2-11 Ostale Mongo funkcije i operatori

Sortiranje

Da bi se sortirao objekat koristi se `sort()` funkcija nad objektima vraćenim koristeći `find()` metodu. Npr.:

```
db.studenti.find({}).sort({ ime: 1, prezime: -1});
```

Ovaj upit će vratiti sve studente sortirano abecedno po imenu i suprotno abecedno po prezimenu.

S obzirom da _id polje u sebi sadrži datum kreiranja dokumenta, interesantna posljedica ovoga je da se može koristiti prilikom sortiranja, tako da se svi studenti, sortirani od zadnje kreiranog mogu pronaći koristeći:

```
db.studenti.find({}).sort({ _id: -1 });
```

Ograničavanje i paginacija

Kako bi se pronašla određena stranica iz izlaznog skupa, mogu se koristiti metode `skip()` i `limit()`. Tako npr. šesta stranica studenata, ako stranica ima 10 elemenata, kombinovano sa sortiranjem:

```
db.studenti.find({}).sort({ ime: 1 }).skip(5*10).limit(10);
```

Tekstualna pretraga

Pošto je moguće imati mnogo tekstualnih polja, potrebno je moći ih pretraživati. Vidjeli smo da možemo jednostavno reći `find({ ime: 'Alisa' })` da pronađemo po imenu, ali ako želimo pretraživati tekstualne podatke po djelimičnom podudaranju. Tako npr. da nađemo sva imena koja počinju sa A, napisat ćemo:

```
db.studenti.find({ ime: /^A[a-z]+/i })
```

Ova komanda je zapravo dio Regex jezika za tekstualnu pretragu, kojim se zapravo definišu šabloni. Pravila ovog jezika su da se šablon po kojem se podudara tekst, definiše unutar dvije kose crte (`/`) nakon koje idu opcije. Tako da ovaj šablon ima sljedeće značenje:

`^` – kapica ili caret simbol, definiše početak teksta
`A` – govori da se očekuje slovo A i s obzirom na prethodni simbol, prvo slovo mora biti A
`[a-z]` – govori da se očekuje bilo koje slovo od a do z
`+` – symbol koji označava da se prethodni iskaz može podudarati jednom ili više puta

Zbog toga što se tekst „Alice“ ili „Alisa“ zadovoljavaju ovaj šablon, onda će biti pronađeni svi objekti gdje ime ima podudaranje sa ovim šablonom.
Neke od ostalih primjera će biti izlistani ali s obzirom da regex može biti komplikovan postoje alati sa kojima se može testirati podudaranje[8]. Npr.:

```
/^[0-9]{3} [0-9]{3}-[0-9]{3,4}$/
```

Šablon kojim se na početku (^) moraju naći tri ({3}) broja ([0-9]) onda mora biti razmak, onda moraju biti tri broja pa crta pa tri ili četri ({3,4}) broja opet i onda je kraj stringa. Ovo je korisno za podudaranje brojeva telefona, npr. 061 123-456 ili 062 123-4567, jer će svi netačno upisani brojevi pasti na testiranju šablona.

```
/^\D+ic$/
```

Šablon kojim se podudaraju svi stringovi koji od početka imaju 1 ili više (+) slova (\D), i završavaju ($) sa ic. Postoje mnogi drugi simboli, jer su ovo samo primjeri te se ovim jezikom mogu napraviti raznorazni šabloni za podudaranje stringova.

Projekcija

S obzirom da objekti mogu biti različite veličine, te imati nizove u svojim atributima, različitih dimenzija, pronalaženje objekata koji mogu biti veliki može predstavljati problem. Postoje situacije u kojima nisu neophodne informacije svakog atributa. Npr. ako tražimo studenta, a student imaju listu hobija, i ima objekat *osobine* sa dugom listom osobina, međutim nas ne interesuju ni osobine ni hobiji već samo *ime*, *prezime*, i *datum rodjenja*, onda možemo iskoristiti projekcije da vratimo samo određenu listu atributa.

```
db.studenti.find({}, { ime: 1, prezime: 1, datum_rodj: 1 });
```

Projekcija se defineše kao drugi argument u `find()` funkciji. Taj objekat ima popis atributa koje želimo da nam se vrate i 1 kao vrijednost ili 0 ako ne želimo te atribute.

Ukoliko je potrebno vratiti sve postojeće atribute, osim jednog onda možemo napisati:

```
db.studenti.find({}, { hobiji: 0 });
```

U ovom slučaju MongoDB će vratiti sve studente i sve njihove atribute osim atributa hobiji. Na ovaj način se mogu optimizovati upiti kako se nepotrebni podaci ne bi prenosili preko mreže i tako bi se ubrzao prenos i smanjili troškovi prenosa podataka koji se obično naplaćuju.

Objektne baze podataka (u slučaju MongoDB dokumentna) faju fleksibilnu mogućnost i fleksibilnu strukturu organizacije podataka. Nedostatak je što nemaju jaku podršku za transakcije te je zbog toga potrebno pažljivo odmjeriti potrebe sistema za koji se projektuje i bira baza podataka.

[8] https://regex101.com/#javascript

Indeksiranje

Kao i sve ostale baze podataka, i MongoDB ima indeksiranje koje omogućava brže pretraživanje po jednom ili više polja. Za razliku od MySQL baze, prilikom kreiranja indeksa, definiše se hoće li indeks biti uzlazni ili silazni. Da se pogledaju postojeći indeksi, koristi se getIndexes().

```
db.studenti.getIndexes();
```

```
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_"
    }
]
```

Po defaultu, svaka kolekcija je indeksirana po _id polju. Međutim, moguće je napraviti indeks po bilo kojem drugom polju koristeći metodu `createIndex()`.

```
db.studenti.createIndex({ ime: 1 }, { name: 'Indeks po imenu' });
```

Na ovaj način smo kreirali indeksiranje po imenu i dali mu naziv. Vrijednost 1 za ime pri kreiranju indeksa znači da će biti kreiran uzlazno sortirani indeks, a ako je -1 taj indeks će biti silazno sortirann. Na isti način možemo kreirati indeks koristeći više atributa.

```
db.studenti.createIndex(
      { ime: 1, prezime: -1 },
      { name: 'Indeks po nazivu' }
);
```

Kako smo kreirali indeks, tako možemo i obrisati indeks nad kolekcijom, identifikovajući ga koristeći njegovo ime ili specifikaciju:

```
db.studenti.dropIndex({ ime: 1});
```

ili

```
db.studenti.dropIndex('Indeks po imenu');
```