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

9. Metody kolekce

verze z 13. listopadu 2023

1 Podmínky

Názvy *operátorů* začínají znakem dolaru. *Komparátory* jsou operátory určené k porovnávání hodnot.

Podmínka

```
{ comparator: argument }
```

je pro hodnotu *value* splněna, pokud *value* a *argument* jsou v relaci určené komparátorem.

Komparátor `$eq` rozhoduje, zda se hodnoty rovnají. Například hodnota 1 splňuje podmínku

```
{ $eq: 1 }
```

Další komparátory: `$gt` (větší než), `$gte` (větší nebo rovno než), `$lt` (menší než), `$lte` (menší nebo rovno než), `$ne` (nerovno).

Komparátor `$in` rozhoduje, zda se hodnota nalézá v zadaném poli, a komparátor `$nin`, zda nikoliv.

Konjunkci podmínek na hodnotu {*comparator*1: *argument*1} ... {*comparator*n: *argument*n}, kde komparátory *comparator*1, ... *comparator*n jsou po dvou různé zapíšeme podmínkou na hodnotu:

```
{comparator1: argument1, ..., comparatorn: argumentn }
```

Například konjunkce podmínek `{ $gt: 1931 }` a `{ $lt: 1992 }` je:

```
{ $gt: 1931, $lt: 1992 }
```

Tedy číslo n splňuje podmínku `{ $gt: 1931, $lt: 1992 }`, právě když $1931 < n < 1992$.

Dokument *document* splňuje podmínku

```
{ field: condition }
```

pokud položka *field* dokumentu *document* splňuje podmínku *condition*. Pokud dokument zadanou položku nemá, použije se (většinou) hodnota `null`.

Například dokument:

```
{
  _id: ObjectId("63623cfe3ee9e28d4c075d98"),
  title: 'The Conversation',
  year: 1974
}
```

splňuje podmínku:

```
{ title: { $eq: "The Conversation" } }
```

Každý dokument splňuje podmínku:

```
{}
```

Regulární výraz[1] (konkrétně PCRE[2]) zapisujeme mezi lomítka:

```
/expression/
```

Regulární výraz je hodnota. Může být tedy součástí dokumentů.

Podmínka, zda řetězec vyhovuje regulárnímu výrazu:

```
{ $regex: regular_expression }
```

Pokud *argument* není regulární výraz, pak

```
{ field: { $eq: argument } }
```

lze zkrátit na:

```
{ field: argument }
```

Podmínku:

[1] https://en.wikipedia.org/wiki/Regular_expression
[2] https://en.wikipedia.org/wiki/Perl_Compatible_Regular_Expressions

```
{ field: { $regex: regular_expression } }
```

lze zkrátit na:

```
{ field: regular_expression }
```

Dokument splňuje podmínku

```
{ field: { $exists: boolean_value } }
```

pokud má položku *field*, právě když *boolean_value* je pravda. Hodnota *boolean_value* je logickou hodnotou (`true` nebo `false`).

Hodnota splňuje podmínku

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

pokud je zadaného typu. Pokud se použije v podmínce pro dokument, tak zadaná položka musí existovat.

Typ zadáváme řetězcem podle následující tabulky.

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"`

2 Metody kolekce

Výrazy tvaru

```
db.collection.method(arg1, ..., argn)
```

nezýváme *vlolání metody* *method* kolekce *collection* s argumenty *arg*1, ..., *arg*n. Jméno *method* se nazývá metoda kolekce.

Metoda `find` tvaru

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

vrátí pole dokumentů v kolekci, které splňují podmínku.

Metoda `find` tvaru

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

vrátí pole projekcí dokumentů v kolekci, které splňují podmínku. Projekcí dokumentu rozumíme dokument, v kterém zůstanou pouze některé z původních položek dokumentu. Projekci dokumentu určuje objekt *projection*.

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*, ...

Metoda

```
db.collection.deleteMany(condition)
```

odstraní z kolekce všechny dokumenty, které splňují podmínku.

Metoda

```
db.collection.updateMany(condition, change)
```

změní všechny dokumenty, které splňují podmínku.

Změnu *change* popíšeme za použití operátoru pro aktualizaci:

```
{ operator: { field1: value1, field2: value2 , ... } }
```

Operátor `$set` nastaví hodnoty zadaných položek. Operátor `$unset` smaže zadané položky. Při mazání se místo hodnot *value1*, *value2*, ... zadají prázdné řetězce (`""`).

Identifikátor dokumentu (položka `_id`) nelze změnit.

3 Logické operátory

Podmínky lze skládat za použití logických operátorů.

Dokument splňuje podmínku

```
{ $and: [ condition1, condition2, ... ] }
```

pokud splňuje všechny podmínky *condition1*, *condition2*, ...

Pokud podmínky *condition1*, *condition2*, ... neobsahují dvě různé položky stejného názvu, pak je možné je spojit do jedné podmínky tak, že sjednotíme všechny položky. Vzniklá podmínka nazývaná *implicitní konjunkce* bude konjunkcí podmínek *condition1*, *condition2*, ... Například:

```
{ $and: [ { title: "Dracula"}, { year: 1992 } ] }
```

lze zkrátit na:

```
{ title: "Dracula", year: 1992 }
```

Dokument splňuje podmínku

```
{ $or: [ condition1, condition2, ... ] }
```

pokud splňuje aspoň jednu z podmínek *condition1*, *condition2*, ...

Hodnota splňuje podmínku

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

pokud nesplňuje podmínku *condition*.

4 Pole

Pole splňuje podmínku:

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

pokud některý jeho prvek splňuje podmínku *condition*.

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

Pole splňuje podmínku

```
{ $eq: value }
```

pokud existuje prvek pole, který splňuje podmínku. Podobně pro ostatní komparátory kromě `$ne` a `$nin`.

Tedy:

```
{ $elemMatch: { $eq: value } }
```

lze zkrátit:

```
{ $eq: value }
```

což lze zkrátit (pokud *value* není regulární výraz):

```
value
```

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*.

Pole splňuje implicitní konjunkci podmínek *condition*1, ..., *condition*n pokud splňuje každou z podmínek *condition*1, ... *condition*n.

Pole splňuje podmínku

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

pokud obsahuje všechny prvky *value*1, ..., *value*n.

Pole splňuje podmínku

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

pokud je jeho velikost *length*.

Projekce dokumentu může ponechat v položce, kde hodnota je pole, pouze některé jeho prvky.

Položka projekce může mít tvar:

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

pak položka *field* bude v projekci, pokud je hodnota *field* pole a existuje prvek splňující podmínku. V kladném případě zůstane v poli pouze první prvek splňující podmínku.

Pokud metoda `find` kolekce má zadanou podmínku na prvky pole v položce *field*, pak můžeme mít v projekci položku:

```
"field.$": 1
```

V projekci zůstane pouze první prvek pole splňující podmínku.

Například:

```
db.movies.find({
    actors: {
        $elemMatch: {
            $regex: /ary/
        }
    }
}, {
    "actors.$": 1
})
```

Změna

```
{ $push: { field : value } }
```

přidá hodnotu *value* nakonec pole v položce *field*. V případě neexistence položky, přidá položku, kde hodnota bude jednoprvkové pole obsahující *value*.

Změna

```
{ $pop: { field : 1 } }
```

odebere prvek z konce pole v zadané položce. V případě neexistence položky se žádná změna neprovede.

Změna

```
{ $pull: { field : condition } }
```

odstraní z pole v položce všechny prvky, které splňují podmínku. V případě neexistence položky se žádná změna neprovede.

Pokud v předchozích změnách není hodnota položky pole, změna končí chybou.