TRYBE
Modulo III – Back-end

Bloco 24 – Updates MongoDB

1) Updates Simples

Updates

updateOne
db.colecao.updateOne(<filtro>, <update>, <opcoes>);
Altera apenas o primeiro documento que satisfaça o critério de filtro.
Atenção, filtro vazio { } resulta em atualização do documento em coleção.

updateMany
db.colecao.updateMany(<filtro>, <update>, <opcoes>);
Permite que vários documentos que satisfaçam o critério de filtro sejam alterados de uma única vez.

A seguir, lista de operadores usados dentro do segundo parâmetro <update>:

Operador $set

Para **alterar o valor** de um campo específico.

Casos:
- Campo inexistante - $set cria;
- Campo existente - $set altera o valor;
- Campo com caminho – $set cria documentos embedados;
- Campo é array – integrar indices no $set para especificar.

```
db.products.update(
  { _id: 100 },
  { $set: {
      "tags.1": "rain gear",
      "ratings.0.rating": 2
    }
  }
);
```

Operador $mul

Para **multiplicar** os valores de campos.

```
db.products.update(
  { _id: 1 },
  { $mul: { price: NumberDecimal("1.25"), qty: 2 } }
);
```

// multiplica price por 1.25 e qty por 2

Referência sobre regras de multiplicação usando por exemplo NumberDecimal|Int|Long.

Operador $inc

Para **incrementar ou decrementar valores** em um campo específico.

```
db.increment.update(
  { sku: "abc123" },
  { $inc: { quantity: -2, "metrics.orders": 1 } }
);
```

// decrementar usando valor negativo

Operadores $min e $max

Para **alterar o valor do campo para o valor especificado** (de vários tipos possíveis) **se esse valor especificado é menor | maior do que o atual** valor do campo.

```
db.scores.update({ _id: 1 }, { $min: { lowScore: 150 } });
```

// altera valor do lowScore para 150 porque é efetivamente menor do valor atual que é 200

Operadores $currentDate

Para **incrementar ou decrementar valores** em um campo específico.

{ $currentDate: { <campo>: <typeSpecification>, ... } }
// typeSpecification sendo true para atribuir valor de data corrente formato date | { $type: "timestamp" } | { $type: "date" }

Operador $rename

Para **renomear um determinado atributo** de um ou mais documentos
{ $rename: { "formerName": "newName"} }

```
db.fruits.updateOne(
  { name: "Banana" },
  { $rename: {
      "name": "productName"
    }
  }
);
```

Operador $unset

Para **remover** um ou mais campos de um documento.

```
db.fruits.updateMany(
  { productName: "Banana" },
  { $unset: { quantity: "" } }
);
```

// remove o campo quantity onde o produto é Banana

<u>Dicas diversas</u>
upsert para inserir se não existir

2) Updates Complexos - Arrays - Parte 1

Arrays enriquecem dados e permitem criar estruturas que simulam o relacionamento 1:N.

Operador $push

Adiciona um valor a um array.

Sintaxe
{ $push: { <campo1>: <valor1>, ... } }

Modificadores
Pode ser combinado com *modificadores: $each, $slice, $sort, $position*

- `$each`: Adiciona múltiplos valores a um *array*;
- `$slice`: Limita o número de elementos do *array*. Requer o uso do modificador `$each`;
- `$sort`: Ordena os elementos do *array*. Requer o uso do modificador `$each`;
- `$position`: Especifica a posição do elemento que está sendo inserido no *array*. Também requer o modificador `$each`. Sem o modificador `$position`, o operador `$push` adiciona o elemento no final do *array*.

Ordem de realização das operações
- adicionar valor;
- sort;
- slice para discriminar onde aplica o sort;
- armazenar ordem.

```
db.supplies.updateOne(
  { _id: 1},
  { $push: {
      items: {
        $each: [
          {
            "name" : "notepad",
            "price" : 35.29,
            "quantity" : 2
          },
          {
            "name": "envelopes",
            "price": 19.95,
            "quantity": 8
          },
          {
            "name": "pens",
            "price": 56.12,
            "quantity": 5
          }
      ],
      $sort: { quantity: -1 },
      $slice: 2
    }
    }
  },
  { upsert: true }
);
```

Operador $pop

Para **remover o primeiro ou o último elemento** de um array.

```
db.supplies.updateOne({ _id: 1 }, { $pop: { items: -1 } });
```

// atribuir valor - 1 para remover primeiro item, 1 para último.

Operador $pull

Remove de um array existente todos os elementos com um ou mais valores que atendam à **condição** especificada.

```
db.supplies.updateMany(
  { },
  {
    $pull: {
      "items": {
        "name": { $in: ["pens", "envelopes"] }
      },
    }
  }
);
```

```
db.profiles.updateOne(
  { _id: 1 },
  {
    $pull: {
      votes: { $gte: 6 }
    }
  }
);
```

```
db.survey.updateMany(
  { },
  {
    $pull: {
      results: { score: 8 , item: "B" }
    }
  }
);
```

Operador $addToSet

Para **garantir que os valores de um array não sejam duplicados**.

Ou seja se não existir cria, se existir não, retorna `WriteResult({ "nMatched": 1, "nUpserted": 0, "nModified": 0 })`

(Basicamente um equivalente de um insert sem dados duplicados e especializado em arrays).

Mesma sintaxe que todos acima.

ArrayFilters

Para **filtrar por valor.**

Sabendo o indice, pode recorrer a ele:

```
db.artists.updateOne( { title: 'Panqueca Simples' }, { $set: { "ingredients.0.unit": "xícara" } } )
```

Sem saber o indice, precisa de arrayFilters:

```
db.recipes.updateMany( // Passamos de updateOne para updateMany.
  {}, // Retiramos a restrição do título.
  { $set : {
    "ingredients.$[elemento].unit": "xícara", // Setamos `unit` como "xícara",
    "ingredients.$[elemento].name": "Farinha Integral"// `name` como "Farinha Integral".
    }
  },
  { arrayFilters: [ { "elemento.name": "Farinha" } ]} // Filtramos os arrays que o valor
)
```

<u>Dicas diversas</u>

Dar pop em mais de um elemento:

```
> for (let i = 1; i <= 5; i++) {
    db.collection.updateOne({ _id:
10 }, { $pop: { array: 1 } })
  }
```

---

3) Updates Complexos - Arrays - Parte 2

Operadores mais avançados, que facilitarão muito a escrita de queries que exijam operações complexas em arrays.

Operador $all

Para filtrar documentos: seleciona todos os documentos **em que o valor do campo é um array que contenha todos os elementos especificados**, independentemente da existência de outros valores ou a ordem em que os elementos estejam.

```
db.inventory.find(
  { tags: { $all: [ "ssl", "security" ] } }
);
```

=

```
db.inventory.find(
  {
    $and: [
      { tags: "ssl" },
      { tags: "security" }
    ]
  }
);
```

Operador $elemMatch

Para filtrar documentos: seleciona os que contêm um campo do tipo array com **pelo menos um elemento que satisfaça todos os critérios de seleção especificados.**

```
db.scores.find(
  { results: { $elemMatch: { $gte: 80, $lt: 85 } } }
);
```

// seleciona documentos em que o array results contém ao menos um elemento gte 80 e lt 85

Operador $size

Para **filtrar documentos pelo tamanho de arrays**.

```
db.products.find(
  { tags: { $size: 2 } }
);
```

// retorna os campos com valor array de exatamente 2 elementos

Operador $where

Para passar uma string contendo uma expressão ou função JavaScript. Hoje substituido por $expr.

Operador $expr

Para **criar expressões de agregação** e construir **queries que comparem campos** no mesmo documento.

```
db.monthlyBudget.find(
  {
    $expr: { $gt: [ "$spent", "$budget" ] }
  }
);
```

// busca os documentos em que o valor de spent exceda o valor de budget

Operador $regex

Fornece os **"poderes" das expressões regulares (regular expressions) para seleção de strings**.

```
db.products.find({ sku: { $regex: /789$/ } });
```

// seleciona documentos em que o campo sku termine com "789"
// adicionar /i para que seja case insensitive

Operador $text

Faz uma **busca "textual" em um campo indexado por um text index** (índice textual).

Sintaxe

```
{
  $text:
    {
      $search: <string>,
      $language: <string>,
      $caseSensitive: <boolean>,
      $diacriticSensitive: <boolean>
    }
}
```

// $language (stop words para tokenizar busca), $caseSensitive and $diacriticSensitive opcionais.
// Por padrão, $test não retorna os resultados ordenados pelas pontuações (score atribuido a cada documento que tenha o termo procurado no campo).

Como usar

- Primeiro criar index de tipo text

```
db.articles.createIndex({ subject: "text" });
```

```
db.produtos.createIndex(
  { campo: "text" },
  { default_language: "portuguese" }
);
```

- Segundo escrever a query

```
db.articles.find({ $text: { $search: "bake coffee cake" } });
```
// mais de uma palavra com espaço

```
db.articles.find({ $text: { $search: "\"coffee shop\"" } });
```
// frase com \"

```
db.produtos.count({ $text: { $search: "\"feito com\"" } });
```

Operador $mod

Executa a **operação módulo**: seleciona todos os documentos em que o **valor do campo dividido por um divisor seja igual ao valor especificado.**

$mod: [divisor, resto]

```
db.inventory.find({ qty: { $mod: [4, 0] } });
```
//seleciona docs em que o valor qty modulo 4 = 0.

Dicas diversas

FindOne => HOF find
FindMany => HOF Filter
Text => LIKE do SQL versão 2.0