Mongo DB

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/DATAB Databáze

Příklad

- Uvažujme kolekci v souboru datab_10_mongodb_3.mongodb.

Cesta

- **Krok** je buď řetězec zapisující nezáporné celé číslo, nebo název položky.
- Například: 2 nebo name.
- **Cesta** je neprázdný řetězec, který se skládá z kroku oddělených tečkou.
- Například: movies.0.title

Příklad

- Najdeme osoby, které jsou z Olomouce:

```
db.osoby.find({"adresa.mesto":"Olomouc"})
```

- Najdeme osoby, které mají splněný první zapsaný předmět:

```
db.osoby.find({"predmety.0.ukoncen":true})
```

Operátor $type

- Hodnota splňuje podmínku

  ```
  { $type: type }
  ```

  pokud je zadaného typu.

Příklad

- Najdeme všechny osoby, jejichž jméno je string:

```
db.osoby.find({jmeno:{$type:"string"}})
```

- Najdeme všechny osoby, jejichž adresa je objekt:

```
db.osoby.find({adresa:{$type:"object"}})
```

- Najdeme všechny osoby, jejichž predmety jsou pole:

```
db.osoby.find({predmety:{$type:"array"}})
```

Projekce

- Metoda find tvaru

```
db.collection.find(condition , projection )
```

  vrátí pole projekcí dokumentu v kolekci, které splňují podmínku.

- Projekcí dokumentu rozumíme dokument, v kterém zůstanou pouze některé z původních položek dokumentu.

Projekce dokumentu

- Projekce

```
{ field1: 1, field2: 1, . . . }
```

ponechá v dokumentu položky _id, field1, field2, ...

- Projekce

```
{ field1: 1, field2: 1, . . ., _id: 0 }
```

ponechá v dokumentu položky field1, field2, . . .

- Projekce

```
{ field1: 0, field2: 0, . . . }
```

ponechá v dokumentu všechny položky, kromě field1, field2, . . .

Příklad

- Zobrazíme u osob jen jméno, příjmení a adresu:

```
db.osoby.find({},{jmeno:1,prijmeni:1,adresa:1})
```

- Zobrazíme u osob jen jméno a město:

```
db.osoby.find({},{jmeno:1,"adresa.mesto":1})
```

Úkol

V kolekci predmety proveďte tyto dotazy:

- U předmětů zobrazte pouze název, zkratku a příjmení vyučujícího.
- Vyhledejte předměty, které vyučuje pan Večerka.
- Vyhledejte předměty, kde je na druhém místě zapsaná Alena.

Příklad

- Změníme osobě Jiří první předmět na neukončený:

```
db.osoby.updateMany({jmeno:"Jiří"},{$set:{"predmety.0.ukoncen":false}})
```

- Pokud chceme aktualizovat prvek pole, který není, vytvoří se a za existující prvky se doplní hodnota null:

```
db.osoby.updateMany({jmeno:"Jiří"},{$set:{"predmety.3.ukoncen":false}})
```

- Prázdnou hotnotu pak můžeme aktualizovat:

```
db.osoby.updateMany({jmeno:"Jiří"},{$set:{"predmety.2":{"zkratka":"KMI/URO","ukoncen":false}}})
```

Příklad

- Nastavíme, že u osoby Jiří se první neukončený předmět ukončí:
- `db.osoby.updateMany({jmeno:"Jiří","predmety.ukoncen":false},{$set:{"predmety.$.ukoncen":true}})`

- Nastavíme, že u osoby Jiří se všechny neukončené předmět ukončí:
- `db.osoby.updateMany({jmeno:"Jiří","predmety.ukoncen":false},{$set:{"predmety.$[].ukoncen":true}})`

Komparátor $in:

- Komparátor
  - `$in` rozhoduje, zda se hodnota nalézá v zadaném poli,
  - `$nin`, zda nikoliv.
- Operátor `$in` slouží k vytvoření podmínky na hodnotu:

```
{ $in: <array> }
```

- kde <array> je pole hodnot.
- Podmínka je v hodnotě splněna, pokud se hodnota nachází v poli <array>.

Příklad

- Vyhledáme všechny osoby se jménem Jiří nebo Martina:

```
db.osoby.find({jmeno:{$in:["Jiří","Martina"]}})
```

- Vyhledáme všechny osoby, které se nejmenují Jiří nebo Martina:

```
db.osoby.find({jmeno:{$nin:["Jiří","Martina"]}})
```

- Vyhledáme všechny osoby, které mají zapsaný některý z předmětů:

```
db.osoby.find({"predmety.zkratka":{$in:["KMI/DATAB","KMI/STRUP"]}})
```

Operátor $elemMatch

- Pole splňuje podmínku:

```
{ $elemMatch: condition }
```

  pokud některý jeho prvek splňuje podmínku condition .

- Položka dokumentu, kde hodnota je pole, je někdy vnímána tak, že dokument má více položek téhož jména, kde hodnoty jsou prvky pole.

Příklad

- Najdeme osobu, která má ukončený předmět KMI/DATAB:

```
db.osoby.find({predmety:{$elemMatch:{$eq:{"zkratka":"KMI/DATAB","ukoncen":true}}}})
```

- Najdeme osobu, která má ukončený předmět KMI/DATAB nebo KMI/WEBA:

```
db.osoby.find({predmety:{$elemMatch:{$in:[{"zkratka":"KMI/DATAB","ukoncen":true},{"zkratka":"KMI/WEBA","ukoncen":true}]}}})
```

Úkol

V kolekci predmety proveďte tyto dotazy:

- U předmětů Algerba 1 změňte druhého zapsaného na Karla Malého.
- Vyhledejte předměty, které mají zapsány Karel nebo Marie (použijte operátor $in).
- Vyhledejte předměty, která má zapsány Patrik Berger (použijte operátor $elemMatch).

Operátory

- Pole splňuje podmínku

```
{ $not: condition }
```

  pokud každý prvek nesplňuje podmínku condition .

- Pole splňuje podmínku

```
{ $ne: value }
```

  pokud se žádný prvek pole nerovná value .

- Pole array1 splňuje podmínku

```
{ $nin: array2 }
```

  pokud se žádný prvek pole array1 nenachází v poli array2.

Příklad

- Najdeme osoby, která se nejmenují Jiří:

```
db.osoby.find({jmeno:{$not:{$eq:"Jiří"}}})
```

- Najdeme osoby, které mají ukončený předmět KMI/ZPP1:

```
db.osoby.find({predmety:{$ne:{"zkratka":"KMI/ZPP1","ukonce
n":false}}})
```

- Totéž, jen jinak:

```
db.osoby.find({predmety:{$nin:[{"zkratka":"KMI/ZPP1","ukon
cen":false}]}})
```

Operátory

- Pole splňuje implicitní konjunkci podmínek condition1, condition2, . . . , pokud splňuje každou z podmínek condition1, condition2, . . .
- Pole splňuje podmínku

```
{ $all: [ value1, value2, . . . ] }
```

  pokud obsahuje všechny prvky value1, value2, . . .
- Pole splňuje podmínku

```
{ $size: length }
```

  pokud je jeho velikost length .

Příklad

- Najdeme osoby, která neukončili předmět KMI/ZPP1:

```
db.osoby.find({predmety:{$all:[{"zkratka":"KMI/ZPP1","ukoncen":false}]}})
```

- Najdeme osoby, které mají zapsány dva předměty:

```
db.osoby.find({predmety:{$size:2}})
```

Úkol

V kolekci predmety proveďte tyto dotazy:

- Najděte předměty, které se nejmenují Algebra 1.
- Vyhledejte předměty, které mají současně zapsány Patrik Berger a Marie Dlouhá.
- Vyhledejte předměty, která mají zapsáni tři studenti.

Operátor $expr

- Slouží k vytváření výrazů.
- Příklad:
  - Zjistíme osoby, které získaly dostatečný počet kreditů:

```
db.osoby.find({
                $expr:{
                  $gt:[
                    "$kredity",59
                    ]
                        }
                })
```

Proměnné

- Provedeme totéž jen pomocí proměnné:

```
db.osoby.find({
                $expr:{
                  $let:{
                    vars:{zapocet:{ $cond: { if:
{$gt:["$kredity",59]}, then: true, else: false } }},
                    in:{$eq:["$$zapocet",true]}
                  }
                }
            })
```

Agregace

- Vybereme muže:

```
db.osoby.aggregate([{
  $match:{"pohlavi":"muž"}
}])
```

- Spočítáme počet mužů:

```
db.osoby.aggregate([{
  $match:{"pohlavi":"muž"}
},
{$count:"celkem"}])
```

Agregace

- Vytvoříme skupiny podle pohlaví:

```
db.osoby.aggregate([{
  $group:{_id:"$pohlavi"}
}])
```

- Vytvoříme skupiny podle pohlaví a ročníku:

```
db.osoby.aggregate([{
  $group:{_id:["$pohlavi","$rocnik"]}
}])

db.osoby.aggregate([{
  $group:{_id:{pohlavi:"$pohlavi",rocnik:"$rocnik"}}
}])
```

Agregace

- Zobrazíme počet osob pro dané pohlaví:

```
db.osoby.aggregate([{
  $group:{_id:"$pohlavi",
          pocet:{$count:{}}
         }
}])
```

- Přidáme celkový počet kreditů:

```
db.osoby.aggregate([{
  $group:{_id:"$pohlavi",
          pocet:{$count:{}},
          celkem:{$sum:"$kredity"}
         }
}])
```

Agregace

- Celkový počet změníme na průměrný:

```
db.osoby.aggregate([{
  $group:{_id:"$pohlavi",
          pocet:{$count:{}},
          prumer:{$avg:"$kredity"}
         }
}])
```

- Přidáme maximální počet kreditů:

```
db.osoby.aggregate([{
  $group:{_id:"$pohlavi",
          pocet:{$count:{}},
          prumer:{$avg:"$kredity"},
          max:{$max:"$kredity"}
         }
}])
```

Úkol

- V kolekci zamestnanci vytvořte skupiny:
  - podle oddělení,
  - podle oddělení a funkce.
- Pro všechny skupiny spočítejte průměrnou hodinovou mzdu.
- Pro všechny skupiny spočítejte součet celkových mezd.

Třídění

- Dokumenty seřadíme podle příjmení vzestupně:

```
db.osoby.aggregate([{
   $sort:{prijmeni:1}
}])
```

- Dokumenty seřadíme podle příjmení sestupně :

```
db.osoby.aggregate([{
   $sort:{jmeno:-1}
}])
```

- Dokumenty seřadíme podle pohlaví vzestupně a podle jména sestupně:

```
db.osoby.aggregate([{
   $sort:{pohlavi:1,jmeno:-1}
}])
```

Třídění

- Zobrazíme první dva dokumenty:

```
db.osoby.aggregate([{
  $sort:{jmeno:-1}
},
{$limit:2}
])
```

- Zobrazíme třetí a čtvrtý dokument:

```
db.osoby.aggregate([{
  $sort:{jmeno:-1}
},
{$skip:2},
{$limit:2}
])

db.osoby.aggregate([{
  $sort:{jmeno:-1}
},
{$limit:2},
{$skip:2}
])
```

Projekce agregace

- Zobrazíme jen jméno a příjmení:

```
db.osoby.aggregate([{
   $project: {
      _id:0,
      jmeno:1,
      prijmeni:1
   }
}])
```

Vypočítaný klíč

- Vytvoříme klíč s chybějícími kredity:

```
db.osoby.aggregate([{
  $addFields: {
    chybi: {$subtract:[60,"$kredity"]}
  }
}])
```

Projekce agregace

- Do projekce doplníme vypočítané pole:

```
db.osoby.aggregate([{
  $project: {
    _id:0,
    jmeno:1,
    prijmeni:1,
    chybi: {$subtract:[60,"$kredity"]}
  }
}])
```

Úkol

- V kolekci zamestnanci setřiďte:
  - podle příjmení sestupně,
  - podle oddělení vzestupně a podle příjmení sestupně.
- Z kolekce zamestnanci zobrazte pouze první dokument.
- Z kolekce zamestnanci zobrazte pouze třetí dokument.
- Zobrazte z kolekce pouze jméno, příjmení a vypočítané pole celková mzda.

Bodovaný úkol

Vytvořte kolekci ze souboru datab_10_mongodb_3_ukol.mongodb.

- Vytvořte tyto dotazy:
  - Dotaz, který zobrazí pouze název knihy, jméno a příjmení autora a vydavatele.
  - Dotaz, který vyhledá knihy, která napsal Terry Pratchett.
  - Dotaz, který zobrazí knihy, které má půjčen jen jeden čtenář.