SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

SEMINARSKI RAD

USPOREDBA SQL SERVER I MONGODB

Lucija Dumančić

Split, lipanj 2023.

Sadržaj

1. Uvod 1

2. Razvojna tehnologija 2

2.1. SQL Server 2

2.2. MongoDB 3

3. Izrada projekta 5

3.1. Baza podataka u SQL Serveru 5

3.1.1. Kreiranje tablica 7

3.1.2. Umetanje podataka 9

3.1.3. Pogledi 15

3.1.4. Procedure 23

3.1.5. Log tablice 29

3.1.6. Okidači 30

3.2. Baza podataka u MongoDB Compass-u 35

3.2.1. Upiti 39

4. Zaključak 52

1. Uvod

Odabir odgovarajuće baze podataka za sustav je bitan korak u izgradnji sustava i one igraju ključnu ulogu u podršci poslovanja i pružanju efikasnog pristupa podacima. Kao takve, baze podataka trebaju biti sposobne brzo i efikasno izvršavati upite kako bi osigurale optimalno korisničko iskustvo. U suvremenom poslovnom okruženju, brzina izvršavanja upita postala je jedan od ključnih faktora prilikom odabira baze podataka. U ovom seminaru uspoređene su dvije popularne baze podataka, MongoDB i SQL Server, kako bi se analizirale njihove performanse izvršavanja upita u kontekstu glazbene trgovine. MongoDB, popularna NoSQL baza podataka, temelji se na fleksibilnom modelu dokumenta, dok je SQL Server, relacijska baza podataka, usmjerena na strukturirane tablice i relacije. Ove razlike u dizajnu mogu imati značajan utjecaj na brzinu izvršavanja upita u različitim scenarijima. Analiza je usredotočena na ključne upite koji su karakteristični za glazbenu trgovinu, poput pretraživanja po nazivu albuma, autoru, žanru, kupcima, zaposlenicima i transakcijama. Mjerene su brzine izvršavanja upita u obje baze podataka kako bismo utvrdili koja pruža bolje performanse u tim scenarijima. Rezultati ove usporedbe mogu pružiti korisne uvide i pomoći prilikom donošenja odluke u odabiru baze podataka za glazbenu trgovinu ili slične industrije. Također, može nam omogućiti bolje razumijevanje karakteristika i prednosti svake od baza podataka u kontekstu brzine izvršavanja upita.

2. Razvojna tehnologija

2.1. SQL Server

SQL Server je sustav za upravljanje relacijskim bazama podataka. Nudi opsežan skup značajki i svojstava koji ga čine moćnim i svestranim rješenjem baze podataka, zbog čega se koristi za upravljanje i pohranu strukturiranih podataka. Temelji se na relacijskom modelu, koji organizira podatke u tablice s redovima i stupcima. Koristi standardni SQL (engl. Structured Query Language) jezik za interakciju s bazom podataka. SQL pruža skup naredbi za upravljanje, postavljanje upita i manipuliranje podacima pohranjenim u SQL Server-u. Osigurava integritet podataka podržavanjem transakcijske obrade i provođenjem svojstava ACID (Atomicity, Consistency, Isolation, Durability). Atomarnost (engl. Atomic) osigurava da se transakcija tretira kao jedna nedjeljiva jedinica rada. To znači da se transakcija provodi u cijelosti ili se uopće ne izvrši, budući da djelomične transakcije nisu moguće. Ako bilo koji dio transakcije ne uspije, cijela transakcija se vraća, a baza podataka se vraća u prethodno stanje. Atomarnost jamči da baza podataka ostane dosljedna čak i u slučaju kvarova ili prekida. Konzistentnost (engl. Consistency) osigurava da transakcija dovodi bazu podataka iz jednog dosljednog stanja u drugo. Izolacija (engl. Isolation) osigurava da istodobne transakcije ne ometaju jedna drugu. Svaka transakcija treba biti izolirana i nesvjesna drugih transakcija koje se izvode istovremeno. Izolacija sprječava nedosljednost podataka i sukobe koji bi mogli nastati zbog istovremenog pristupa. Trajnost (engl. Durability) jamči da se nakon izvršenja transakcije njezine promjene trajno spremaju. Osigurava da se baza podataka može oporaviti i obnoviti izvršene transakcije čak i nakon neočekivanog događaja. Usklađenost s ACID-om ključna je u scenarijima u kojima su dosljednost i pouzdanost podataka najvažniji, kao što su financijski sustavi, platforme za internet trgovinu i poslovne aplikacije. SQL Server pruža robusne sigurnosne značajke za zaštitu podataka. Podržava mehanizme provjere autentičnosti, kontrolu pristupa temeljenu na ulogama, enkripciju i reviziju kako bi se osigurala povjerljivost i integritet podataka. Ovo su neke od značajki zbog kojih je SQL Server odabran za realizaciju ove baze podataka.

2.2. MongoDB

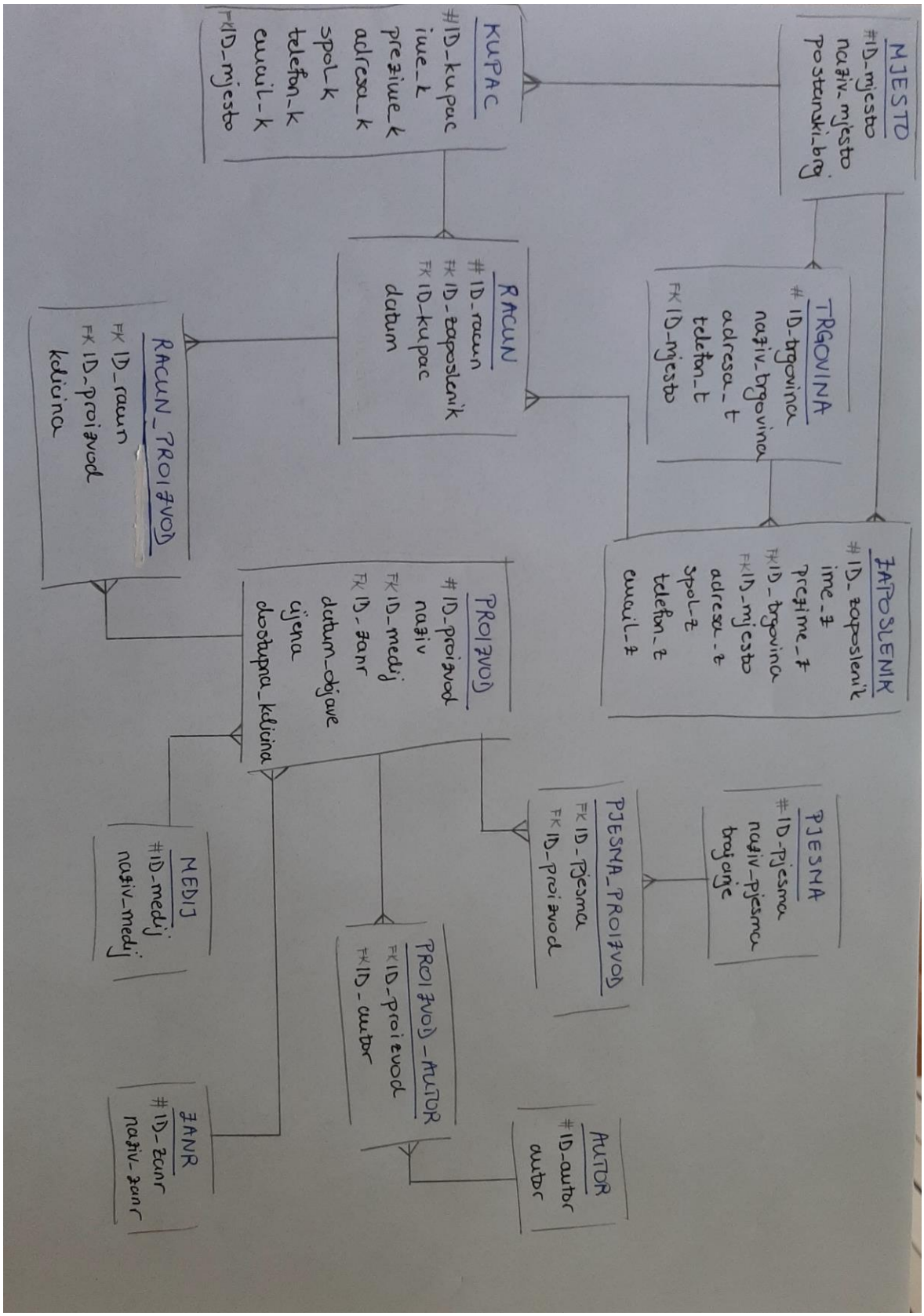
MongoDB je popularan „open-source“ NoSQL sustav za upravljanje bazom podataka koji nudi jedinstven skup značajki i svojstava. MongoDB je baza podataka orijentirana na dokumente, što znači da pohranjuje i dohvaća podatke u fleksibilnim dokumentima sličnim JSON-u, poznatim kao BSON (engl. Binary JSON). To omogućuje pohranjivanje i organiziranje podataka na prirodniji i fleksibilniji način u usporedbi s tradicionalnim relacijskim bazama podataka temeljenim na tablici. Kombinira sposobnost horizontalnog skaliranja sa značajkama kao što su sekundarni indeksi, upiti raspona, sortiranje, sakupljanje i geoprostorni indeksi. MongoDB je dizajniran za horizontalno skaliranje dopuštajući distribuciju podataka na više poslužitelja i omogućavajući visokoučinkovito dohvaćanje i obradu podataka. Njegov model podataka koji je orijentiran na dokumente olakšava podjelu i dohvaćanje podataka preko klastera, automatsku raspodjelu dokumenata i usmjeravanje korisničkih zahtjeva na ispravne uređaje. Kad klaster zatreba veći kapacitet, mogu se dodati novi uređaji i MongoDB će shvatiti kako se postojeći podaci trebaju raspodijeliti na njima. MongoDB-ova fleksibilna shema omogućuje dinamičko modeliranje podataka bez shema. Svaki dokument u zbirci može imati različitu strukturu, a polja se mogu dodavati ili mijenjati u hodu. Ova fleksibilnost je korisna u scenarijima u kojima se podaci razvijaju tijekom vremena ili imaju različite strukture. MongoDB podržava različite vrste indeksa, uključujući indekse s jednim poljem, složene indekse, indekse s više ključeva i geoprostorne indekse. Indeksiranje poboljšava izvedbu upita omogućavanjem brzog pretraživanja podataka. Za razliku od SQL-a koji koristi ACID osobine transakcija, NoSQL sustavi ih pokušavaju zaobići kako bi se postigla dosta veća brzina i skalabilnost. CAP (Consistency, Availability, Partitioning) teorem dokazuje da se u distribuiranom sustavu mogu postići samo dva od tri navedena svojstva u isto vrijeme. Konzistentnost (engl. Consistency) zahtjeva da se u sustavu koji ima više kopija istih podataka, te kopije se moraju međusobno podudarati. U horizontalno skaliranoj bazi može se jamčiti da se sve kopije podudaraju ako se žrtvuje dostupnost ili parcijalni rad. Dostupnost (engl. Availability) podrazumijeva da za bilo koji zahtjev za podacima, postoji velika vjerojatnost da neko radno računalo može pristupiti nekoj kopiji podataka i zadovoljiti zahtjev, čak ako su neka od računala ili dijelovi podataka nedostupni. Potpuna dostupnost može se ostvariti samo ako se žrtvuje konzistentnost ili parcijalni rad. Parcijalni rad (engl. Partitioning) je slučaj kada mrežni kvar uzrokuje da jedan podskup računala u distribuiranom sustavu ne može komunicirati s ostalim računalima. Rezultat toga je da kopije baze podataka

održavane od strane različitih računala mogu se međusobno uskladiti. Kako bi ovo bilo moguće u sustavu, nije moguće realizirati konzistentnost ili dostupnost.

3. Izrada projekta

3.1. Baza podataka u SQL Serveru

Kako je zamišljeno, radi se o tvrtki koja ima lanac trgovina u jednoj državi. Glazbena trgovina specijalizirana je za prodaju fizičkih CD-ova, DVD-ova i ploča, zajedno s digitalnim glazbenim zapisima, odnosno kazetama. Vodi se evidencija o lokaciji i informacijama svake od trgovina, zaposlenicima, kupcima te proizvodima koji se prodaju. Izdaje se račun za svaku transakciju s podacima o kupcu, zaposleniku koji je uslužio kupca, vremenu izdavanja računa i kupljenim proizvodima. Kako se jedan proizvod može naći na više računa, a jedan račun može sadržavati više proizvoda, račun i proizvod su realizirani vezom više na više. Proizvod predstavlja glazbeni album iz kojeg možemo saznati naziv albuma, medij na kojem se nalazi, njegov žanr, datum objave, cijenu i dostupnu količinu tog proizvoda. Autor proizvoda i proizvod su prikazani vezom više na više zato što može biti više autora jednog albuma i jedan autor može izdati više albuma. Također, pjesme i proizvodi su realizirani vezom više na više zbog mogućnosti da se jedan album objavi dvaput zbog npr. posebnog izdanja ili se objavi na različitim medijima, pa stoga jedan album sadrži više pjesama i jedna pjesma se može naći na više albuma. Zbog jednostavnosti modela, jedan zaposlenik može raditi samo u jednoj trgovini. Sve prethodno opisano možemo vidjeti na konceptualnom modelu baze podataka na Slici 1.



Slika 1 Konceptualni model

3.1.1. Kreiranje tablica

U ovom dijelu su prikazani upiti za izradu svake od tablica iz konceptualnog modela za relacijsku bazu podataka.

1. MJESTO

```
CREATE TABLE MJESTO(
	ID_mjesto INT IDENTITY PRIMARY KEY,
	naziv_mjesto nvarchar(255) NOT NULL,
	postanski_broj INT NOT NULL

);
```

2. TRGOVINA

```
CREATE TABLE TRGOVINA(
	ID_trgovina INT IDENTITY PRIMARY KEY,
	naziv_trgovina nvarchar(255) NOT NULL,
	adresa_t nvarchar(255) NOT NULL,
	telefon_t nvarchar(255) NOT NULL,
	ID_mjesto INT FOREIGN KEY REFERENCES MJESTO(ID_mjesto) NOT NULL

);
```

3. KUPAC

```
CREATE TABLE KUPAC (
	ID_kupac INT IDENTITY PRIMARY KEY,
	ime_k nvarchar(255) NOT NULL,
	prezime_k nvarchar(255) NOT NULL,
	adresa_k nvarchar(255) NOT NULL,
	spol_k char(1) NOT NULL,
	telefon_k nvarchar(255) NOT NULL,
	email_k nvarchar (255) NOT NULL,
	ID_mjesto INT FOREIGN KEY REFERENCES MJESTO(ID_mjesto) NOT NULL

);
```

4. ZAPOSLENIK

```
CREATE TABLE ZAPOSLENIK(
	ID_zaposlenik INT IDENTITY PRIMARY KEY,
	ime_z nvarchar(255) NOT NULL,
	prezime_z nvarchar(255) NOT NULL,
	ID_trgovina INT FOREIGN KEY REFERENCES TRGOVINA(ID_trgovina) NOT NULL,
	ID_mjesto INT FOREIGN KEY REFERENCES MJESTO(ID_mjesto) NOT NULL,
	adresa_z nvarchar(255) NOT NULL,
	spol_z char(1) NOT NULL,
	telefon_z nvarchar(255) NOT NULL,
	email_z nvarchar (255) NOT NULL

);
```

5. ZANR

```
CREATE TABLE ZANR(
	ID_zanr INT IDENTITY PRIMARY KEY,
	naziv_zanr nvarchar(255) NOT NULL

);
```

6. MEDIJ

```
CREATE TABLE MEDIJ(
	ID_medij INT IDENTITY PRIMARY KEY,
	naziv_medij nvarchar(255) NOT NULL

);
```

7. AUTOR

```
CREATE TABLE AUTOR(
	ID_autor INT IDENTITY PRIMARY KEY,
	autor nvarchar(255) NOT NULL

);
```

8. PJESMA

```
CREATE TABLE PJESMA(
	ID_pjesma INT IDENTITY PRIMARY KEY,
	naziv_pjesma nvarchar(255) NOT NULL,
	trajanje INT NOT NULL

);
```

9. PROIZVOD

```
CREATE TABLE PROIZVOD(
	ID_proizvod INT IDENTITY PRIMARY KEY,
	naziv nvarchar(255) NOT NULL,
	ID_medij INT FOREIGN KEY REFERENCES MEDIJ(ID_medij) NOT NULL,
	ID_zanr INT FOREIGN KEY REFERENCES ZANR(ID_zanr) NOT NULL,
	datum_objave DATETIME NOT NULL,
	cijena INT NOT NULL,
	dostupna_kolicina INT NOT NULL

);
```

10. PJESMA_PROIZVOD

```
CREATE TABLE PJESMA_PROIZVOD(
	ID_pjesma INT FOREIGN KEY REFERENCES PJESMA(ID_pjesma) NOT NULL,
	ID_proizvod INT FOREIGN KEY REFERENCES PROIZVOD(ID_proizvod) NOT NULL

);
```

11. PROIZVOD_AUTOR

```
CREATE TABLE PROIZVOD_AUTOR(
	ID_proizvod INT FOREIGN KEY REFERENCES PROIZVOD(ID_proizvod) NOT NULL,
	ID_autor INT FOREIGN KEY REFERENCES AUTOR(ID_autor) NOT NULL

);
```

12. RACUN

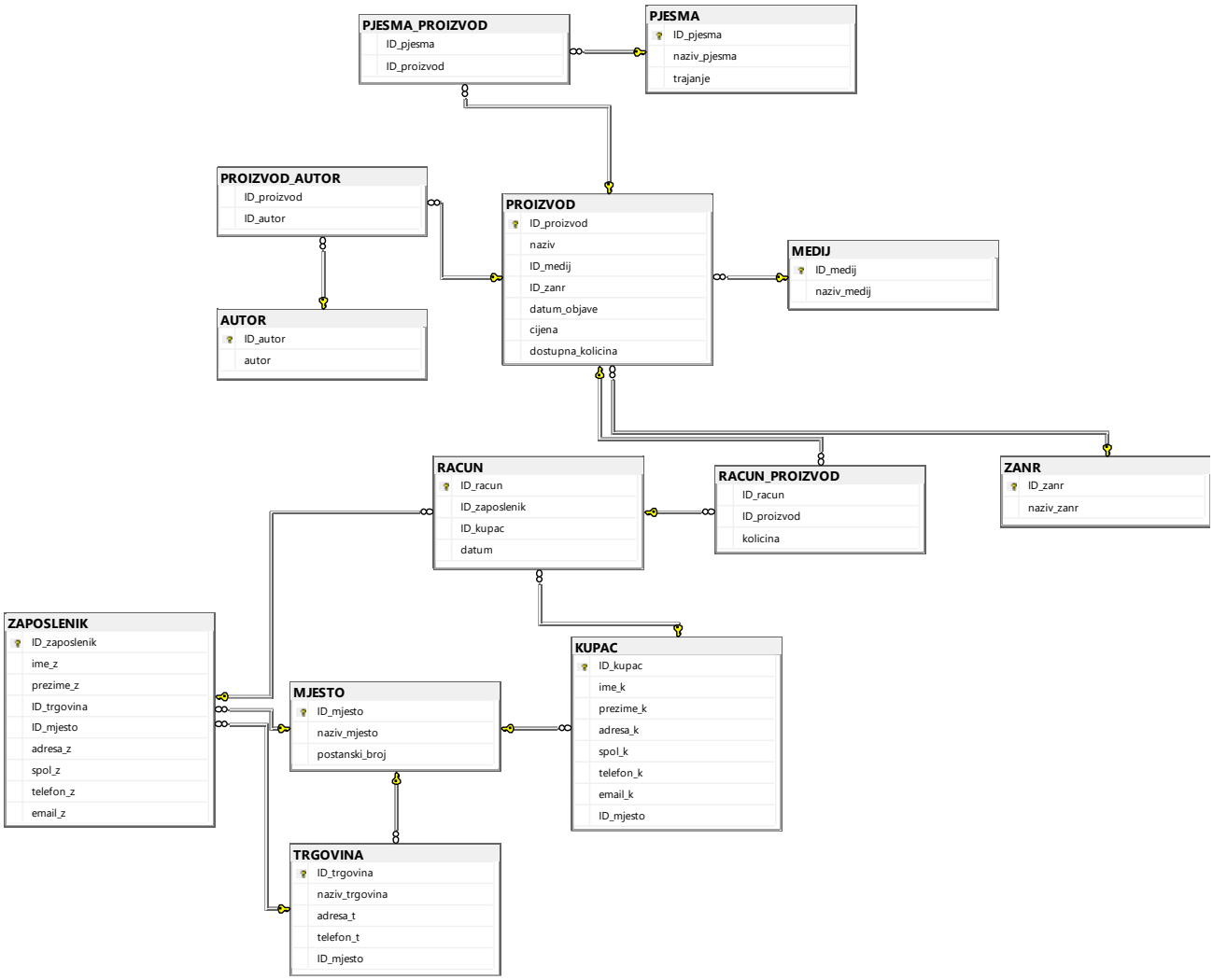
```
CREATE TABLE RACUN(
	ID_racun INT IDENTITY PRIMARY KEY,
	ID_zaposlenik INT FOREIGN KEY REFERENCES ZAPOSLENIK(ID_zaposlenik) NOT
NULL,
	ID_kupac INT FOREIGN KEY REFERENCES KUPAC(ID_kupac) NOT NULL,
	datum DATETIME NOT NULL

);
```

13. RACUN_PROIZVOD

```
CREATE TABLE RACUN_PROIZVOD(
        ID_racun INT FOREIGN KEY REFERENCES RACUN(ID_racun) NOT NULL,
        ID_proizvod INT FOREIGN KEY REFERENCES PROIZVOD(ID_proizvod) NOT NULL,
        kolicina INT NOT NULL
);
```

Nakon izrade svih tablica možemo prikazati dijagram entiteti-veze baze podataka (Slika 2).



Slika 2 Dijagram entiteti-veze

3.1.2. Umetanje podataka

Zbog veće količine podataka, podaci su umetani iz JSON datoteka koristeći „SQL BULK INSERT“ metodu. Podaci su nasumično generirani u online „data generatoru“ te nisu validni u ovom kontekstu.

U nastavku je prikaz upita za unos podataka u svaku od tablica.

1. MJESTO

```
declare @json_data varchar(max);

select @json_data = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\MJESTO.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into MJESTO (naziv_mjesto, postanski_broj)
select naziv_mjesto, postanski_broj from openjson (@json_data)
WITH
(
	naziv_mjesto nvarchar(255),
	postanski_broj int
)


select * from MJESTO
```

2. TRGOVINA

```
declare @json_data_trg varchar(max);

select @json_data_trg = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\TRGOVINA.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into TRGOVINA (naziv_trgovina, adresa_t, telefon_t, ID_mjesto)
select naziv_trgovina, adresa_t, telefon_t, ID_mjesto from openjson
(@json_data_trg)
WITH
(
	naziv_trgovina nvarchar(255),
	adresa_t nvarchar(255),
	adresa_t nvarchar(255),
	ID_mjesto int
)
select * from TRGOVINA
```

3. KUPAC

```
declare @json_data_kup varchar(max);

select @json_data_kup = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\KUPAC.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into KUPAC (ime_k, prezime_k, adresa_k, spol_k, telefon_k, email_k,
ID_mjesto)
```

```sql
select ime_k, prezime_k, adresa_k, spol_k, telefon_k, email_k, ID_mjesto from
openjson (@json_data_kup)
WITH
(
	ime_k nvarchar(255),
	prezime_k nvarchar(255),
	adresa_k nvarchar(255),
	spol_k char(1),
	telefon_k nvarchar(255),
	email_k nvarchar (255),
	ID_mjesto int
)

select * from KUPAC
```

4. ZAPOSLENIK

```sql
declare @json_data_zap varchar(max);

select @json_data_zap = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\ZAPOSLENIK.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into ZAPOSLENIK (ime_z, prezime_z, ID_trgovina, ID_mjesto, adresa_z,
spol_z, telefon_z, email_z)
select ime_z, prezime_z, ID_trgovina, ID_mjesto, adresa_z, spol_z, telefon_z,
email_z from openjson (@json_data_zap)
WITH
(
	ime_z nvarchar(255),
	prezime_z nvarchar(255),
	ID_trgovina INT,
	ID_mjesto INT,
	adresa_z nvarchar(255),
	spol_z char(1),
	telefon_z nvarchar(255),
	email_z nvarchar (255)
)

select * from ZAPOSLENIK;
```

5. ZANR

```sql
declare @json_data_zanr varchar(max);

select @json_data_zanr = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\ZANR.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into ZANR (naziv_zanr)
select naziv_zanr from openjson (@json_data_zanr)
WITH
(
	naziv_zanr nvarchar(255)
)
```

```
select * from ZANR;
```

6. MEDIJ

```
declare @json_data_medij varchar(max);

select @json_data_medij = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\MEDIJ.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into MEDIJ (naziv_medij)
select naziv_medij from openjson (@json_data_medij)
WITH
(
	naziv_medij nvarchar(255)
)
select * from MEDIJ;
```

7. AUTOR

```
declare @json_data_autor varchar(max);

select @json_data_autor = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\AUTOR.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into AUTOR (autor)
select autor from openjson (@json_data_autor)
WITH
(
	autor nvarchar(255)
)
select * from AUTOR;
```

8. PJESMA

```
declare @json_data_pjesma varchar(max);

select @json_data_pjesma = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\PJESMA.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into PJESMA (naziv_pjesma, trajanje)
select naziv_pjesma, trajanje from openjson (@json_data_pjesma)
WITH
(
	naziv_pjesma nvarchar(255),
	trajanje INT
)
```

```sql
select * from PJESMA;
```

9. PROIZVOD

```sql
declare @json_data_proz varchar(max);

select @json_data_proz = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\PROIZVOD.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into PROIZVOD (naziv, ID_medij, ID_zanr, datum_objave, cijena,
dostupna_kolicina)
select naziv, ID_medij, ID_zanr, datum_objave, cijena, dostupna_kolicina from
openjson (@json_data_proz)
WITH
(
	naziv nvarchar(255),
	ID_medij INT,
	ID_zanr INT,
	datum_objave DATETIME,
	cijena INT,
	dostupna_kolicina INT
)

select * from PROIZVOD;
```

10. PJESMA_PROIZVOD

```sql
declare @json_data_pje_proz varchar(max);

select @json_data_pje_proz = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\PJESMA_PROIZVOD.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into PJESMA_PROIZVOD (ID_pjesma, ID_proizvod)
select ID_pjesma, ID_proizvod from openjson (@json_data_pje_proz)
WITH
(
	ID_pjesma INT,
	ID_proizvod INT
)

select * from PJESMA_PROIZVOD;
```

11. PROIZVOD_AUTOR

```sql
declare @json_data_proz_aut varchar(max);

select @json_data_proz_aut = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\PROIZVOD_AUTOR.json', SINGLE_CLOB
) as datasource;

--print @json_data
```

```sql
insert into PROIZVOD_AUTOR (ID_autor, ID_proizvod)
select ID_autor, ID_proizvod from openjson (@json_data_proz_aut)
WITH
(
	ID_autor INT,
	ID_proizvod INT
)

select * from PROIZVOD_AUTOR;
```

12. RACUN

```sql
declare @json_data_rcn varchar(max);

select @json_data_rcn = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\RACUN.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into RACUN (ID_zaposlenik, ID_kupac, datum)
select ID_zaposlenik, ID_kupac, datum from openjson (@json_data_rcn)
WITH
(
	ID_zaposlenik INT,
	ID_kupac INT,
	datum DATETIME
)

select * from RACUN
```

13. RACUN_PROIZVOD

```sql
declare @json_data_rcn_proz varchar(max);

select @json_data_rcn_proz = BulkColumn
from openrowset
(
	BULK 'C:\DATA\podaci\RACUN_PROIZVOD.json', SINGLE_CLOB
) as datasource;

--print @json_data

insert into RACUN_PROIZVOD (ID_racun, ID_proizvod, kolicina)
select ID_racun, ID_proizvod, kolicina from openjson (@json_data_rcn_proz)
WITH
(
	ID_racun INT,
	ID_proizvod INT,
	kolicina INT
)

select * from RACUN_PROIZVOD;
```

3.1.3. Pogledi

Napravljena je nekolicina upita koji su spremljeni kao pogledi (engl. views). Mjereno je vrijeme izvršavanja svakog upita koristeći naredbu „SET STATISTICS TIME ON“.

Napravljen je pogled koji vraća ime i prezime zaposlenika, njegov broj obavljenih transakcija i trgovina u kojoj je zaposlen.

```
CREATE VIEW v_zaposlenik_prodaja
AS
select Z.ime_z as ime_zaposlenika, Z.prezime_z as prezime_zaposlenika,
COUNT(R.ID_zaposlenik) as broj, t.naziv_trgovina as naziv_trgovina
from RACUN R
inner join ZAPOSLENIK Z ON Z.ID_zaposlenik=R.ID_zaposlenik
inner join TRGOVINA T ON Z.ID_trgovina = T.ID_trgovina
group by  Z.ime_z, Z.prezime_z, t.naziv_trgovina
;

select * from v_zaposlenik_prodaja order by broj desc;
```

	ime_zaposlenika	prezime_zaposlenika	broj	naziv_trgovina
1	Matt	Morgan	50	Comodo
2	Daron	Smith	49	Biolife Grup
3	Enoch	Collins	48	Biolife Grup
4	Chris	Logan	48	Apple Inc.
5	Matt	Shaw	48	21st Century Fox
6	Luke	Knight	48	Vodafone
7	Ronald	Reid	48	Erickson
8	Summer	John	48	Metro Cash&Carry
9	Rosalee	Clark	47	Metro Cash&Carry
10	Kieth	Attwood	47	ExxonMobil
11	Mary	Lyon	47	Metro Cash&Carry
12	Christy	Eyres	47	Comcast
13	Bob	Judd	47	Vodafone
14	Bethany	Rose	46	Areon Impex
15	Bree	Phillips	46	Erickson
16	Alan	Buckley	46	Danone
17	Angelique	Boulton	46	Global Print

Slika 3 Rezultat pogleda v_zaposlenik_prodaja

```
SQL Server parse and compile time:
   CPU time = 9 ms, elapsed time = 9 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(3174 rows affected)

 SQL Server Execution Times:
   CPU time = 47 ms,  elapsed time = 115 ms.

Completion time: 2023-06-16T15:29:13.4267529+02:00
```

Slika 4 Vrijeme izvršavanja

Pogled v_proizvod_prodaja ispisuje naziv proizvoda i dosad prodanu količinu tog proizvoda.

```
CREATE VIEW v_proizvod_prodaja
AS
select P.naziv, count(R.ID_proizvod) as broj
from RACUN_PROIZVOD R
inner join PROIZVOD P ON P.ID_proizvod=R.ID_proizvod
group by P.naziv;

SELECT * FROM v_proizvod_prodaja ORDER BY broj DESC;
```

	naziv	broj
1	Elvis 30 #1 Hits	739
2	Born to Run	446
3	A Night at the Opera	429
4	The Temptations Sing Smokey	428
5	The Best of the Everly Brothers	422
6	The Ultimate Collection	421
7	A Rush of Blood to the Head	420
8	Ill Communication	418
9	Sail Away	418
10	The Ritchie Valens Story	418
11	The Screaming End: The Best of Gene Vincent and ...	417
12	Layla And Other Assorted Love Songs	417
13	For the Lonely: 18 Greatest Hits	417
14	Burnin'	417
15	Sounds of Summer: The Very Best of the Beach Boys	416
16	The Best of Frankie Lymon and the Teenagers	415
17	All That You Can't Leave Behind	415

Slika 5 Rezultat pogleda v_proizvod_prodaja

```
SQL Server parse and compile time:
   CPU time = 6 ms, elapsed time = 6 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(367 rows affected)

 SQL Server Execution Times:
   CPU time = 31 ms,  elapsed time = 94 ms.

Completion time: 2023-06-16T15:30:41.2947549+02:00
```

Slika 6 Vrijeme izvršavanja

Napravljen je i pogled koji računa ukupni broj narudžbi za svaku godinu poslovanja tvrtke. Ispisuje se godina poslovanja i broj narudžbi za tu godinu.

```
CREATE VIEW v_narudzbe_godine
AS
select YEAR(datum) as godina, COUNT(ID_racun) as broj_narudzbi
from RACUN
group by YEAR(datum);

select * from v_narudzbe_godine order by godina desc;
```

	godina	broj_narudzbi
1	2023	3317
2	2022	7446
3	2021	7387
4	2020	7418
5	2019	7460
6	2018	7294
7	2017	7441
8	2016	7412
9	2015	7481
10	2014	7420
11	2013	7420
12	2012	7363
13	2011	7541
14	2010	7600

j

Slika 7 Rezultat pogleda v_narudzbe_godine

```
SQL Server parse and compile time:
   CPU time = 5 ms, elapsed time = 5 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(14 rows affected)

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 25 ms.

Completion time: 2023-06-16T15:32:45.1769535+02:00
```

Slika 8 Vrijeme izvršavanja

Pogled v_proizvod_medij ispisuje koliko se proizvoda nalazi na svakom od 4 moguća medija.

```
CREATE VIEW v_proizvod_medij
AS
select M.naziv_medij, count(P.ID_medij) as broj
```

```
from PROIZVOD P
inner join MEDIJ M on M.ID_medij = P.ID_medij
group by M.naziv_medij;

select * from v_proizvod_medij;
```

	naziv_medij	broj
1	CD	92
2	DVD	106
3	Record	88
4	Track	82

Slika 9 Rezultat pogleda v_proizvod_medij

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 4 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(4 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2023-06-16T15:33:51.5683716+02:00
```

Slika 10 Vrijeme izvršavanja

Kako bi se vodila evidencija o kupcima napravljen je pogled v_najcesci_kupac koji prikazuje imena i prezimena najčešćih kupaca i broj njihovih kupnji.

```
CREATE VIEW v_najcesci_kupac
AS
select K.ime_k as ime_kupca, K.prezime_k as prezime_kupca, COUNT(R.ID_kupac) as broj
from RACUN R
inner join KUPAC K on K.ID_kupac=R.ID_kupac
group by K.ime_k, K.prezime_k


select * from v_najcesci_kupac order by broj desc;
```

	ime_kupca	prezime_kupca	broj
1	Sabrina	Vangness	15
2	Matthew	Waterson	12
3	Monica	Chester	12
4	Eduardo	Dillon	12
5	Harriet	Forester	12
6	Beatrice	Freeburn	12
7	Skylar	Reid	12
8	Josh	Richards	11
9	Denny	Pratt	11
10	Rick	Preston	11
11	Sofia	Purvis	11
12	Nathan	Penn	11
13	Willow	Pitt	11
14	Ryan	Poulton	11
15	Liam	Nayler	11
16	John	Harrison	11
17	Peyton	Martin	11

Slika 11 Rezultat pogleda v_najcesci_kupac

```
SQL Server parse and compile time:
   CPU time = 5 ms, elapsed time = 5 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(26961 rows affected)

 SQL Server Execution Times:
   CPU time = 438 ms,  elapsed time = 591 ms.

Completion time: 2023-06-16T15:32:10.6130296+02:00
```

Slika 12 Vrijeme izvršavanja

Za pregled svakog od dostupnih albuma i pjesama koje se nalaze na njemu, njihovo trajanje u sekundama i autora napravljen je pogled v_album_pjesma_autor.

```
CREATE VIEW v_album_pjesma_autor
AS
SELECT P.naziv, PJ.naziv_pjesma, PJ.trajanje, A.autor
FROM PJESMA_PROIZVOD PP
INNER JOIN PROIZVOD P ON PP.ID_proizvod=P.ID_proizvod
INNER JOIN PJESMA PJ ON PP.ID_pjesma=PJ.ID_pjesma
INNER JOIN PROIZVOD_AUTOR AP ON AP.ID_proizvod=P.ID_proizvod
INNER JOIN AUTOR A ON AP.ID_autor=A.ID_autor
GROUP BY P.naziv, PJ.naziv_pjesma, PJ.trajanje, A.autor

select * from v_album_pjesma_autor;
```

	naziv	naziv_pjesma	trajanje	autor
1	#1 Record/Radio City	Baby Watcha Want Me To Do - Live	269	Cheap Trick
2	#1 Record/Radio City	Cuando Vuelan Las Gaviotas (with Mariachi Vargas...	261	Cheap Trick
3	#1 Record/Radio City	Don Palabras	209	Cheap Trick
4	#1 Record/Radio City	Goodbye Forever	172	Cheap Trick
5	#1 Record/Radio City	I Think I'll Disappear Now	270	Cheap Trick
6	#1 Record/Radio City	Les poissons	181	Cheap Trick
7	#1 Record/Radio City	My Maria	140	Cheap Trick
8	#1 Record/Radio City	No Depression	196	Cheap Trick
9	#1 Record/Radio City	Preludium	265	Cheap Trick
10	#1 Record/Radio City	Ringleader	201	Cheap Trick
11	#1 Record/Radio City	Shoot Yo Mans	130	Cheap Trick
12	#1 Record/Radio City	Your Name on a Grain of Rice	246	Cheap Trick
13	12 Greatest Hits	Con la Muerte Entre los PuĂ±os (El Boxeador)	227	Bill Haley & His Comets
14	12 Greatest Hits	Cupid In Me	222	Bill Haley & His Comets
15	12 Greatest Hits	Dark Alley	166	Bill Haley & His Comets
16	12 Greatest Hits	Dry-land Fish	247	Bill Haley & His Comets
17	12 Greatest Hits	Fistful of Steel	135	Bill Haley & His Comets
18	12 Greatest Hits	Further Away	218	Bill Haley & His Comets
19	12 Greatest Hits	Gabrielle Douglas	234	Bill Haley & His Comets
20	12 Greatest Hits	I Do' Wanna Know - Live at Kemper Arena, Kansas ...	268	Bill Haley & His Comets
21	12 Greatest Hits	Mr.HorseOpera	251	Bill Haley & His Comets
22	12 Greatest Hits	Psycho	284	Bill Haley & His Comets
23	1999	Before The Next Teardrop Falls	178	Green day
24	1999	Concerto for Piano & Orchestra No. 2 in B-Flat Majo...	281	Green day
25	1999	Damaged	353	Green day
26	1999	Eternal-Hide-And-Seek	305	Green day
27	1999	For Cantilever 1	174	Green day
28	1999	I Need You - Live From The Eleventh Hour Tour	143	Green day
29	1999	Les Ă®les au soleil (los)	156	Green day
30	1999	Roly Poly	212	Green day

Slika 13 Rezultat pogleda v_album_pjesma_autor

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 10 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(3914 rows affected)

 SQL Server Execution Times:
   CPU time = 32 ms,  elapsed time = 112 ms.

Completion time: 2023-06-16T15:35:02.1797798+02:00
```

Slika 14 Vrijeme izvršavanja

Ako nas zanima koliko svaki od autora ima objavljenih albuma, možemo saznati iz pogleda v_album_autor.

```
CREATE VIEW v_album_autor
AS
SELECT A.autor, COUNT (PA.ID_autor) as broj
FROM PROIZVOD_AUTOR PA
INNER JOIN PROIZVOD P ON PA.ID_proizvod=P.ID_proizvod
INNER JOIN AUTOR A ON PA.ID_autor=A.ID_autor
GROUP BY A.autor
```

```
SELECT * FROM v_album_autor ORDER BY broj desc;
```

	autor	broj
1	Afrika Bambaataa & the Soulsonic Force	5
2	Buffalo Springfield	5
3	Lavern Baker	5
4	Love	5
5	The Drifters	5
6	R. Kelly	4
7	Radiohead	4
8	Bobbie Gentry	4
9	BeyoncĂ©	4
10	George Harrison	4
11	Jefferson Airplane	4
12	John Lee Hooker	3
13	John Lennon	3
14	Johnny Cash	3
15	Joni Mitchell	3
16	Joy Division	3
17	Justin Timberlake	3
18	George Jones	3
19	James Taylor	3
20	Gloria Gaynor	3
21	Daft Punk	3
22	Eagles	3

Slika 15 Rezultat pogleda v_album_autor

```
SQL Server parse and compile time:
   CPU time = 5 ms, elapsed time = 5 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(196 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 1 ms.

Completion time: 2023-06-16T15:37:50.3543616+02:00
```

Slika 16 Vrijeme izvršavanja

Pogled v_proizvod_zanr prebrojava koliko proizvoda, odnosno albuma ima pod kojim žanrom.

```
CREATE VIEW v_proizvod_zanr
AS
select Z.naziv_zanr, count(P.ID_zanr) as broj
from PROIZVOD P
inner join ZANR Z on Z.ID_zanr = P.ID_zanr
group by Z.naziv_zanr;
```

```
select * from v_proizvod_zanr order by broj desc;
```

	naziv_zanr	broj
1	Hard Rock	32
2	Country	29
3	Jazz	29
4	Pop	25
5	Soul	25
6	Opera	24
7	Reggae	24
8	Blues	24
9	Dubstep	24
10	House	24
11	Baroque	23
12	Latin	23
13	Techno	23
14	Hip-Hop	20
15	Drum & Bass	19

Slika 17 Rezultat pogleda v_proizvod_zanr

```
SQL Server parse and compile time:
   CPU time = 8 ms, elapsed time = 8 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(15 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2023-06-16T15:35:57.7163928+02:00
```

Slika 18 Vrijeme izvršavanja

Pogled v_racun_kupac računa ukupan iznos svakog od računa i tko je bio njegov kupac.

```
CREATE VIEW v_racun_kupac
AS
SELECT R.ID_racun, K.ime_k + ' ' + K.prezime_k as ime_prezime_kupca,
SUM(RP.kolicina*P.cijena) as ukupan_iznos
FROM RACUN_PROIZVOD RP
INNER JOIN RACUN R ON RP.ID_racun=R.ID_racun
INNER JOIN PROIZVOD P ON RP.ID_proizvod=P.ID_proizvod
INNER JOIN KUPAC K ON R.ID_kupac=K.ID_kupac
GROUP BY R.ID_racun, K.ime_k, K.prezime_k

SELECT * FROM v_racun_kupac ORDER BY ukupan_iznos desc;
```

	ID_racun	ime_prezime_kupca	ukupan_iznos
1	25027	Christy Lane	17859
2	55774	Chris Noon	17549
3	8231	Lauren Mackenzie	16374
4	57663	Madelyn Fox	15721
5	60848	Gwenyth Tait	15693
6	87325	Sebastian Nicolas	15282
7	57189	Dalia Fox	15279
8	8422	Ivy Evans	15188
9	18615	Sienna Nicolas	15148
10	59095	Daron Rivers	15046
11	36816	Ronald Russell	14972
12	56670	Cedrick Lynch	14922
13	72014	Ethan Snell	14797
14	39496	Leilani Wilde	14766
15	9198	Harry Fields	14708
16	98475	Candice Wilcox	14463
17	61792	Carina Addison	14432
18	57952	Russel Shaw	14358
19	48914	Dasha Gilmour	14356
20	37081	Maxwell Wild	14351
21	24296	Roger Morris	14321
22	55378	Rachael Nelson	14166

Slika 19 Rezultat pogleda v_racun_kupac

```
SQL Server parse and compile time:
   CPU time = 16 ms, elapsed time = 22 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(75340 rows affected)

 SQL Server Execution Times:
   CPU time = 359 ms,  elapsed time = 664 ms.

Completion time: 2023-06-16T15:38:35.8033526+02:00
```

Slika 20 Vrijeme izvršavanja

3.1.4. Procedure

Kao i kod pogleda, mjereno je vrijeme izvršavanja procedura.

Napravljena je procedura koja vraća naziv i cijenu svakog proizvoda u silaznom poretku prema cijeni.

```
CREATE PROCEDURE sp_get_proizvod_cijena
AS
        BEGIN
                SELECT naziv, cijena FROM PROIZVOD ORDER BY 2 DESC;
        END

EXEC sp_get_proizvod_cijena;
```

	naziv	cijena
1	Murmur	999
2	The Very Best of the Impressions	995
3	The Best of the Shangri-Las	993
4	All Things Must Pass	980
5	Music From Big Pink	979
6	Sound Affects	979
7	Out of Time	978
8	2nd to None	978
9	Greatest Hits	976
10	Straight Outta Compton	975
11	Please Please Me	970
12	Sounds of Summer	969
13	Achtung Baby	961
14	Soul Men	960
15	Marquee Moon	958
16	Red Headed Stranger	957
17	Layla And Other Assorted Love Songs	951

Slika 21 Rezultat procedure sp_get_proizvod_cijena

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 2 ms.

(368 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 36 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 38 ms.

Completion time: 2023-06-16T15:52:22.6395510+02:00
```

Slika 22 Vrijeme izvršavanja

Nastavno na prethodnu proceduru napravljena je i procedura koja vraća naziv proizvoda i cijenu silazno u određenom rasponu cijene.

```
CREATE PROCEDURE sp_get_proizvod_cijena_range(@najmanja_cijena INT, @najveca_cijena
INT)
AS
	BEGIN
		SELECT naziv, cijena
		FROM PROIZVOD
		WHERE cijena>@najmanja_cijena and cijena<@najveca_cijena
		ORDER BY 2 DESC;

	END
```

```
EXEC sp_get_proizvod_cijena_range 500, 600;
```

	naziv	cijena
1	Kala	586
2	That's All	586
3	Harvest	585
4	Surrealistic Pillow	583
5	#1 Record/Radio City	579
6	Otis Blue: Otis Redding Sings Soul	577
7	Risqué	571
8	Blonde on Blonde	571
9	Buddy Holly: Greatest Hits	566
10	Violator	563
11	Best of the Crystals	563
12	Aerosmith	562
13	Killing Me Softly	560
14	The Lovin' Spoonful Greatest Hits	558
15	Miss E ... So Addictive	557
16	The Best of the Ronettes	553
17	Who's Next	550

Slika 23 Rezultat procedure sp_get_proizvod_cijena_range

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

(36 rows affected)

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 2 ms.

Completion time: 2023-06-16T15:51:52.8333772+02:00
```

Slika 24 Vrijeme izvršavanja

Procedura sp_proizvod_po_godini vraća sve albume koji su objavljeni u godini koja je unesena.

```
CREATE PROCEDURE sp_proizvod_po_godini(@godina INT)
AS
	BEGIN
```

```
        SELECT P.naziv, (YEAR(P.datum_objave)) as godina_objave
        FROM PROIZVOD P
        WHERE (YEAR(P.datum_objave))=@godina
    END

EXEC sp_proizvod_po_godini 2000
```

	naziv	godina_objave
1	Highway 61 Revisited	2000
2	The Dock of the Bay	2000
3	Retrospective	2000
4	Like a Prayer	2000
5	The Wall	2000
6	Zuma	2000

Slika 25 Rezultat procedure sp_proizvod_po_godini

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

(6 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 2 ms.

Completion time: 2023-06-16T15:51:25.8588924+02:00
```

Slika 26 Vrijeme izvršavanja

Ako želimo pretražiti sve dostupne albume jednog autora, možemo koristiti proceduru sp_proizvod_autor koja vraća autora, naziv albuma, medij na kojem je objavljen i cijenu.

```
CREATE PROCEDURE sp_proizvod_autor(@autor nvarchar(255))
AS
    BEGIN
        SELECT  A.autor, P.naziv, M.naziv_medij, P.cijena
        FROM PROIZVOD_AUTOR PA
        INNER JOIN PROIZVOD P ON PA.ID_proizvod=P.ID_proizvod
        INNER JOIN AUTOR A ON PA.ID_autor=A.ID_autor
        INNER JOIN MEDIJ M ON P.ID_medij=M.ID_medij
        WHERE A.autor like '%'+@autor+'%'
        GROUP BY A.autor, P.naziv, M.naziv_medij, P.cijena
    END

EXEC sp_proizvod_autor 'John'
```

	autor	naziv	naziv_medij	cijena
1	Elton John	Crosby, Stills and Nash	Record	478
2	Elton John	Zuma	CD	258
3	John Lee Hooker	The Beatles 1	CD	83
4	John Lee Hooker	The Freewheelin' Bob Dylan	DVD	677
5	John Lee Hooker	The Georgia Peach	Record	735
6	John Lennon	Meet the Beatles!	Track	874
7	John Lennon	Past Masters	Record	267
8	John Lennon	Violator	Track	563
9	John Mellencamp	Achtung Baby	Record	961
10	Johnny Cash	New York Dolls	CD	781
11	Johnny Cash	Street Songs	CD	300
12	Johnny Cash	The Anthology	Track	734

Slika 27 Rezultat procedure sp_proizvod_autor

```
SQL Server parse and compile time:
   CPU time = 8 ms, elapsed time = 8 ms.

(12 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 2 ms.

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 11 ms.

Completion time: 2023-06-16T15:50:52.9780339+02:00
```

Slika 28 Vrijeme izvršavanja

Iz procedure sp_profit_proizvod možemo saznati koliko profita je tvrtka dobila od prodaje svakog proizvoda.

```
CREATE PROCEDURE sp_profit_proizvod
AS
       BEGIN
              SELECT A.autor, P.naziv, SUM(RP.kolicina*P.cijena) as profit
              FROM PROIZVOD P
              INNER JOIN PROIZVOD_AUTOR PA ON P.ID_proizvod=PA.ID_proizvod
              INNER JOIN AUTOR A ON PA.ID_autor=A.ID_autor
              INNER JOIN RACUN_PROIZVOD RP ON P.ID_proizvod=RP.ID_proizvod
              GROUP BY A.autor, P.naziv
              ORDER BY profit desc

       END

EXEC sp_profit_proizvod
```

	autor	naziv	profit
1	Simon & Garfunkel	Sound Affects	1216897
2	Sinead O'Connor	Sounds of Summer	1193808
3	Madonna	The Best of the Everly Brothers	1176600
4	John Mellencamp	Achtung Baby	1171459
5	Radiohead	Layla And Other Assorted Love Songs	1156416
6	MGMT	My Aim Is True	1155884
7	The Jam	Born to Run	1145520
8	The Notorious B.I.G	Soul Men	1132800
9	Neil Young	The Very Best of the Impressions	1123355
10	The Platters	Marquee Moon	1121818
11	Creedence Clearwater Revival	All Things Must Pass	1119160
12	Pink Floyd	Murmur	1118880
13	Lou Reed	2nd to None	1098294
14	John Lennon	Meet the Beatles!	1097744
15	Bill Haley & His Comets	The Ultimate Collection	1091850
16	The Allman Brothers Band	Blondes Have More Fun	1083240
17	Ritchie Valens	Straight Outta Compton	1081275

Slika 29 Rezultat procedure sp_profit_proizvod

```
SQL Server parse and compile time:
   CPU time = 16 ms, elapsed time = 20 ms.

(368 rows affected)

 SQL Server Execution Times:
   CPU time = 218 ms,  elapsed time = 248 ms.

 SQL Server Execution Times:
   CPU time = 234 ms,  elapsed time = 269 ms.

Completion time: 2023-06-16T15:50:11.5936775+02:00
```

Slika 30 Vrijeme izvršavanja

Koristeći proceduru sp_najstariji_proizvod možemo saznati koji je najstariji album koji tvrtka posjeduje i na kojem mediju je objavljen.

```
CREATE PROCEDURE sp_najstariji_proizvod
AS
	BEGIN
		SELECT P.naziv, M.naziv_medij, YEAR(P.datum_objave) as godina_objave
		FROM PROIZVOD P
		INNER JOIN MEDIJ M ON P.ID_medij=M.ID_medij
		GROUP BY P.naziv, M.naziv_medij, YEAR(P.datum_objave)
		ORDER BY godina_objave asc

	END

EXEC sp_najstariji_proizvod
```

	naziv	naziv_medij	godina_objave
1	Breakaway	Track	1960
2	Fear of a Black Planet	CD	1960
3	Heroes	DVD	1960
4	Sounds of Summer: The Very Best of the Beach Boys	DVD	1960
5	The Best of the Chantels	Record	1960
6	The Very Best of Jackie Wilson	Track	1960
7	Love and Theft	Track	1961
8	Somethin' Else	DVD	1961
9	Something Else By The Kinks	CD	1961
10	Soul Men	Track	1961
11	The Anthology	Track	1961
12	The Black Album	Track	1961
13	Are You Experienced?	DVD	1962
14	Johnny B. Goode: His Complete '50s Chess Recordin...	Record	1962
15	Lady Soul	Track	1962
16	Outlandos d'Amour	DVD	1962
17	Parallel Lines	Record	1962

Slika 31 Rezultat procedure sp_najstariji_proizvod

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 4 ms.

(368 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 38 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 43 ms.

Completion time: 2023-06-16T15:49:30.7438791+02:00
```

Slika 32 Vrijeme izvršavanja

3.1.5. Log tablice

Log tablice se koriste za pohranjivanje promjena koje se događaju unutar baze podataka. Služe kao povijesni zapis aktivnosti kao što su izmjene podataka i mogu biti korisne za potrebe revizije. Napravljene su log tablice za entitete „PROIZVOD“, „RACUN“, „RACUN_PROIZVOD“.

```
CREATE TABLE PROIZVOD_LOG(
       change_id INT IDENTITY PRIMARY KEY,
       ID_proizvod INT NOT NULL,
       naziv NVARCHAR(255) NOT NULL,
       ID_medij INT NOT NULL,
       ID_zanr INT NOT NULL,
       datum_objave DATETIME NOT NULL,
```

```
        cijena INT NOT NULL,
        dostupna_kolicina INT NOT NULL,
        updated_at DATETIME NOT NULL,
        operation CHAR(3) NOT NULL,
        CHECK(operation='INS' or operation='DEL' or operation='UPD')
);
CREATE TABLE RACUN_LOG(
        change_id INT IDENTITY PRIMARY KEY,
        ID_racun INT NOT NULL,
        ID_zaposlenik INT NOT NULL,
        ID_kupac INT NOT NULL,
        datum DATETIME NOT NULL,
        updated_at DATETIME NOT NULL,
        operation CHAR(3) NOT NULL,
        CHECK(operation='INS' or operation='DEL')
);

CREATE TABLE RACUN_PROIZVOD_LOG(
        change_id INT IDENTITY PRIMARY KEY,
        ID_racun INT NOT NULL,
        ID_proizvod INT NOT NULL,
        kolicina INT,
        updated_at DATETIME NOT NULL,
        operation CHAR(3) NOT NULL,
        CHECK(operation='INS' or operation='DEL')
);
```

Log tablice su povezane sa okidačima, koji su prikazani u nastavku.

3.1.6. Okidači

Kada se operacija izmjene podataka dogodi na tablici s pridruženim okidačem, kod okidača se izvršava prije ili nakon izmjene. Time se pruža prilika za bilježenje promjena napravljenih na podacima. Okidači mogu sadržavati logiku za umetanje zapisa u odgovarajuće log tablice kad god se dogodi operacija izmjene podataka.

Kreiran je okidač koji svaki put kad se naruči određeni proizvod provjerava ima li dovoljno tog proizvoda na stanju. Ako proizvoda ima dovoljno, dostupna količina proizvoda se umanjuje za naručenu količinu i promjene se zapisuju u log tablicu „RACUN_PROIZVOD_LOG“, u protivnom se javlja greška i transakcija se poništava.

```sql
CREATE TRIGGER trg_kolicina_proizvod_upd
ON RACUN_PROIZVOD
AFTER INSERT
AS
BEGIN
	declare @ID_proizvod as INT;
	declare @dostupna_kolicina as INT;
	declare @narucena_kolicina as INT;

	SELECT @ID_proizvod = i.ID_proizvod,
		@dostupna_kolicina = p.dostupna_kolicina,
		@narucena_kolicina = i.kolicina
	FROM inserted i
	INNER JOIN PROIZVOD p ON i.ID_proizvod = p.ID_proizvod;

	BEGIN TRANSACTION;

	IF @dostupna_kolicina >= @narucena_kolicina
	BEGIN
		update PROIZVOD
		SET dostupna_kolicina = dostupna_kolicina - (SELECT i.kolicina from
inserted i)
		WHERE ID_proizvod = @ID_proizvod;

		COMMIT;

		INSERT INTO RACUN_PROIZVOD_LOG(
			ID_racun,
			ID_proizvod,
			kolicina,
			updated_at,
			operation
		)
		SELECT
			i.ID_racun,
			i.ID_proizvod,
			i.kolicina,
			GETDATE(),
			'INS'
		FROM inserted i
	END

	ELSE
	BEGIN
		ROLLBACK;
		RAISERROR('Greška!', 0, 1) WITH NOWAIT;
	END

END
```

	change_id	ID_racun	ID_proizvod	kolicina	updated_at	operation
1	1	100000	55	1	2023-06-13 16:52:19.367	INS

Slika 33 Primjer zapisa u tablici "RACUN_PROIZVOD_LOG"

Sljedeći okidač registrira promjene na tablici „PROIZVOD“ i nakon unosa, brisanja ili ažuriranja podataka sve promjene zapisuje u tablicu „PROIZVOD_LOG“.

```
CREATE TRIGGER trg_proizvod_log
ON PROIZVOD
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
	SET NOCOUNT ON;

	IF EXISTS (SELECT * FROM inserted)
	BEGIN
		INSERT INTO PROIZVOD_LOG (
			ID_proizvod,
			naziv,
			ID_medij,
			ID_zanr,
			datum_objave,
			cijena,
			dostupna_kolicina,
			updated_at,
			operation
		)
		SELECT
			ID_proizvod,
			naziv,
			ID_medij,
			ID_zanr,
			datum_objave,
			cijena,
			dostupna_kolicina,
			GETDATE(),
			'INS'
		FROM inserted;
	END;

	IF EXISTS (SELECT * FROM deleted)
	BEGIN
		INSERT INTO PROIZVOD_LOG (
			ID_proizvod,
			naziv,
			ID_medij,
			ID_zanr,
			datum_objave,
			cijena,
			dostupna_kolicina,
			updated_at,
			operation
		)
		SELECT
			ID_proizvod,
			naziv,
			ID_medij,
			ID_zanr,
			datum_objave,
			cijena,
			dostupna_kolicina,
			GETDATE(),
			'DEL'
		FROM deleted;
	END;
```

```
        IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
        BEGIN
                INSERT INTO PROIZVOD_LOG (
                        ID_proizvod,
                        naziv,
                        ID_medij,
                        ID_zanr,
                        datum_objave,
                        cijena,
                        dostupna_kolicina,
                        updated_at,
                        operation
                )
                SELECT
                        i.ID_proizvod,
                        i.naziv,
                        i.ID_medij,
                        i.ID_zanr,
                        i.datum_objave,
                        i.cijena,
                        i.dostupna_kolicina,
                        GETDATE(),
                        'UPD'
                FROM inserted i;
        END;

END;
```

	change_id	ID_proizvod	naziv	ID_medij	ID_zanr	datum_objave	cijena	dostupna_kolicina	updated_at	operation
1	1	369	Dalmatinski hitovi	2	5	2023-06-13 12:55:18.213	150	15	2023-06-13 12:55:18.217	INS
2	2	369	Dalmatinski hitovi	2	5	2023-06-13 12:55:18.213	150	15	2023-06-13 12:55:18.217	UPD
3	3	320	The Ultimate Isley Brothers	1	9	1986-03-06 00:00:00.000	362	2	2023-06-13 16:30:06.077	INS
4	4	320	The Ultimate Isley Brothers	1	9	1986-03-06 00:00:00.000	362	3	2023-06-13 16:30:06.077	DEL
5	5	320	The Ultimate Isley Brothers	1	9	1986-03-06 00:00:00.000	362	2	2023-06-13 16:30:06.077	UPD
6	6	55	The Times They Are A-Changin'	1	13	1983-04-21 00:00:00.000	859	6	2023-06-13 16:52:19.367	INS
7	7	55	The Times They Are A-Changin'	1	13	1983-04-21 00:00:00.000	859	7	2023-06-13 16:52:19.367	DEL
8	8	55	The Times They Are A-Changin'	1	13	1983-04-21 00:00:00.000	859	6	2023-06-13 16:52:19.367	UPD

Slika 34 Primjer zapisa u tablici "PROIZVOD_LOG"

Zadnji okidač nakon unosa ili brisanja podataka iz tablice „RACUN“ promjene upisuje u tablicu „RACUN_LOG“.

```
CREATE TRIGGER trg_racun_log
ON RACUN
AFTER INSERT, DELETE
AS
BEGIN
        SET NOCOUNT ON;
        INSERT INTO RACUN_LOG(
                ID_racun,
                ID_zaposlenik,
                ID_kupac,
                datum,
                updated_at,
                operation
        )
        SELECT
                i.ID_racun,
```

```
            i.ID_zaposlenik,
            i.ID_kupac,
            i.datum,
            GETDATE(),
            'INS'
        FROM inserted i
        UNION ALL
        SELECT
            d.ID_racun,
            d.ID_zaposlenik,
            d.ID_kupac,
            d.datum,
            GETDATE(),
            'DEL'
        FROM deleted d

END
```

	change_id	ID_racun	ID_zaposlenik	ID_kupac	datum	updated_at	operation
1	1	100004	5	6	2023-06-13 16:42:04.037	2023-06-13 16:42:04.047	INS
2	2	100004	5	6	2023-06-13 16:42:04.037	2023-06-13 16:42:53.480	DEL

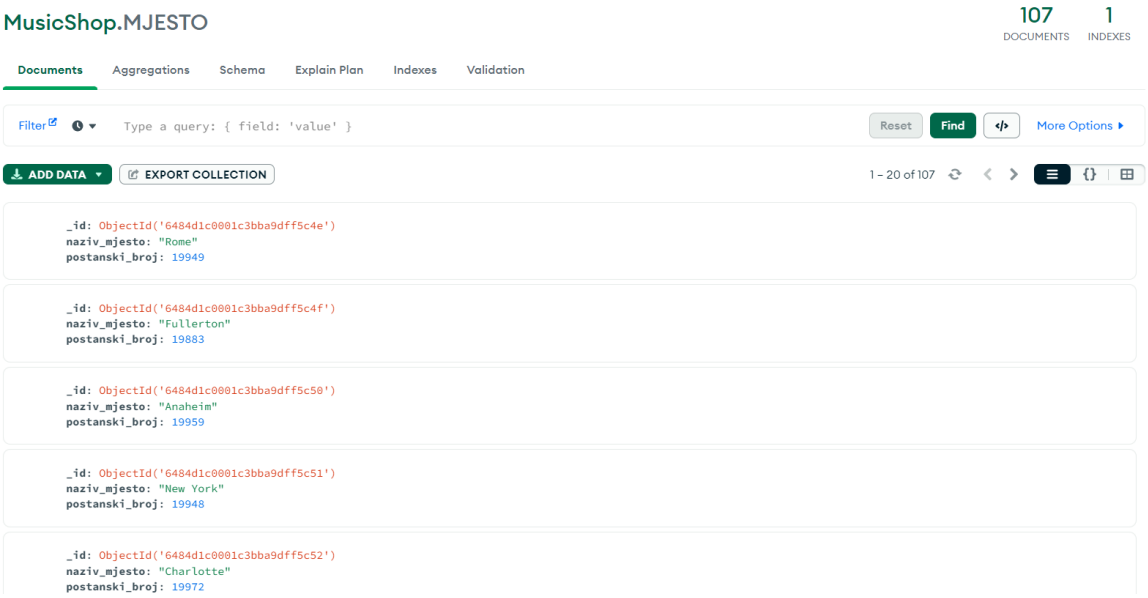
Slika 35 Primjer zapisa u tablici "RACUN_LOG"

3.2. Baza podataka u MongoDB Compass-u

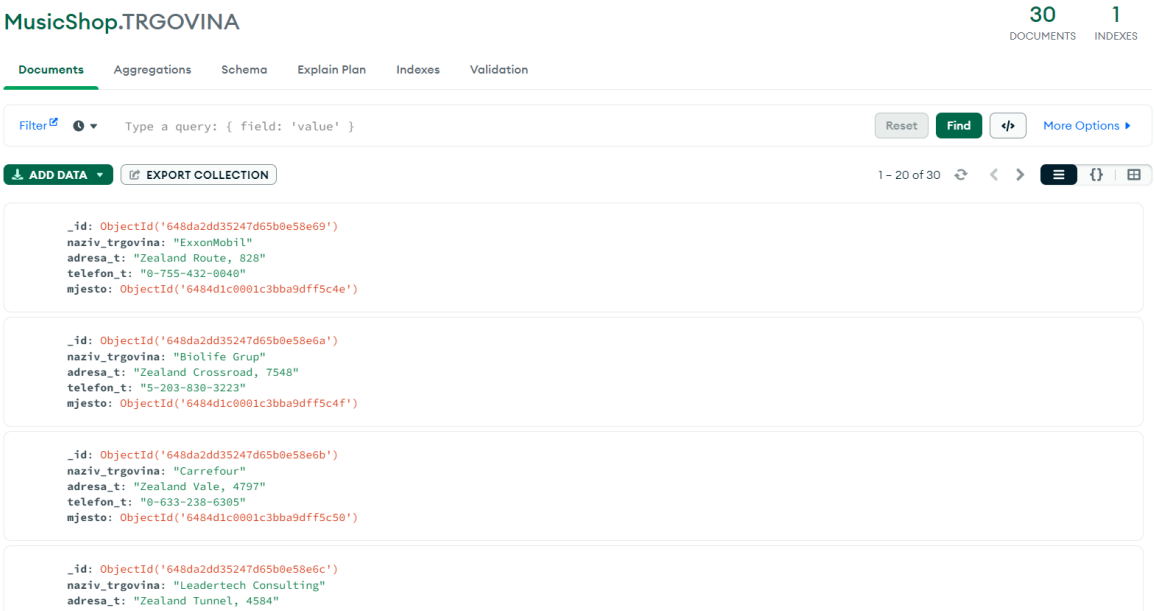
Kao prvi korak izrade baze podataka u MongoDB Compass-u je kreiranje kolekcija. Kreirano je 9 kolekcija, a to su:

1. MJESTO
2. TRGOVINA
3. ZAPOSLENIK
4. PROIZVOD
5. PJESMA
6. AUTOR
7. KUPAC
8. RACUN
9. RACUN_PROIZVOD

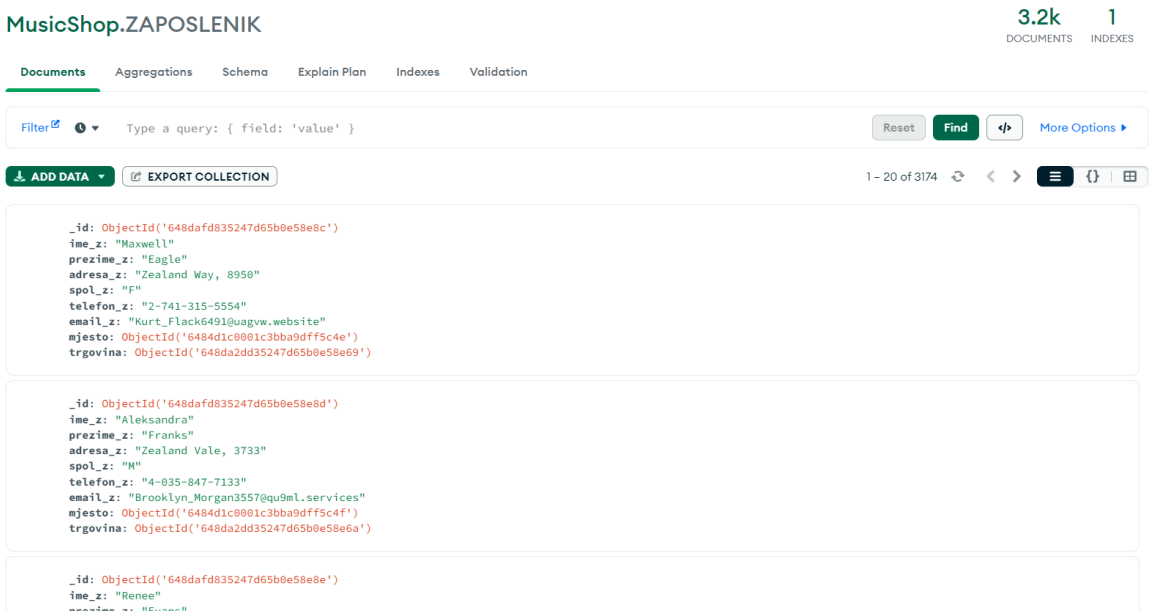
Podaci su umetani iz CSV datoteka koristeći dugme „ADD DATA“ u kolekciji. Jednaka količina podataka je umetnuta kao i u SQL bazi podataka. Neke od kolekcija sadrže „ObjectId“ referentnog dokumenta. Izgled kolekcija je prikazan u nastavku.



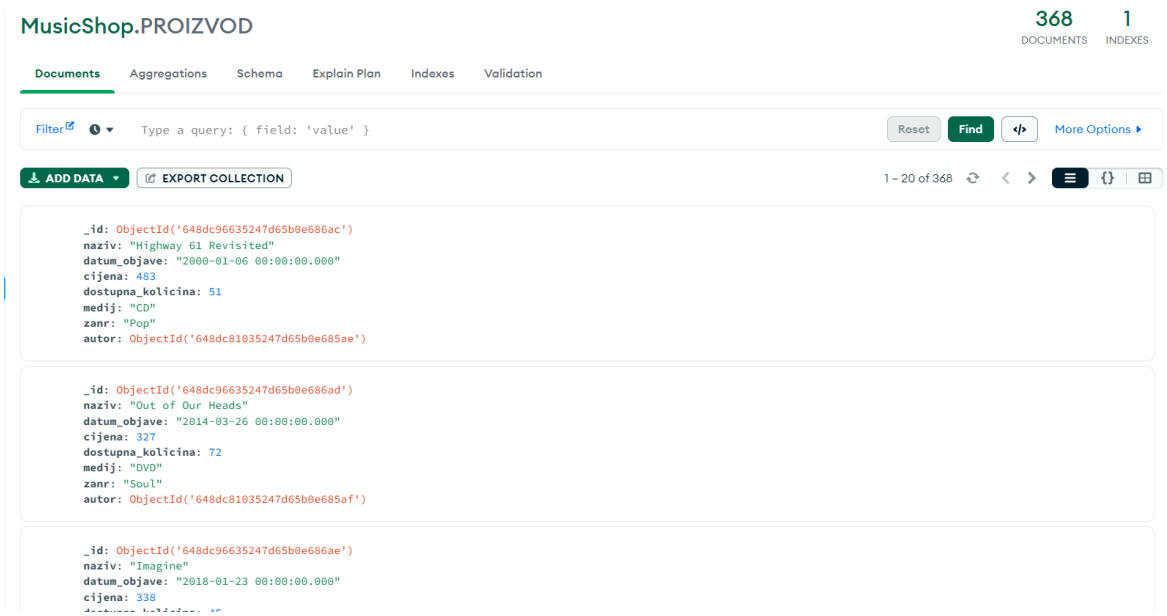
Slika 36 Kolekcija MJESTO



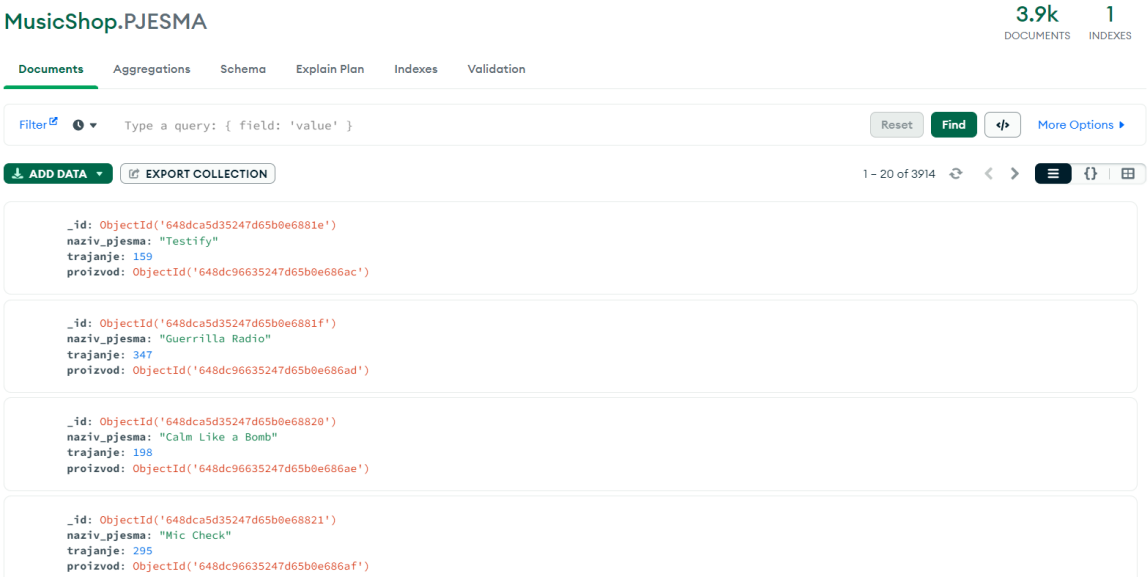
Slika 37 Kolekcija TRGOVINA



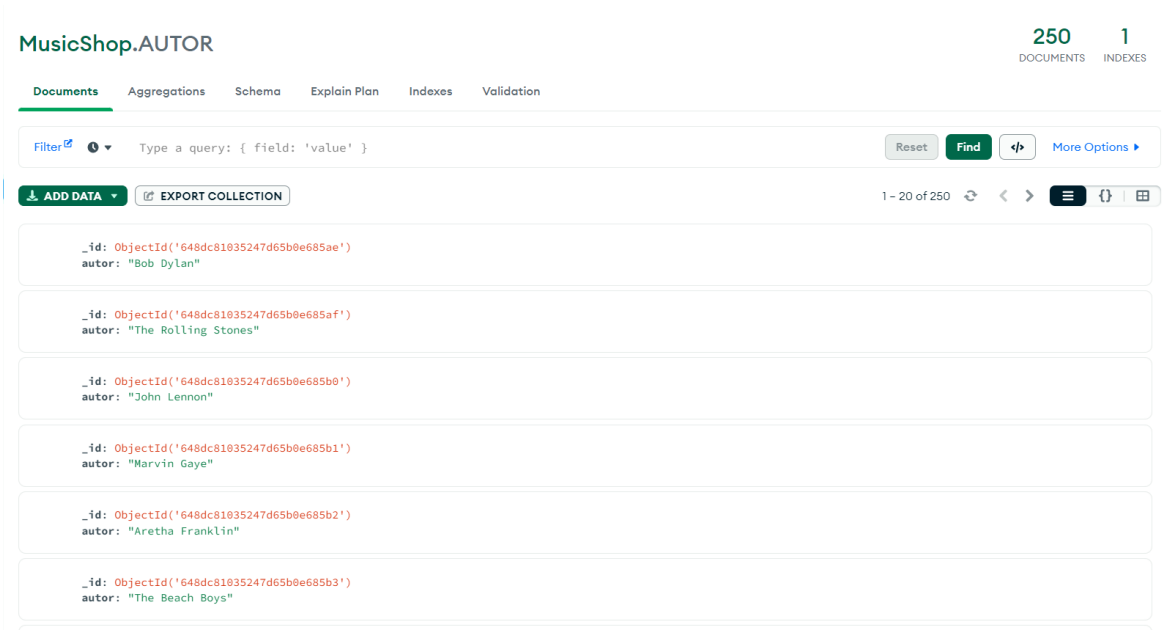
Slika 38 Kolekcija ZAPOSLENIK



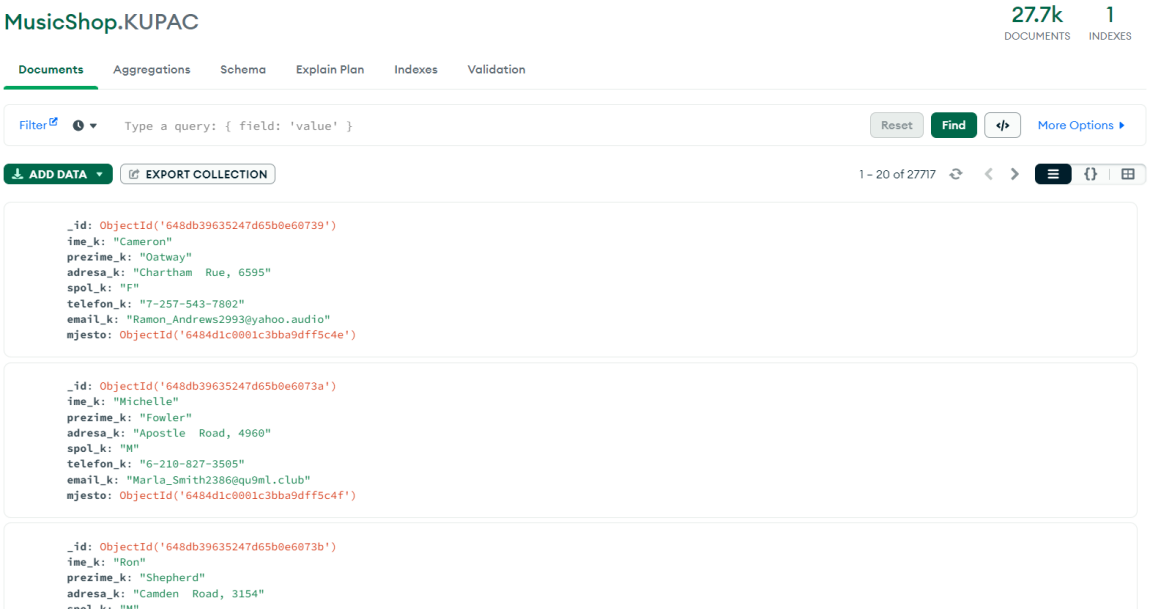
Slika 39 Kolekcija PROIZVOD



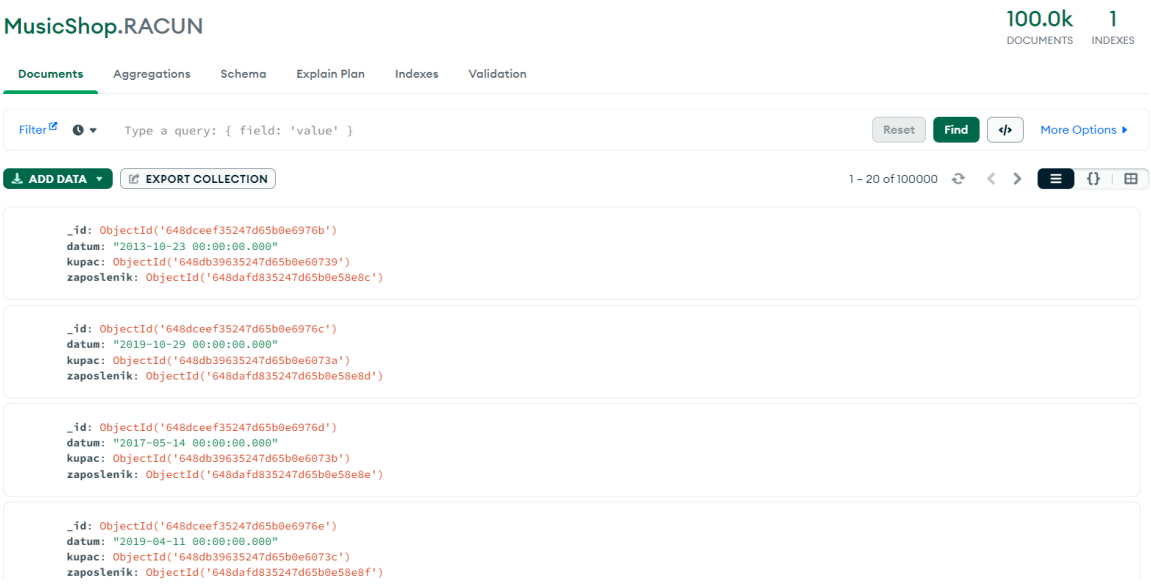
Slika 40 Kolekcija PJESMA



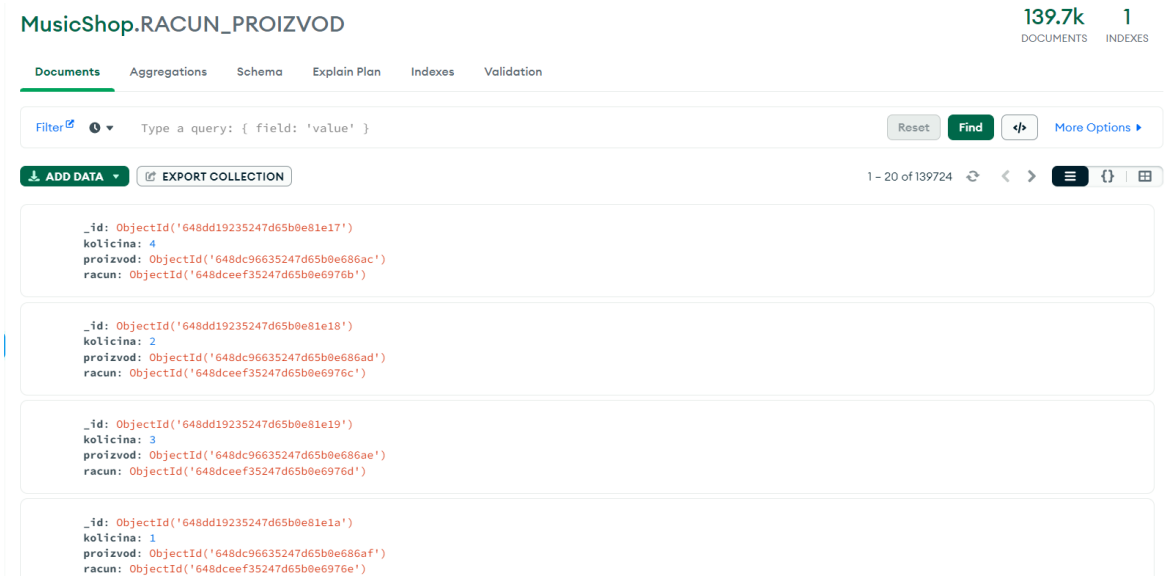
Slika 41 Kolekcija AUTOR



Slika 42 Kolekcija KUPAC



Slika 43 Kolekcija RACUN



Slika 44 Kolekcija RACUN_PROIZVOD

3.2.1. Upiti

Napravljeni su isti upiti kao i u relacijskoj bazi podataka kako bi se moglo usporediti vrijeme izvršavanja. Vrijeme izvršavanja upita mjereno je koristeći „explain()“ metodu nad agregatnim operacijama.

Napravljen je upit koji ispisuje ime i prezime zaposlenika, trgovinu u kojoj radi i broj računa koje je izdao taj zaposlenik.

```
db.RACUN.aggregate([
  {
    $lookup: {
      from: "ZAPOSLENIK",
      localField: "zaposlenik",
      foreignField: "_id",
      as: "zaposlenik"
    }
  },
  {
    $unwind: "$zaposlenik"
  },
  {
    $lookup: {
      from: "TRGOVINA",
      localField: "zaposlenik.trgovina",
      foreignField: "_id",
      as: "trgovina"
    }
  },
  {
    $unwind: "$trgovina"
  },
  {
    $group: {
      _id: {
        ime_z: "$zaposlenik.ime_z",
        prezime_z: "$zaposlenik.prezime_z",
        naziv_trgovina: "$trgovina.naziv_trgovina"
      },
      broj: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      ime_z: "$_id.ime_z",
      prezime_z: "$_id.prezime_z",
      broj: 1,
      naziv_trgovina: "$_id.naziv_trgovina"
    }
  }
])
```

```
<  {
      broj: 32,
      ime_z: 'Fred',
      prezime_z: 'Kelly',
      naziv_trgovina: 'UPC'
    }
    {
      broj: 31,
      ime_z: 'Eve',
      prezime_z: 'Ianson',
      naziv_trgovina: 'UPC'
    }
    {
      broj: 31,
      ime_z: 'Noah',
      prezime_z: 'Selby',
      naziv_trgovina: 'DynCorp'
    }
```

Slika 45 Ispis upita

```
    executionStats: {
      executionSuccess: true,
      nReturned: 100000,
      executionTimeMillis: 21838,
      totalKeysExamined: 0,
      totalDocsExamined: 100000,
```

Slika 46 Vrijeme izvršavanja

Sljedeći upit računa dosad prodanu količinu svakog od proizvoda te ispisuje naziv istog.

```
db.RACUN_PROIZVOD.aggregate([
  {
    $lookup: {
      from: "PROIZVOD",
      localField: "proizvod",
      foreignField: "_id",
      as: "proizvod"
    }
  },
  {
    $unwind: "$proizvod"
  },
  {
    $group: {
      _id: "$proizvod.naziv",
      broj: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
```

```
      naziv: "$_id",
      broj: 1
    }
  }
])
```

```
<  {
      broj: 380,
      naziv: 'Beggars Banquet'
    }
    {
      broj: 379,
      naziv: 'Justified'
    }
    {
      broj: 379,
      naziv: 'Soul Men'
    }
    {
      broj: 380,
      naziv: 'Combat Rock'
    }
```

Slika 47 Ispis upita

```
  executionStats: {
    executionSuccess: true,
    nReturned: 139724,
    executionTimeMillis: 12956,
    totalKeysExamined: 0,
    totalDocsExamined: 139724,
```

Slika 48 Vrijeme izvršavanja

Ponovljen je i upit koji računa ukupan broj narudžbi za svaku godinu poslovanja tvrtke. Ispisuje se broj narudžbi i godina poslovanja.

```
db.RACUN.aggregate([
  {
    $group: {
      _id: { $year: "$datum" },
      broj_narudzbi: { $sum: 1 }
    }
  },
  {
    $project: {
      godina: "$_id",
      broj_narudzbi: 1,
      _id: 0
```

```
    }
  },
  {
    $sort: {
      godina: -1
    }
  }
])
```

```
< {
    broj_narudzbi: 3303,
    godina: 2023
  }
  {
    broj_narudzbi: 7436,
    godina: 2022
  }
  {
    broj_narudzbi: 7388,
    godina: 2021
  }
  {
    broj_narudzbi: 7419,
    godina: 2020
  }
  {
    broj_narudzbi: 7460,
    godina: 2019
  }
  {
    broj_narudzbi: 7296,
    godina: 2018
  }
```

```
    broj_narudzbi: 7446,
    godina: 2017
  }
  {
    broj_narudzbi: 7404,
    godina: 2016
  }
  {
    broj_narudzbi: 7492,
    godina: 2015
  }
  {
    broj_narudzbi: 7422,
    godina: 2014
  }
  {
    broj_narudzbi: 7412,
    godina: 2013
  }
```

```
  {
    broj_narudzbi: 7361,
    godina: 2012
  }
  {
    broj_narudzbi: 7538,
    godina: 2011
  }
  {
    broj_narudzbi: 7609,
    godina: 2010
  }
  {
    broj_narudzbi: 14,
    godina: 2009
  }
```

Slika 49 Ispis upita

```
executionStats: {
  executionSuccess: true,
  nReturned: 100000,
  executionTimeMillis: 212,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
```

Slika 50 Vrijeme izvršavanja

Upit koji je prikazan u nastavku ispisuje naziv medija i broj koliko se proizvoda nalazi na svakoj vrsti medija.

```
db.PROIZVOD.aggregate([
  {
    $group: {
      _id: "$medij",
      broj: { $sum: 1 }
    }
  },
  {
    $project: {
```

```
      medij: "$_id",
      broj: 1,
      _id: 0
    }
  }
])
```

```
<  {
     broj: 92,
     medij: 'CD'
   }
   {
     broj: 92,
     medij: 'DVD'
   }
   {
     broj: 92,
     medij: 'Record'
   }
   {
     broj: 92,
     medij: 'Track'
   }
```

Slika 51 Ispis upita

```
  executionStats: {
    executionSuccess: true,
    nReturned: 368,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 368,
```

Slika 52 Vrijeme izvršavanja

Napravljen je i upit koji prikazuje koliko albuma je pod kojim žanrom.

```
db.PROIZVOD.aggregate([
  {
    $group: {
      _id: "$zanr",
      broj: { $sum: 1 }
    }
  },
  {
    $project: {
```

```
      zanr: "$_id",
      broj: 1,
      _id: 0
    }
  }
])
```

```
< {
    broj: 24,
    zanr: 'Techno'
  }
  {
    broj: 24,
    zanr: 'House'
  }
  {
    broj: 24,
    zanr: 'Blues'
  }
  {
    broj: 24,
    zanr: 'Jazz'
  }
  {
    broj: 24,
    zanr: 'Latin'
  }
  {
    broj: 25,
    zanr: 'Reggae'
  }
```

```
{
  broj: 25,
  zanr: 'Opera'
}
{
  broj: 25,
  zanr: 'Baroque'
}
{
  broj: 24,
  zanr: 'Hard Rock'
}
{
  broj: 25,
  zanr: 'Drum & Bass'
}
{
  broj: 25,
  zanr: 'Country'
}
```

```
{
  broj: 25,
  zanr: 'Soul'
}
{
  broj: 25,
  zanr: 'Pop'
}
{
  broj: 25,
  zanr: 'Dubstep'
}
{
  broj: 24,
  zanr: 'Hip-Hop'
}
```

Slika 53 Ispis upita

```
executionStats: {
  executionSuccess: true,
  nReturned: 368,
  executionTimeMillis: 0,
  totalKeysExamined: 0,
  totalDocsExamined: 368,
```

Slika 54 Vrijeme izvršavanja

Sljedeći upit prikazuje imena i prezimena kupaca te broj njihovih kupnji.

```
db.RACUN.aggregate([
  {
    $lookup: {
      from: "KUPAC",
      localField: "kupac",
      foreignField: "_id",
      as: "kupac"
    }
  },
  {
```

```
    $unwind: "$kupac"
  },
  {
    $group: {
      _id: {
        ime_kupca: "$kupac.ime_k",
        prezime_kupca: "$kupac.prezime_k"
      },
      broj: { $sum: 1 }
    }
  },
  {
    $project: {
      ime_kupca: "$_id.ime_kupca",
      prezime_kupca: "$_id.prezime_kupca",
      broj: 1,
      _id: 0
    }
  },
  {
    $sort: {
      broj: -1
    }
  }
])
```

```
< {
    broj: 8,
    ime_kupca: 'Bart',
    prezime_kupca: 'Stone'
  }
  {
    broj: 8,
    ime_kupca: 'Carter',
    prezime_kupca: 'Stone'
  }
  {
    broj: 8,
    ime_kupca: 'Cedrick',
    prezime_kupca: 'Stone'
  }
  {
    broj: 8,
    ime_kupca: 'Hayden',
    prezime_kupca: 'Stone'
  }
  {
    broj: 8,
    ime_kupca: 'Alexander',
    prezime_kupca: 'Stone'
  }
```

```
  {
    broj: 7,
    ime_kupca: 'Roger',
    prezime_kupca: 'Stone'
  }
  {
    broj: 7,
    ime_kupca: 'Ryan',
    prezime_kupca: 'Stone'
  }
  {
    broj: 4,
    ime_kupca: 'Summer',
    prezime_kupca: 'Allen'
  }
  {
    broj: 4,
    ime_kupca: 'David',
    prezime_kupca: 'Connell'
  }
  {
    broj: 4,
    ime_kupca: 'Leroy',
    prezime_kupca: 'Simpson'
  }
```

Slika 55 Ispis upita

```
executionStats: {
  executionSuccess: true,
  nReturned: 100000,
  executionTimeMillis: 9570,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
```

Slika 56 Vrijeme izvršavanja

Kako bi saznali koje pjesme se nalaze na kojem proizvodu, njihovo trajanje u sekundama i autor, napravljen je sljedeći upit.

```
db.PROIZVOD.aggregate([
  {
    $lookup: {
      from: "PJESMA",
      localField: "_id",
      foreignField: "proizvod",
      as: "pjesme"
    }
  },
  {
    $unwind: "$pjesme"
  },
  {
    $lookup: {
      from: "AUTOR",
      localField: "autor",
      foreignField: "_id",
      as: "autor"
    }
  },
  {
    $unwind: "$autor"
  },
  {
    $project: {
      naziv: "$naziv",
      naziv_pjesma: "$pjesme.naziv_pjesma",
      trajanje: "$pjesme.trajanje",
      autor: "$autor.autor",
      _id: 0
    }
  }
])
```

```
< {
    naziv: 'Highway 61 Revisited',
    naziv_pjesma: 'Testify',
    trajanje: 159,
    autor: 'Bob Dylan'
  }
  {
    naziv: 'Highway 61 Revisited',
    naziv_pjesma: 'Still Alive and Well',
    trajanje: 314,
    autor: 'Bob Dylan'
  }
  {
    naziv: 'Highway 61 Revisited',
    naziv_pjesma: 'Lost',
    trajanje: 187,
    autor: 'Bob Dylan'
  }
  {
    naziv: 'Highway 61 Revisited',
    naziv_pjesma: 'Choke',
    trajanje: 315,
    autor: 'Bob Dylan'
  }
```

```
  {
    naziv: 'Out of Our Heads',
    naziv_pjesma: 'Guerrilla Radio',
    trajanje: 347,
    autor: 'The Rolling Stones'
  }
  {
    naziv: 'Out of Our Heads',
    naziv_pjesma: 'Too Much Seconal',
    trajanje: 252,
    autor: 'The Rolling Stones'
  }
  {
    naziv: 'Out of Our Heads',
    naziv_pjesma: 'Ecstasy',
    trajanje: 140,
    autor: 'The Rolling Stones'
  }
  {
    naziv: 'Out of Our Heads',
    naziv_pjesma: 'Blackout',
    trajanje: 269,
    autor: 'The Rolling Stones'
  }
```

Slika 57 Ispis upita

```
executionStats: {
  executionSuccess: true,
  nReturned: 368,
  executionTimeMillis: 2019,
  totalKeysExamined: 0,
  totalDocsExamined: 368,
```

Slika 58 Vrijeme izvršavanja

Iz sljedećeg upita možemo vidjeti koliko svaki od autora ima objavljenih albuma.

```
db.PROIZVOD.aggregate([
  {
    $lookup: {
      from: "AUTOR",
      localField: "autor",
      foreignField: "_id",
      as: "autor"
    }
  },
  {
    $unwind: "$autor"
  },
  {
    $group: {
      _id: "$autor.autor",
      broj: { $sum: 1 }
    }
  },
  {
```

```
    $sort: {
      broj: -1
    }
  },
  {
    $project: {
      autor: "$_id",
      broj: 1,
      _id: 0
    }
  }
])
```

```
<  {
     broj: 2,
     autor: 'Sly & The Family Stone'
   }
   {
     broj: 2,
     autor: 'Ike & Tina Turner'
   }
   {
     broj: 2,
     autor: 'Gnarls Barkley'
   }
   {
     broj: 2,
     autor: 'Lynyrd Skynyrd'
   }
   {
     broj: 2,
     autor: 'Cream'
   }
   {
     broj: 2,
     autor: 'Sex Pistols'
   }
```

Slika 59 Ispis upita

```
executionStats: {
  executionSuccess: true,
  nReturned: 368,
  executionTimeMillis: 25,
  totalKeysExamined: 0,
  totalDocsExamined: 368,
```

Slika 60 Vrijeme izvršavanja

Na samom kraju, napravljen je upit koji ispisuje ukupan iznos računa i tko je bio kupac.

```
db.RACUN_PROIZVOD.aggregate([
  {
    $lookup: {
      from: "RACUN",
      localField: "racun",
      foreignField: "_id",
      as: "racun"
    }
  },
  {
    $unwind: "$racun"
  },
  {
    $lookup: {
      from: "PROIZVOD",
      localField: "proizvod",
      foreignField: "_id",
      as: "proizvod"
    }
  },
  {
    $unwind: "$proizvod"
  },
  {
    $lookup: {
      from: "KUPAC",
      localField: "racun.kupac",
      foreignField: "_id",
      as: "kupac"
    }
  },
  {
    $unwind: "$kupac"
  },
  {
    $group: {
      _id: {
        racun: "$racun._id",
        ime_k: "$kupac.ime_k",
        prezime_k: "$kupac.prezime_k"
      },
      ukupan_iznos: { $sum: { $multiply: ["$kolicina", "$proizvod.cijena"] } }
    }
  },
  {
    $sort: {
      ukupan_iznos: -1
    }
  },
  {
    $project: {
      racun: "$_id.racun",
      ime_k: "$_id.ime_k",
      prezime_k: "$_id.prezime_k",
      ukupan_iznos: 1,
      _id: 0
    }
  }
])
```

```
<  {
      ukupan_iznos: 9510,
      racun: ObjectId("648ec8ebe69f41f8c6712011"),
      ime_k: 'Camden',
      prezime_k: 'Johnson'
    }
    {
      ukupan_iznos: 9510,
      racun: ObjectId("648ec8ece69f41f8c6717c11"),
      ime_k: 'Elijah',
      prezime_k: 'Norman'
    }
    {
      ukupan_iznos: 9510,
      racun: ObjectId("648ec8ebe69f41f8c6713881"),
      ime_k: 'Aiden',
      prezime_k: 'Samuel'
    }
    {
      ukupan_iznos: 9480,
      racun: ObjectId("648ec8ece69f41f8c6718132"),
      ime_k: 'Summer',
      prezime_k: 'Miller'
    }
```

Slika 61 Ispis upita

```
executionStats: {
  executionSuccess: true,
  nReturned: 139724,
  executionTimeMillis: 45850,
  totalKeysExamined: 0,
  totalDocsExamined: 139724,
```

Slika 62 Vrijeme izvršavanja

4. Zaključak

Iz rezultata možemo vidjeti da brže vrijeme izvršavanja upita koji zahtijevaju obradu veće količine podataka ima SQL Server u odnosu na MongoDB. Upiti koji obrađuju manju količinu podataka se gotovo jednako brzo izvršavaju u obje baze podataka. Postoji nekoliko razloga zašto SQL Server može imati bolje vrijeme izvršavanja od MongoDB-a kada se radi o velikoj količini podataka. SQL Server ima zreo i robustan alat za optimizaciju upita koji može učinkovito optimizirati i izvršavati složene upite. Iskorištava različite tehnike optimizacije, kao što su složenost upita, indeksiranje i statistika kako bi se generirao plan izvršavanja koji minimizira upotrebu resursa i maksimizira izvedbu upita. Ovaj proces optimizacije pomaže SQL Server-u da učinkovitije rukuje velikim skupovima podataka. SQL Server podržava različite vrste indeksa, kao što su grupirani i neklasterirani indeksi, koji poboljšavaju izvedbu dohvaćanja podataka. Ispravno dizajnirani i održavani indeksi mogu značajno ubrzati izvođenje upita, posebno kada su u pitanju velike količine podataka. MongoDB također podržava indeksiranje, ali su mogućnosti indeksiranja SQL Server-a usavršavane tijekom mnogo godina, budući da je SQL Server dosta stariji. SQL Server temelji se na modelu relacijske baze podataka, koji pruža strukturirani pristup organizaciji podataka i vezama. Budući da su testirani upiti uključivali spajanja i složene operacije koje zahtijevaju podatke iz više tablica, relacijski model SQL Server-a pruža prednost. Sposobnost SQL Server-a da izvodi učinkovite operacije spajanja na povezanim tablicama može pridonijeti bržem izvršavanju upita. MongoDB je, s druge strane, NoSQL baza podataka koja koristi model dokumenta, koji možda nije toliko optimiziran za složene relacijske upite.

MongoDB i SQL Server imaju svoje prednosti i mane te su odgovarajući odabir u različitim scenarijima. Razumijevanje korisničkih zahtjeva i odabir odgovarajućeg sustava baze podataka ključan je korak u postizanju najboljih performansi sustava.