohvaćanje svih zapisa | 51 ms | 932 ms |
| Izračun prosječne vrijednosti | 127 ms | 44 ms |
| Dodavanje novog zapisa | 12 ms | 22 ms |
| Uklanjanje jednog zapisa | 9 ms | 28 ms |
| Sortiranje podataka | 117 ms | 487 ms |

| | | |
|---|---|---|
| Uklanjanje svih zapisa | 1.277s | 883ms |
| Veličina baze | 14.844MB | 15.094MB |

U tablici se može zamijetit da performanse variraju ovisno o operaciji koja se izvodi. Pri količini od 99 tisuća podataka primjećuje se da *MongoDB* u pravilu brže izvršava operacije koje zahtijevaju rad sa više od jednog objekta. Jedine iznimke su brisanje svih zapisa iz tablice i sortiranje podataka, gdje je *MsSQL* više puta brži. Također uočljivo je da je veličina same baze nešto manja u dokumentnom obliku.

- **Testovi performansi pri radu sa 500,000[Prilog 1.2 i 2.2] zapisa**

Tablica 4.2. Uspoređivanje prosječnih trajanja osnovnih operacija nad relacijskim i dokumentnim bazama podataka te njihovih svojstava na 500 tisuća podataka

| Test | *MongoDB* | *MsSQL* |
|---|---|---|
| Unošenje podataka | 13.23 s | 98.92 s |
| Dohvaćanje jednog zapisa | 4 ms | 8 ms |
| Dohvaćanje svih zapisa | 326 ms | 4.077 s |
| Izračun prosječne vrijednosti | 474 ms | 195 ms |
| Dodavanje novog zapisa | 14 ms | 22 ms |
| Uklanjanje jednog zapisa | 10 ms | 30 ms |
| Sortiranje podataka | 1.117 s | 632 ms |
| Uklanjanje svih zapisa | 4.138 s | 9.436 s |
| Veličina baze | 49.913 MB | 68.024 MB |

Kao što se moglo očekivati, povećavanjem veličine baze i broja korisnika koji se u njoj nalaze, vremena potrebna za izvođenje upita se povećavaju. No, taj rast je nešto manji u dokumentno orijentiranim bazama naspram relacijskih. Iako se i dalje radi sa relativno malom količinom podataka, može se vidjeti kako vrijeme obavljanja upita u *MsSQL*u raste puno više s

obzirom na količinu podataka. Također primjećuje se da potrebna memorija za spremanje podataka raste više za relacijski model, nego za objektno orijentirani.

- **Testovi performansi pri radu sa 1,000,000[Prilog 1.3 i 2.3] zapisa**

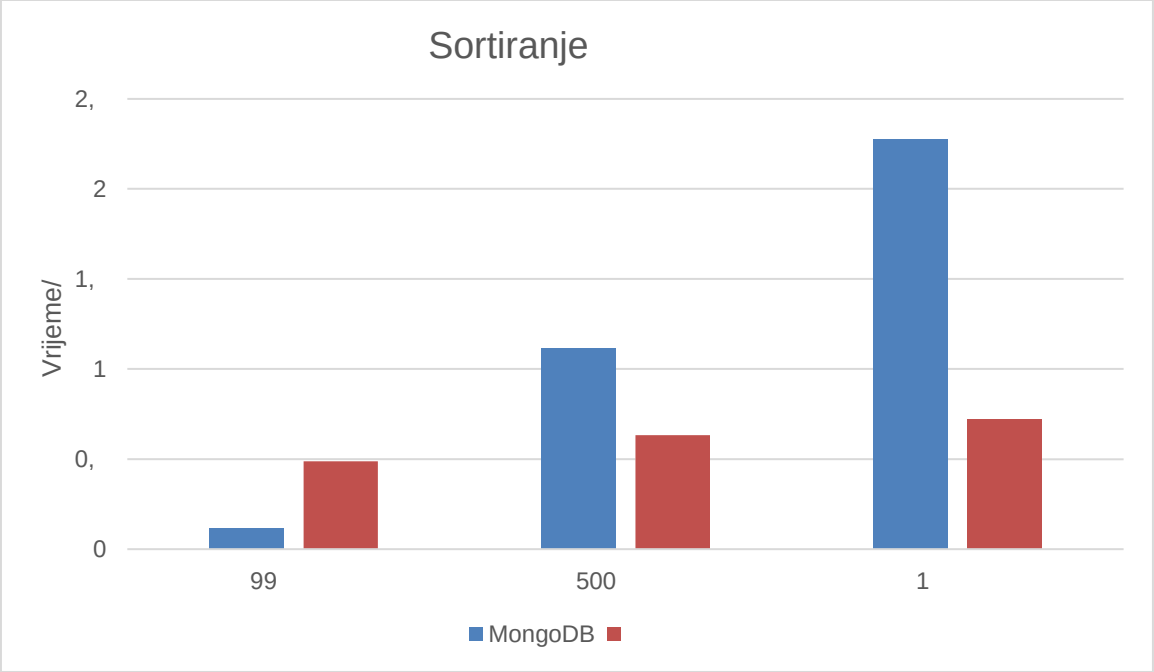
Tablica 4.3. Uspoređivanje prosječnih trajanja osnovnih operacija nad relacijskim i dokumentnim bazama podataka te njihovih svojstava na milijun podataka

| Test | *MongoDB* | *MsSQL* |
|---|---|---|
| Unošenje podataka | 27.23 s | 204.63 s |
| Dohvaćanje jednog zapisa | 7 ms | 11 ms |
| Dohvaćanje svih zapisa | 411 ms | 27.144 s |
| Izračun prosječne vrijednosti | 1.077 s | 411 ms |
| Dodavanje novog zapisa | 14 ms | 24 ms |
| Uklanjanje jednog zapisa | 11 ms | 30 ms |
| Sortiranje podataka | 2.277 s | 724 ms |
| Uklanjanje svih zapisa | 10.454 s | 33.912 s |
| Veličina baze | 92.721 MB | 137.271 MB |

Nakon izvedenog testa nad milijun podataka može se primijetiti da *MongoDB* ima prednost u odrađivanju operacija stvaranja, čitanja, ažuriranja i brisanja podataka. Intuitivno se da zaključiti da potrebno vrijeme obavljanja operacija raste sa brojem podataka koji se nalazi u bazi. Također se može zamijetit svojstvo linearnosti u porastu vremena u odnosu na količinu podataka koji se nalazi u bazi. Kao primjer mogu se promotriti vrijednosti pri unošenju podataka u bazu, gdje je vrijeme potrebno za izvođenje te operacije bilo 2.34 sekunde u dokumentno orijentiranoj bazi, te 204.63 sekunde u relacijskoj bazi pri količini od 99 tisuća podataka. Ako se za isti upit obrati pozornost na vremenske vrijednosti u tablici koja sadrži milijun podataka, primjećuje se da *MongoDB* izvodi operaciju za 27.23 sekunde, dok *MsSQL* za istu zahtijeva 204.63 sekunde. Vrijednosti su u obje baze

otprilike deset puta veće te sa kraćim početno potrebnim vremenom vidimo da porastom količine podataka vrijeme izvođenja takvih upita osobito dolazi do izražaja.

- **Operacija sortiranja podataka**

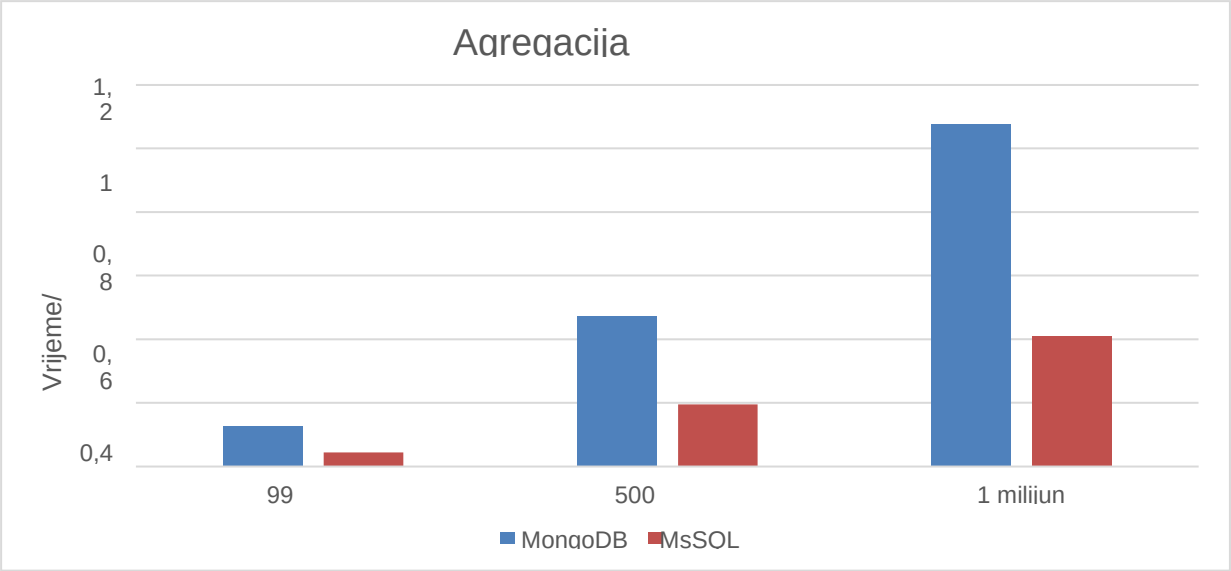



*Slika 4.20. Uspoređivanje operacije sortiranja*

Na slici 4.20. može se vidjeti da *MsSQL*, odnosno relacijska baza podataka učinkovitije rješava probleme sortiranja podataka u odnosu na *MongoDB*. Vrijeme izvođenja te operacije je u početku veće, no s porastom količine podataka *MsSQL* postaje puno bolje rješenje za ovakav tip upita.

- **Agregiranje podataka**

Još jedna prednost koju relacijske baze podataka imaju u odnosu na dokumentne je kod operacija agregiranja podataka. Kao i kod operacije sortiranja, porastom količine podataka, *MsSQL* efikasnije agregira dane podatke te su razlike u vremenu manje nego one kod *MongoDBa*. Usporedba se može vidjeti na slici 4.21.




*Slika 4.21. Uspoređivanje operacije agregiranja podataka*

Tablica 4.4. Prednosti i nedostaci dokumentno orijentiranih baza podataka

| **Prednosti dokumentno orijentiranih baza podataka** | **Nedostaci dokumentno orijentiranih baza podataka** |
|---|---|
| Skalabilnost | Rana faza tehnologije rezultira manjim iskustvom u radu sa ovakvim tipom baze |
| Bez-shematska struktura koja omogućava promjenu strukture bez prekidanja rada baze podataka | Nisu adresirani svi nedostaci u koji trenutno ograničavaju rad i veću implementaciju u korištenju ovog tipa baze podataka |
| Rad sa manjom količinom resursa | Slaba konzistentnost podataka koja ponekad dovodi do gubitka istih |
| Visoka razina fleksibilnosti koja se prilagođava zahtjevima u projektu | Ne postoji mogućnost kriptiranja podataka |
| Podržava pohranu bilo kakvog tipa podataka bilo koje veličine bez utjecaja na samu hrpu | Ne podržava mnogo programskih jezika |
| Brže izvođenje velikog broja operacija nad velikim bazama podataka | Manja pouzdanost naspram relacijskih baza podataka |

# 4. ZAKLJUČAK

U ovom radu napravljena je usporedba relacijskih i dokumentno orijentiranih baza podataka. Provedeno je testiranje performansi jednostavnih i često korištenih upita i promatrana su prosječna vremena potrebna za njihovo izvođenje. Može se zamijetit da *MongoDB* i *MsSQL* nude različita rješenje te posjeduje različite prednosti i nedostatke pri rješavanju problema i zahtjeva korisnika.

Uspoređene su dvije baze popunjene istim podacima u različitom obliku. Koristeći *MongoDB* napravljena je dokumentno orijentirana baza nad kojoj su se provodili isti testovi kao u relacijskoj bazi konfiguriranoj putem *MsSQLa*. Praktični dio rada pokazao je da dokumentno orijentirane baze imaju prednost pri izvođenju operacija stvaranja, čitanja, ažuriranja i brisanja podataka, te da se ta prednost povećava sa brojem zapisa u bazi. Također, format dokumenata i bez-shematska struktura objektno orijentiranih baza je rezultirala u korištenju manje prostora za pohranu podataka. *MsSQL*, odnosno relacijski oblik baze podataka je zadržao prednost u operacijama sortiranja upisanih podataka te u agregiranju istih.

Može se doći do zaključka da takozvane *NoSQL* baze podataka više pogoduju radu sa podacima bez strukture te omogućuju veću fleksibilnost. S druge strane relacijske baze podataka sa njihovom definiranom strukturom omogućuju višu razinu sigurnosti pri održavanju i zaštiti podataka.

Objektno orijentirane baze podataka pružaju lakši i brži rad sa podacima u određenim slučajevima, te svoje nedostatke trebaju adresirati u budućem razvoju. Ovaj tip baze ovisi o zahtjevima korisnika, te prema tome, sa informacijama koje su do sada prikupljene, potrebno je donijeti odluku za optimalan način rukovanja bazom podataka.

# LITERATURA


[1] INAP, „Microsoft SQL Server“, *SQL Server*, https://www.inap.com/blog/microsoft-sql-server-architecture/ (accessed Jun. 29, 2021).

[2] MongoDB, “MongoDB,” MongoDB. https://en.wikipedia.org/wiki/MongoDB (accessed Jun. 29, 2021).

[3] TechTarget, “SQL,” *SQL*. https://searchdatamanagement.techtarget.com/definition/SQL (accessed Jul. 1, 2021).

[4] MongoDB, “BSON types,” *BSON Types*, https://docs.mongodb.com/manual/reference/bson-types/ (accessed Jul. 1, 2021).

[5] I. Lukić, “predavanja s Baze podataka, studij: Računalno inženjerstvo, ak. God. 2020/2021.” Accessed: Jul. 02, 2021. [Online]. Available: https://moodle.srce.hr/2020-2021/pluginfile.php/4468276/mod_resource/content/6/BP%2001%20Uvodno%20predavanje.pdf.

[6] Microsoft, „Microsoft SQL documentation“ SQL Documentation, https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15 (accessed Sep. 4, 2021).

[7] MongoDB, „MongoDB documentation“, MongoDB documentation, https://docs.mongodb.com (accessed Sep. 5, 2021).

[8] MongoDB, „MongoDB tutorials“, *MongoDB tutorials*, https://www.softwaretestinghelp.com/mongodb/ (accessed Sep. 5, 2021).

[9] Youtube channel Traversy Media, „MongoDB Crash Course“, *MongoDB tutorials*, https://www.youtube.com/watch?v=-56x56UppqQ (accessed Sep. 7, 2021).

[10] Microsoft, „SQL Server Management Studio“, *SQL Server Configuration*, https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms (accessed Sep. 7, 2021).

[11] Guru99, „MsSQL vs MongoDB“, *MsSQL and MongoDB comparison*, https://www.guru99.com/mongodb-vs-mysql.html (accessed Sep. 9, 2021).

## SAŽETAK

U ovom radu su promatrane prednosti i nedostaci dokumentno orijentiranih baza podataka na način da su se testirala prosječna vremena izvođenja različitih jednostavnih i često korištenih upita. Na početku rada dana je teorijska podloga o različitim tipovima baza koje se danas koriste za rukovanje podacima. Ukratko je objašnjen način funkcioniranja relacijskih baza podataka te su dani primjeri i načini njihovog korištenja. Potom su pogledani tipove podataka i neke osnovne upite za rukovanje podacima unutar dokumentno orijentiranih ili *NoSQL* baza podataka te su objašnjene neke karakteristike koje ih opisuju. U poglavlju praktičnog rada konfigurirana sa dva različita tipa baza korištenjem softvera *MongoDB* i *MsSQL* i nad njima su se radili jednostavne i često korištene operacije te uspoređivali vremenski rezultati njihovog izvođenja. Rad je zaključen sa prednostima i nedostacima dokumentno orijentiranih baza podataka prema rezultatima koje su dobiveni iz njihovog testiranja i uspoređivanja sa relacijskim bazama podataka.



## KLJUČNE RIJEČI:

*Baza podataka, MongoDB, MsSQL, relacijske baze podataka, dokumentno orijentirane baze podataka*

# ADVANTAGES AND DISADVANTAGES OF DOCUMENT ORIENTED DATABASES


## ABSTRACT

Topic of this paper is “Advantages and disadvantages of document-oriented databases”. In the beginning of this paper the reader is given a theoretical background on different database types most commonly used nowadays for data handling. Function of such databases is briefly explained, along with examples of their usage. Furthermore, data types and some of the basic queries for handling data in document oriented or NoSQL databases were analysed, as well as interpreted characteristics which describe them. In the chapter which focuses on practical work using *MongoDB* and *MsSQL*, two different types of databases were configured. Afterwards they were used to conduct frequently used operation and compared the results of their performance. The paper is concluded with advantages and disadvantages of document oriented databases based on acquired results from performed tests and their comparison with relational databases.



## KEYWORDS:

*Database, MongoDB, MSSQL, relational databases, document oriented databases.*

# ŽIVOTOPIS

Matija Barić rođen 16.05.1999. godinu u Vinkovcima u Hrvatskoj. Osnovnoškolsko obrazovanje sa odličnim uspjehom tokom svih godina stječe u Gunji gdje ujedno živi. Godine 2014. upisuje Srednju Školu u Županji gdje pohađa Matematičku Gimnaziju i završava ju sa vrlo dobrim uspjehom. 2018./2019. upisuje prvu godinu preddiplomskog studija računarstva na Fakultetu Elektrotehnike, Računarstva i Informacijskih Tehnologija.

# PRILOZI

https://drive.google.com/drive/folders/1_OCJgygrmrhHNlsmdi4wWU35XJ99hv8F?usp=sharing

Na *Google Drive* linku priloženom uz završni rad nalaze se:

[1] **Prilog 1.1** 99 tisuća *JSON* podataka

[2] **Prilog 1.2** 500 tisuća *JSON* podataka

[3] **Prilog 1.3** 1 milijun *JSON* podataka

[4] **Prilog 2.1** 99 tisuća *SQL* podataka

[5] **Prilog 2.2** 500 tisuća *SQL* podataka

[6] **Prilog 2.3** 1 milijun *SQL* podataka