Agregacije I šeme

Predmetni asistent:

Jovana Stojanović, dipl. inž. el. i rač.

jovana.stojanovic@akademijanis.edu.rs

Šta je Aggregation?

- Mehanizam za obradu podataka kroz faze (pipeline).
- Umesto prostog pronalaženja dokumenata, podaci prolaze kroz "traku" gde se transformišu, grupišu i filtriraju.

- Analogija: Kao fabrika gde svaka mašina (faza) radi jedan specifičan posao i šalje proizvod sledećoj mašini.

Koncept Pipeline-a

- Struktura:
  - **db.collection.aggregate([{ faza1 }, { faza2 }, ...])**

- Ključna pravila:
  - Redosled faza je presudan.
  - Izlaz jedne faze je ulaz za sledeću.
  - Originalna kolekcija se nikada ne menja.

Osnovne faze (I deo)

- **$match**: Filtriranje dokumenata (identično kao find). Uvek ga stavljamo na početak da smanjimo količinu podataka za obradu.
- **$project**: Projekcija, ali u pipeline-u.

```
db.ucenici.aggregate([
  {
    $project: {
      _id: 0,       // Isključujemo _id (on je jedini po defaultu 1)
      ime: 1,       // Uključujemo ime
      smer: 1       // Uključujemo smer
    }
  }
])
```

- **$sort**
- **$limit**

Grupisanje podataka (**$group**)

- Svrha: Grupisanje dokumenata po određenom ključu (npr. po smeru ili gradu).
- Sintaksa:

```
{ $group: { _id: "$polje_po_kome_grupišemo", total: { $sum: 1 } } }
```

- Akumulatori: **$sum**, **$avg**, **$min**, **$max**.

- Primer: Izračunavanje prosečne ocene po studentu.

Razbijanje nizova (**$unwind**)

```
// Početni dokument u kolekciji "studenti"
{
  "_id": 1,
  "ime": "Marko",
  "hobiji": ["fudbal", "gitara", "trčanje"]
}
```

- Jedan red postaje više redova.

- Ako želimo da "razbijemo" ovaj niz tako da svaki hobi dobije svoj zaseban privremeni dokument (kako bismo ih npr. lakše brojali)

```
db.studenti.aggregate([
  { $unwind: "$hobiji" }
])
```

- Rezultat:

```
{ "_id": 1, "ime": "Marko", "hobiji": "fudbal" }
{ "_id": 1, "ime": "Marko", "hobiji": "gitara" }
{ "_id": 1, "ime": "Marko", "hobiji": "trčanje" }
```

Veliki primer (Pipeline u praksi)

- Kolekcija: studenti (polja: ime, godine, smer, ocene[niz])
- Zadatak: Pronađi prosečnu ocenu studenata na IT smeru koji imaju više od 20 godina, sortiraj ih od najboljeg.

```
db.studenti.aggregate([
  { $match: { smer: "IT", godine: { $gt: 20 } } }, // 1. Filtriraj
  { $unwind: "$ocene" },                           // 2. Razbij niz ocena
  { $group: {                                      // 3. Grupiši po imenu
      _id: "$ime",
      prosek: { $avg: "$ocene" }
  }},
  { $sort: { prosek: -1 } }                        // 4. Sortiraj
])
```

Spajanje kolekcija (**$lookup**)

- Poređenje sa SQL JOIN:
  - U SQL-u radimo JOIN.
  - U MongoDB-u koristimo **$lookup** (Left Outer Join).

- Parametri:
  - **from**: Ciljna kolekcija.
  - **localField**: Polje iz trenutne kolekcije.
  - **foreignField**: Polje iz ciljne kolekcije.
  - **as**: Ime novog niza gde će se smestiti spojeni podaci.

Intuitivni primer (**$lookup**)

- Scenario: Imamo kolekcije studenti i prijave_ispita (sadrži id studenta).

```
db.studenti.aggregate([
  {
    $lookup: {
      from: "prijave_ispita",
      localField: "_id",
      foreignField: "student_id",
      as: "sve_prijave"
    }
  }
])
```

  - **Objašnjenje:** Svaki student u rezultatu dobija novi niz **`sve_prijave`** koji sadrži sve njegove prijave iz druge kolekcije.

Kompletan **$lookup** model (Studenti - Predmeti - Prijave)

- Model:
  - studenti (_id, ime, prezime)
  - predmeti (_id, naziv_predmeta, espb)
  - prijave (student_id, predmet_id, ocena, datum)

- Logika: Prijave su "most" (kao tabela veze u SQL-u).

- Zadatak: Kako pomoću dva $lookup-a izvući ime studenta i naziv predmeta koji je položio?

Schema Validation

- MongoDB koristi **$jsonSchema** operator za definisanje pravila. To je dokument koji opisuje kako drugi dokumenti treba da izgledaju.
- Glavni gradivni blokovi:
  - **bsonType**: Određuje tip (string, int, double, object, array, bool).
  - **required**: Niz naziva polja koja su obavezna.
  - **properties**: Detaljna pravila za svako pojedinačno polje.

Osnovni tipovi podataka (**bsonType**)

- Ovim određujete šta polje sme da sadrži:
  - **"string"** – Tekst (ime, grad, opis).
  - **"int"** – Ceo broj (godine, ocena, broj poena).
  - **"double"** – Decimalni broj (prosek, plata, cena).
  - **"bool"** – Logička vrednost (**true** ili **false**).
  - **"array"** – Niz elemenata (hobiji, ispiti, tagovi).
  - **"object"** – Ugnježdeni dokument (adresa, detalji).

Obavezna polja (**required**)

- Niz koji sadrži nazive polja koja moraju postojati u svakom dokumentu:
  - **required: ["ime", "prezime", "email"]**

Obavezna polja (**required**)

- Niz koji sadrži nazive polja koja moraju postojati u svakom dokumentu:
  - **required: ["ime", "prezime", "email"]**

Dodatna ograničenja - Constraints (**properties**)

- Za svaki tip podatka možete dodati specifična pravila:
  - Za brojeve: **minimum** i **maximum** (npr. ocena od 5 do 10).
  - Za stringove: **enum** (lista dozvoljenih vrednosti, npr. ["IT", "Dizajn"]).
  - Za nizove: **minItems** i **maxItems** (npr. mora imati bar 1 hobi).

Primer "Šablona" za vežbu:

```
db.createCollection("kolekcija", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["polje1"],
      properties: {
        polje1: {
          bsonType: "string",
          description: "Kratak opis za slučaj greške"
        },
        polje2: {
          bsonType: "int",
          minimum: 0
        }
      }
    }
  }
});
```

Zlatni standard

- Kad god pišemo šemu, uvek idemo ovim redom:
  - Otvorimo $jsonSchema.
  - Kažemo da je koren bsonType: "object".
  - Napišemo required niz.
  - U properties definišemo pravila za svako polje.

Praktičan primer (Kreiranje kolekcije)

- Evo kako kreiramo kolekciju studenti sa strogim pravilima:

```
db.createCollection("studenti", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: ["ime", "prezime", "godine"],
         properties: {
            ime: {
               bsonType: "string",
               description: "Mora biti string i obavezno je"
            },
            godine: {
               bsonType: "int",
               minimum: 18,
               maximum: 100,
               description: "Broj između 18 i 100"
            },
            smer: {
               enum: ["IT", "Dizajn", "Menadžment"],
               description: "Može biti samo jedna od ponuđenih vrednosti"
            }
         }
      }
   }
});
```

Validacija složenih tipova (Nizovi i Objekti)

- Svaki element u nizu polozeni_ispiti mora biti objekat koji ima naziv i ocenu koja nije manja od 6:

```
properties: {
   polozeni_ispiti: {
      bsonType: "array",
      items: {
         bsonType: "object",
         required: ["naziv", "ocena"],
         properties: {
            ocena: { bsonType: "int", minimum: 6 }
         }
      }
   }
}
```

Šta se desi kada validacija ne uspe?

- Kada korisnik pokuša da unese loš podatak:
  - MongoDB odbija operaciju.
  - Vraća WriteConcernError (ili DocumentValidationFailure).
  - ValidationAction:
    - error (default): Blokira upis.
    - warn: Upisuje, ali beleži grešku u sistemski log (korisno za migraciju starih podataka).