KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

Databáze ◇ poznámky k přednášce

10. Vnořování

verze z 20. listopadu 2023

1 Rovnost hodnot

Na atomických typech je dána rovnost. Máme například rovnost řetězců nebo čísel. Například 8 se rovná 8.0.

Pole A_1 se rovná poli A_2, pokud jsou obě pole stejně dlouhá a pro každý index $0 \leq i < n$, kde n je společná délka obou polí, platí, že prvek na indexu i pole A_1 se rovná prvku na indexu i pole A_2. Například pole:

```
[1, 2, 2]
```

se nerovná poli:

```
[2, 1, 2]
```

Dokument D_1 se rovná dokumentu D_2, pokud dokumenty mají stejné názvy položek uvedené ve stejném pořadí a pro každý společný název položky p platí, že hodnota položky p dokumentu D_1 se rovná hodnotě položky p dokumentu D_2.

Například dokument:

```
{ mean: 8, median: 8 }
```

se nerovná dokumentu:

```
{ median: 8, mean: 8 }
```

2 Cesty

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

Cesta určuje *místo* v hodnotě. Předpokládejme nejprve, že cesta má jediný krok S. Pokud je hodnota dokument, pak cesta určuje položku dokumentu jménem S. Pokud je hodnota polem a krok S indexem, pak cesta určuje prvek pole na indexu S.

Například cesta `first` určuje v dokumentu:

```
{
  first: "Andrei",
  last: "Tarkovsky"
}
```

položku `first`. Položka v dokumentu nemusí existovat. Například cesta `middle` v předchozím dokumentu určuje položku, která neexistuje. Podobně cesta `1` určuje v poli

```
["Francis", "Ford", "Coppola"]
```

prvek na indexu 1. I zde nemusí prvek na indexu existovat. Protože název položky může být i číslo, pak cesta `1` určuje v dokumentu:

```
{
  "1": "Ford"
}
```

položku s názvem `1`.

Je-li dána hodnota a cesta, můžeme chtít získat hodnotu na cestě, změnit hodnotu na cestě, nebo se zeptat, zda existuje. Například nastavení hodnoty na cestě `middle` na `"Ford` v dokumentu:

```
{
  first: "Francis",
  last: "Coppola"
}
```

povede na dokument:

```
{
  first: "Francis",
  last: "Coppola",
  middle: "Ford"
}
```

Nyní předpokládejme, že má cesta tvar

$$S.P$$

kde S je krok a P cesta. Pokud existuje hodnota na cestě S v hodnotě U, pak cesta $S.P$ určuje v hodnotě U místo dané cestou P v hodnotě V, kde V je hodnota na cestě S v hodnotě U.

Například cesta `movies.0.title` v dokumentu

```
{
    name: {
        first: "Andrei",
        last: "Tarkovsky"
    },
    movies: [{
        title: "Solaris",
        year: 1972,
        rating: 8
    }, {
        title: "Stalker",
        year: 1979,
        rating: 9
    }]
}
```

určuje místo v poli

```
[{
  title: "Solaris",
  year: 1972,
  rating: 8
}, {
  title: "Stalker",
  year: 1979,
  rating: 9
}]
```

na cestě `0.title`.

Cestu lze používat místo názvů položek v podmínkách, projekcích a změnách. Pokud cesta obsahuje tečku, musí být psána do dvojitých uvozovek (nejedná se o identifikátor JavaScriptu).

Například dokument splňuje podmínku `{ "ratings.mean": 8 }`, pokud má na místě určeném cestou `ratings.mean` hodnotu 8. Tedy má položku `ratings`, která obsahuje dokument s položkou `mean` a hodnotou 8. Podmínku splňuje například dokument:

```
{ title: "Solyaris", ratings: { mean: 8, median: 8  } }
```

Pokud místo na cestě není definováno, pak hodnota na místě neexistuje.

Například v dokumentu

```
{ title: "Cloud Atlas" }
```

cesta `ratings.mean` neurčuje žádné místo.

Pokud by u změny na cestě neexistovala požadované položka a byla by potřeba, pak se vytvoří nová a hodnotou bude prázdný dokument. Například změna

```
{ $set: { "ratings.mean": 8 } }
```

dokumentu

```
{ title: "Cloud Atlas" }
```

povede na dokument

```
{
  title: 'Cloud Atlas',
  ratings: { mean: 8 }
}
```

Změna `{ $set: { "movies.0.title": "Slaughterhouse-Five"} }` dokumentu:

```
{
  name: { first: 'George', middle: 'Roy', last: 'Hill' }
}
```

vede překvapivě na dokument:

```
{
  name: { first: 'George', middle: 'Roy', last: 'Hill' },
  movies: { '0': { title: 'Slaughterhouse-Five' } }
}
```

Při neexistenci položky se vždy vytváří dokument.

Pokud u změny na cestě neexistuje prvek pole na zadaném indexu a byl by potřeba, nakonec pole se přidá dostatečný počet hodnot `null`. Například změna

```
{ $set: { "name.2": "Coppola" }
```

dokumentu

```
{ name: [ "Francis" ] }
```

povede na

```
{ name: [ "Francis", null, "Coppola" ] }
```

V poli cesta $S.P$, kde S není číslo, určuje místa cest P ve všech prvcích pole. Například dokument:

```
{
  name: { first: 'Francis', middle: 'Ford', last: 'Coppola' },
  movies: [
    { title: 'The Godfather', year: 1972, rating: 9 },
    { title: "Bram Stoker's Dracula", year: 1992, rating: 7 }
  ]
}
```

splňuje podmínku `{ "movies.year": 1992 }`, protože cesta `movies.year` určuje místa, která mají hodnoty: 1972 a 1992. Jak víme, pro splnění této podmínky stačí, aby existovala aspoň jedna hodnota rovna 1992.

Cesty použité v projekci nesmí obsahovat číselné kroky. Projekcí `{ "name.last": 1, "movies.title": 1 }` dokumentu

```
{
  name: { first: 'Andrei', last: 'Tarkovsky' },
  movies: [
    { title: 'Solaris', year: 1972, rating: 8 },
    { title: 'Stalker', year: 1979, rating: 9 }
  ]
}
```

získáme:

```
{
  name: { last: 'Tarkovsky' },
  movies: [ { title: 'Solaris' }, { title: 'Stalker' } ]
}
```

Dokument v položce dokumentu D nebo v poli v položce dokumentu D se nazývá poddokument dokumentu D.

Pokud položka dokumentu obsahuje dokument nebo pole dokumentů, můžeme v projekci místo hodnoty 1 zadat objekt popisující projekci poddokumentů. Například:

```
{ name: { last: 1 }, movies: { title: 1 } }
```

Pokud máme dánu podmínku na prvek pole na cestě, můžeme u změny místo kroku cesty v poli zadat znak dolar (`$`). Změna je pak aplikována na první prvek pole splňující odpovídající podmínku. Například:

```
db.directors.updateMany({
    "movies.title": "Bram Stoker's Dracula"
}, {
    $set: {
        "movies.$.rating": 9
    }
})
```

Ve změně můžeme jako krok cesty v poli zadat `$[]`. Změna se pak aplikuje na každý prvek pole. Například změna

```
{
  $set: {
```

```
    "movies.$[].own": true
  }
}
```

nastaví v poli na cestě `movies` každému dokumentu položku `own` na `true`.

3 Schéma

Připomeňme si následující tabulku typů hodnot.

číslo s desetinou čárkou	`"double"`
řetězec	`"string"`
dokument	`"object"`
pole	`"array"`
logická hodnota	`"bool"`
regulární výraz	`"regex"`
celé číslo (32-bitů)	`"int"`
celé číslo (64-bitů)	`"long"`
identifikátor	`"objectId"`

Objekt s položkou `bsonType`, kde hodnota položky je typ hodnoty T , se nazývá *schéma* pro T. Schéma může obsahovat i další níže uvedené položky.

Například:

```
{ bsonType: "string" }
```

je schéma pro řetězec.

Říkáme, že hodnota *vyhovuje schématu*, jestliže splňuje všechny *požadavky*, které schéma *klade*.

Každé schéma klade na hodnotu požadavek, aby byla typu určeného položkou `bsonType`. Například:

```
"Atlas Mraků"
```

vyhovuje schématu:

```
{ bsonType: "string" }
```

Schéma pro objekt může mít položku `properties`, jejíž hodnotou je objekt P. Položky objektu P jsou opět schémata. Například:

```
{
  bsonType: "object",
  properties: {
```

```
    title: { bsonType: "string" },
    year: { bsonType: "int" },
    length: { bsonType: "int" }
  }
}
```

Schéma pro objekt s položkou `properties` rovnou P klade na objekt O požadavek, aby pro každou položku *`field`* rovnou F objektu P platilo, že pokud položka *`field`* objektu O existuje, pak vyhovuje schématu F.

Například objekty:

```
{
  title: "The Matrix",
  year: 1999,
  length: 136
}

{
   title: "Cloud Atlas",
   year: 2012
}
```

vyhovují výše uvedenému schématu.

Schéma pro objekt může mít položku `required` rovnou poli řetězců.

Schéma pro objekt s položkou `required` rovnou A klade na objekt O požadavek, aby měl všechny položky uvedené v A. Například objekty:

```
{
  title: "The Matrix",
  year: 1999,
  length: 136
}

{
   title: "Cloud Atlas",
   year: 2012
}
```

vyhovují schématu:

```
{
  bsonType: "object",
  required: [ "title", "year" ]
}
```

Pokud má schéma pro objekt položku `properties`, pak může mít logickou položku `additionalProperties`.

Schéma s nepravdivou položkou `additionalProperties` klade na objekt požadavek, aby neměl jiné položky, než ty uvedené v `properties`.

Například:

```
{
  title: 'The Avengers',
  year: 1998,
  length: 89,
  _id: ObjectId("655a3c893d08deab7dc15684")
}
```

nevyhovuje schématu:

```
{
  bsonType: "object",
  additionalProperties: false,
  properties: {
    _id: { bsonType: "objectId" },
    title: { bsonType: "string" },
    year: { bsonType: "int" }
  }
}
```

Schéma pro pole může mít položku `items`, jejíž hodnotou je opět schéma.

Schéma pro pole s položkou `items` rovnou I klade na pole požadavek, aby každý prvek pole vyhovoval schématu I.

Například pole:

```
["Francis", "Ford", "Coppola"]
```

vyhovuje schématu:

```
{
  bsonType: "array",
  items: { bsonType: "string" }
}
```

ale pole:

```
["Andrei", null, "Tarkovsky"]
```

nikoliv.

Na prvek pole můžeme klást další požadavky. Například pole:

```
[{
  title: "Solaris",
  year: 1972,
  rating: 8
}, {
  title: "Stalker",
  year: 1979,
  rating: 9
}]
```

vyhovuje schématu:

```
{
  bsonType: "array",
  items: {
    bsonType: "object",
    properties: {
      title: { bsonType: "string" },
      year: { bsonType: "int" },
      rating: { bsonType: "int" }
    }
  }
}
```

Vyhodnocením výrazu

```
db.createCollection(collection, {
  validator: {
    $jsonSchema: schema
  }
})
```

vytvoříme kolekci *collection*, kde každý dokument musí vyhovovat schématu *schema*.