NoSQL

FERIT

MongoDB
Upiti i agregacije

Predavanje 7

Sadržaj

- Podsjetnik: CRUD operacije
- **Čitanje (Read):**
  - Filtriranje dokumenata (Query filter)
  - Operatori za upite ($eq, $ne, $gt, $lt, $in, $nin)
  - Logički operatori ($and, $or, $not, $nor)
  - Projekcija (Projection): Vraćanje samo potrebnih polja
- **Ažuriranje (Update):**
  - Operatori za ažuriranje ($set, $unset, $inc, $rename)
  - Upsert opcija
- **Brisanje (Delete):**
- Sortiranje, preskakanje i limitiranje rezultata (sort, skip, limit)
- **Uvod u Aggregation Framework:**
  - Koncept pipeline-a (cjevovoda)
  - Osnovne faze (stages): $match, $group, $sort, $project

Podsjetnik: CRUD Operacije

- **Definicija:**
  - Četiri osnovne funkcije perzistentne pohrane podataka.
- **MongoDB CRUD naredbe (u mongosh):**
  - **C**reate: insertOne(), insertMany()
  - **R**ead: find(), findOne()
  - **U**pdate: updateOne(), updateMany(), replaceOne()
  - **D**elete: deleteOne(), deleteMany()
- Sve naredbe se izvršavaju na razini kolekcije: db.<kolekcija>.<naredba>().

Čitanje podataka: `find()` i `findOne()`

- **Sintaksa:**
  - db.collection.find(<filter>, <projection>)
  - db.collection.findOne(<filter>, <projection>)
- **Argumenti:**
  - <filter> (dokument za filtriranje): Specificira kriterije za odabir dokumenata. Prazan dokument {} vraća sve dokumente.
  - <projection> (dokument za projekciju): Specificira koja polja treba vratiti. Opcionalan.
- **Razlika:**
  - find(): Vraća **kursor (cursor)** – pointer na rezultate. Shell iterira kroz kursor i prikazuje dokumente.
  - findOne(): Vraća **jedan dokument** koji zadovoljava kriterij (ili null ako ga nema).

Filtriranje

- **Osnovno podudaranje (*Equality Match*):**
  - { <polje>: <vrijednost> }
  - Primjer: Pronađi sve proizvode s brendom "TechCorp".

```
db.products.find({ brand: "TechCorp" })
```

- **Podudaranje u ugniježđenim dokumentima (*Dot Notation*):**
  - { "<ugniježđeno.polje>": <vrijednost> }
  - Primjer: Pronađi sve proizvode s CPU-om "i7".

```
db.products.find({ "specifications.cpu": "i7" })
```

- **Podudaranje elemenata u nizu:**
  - Primjer: Pronađi sve proizvode s tagom "laptops".

```
db.products.find({ tags: "laptops" })
```

Operatori usporedbe za upite (*Comparison Operators*)

- **Sintaksa:** { <polje>: { <operator>: <vrijednost> } }
- **Najčešći operatori:**
  - $eq: jednako (equals)
  - $ne: nije jednako (not equal)
  - $gt: veće od (greater than)
  - $gte: veće ili jednako (greater than or equal)
  - $lt: manje od (less than)
  - $lte: manje ili jednako (less than or equal)
  - $in: vrijednost je u nizu (in)
  - $nin: vrijednost nije u nizu (not in)

Primjer: Pronađi proizvode s cijenom većom od 500 i manjom ili jednakom 1000.

```
db.products.find({
    price: { $gt: 500, $lte: 1000 }
})
```

Logički Operatori
(Logical Operators)

- **Povezivanje više uvjeta:**
  - $and: Svi uvjeti moraju biti zadovoljeni
  - $or: Barem jedan uvjet mora biti zadovoljen.
  - $nor: Nijedan od uvjeta ne smije biti zadovoljen.
  - $not: Negira uvjet operatora.

Primjer: Pronađi proizvode čija je cijena manja od 100 ILI koji imaju tag "sale".

```
db.products.find({
    $or: [
        { price: { $lt: 100 } },
        { tags: "sale" }
    ]
})
```

Implicitni AND ako navedete više polja u filteru:

```
db.products.find({ brand: "TechCorp", price: { $lt: 1000 } })
```

Ovo je isto kao:

```
db.products.find({ $and: [ { brand: "TechCorp" }, { price: { $lt: 1000 } } ] })
```

Projekcija (*Projection*)

- **Svrha:** Ograničiti koja polja će biti vraćena u rezultatu.
  - Smanjuje količinu podataka koji putuju preko mreže.
  - Pojednostavljuje rezultat.
- **Sintaksa:**
  - Drugi argument find() naredbe.
  - { <polje>: 1 } ili { <polje>: true }: Uključi polje.
  - { <polje>: 0 } ili { <polje>: false }: Isključi polje.
- **Pravilo:** Ne možete miješati uključivanje (1) i isključivanje (0) u istoj projekciji (osim za isključivanje _id).

Primjer: Dohvati samo ime i cijenu svih proizvoda, bez _id.

```
db.products.find(
    //filter
    { brand: "TechCorp" },
    //projekcija
    { name: 1, price: 1, _id: 0 }
)
```

Ažuriranje Dokumenata: `updateOne()` i `updateMany()`

- **Sintaksa:**
  - db.collection.updateOne(<filter>, <update>, <options>)
  - db.collection.updateMany(<filter>, <update>, <options>)
- **Argumenti:**
  - <filter>: Određuje koji dokument(e) će biti ažurirani.
  - <update>: Specificira modifikacije koje treba primijeniti. **Mora koristiti operatore za ažuriranje!**
  - <options>: Dodatne opcije, npr. upsert.
- **Razlika:**
  - updateOne(): Ažurira **prvi dokument** koji zadovoljava filter.
  - updateMany(): Ažurira **sve dokumente** koji zadovoljavaju filter.

Operatori za Ažuriranje (*Update Operators*)

- **Najčešći operatori:**
  - $set: Postavlja ili mijenja vrijednost polja. Ako polje ne postoji, kreira ga.
  - $unset: Uklanja (briše) polje iz dokumenta.
  - $inc: Inkrementira (ili dekrementira s negativnom vrijednošću) numeričko polje.
  - $rename: Preimenuje polje.
  - $push: Dodaje element u niz.
  - $pull: Uklanja element(e) iz niza koji zadovoljavaju uvjet.
  - $addToSet: Dodaje element u niz samo ako već ne postoji.

Primjer: Smanji cijenu proizvoda s ID-om 1 za 50 i dodaj tag "on_sale"

```
db.products.updateOne(
    { _id: 1 },
    {
        $inc: { price: -50 },
        $push: { tags: "on_sale" }
    }
)
```

Upsert: Update or Insert

- **Koncept:**
  - Ako dokument koji odgovara filteru postoji, ažuriraj ga.
  - Ako ne postoji, umetni novi dokument.
- **Kako:**
  - Postavljanjem opcije upsert: true.

Primjer: Zabilježi posjet korisnika "pperic". Ako ne postoji, kreiraj ga s 1 posjetom. Ako postoji, inkrementiraj broj posjeta.

```
db.user_stats.updateOne(
    { username: "pperic" },
    { $inc: { visits: 1 } },
    { upsert: true }
)
```

Ako dokument { username: "pperic" } ne postoji, MongoDB će kreirati novi dokument { username: "pperic", visits: 1 }.

Brisanje Dokumenata: `deleteOne()` i `deleteMany()`

- **Sintaksa:**
  - db.collection.deleteOne(<filter>)
  - db.collection.deleteMany(<filter>)
- **Argumenti:**
  - <filter>: Određuje koji dokument(i) će biti obrisani.
- **Ponašanje:**
  - deleteOne(): Briše **prvi dokument** koji zadovoljava filter.
  - deleteMany(): Briše **sve dokumente** koji zadovoljavaju filter.
- **Brisanje svih dokumenata u kolekciji:**
  - db.products.deleteMany({})

Sortiranje, preskakanje i limitiranje

- **Svrha:** Kontrola nad skupom rezultata (paginacija, top N rezultata).
- Ove metode se "lančano" pozivaju na kursoru koji vraća find().
- **Metode:**
  - .sort({ <polje>: 1 ili -1 }): Sortira rezultate. 1 za uzlazno (*ascending*), -1 za silazno (*descending*).
  - .limit(<broj>): Ograničava broj vraćenih dokumenata.
  - .skip(<broj>): Preskače određeni broj dokumenata.

Primjer: Dohvati 5 najskupljih proizvoda, sortiranih po cijeni silazno.

```
db.products.find().sort({ price: -1 }).limit(5)
```

Primjer paginacije (stranica 3, 10 rezultata po stranici):

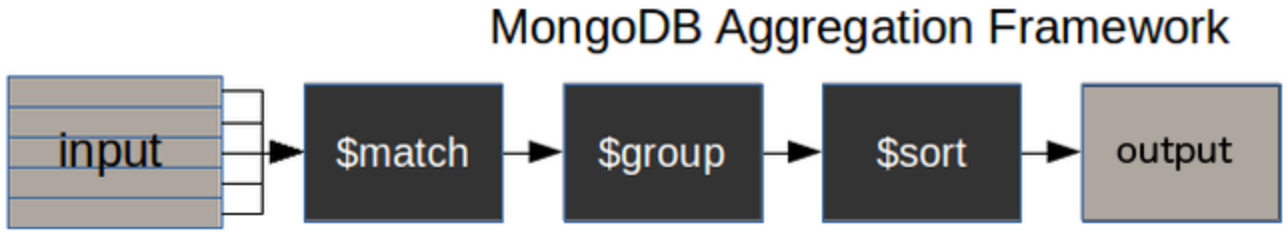
```
db.products.find().sort({ name: 1 }).skip(20).limit(10)
```

Problem: Kako analizirati podatke?

- **Pitanje:** Kako dobiti odgovore na pitanja kao što su:
  - "Koliko proizvoda ima svaki brend?"
  - "Koja je prosječna cijena proizvoda po brendu?"
  - "Koji su najpopularniji tagovi?"
- find() dohvaća cijele dokumente. Nije dovoljan za agregacije.
- U SQL-u bismo koristili COUNT(), AVG(), GROUP BY.
- **Rješenje u MongoDB-u: Aggregation Framework**.

Uvod u Aggregation Framework

- **Koncept: *Pipeline* (cjevovod)**
  - Dokumenti iz kolekcije prolaze kroz niz **faza (stages)**.
  - Svaka faza transformira dokumente i prosljeđuje rezultat idućoj fazi.
- **Sintaksa:**
  - db.collection.aggregate([<faza1>, <faza2>, ...])
- **Prednosti:**
  - Složena obrada podataka se izvršava na serveru.
  - Vrlo moćno i fleksibilno.



Ključne faze agregacije

- **$match:** Filtrira dokumente. Slično kao find() filter. Treba biti što ranije u pipeline-u radi performansi.
- **$group:** Grupira dokumente po nekom ključu i omogućava izračunavanje agregatnih vrijednosti (npr. sum, avg, count).
- **$sort:** Sortira rezultirajuće dokumente.
- **$project:** Preoblikuje dokumente: mijenja imena polja, dodaje izračunata polja, uklanja postojeća. Slično projekciji u find().
- **$limit:** Ograničava broj dokumenata.
- **$skip:** Preskače određeni broj dokumenata.
- **$unwind:** "Odmotava" dokumente koji imaju niz. Za svaki element u nizu, kreira novi dokument.

Primjer Agregacije (1/2)

- **Problem:** Izračunati broj proizvoda i prosječnu cijenu za svaki brend.
- **Pipeline:**
- **Faza 1: $group**
  - Grupiraj po polju brand.
  - Za svaku grupu (brend), izračunaj:
    - totalProducts: broj dokumenata u grupi.
    - avgPrice: prosjek price polja u grupi.

```
db.products.aggregate([
    {
        $group: {
            // Grupiraj po vrijednosti polja 'brand'
            _id: "$brand",
            // Za svaki dokument, dodaj 1 na sumu
            totalProducts: { $sum: 1 },
            // Izračunaj prosjek polja 'price'
            avgPrice: { $avg: "$price" }
        }
    }
])
```

Znak $ ispred imena polja ("$brand", "$price") unutar agregacije označava da se koristi vrijednost tog polja iz dokumenta.

Primjer Agregacije (2/2)

- **Problem:** Isto kao prije, ali samo za proizvode skuplje od 50, i rezultate sortirati po broju proizvoda silazno.
- **Pipeline:**
  - **Faza 1: $match** - Filtriraj samo željene proizvode.
  - **Faza 2: $group** - Grupiraj kao i prije.
  - **Faza 3: $sort** - Sortiraj rezultate.

```
db.products.aggregate([
    { $match: { price: { $gt: 50 } } },
    {
        $group: {
            _id: "$brand",
            totalProducts: { $sum: 1 },
            avgPrice: { $avg: "$price" }
        }
    },
    { $sort: { totalProducts: -1 } }
])
```

Primjer faze `$project`

- **Svrha:** Preoblikovanje izlaznih dokumenata. Više od obične projekcije.
- **Mogućnosti:**
  - Uključivanje postojećih polja (<polje>: 1).
  - Isključivanje postojećih polja (<polje>: 0).
  - Preimenovanje polja (<novo_ime>: "$<staro_ime>").
  - Kreiranje **novih, izračunatih polja** pomoću izraza i operatora.
- **Primjer:** Iz našeg ranijeg primjera agregacije, želimo preimenovati _id u brand, formatirati prosječnu cijenu na dvije decimale i dodati polje report_date.

Primjer faze `$project`

```
db.products.aggregate([
    // ... prethodne faze $match i $group ...
    // Rezultat iz $group: { _id: "TechCorp", totalProducts: 5, avgPrice: 855.498 }
    {
        $project: {
            _id: 0, // Isključi staro _id polje
            brand: "$_id", // Preimenuj _id u brand
            count: "$totalProducts", // Preimenuj totalProducts u count
            averagePrice: { $round: ["$avgPrice", 2] }, // Kreiraj novo polje i zaokruži
            reportDate: new Date() // Dodaj novo polje s trenutnim datumom
        }
    }
])
```

Izlazni dokument:

```
{
    "brand": "TechCorp",
    "count": 5,
    "averagePrice": 855.50,
    "reportDate": "2023-10-27T10:30:00Z"
}
```

Primjer faze `$unwind`

- **Svrha:** "Odmotava" niz unutar dokumenta. Za svaki element niza, kreira se nova kopija dokumenta.
- **Kada se koristi?** Kada želite izvršiti operacije (npr. grupiranje) na pojedinačnim elementima niza.
- **Primjer upotrebe:** Izračunati koliko puta se svaki tag pojavljuje u svim proizvodima.
  - **$unwind: "$tags"**: Kreira poseban dokument za svaki tag svakog proizvoda.
  - **$group: { _id: "$tags", count: { $sum: 1 } }**: Grupa po tagovima i prebroji ih.
  - **$sort: { count: -1 }**: Sortira da bi najpopularniji tagovi bili na vrhu.

Ulazni dokument:

```
{
    "_id": 1,
    "name": "Laptop Pro X1",
    "tags": ["electronics", "laptops", "powerful"]
}
```

Agregacijska faza:

```
{ $unwind: "$tags" }
```

Izlaz (3 dokumenta):

```
// Dokument 1
{ "_id": 1, "name": "Laptop Pro X1", "tags": "electronics" }
// Dokument 2
{ "_id": 1, "name": "Laptop Pro X1", "tags": "laptops" }
// Dokument 3
{ "_id": 1, "name": "Laptop Pro X1", "tags": "powerful" }
```

Sažetak današnjeg predavanja

- Savladali smo osnovne CRUD operacije u MongoDB-u (find, updateOne, deleteMany...).
- Naučili smo koristiti operatore za upite ($gt, $in, $or...) za precizno filtriranje podataka.
- Projekcija (projection) nam omogućava optimizaciju upita vraćanjem samo potrebnih polja.
- sort(), limit() i skip() se koriste za kontrolu nad skupom rezultata.
- Aggregation Framework je moćan alat za analizu i transformaciju podataka pomoću pipeline-a i faza ($match, $group...).