Uvod u MongoDB

Stefan Kapunac

Univerzitet u Beogradu, Matematički fakultet

3.5.2020.

Pregled

1 Uvod

2 Upitni jezik

3 Projektovanje

4 Literatura

Karakteristike

- Baze dokumenata
- Podaci se čuvaju u dokumentima nalik JSON-u (BSON - Binary JSON)
- Upitni jezik - i sami upiti su JSON
- Podrska za (ACID) transakcije
- Distribuiranost

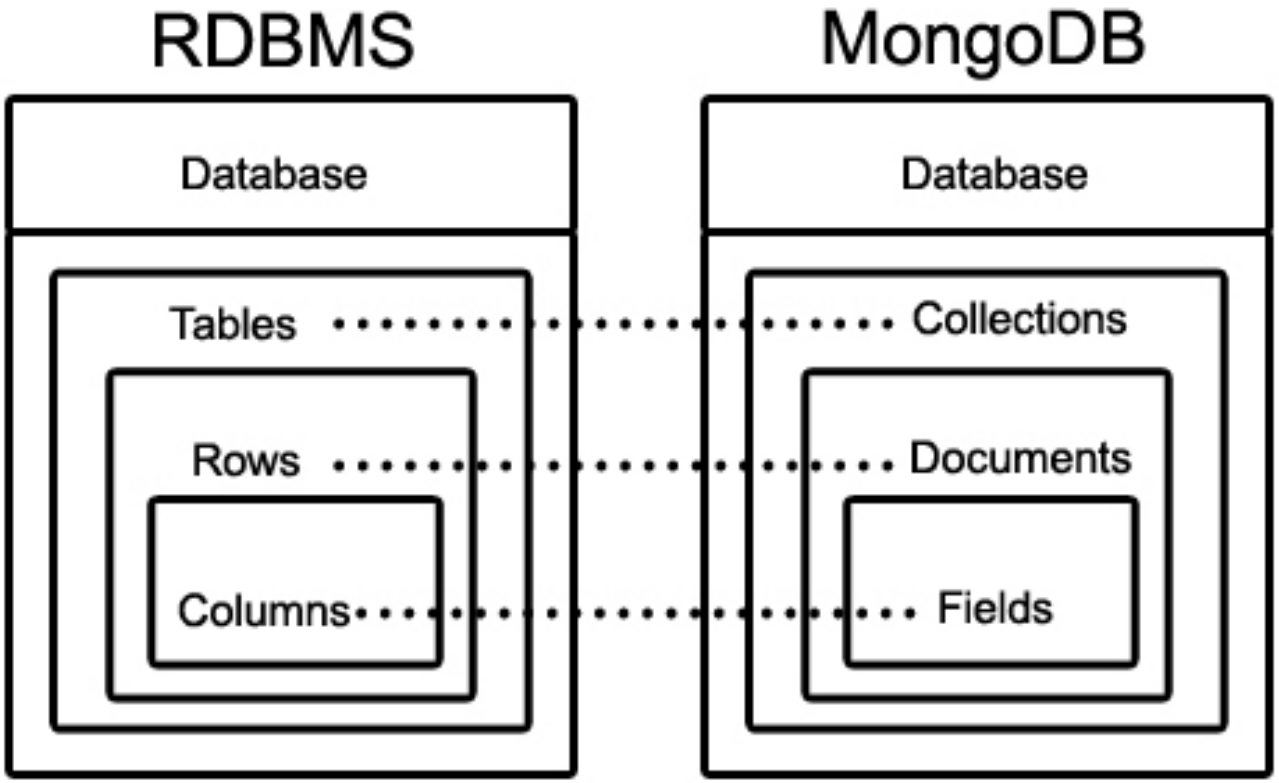
MongoDB Atlas

- Baza podataka kao servis (database as a service)
- Oblak (cloud)
- Replikacija
- Klaster - grupa servera koji čuvaju podatke
- Replica set - klaster gde svaki server čuva iste podatke
- Redundantnost (tj. ponavljanje podataka) povećava dostupnost
- Prestanak rada jednog servera - i dalje imamo podatke na raspolaganju

Klijenti

- mongo shell - interaktivni javaskript interfejs
- MongoDB Compass - grafičko korisničko okruženje

Poređenje sa relacionim bazama



Dokumenti

- Parovi "ključ": vrednost, ...
- Kolekcija sadrži slične dokumente
- Terminologija: JSON objekat = JSON dokument

Tipovi polja u dokumentima

- Skalarni - string, int32, double, date...
- Nizovi - [element, ...]
- Objekti - {"ključ": vrednost, ...}
- Kombinacije
  - Niz objekata
  - Objekat čije je polje niz
  - ...

Fleksibilnost

- Jedno polje može da bude različitog tipa u različitim dokumentima iste kolekcije
- Npr. polje "viewerRating": double, int32, undefined

_id

- Svaki dokument ima polje _id - jedinstveni identifikator
- Automatsko postavljanje ovog polja - tip ObjectId
- Moguće i ručno postavljanje ovog polja - bilo koji tip

Primer dokumenta


```
{
        "_id" : ObjectId("58c59c7599d4ee0af9e25634"),
        "title" : "The Shawshank Redemption",
        "year" : 1994,
        "mpaaRating" : "R",
        "genre" : "Crime, Drama",
        "viewerRating" : 9.3,
        "viewerVotes" : 1521105,
        "runtime" : 142,
        "director" : "Frank Darabont",
        "cast" : [
                "Tim Robbins",
                "Morgan Freeman",
                "Bob Gunton",
                "William Sadler"
        ],
        "awards": {
          "wins" : 21,
          "nominations" : 35
        }
}
```

Fleksibilna shema

- U jednoj kolekciji mogu da postoje dokumenti sa različitom shemom
- Nemaju svi ista polja - različito od relacionih baza

Fleksibilna shema - primer

- Tabela sa podacima o filmovima
- Neki filmovi su dobili neku nagradu
- Većina nije nijednu

Relacione baze

- Jedna tabela
- Svi redovi imaju iste kolone
- U koloni nagrada - puno NULL vrednosti

Mongo

- Filmovi koji su dobili nagradu imaju polje za nagradu
- Filmovi koji nisu dobili nagradu nemaju polje za nagradu

Upiti

- Upit - JSON objekat
- mongo shell - db.<kolekcija>.find(<upit>)
- MongoDB Compass - samo upit (i klik...)

Primer 1

- `db.movies.find({director: "Sergio Leone"})`
- Nad kolekcijom movies pozivamo metod find sa jednim argumentom
- Rezultat - svi dokumenti u kolekciji movies koji za vrednost polja director imaju Sergio Leone
- SQL:

```
select *
from movies
where director = "Sergio Leone"
```

Primer 2

- Konjunkcija uslova
- `db.movies.find({director: "Sergio Leone", year: 1971})`
- SQL:

```
select *
from movies
where director = "Sergio Leone"
  and year = 1971
```

Operatori poređenja

Mongo

- $eq
- $gt
- $lt
- $gte
- $lte
- $ne
- $in
- $nin

SQL

- =
- >
- <
- >=
- <=
- <>
- in
- not in

Primer 3

- ```
  db.movies.find({director: "Sergio Leone",
                  year: {$gte: 1970, $lt: 1975}})
  ```
- SQL:

```
select *
from movies
where director = "Sergio Leone"
  and year >= 1970 and year < 1975
```
```

Primer 4

- Vrednost za $in je niz
- ```
  db.movies.find({director: {$in: ["Sergio Leone",
                                   "Francis Ford Coppola"]},
                  year: {$gte: 1970, $lt: 1975}})
  ```
- SQL:

```
select *
from movies
where director in ("Sergio Leone",
                   "Francis Ford Coppola")
  and year >= 1970 and year < 1975
```
```

Primer 5

- `db.movies.find({director: {$ne: "Sergio Leone"}})`
- Rezultat = svi dokumenti čija vrednost polja director nije Sergio Leone + svi dokumenti koji nemaju polje director

Logički operatori

Mongo

- $and
- $or
- $not
- $nor

SQL

- and
- or
- not
- not (<kolona1> or <kolona2>)

Zašto postoji $and?

- Videli smo primer konjunkcije
- `db.movies.find({director: "Sergio Leone", year: 1971})`
- $and u ovom slučaju nije potreban

Kada $and jeste potreban?

Kada hoćemo u više izraza da koristimo:

- isto polje
- ```
  db.movies.find({$and: [
                          {director: {$ne: "Sergio Leone"}},
                          {director: {$exists: true}}
                        ]})
  ```
- ili isti operator
- ```
  db.movies.find({$and: [
                           {$or: [{director: "Sergio Leone"},
                                  {director: "Francis
                                              Ford Coppola"}]},
                           {$or: [{year: 1971}, {year: 1981}]}
                         ]})
  ```

Ne baš...

- Postoji drugi način za korišćenje istog polja
- `db.movies.find({director: {$ne: "Sergio Leone", $exists: true}})`
- Ali za korišćenje istog operatora ne postoji drugi način

Još operatora

- Više o ovim i drugim operatorima ovde

Upiti nad objektima

- Prethodni upiti nad poljima skalarnih tipova
- Objekti - `"<polje>.<potpolje>"`
- `db.movies.find({"awards.wins": 2})`
- Navodnici su obavezni

Upiti nad nizovima - 1

- Ceo niz - <polje>: [<element1>, ..., <elementN>]
- db.movies.find({cast: ["Morgan Freeman", "Tim Robins"]})
- Rezultat = svi dokumenti čija je vrednost polja cast identična kao u upitu - samo ta dva glumca, u tom redosledu

Upiti nad nizovima - 2

- Jedan element - <polje>: <element>
- `db.movies.find({cast: "Morgan Freeman"})`
- Rezultat = svi dokumenti kod kojih je "Morgan Freeman"jedan od elemenata niza cast

Upiti nad nizovima - 3

- Indeksiranje niza - "<polje>.<indeks>"
- `db.movies.find({"cast.0": "Morgan Freeman"})`
- Rezultat = svi dokumenti kod kojih je "Morgan Freeman"prvi element niza cast
- Navodnici su obavezni

Projekcija

- Izdvajanje polja koje želimo u rezultatu
- Drugi argument metoda find
  - uključujemo - <polje>: 1
  - isključujemo - <polje>: 0
- _id podrazumevano uključen

Projekcija - primer

- `db.movies.find({year: 2020}, {title: 1, _id: 0})`
- SQL:

```
select title
from movies
where year = 2020
```

Insert

Dva osnovna načina:

- Metod insertOne
- Metod insertMany

insertOne

- Jedan argument - dokument koji želimo da dodamo u kolekciju nad kojom pozivamo metod
- `db.movies.insertOne({title: "The Godfather", year: 1972})`
- _id se automatski dodaje

insertMany

- Jedan argument - niz dokumenata koje želimo da dodamo u kolekciju nad kojom pozivamo metod
- ```
  db.movies.insertMany([
      {_id: 1, title: "12 Angry Men", year: 1957},
      {_id: 2, title: "Rocky", year: 1976},
      {_id: 1, title: "12 Angry Men", year: 1957},
      {_id: 3, title: "The Godfather", year: 1972}
                                  ])
  ```
- Prvi i treći dokument identični (isti _id) - zaustavlja se sa unošenjem dokumenata
- Rezultat - uneta su prva dva dokumenta u kolekciju
  ```

insertMany - ordered

- Drugi argument - "ordered": false
- ```
  db.movies.insertMany([
    {_id: 1, title: "12 Angry Men", year: 1957},
    {_id: 2, title: "Rocky", year: 1976},
    {_id: 1, title: "12 Angry Men", year: 1957},
    {_id: 3, title: "The Godfather", year: 1972}
                          ],
                          {"ordered": false})
  ```
- Poruka o grešci za treći dokument, ali se nastavlja sa unošenjem sledećih
- Rezultat - unet je (pored prva dva) i četvrti film
  ```

Update

Dva osnovna načina:

- Metod updateOne
- Metod updateMany

updateOne

- 2 argumenta
- Prvi argument - filter kojim biramo dokument koji želimo da izmenimo
- Drugi argument - kako menjamo dokument

Primer

- Za film Rocky nemamo režisera - ne postoji polje director u tom dokumentu
- ```
  db.movies.updateOne({title: "Rocky"},
                      {$set: {director:
                          "John G. Avildsen"}})
  ```
- Operator $set
- U ovom primeru ne postoji polje director, pa ga dodajemo
- Ako bi postojalo, ovako bismo promenili njegovu vrednost
  ```

updateMany

- Isti argumenti kao za updateOne
- Razlika - menjamo SVE dokumente koji ispunjavaju uslov

Primer

- Čišćenje podataka - brišemo polje viewerRating iz svih dokumenata u kojima je njegova vrednost null
- `db.movies.updateMany({viewerRating: null}, {$unset: {viewerRating: ""}})`
- Operator $unset - briše polje (ili polja)
- Vrednost "" je nebitna - ne utiče na brisanje

Upsert

- updateOne može i da doda novi dokument
- Dodatni argument - upsert: true
- updateOne(<uslov>, $set: <izmena>, upsert: true)
- Automatska provera da li postoji dokument koji ispunjava uslov
  - Ako da - menja ga
  - Ako ne - dodaje nov

Delete

- deleteOne
- deleteMany
- Oba metoda - 1 argument - filter kojim biramo koji dokument(e) želimo da obrišemo

$lookup

- $lookup ≈ JOIN
- {

```
    $lookup:
      {
        from: <kolekcija sa kojom se spaja>,
        localField: <polje ulaznih dokumenata>,
        foreignField: <polje dokumenata iz
                       "from" kolekcije>,
        as: <izlazno polje - niz>
      }
}
```

Primer

- Prva kolekcija - orders
- ```
  db.orders.insert([
      { "_id" : 1, "item" : "almonds",
        "price" : 12, "quantity" : 2 },
      { "_id" : 2, "item" : "pecans",
        "price" : 20, "quantity" : 1 },
      { "_id" : 3  }
  ])
  ```
  ```

Primer

- Druga kolekcija - inventory
- db.inventory.insert([

```
{ "_id" : 1, "sku" : "almonds",
  description: "product 1", "instock" : 120 },
{ "_id" : 2, "sku" : "bread",
  description: "product 2", "instock" : 80 },
{ "_id" : 3, "sku" : "cashews",
  description: "product 3", "instock" : 60 },
{ "_id" : 4, "sku" : "pecans",
  description: "product 4", "instock" : 70 },
{ "_id" : 5, "sku": null,
  description: "Incomplete" },
{ "_id" : 6 } ])
```

Primer

- Spajanje na osnovu polja item i sku
- ```
  db.orders.aggregate([
      {
        $lookup:
          {
            from: "inventory",
            localField: "item",
            foreignField: "sku",
            as: "inventory_docs"
          }
      }
  ])
  ```
  ```

Primer - rezultat

```
{
   "_id" : 1,
   "item" : "almonds",
   "price" : 12,
   "quantity" : 2,
   "inventory_docs" : [
      { "_id" : 1, "sku" : "almonds", "description" : "product 1", "instock" : 120 }
   ]
}
{
   "_id" : 2,
   "item" : "pecans",
   "price" : 20,
   "quantity" : 1,
   "inventory_docs" : [
      { "_id" : 4, "sku" : "pecans", "description" : "product 4", "instock" : 70 }
   ]
}
{
   "_id" : 3,
   "inventory_docs" : [
      { "_id" : 5, "sku" : null, "description" : "Incomplete" },
      { "_id" : 6 }
   ]
}
```

Primer

- Ista stvar u (pseudo)SQL-u:
- ```
  SELECT *, inventory_docs
  FROM orders
  WHERE inventory_docs IN (SELECT *
                           FROM inventory
                           WHERE sku = orders.item);
  ```
  ```

3 faze

- Opis radnog opterećenja
- Određivanje odnosa između entiteta
- Primena obrazaca za projektovanje

Radno opterećenje

- Količina podataka
- Broj operacija (čitanja i pisanja) u jedinici vremena
- Dodatni zahtevi:
  - Brzina
  - Tolerancija na "ustajale"podatke
  - ...

Trajnost podataka

- Podaci vezani za novac (ili bilo šta što ne smemo da izgubimo) - čekamo da većina čvorova upiše podatke
- Podaci iz senzora, logovi... - ne čekamo većinu čvorova
- Neke podatke smemo da izgubimo zarad boljih performansi

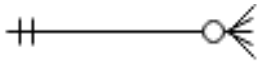
Odnosi

Isto kao kod ER modela

- Jedan prema jedan: 1-1
- Jedan prema više: 1-N
- Više prema više: N-N

Dodatno

- Jedan prema zilion - na dijagramu naglašavamo veliki broj entiteta (npr. veći od 10000)



- Kardinalnost: [minimum, najverovatnija vrednost, maksimum]

Jedan prema jedan

- Ugrađivanje - jedan dokument za dva entiteta
  - Polja na istom nivou
  - Grupisanje u poddokumente
- Referenciranje - dva dokumenta za dva entiteta
  - Isti identifikator u oba dokumenta
  - Referenca na jednoj strani
  - Referenca na drugoj strani

Polja na istom nivou

```
{
        _id: <objectId>,
        name: <string>,
        street: <string>,
        city: <string>,
        zip: <string>,
        shipping_street: <string>,
        shipping_city: <string>,
        shipping_zip: <string>
}
```

Grupisanje u poddokumente

- Preporučeni način - bolja organizacija

```
{
        _id: <objectId>,
        name: <string>,
        address: {
          street: <string>,
          city: <string>,
          zip: <string>
        },
        shipping_address: {
          shipping_street: <string>,
          shipping_city: <string>,
          shipping_zip: <string>
        }
}
```

Referenciranje

- Kompleksnije nego prethodno
- Moguće poboljšanje performansi - ako nam često nisu potrebni podaci o zaposlenima

```
{
        _id: <objectId>,
        storeId: <string>,
        name: <string>,
        address: {
          street: <string>,
          city: <string>,
          zip: <string>
        }
}
```

```
{
        _id: <objectId>,
        storeId: <string>,
        manager: {
          name: <string>,
          address: <string>,
        },
        staff: [{
          name: <string>,
          address: <string>,
          title: <string>
        }]
}
```

Jedan prema više

- Ugrađivanje
  - Na strani "jedan"
  - Na strani "više"
- Referenciranje
  - Na strani "jedan"
  - Na strani "više"

Ugrađivanje na strani "jedan"

- Niz poddokumenata
- Najčešći način kada je mali broj ugrađenih poddokumenata

```
{
        _id: <objectId>,
        title: <string>,
        category: <string>,
        price: <decimal>,
        img_url: <string>,
        top_reviews: [{
          user_id: <string>,
          user_name: <string>,
          date: <date>,
          body: <string>
        }]
}
```

Ugrađivanje na strani "više"

- Ređe se koristi
- Ima smisla ako imamo više upita na strani "više"(u primeru: fokus je na porudžbinama, a ne na adresama)
- Mana - redundantnost, ponavljanje iste adrese u svim porudžbinama

```
{
        _id: <objectId>,
        customer_id: <string>,
        date: <date>,
        items: [{
          item_id: <string>,
          qty: <int>,
          price: <double>
        }],
        shipping_address: {
          street: <string>,
          city: <string>,
          zip: <string>
        }
}
```

Referenca na strani "jedan"

- Niz referenci
- MongoDB ne podržava strane ključeve i kaskadno brisanje - moramo sami da se brinemo o tome
- Za svaki grad čuvamo niz prodavnica koje se nalaze u tom gradu

Gradovi

```
{
        _id: <objectId>,
        name: <string>,
        county: <string>,
        population: <long>,
        stores: [<string>]
}
```

Prodavnice

```
{
        _id: <objectId>,
        storeId: <string>,
        name: <string>,
        address: <string>
}
```

Referenca na strani "više"

- Ako se koriste reference - preferiran način
- Nema problema kod brisanja - ako obrišemo prodavnicu, ne menjamo ništa u gradu

Gradovi

```
{
        _id: <objectId>,
        name: <string>,
        county: <string>,
        population: <long>
}
```

Prodavnice

```
{
        _id: <objectId>,
        storeId: <string>,
        name: <string>,
        address: <string>,
        city: <string>
}
```

Jedan prema zilion

- Podtip veze jedan prema više
- U ovom slučaju samo jedna opcija ima smisla - referenca na strani više

Proizvodi

```
{
        _id: <objectId>,
        title: <string>,
        category: <string>,
        price: <decimal>,
        img_url: <string>,
        top_reviews: [{
          user_id: <string>,
          user_name: <string>,
          date: <date>,
          body: <string>
        }]
}
```

Pregledi proizvoda na sajtu prodavnice

```
{
        _id: <objectId>,
        item_id: <int>,
        user_id: <objectId>,
        view_date: <date>
}
```

Više prema više

- Ugrađivanje
  - Niz poddokumenata na jednoj strani
  - Niz poddokumenata na drugoj strani
- Referenciranje
  - Niz referenci na jednoj strani
  - Niz referenci na drugoj strani

Ugrađivanje

- Dokumente nad kojima imamo manje upita ugrađujemo u one nad kojima imamo više upita
- Ali čuvamo i originale u drugoj kolekciji - proizvod može da postoji i ako nije ni u jednoj korpi

Korpe

```
{
        _id: <objectId>,
        date: <date>,
        items: [
          qty: <int>,
          {
          _id: <int>,
          title: <string>,
          category: <string>,
          price: <decimal>
          }
        ]
}
```

Proizvodi

```
{
        _id: <int>,
        title: <string>,
        category: <string>,
        price: <decimal>
}
```

Referenciranje

- Izbegavamo ponavljanje podataka - čuvamo samo niz referenci (u primeru: sold_at)
- Ali sada su nam potrebna dva upita ili $lookup

Proizvodi

```
{
        _id: <objectId>,
        title: <string>,
        category: <string>,
        price: <decimal>,
        img_url: <string>,
        sold_at: [<string>],
        top_reviews: [{
          user_id: <string>,
          user_name: <string>,
          date: <date>,
          body: <string>
        }]
}
```

Prodavnice

```
{
        _id: <objectId>,
        storeId: <string>,
        name: <string>,
        address: <string>,
        city: <string>
}
```

Obrasci za projektovanje

- Važni
- Korisni
- ...
- Prevelika tema za ovaj "uvod"

Neki često korišćeni obrasci za projektovanje

- Attribute
- Extended Reference
- Subset
- Computed
- Bucket
- Schema Versioning
- Tree
- Polymorphic

Literatura

- Dokumentacija
- MongoDB University