MongoDB

Aula 4:
Operadores de UPDATE

Banco de Dados 2 (BD2)

Instituto J&F Germinare TECH

NoSQL

Até agora já aprendemos as principais funções para operações de inserção, atualização, leitura e exclusão.

Também já aprendemos os principais operadores de consulta, para usar em filtros dentro de um **find().**

MongoDB – Update

Além de consultar algo a partir de filtros e operadores, será muito comum realizar operações de atualizações dos dados, certo? Já sabemos que podemos usar as funções:

- **db.collection.updateOne();**
- **db.collection.updateMany();**
- **db.collection.findAndUpdate**().

Operadores de Update

O MongoDB fornece diferentes tipos de operadores de atualização de campo para atualizar os valores dos campos dos documentos que correspondem à condição especificada. A tabela a seguir contém os operadores de atualização de campo:

São muitos então você pode ir na documentação:

https://www.mongodb.com/docs/manual/reference/operator/update/

Vamos criar o banco de dbBank

Crie um banco chamado dbBank e copie o seguinte script para criar a collection clientes:

O script está na aula.

Exibindo campos no .find()

Assim como no SQL, a gente pode querer mostrar apenas alguns campos em um filtro, para isso basta adicionar um documento {} depois do filtro no find(), mas nesse documento, você precisa especificar o nome do campo e definir se ele vai aparecer (1) ou não (0)

Mostrando apenas os nomes de todos os clientes.

```
dbBank> db.clientes.find({},{nome:1})
[
  { _id: ObjectId("6464d2f6c2d23bb5609bf29b"), nome: 'Jucelio' },
  { _id: ObjectId("6464d2f6c2d23bb5609bf29c"), nome: 'Lamelo' },
  { _id: ObjectId("6464d2f6c2d23bb5609bf29d"), nome: 'Juliano' },
  { _id: ObjectId("6464d2f6c2d23bb5609bf29e"), nome: 'Emiliano' },
  { _id: ObjectId("6464d2f6c2d23bb5609bf29f"), nome: 'Iago' },
  { _id: ObjectId("6464d2f6c2d23bb5609bf2a0"), nome: 'Verônica' }
]
```

Exibindo campos no .find()

Mostrando o nome e os endereços dos clientes que tenham empréstimo:

```
dbBank> db.clientes.find({emprestimo_disponivel:true},{"endereço":1,nome:1})
[
  {
    _id: ObjectId('65e4a9e74dd20c9a1dd1240e'),
    nome: 'Jucelio',
    'endereço': { cidade: 'São Paulo', estado: 'SP', cep: '01001-50' }
  },
  {
    _id: ObjectId('65e4a9e74dd20c9a1dd1240f'),
    nome: 'Lamelo',
    'endereço': { cidade: 'São Paulo', estado: 'SP', cep: '01002-10' }
  },
  {
    _id: ObjectId('65e4a9e74dd20c9a1dd12410'),
    nome: 'Juliano',
    'endereço': { cidade: 'Rio de Janeiro', estado: 'SP', cep: '05100-70' }
  },
  {
    _id: ObjectId('65e4a9e74dd20c9a1dd12412'),
    nome: 'Iago',
    'endereço': { cidade: 'Campinas', estado: 'SP', cep: '88098-11' }
  }
]
```

Use aspas duplas quando o campo tem caractere especial

Exibindo campos no .find()

Verificando todas as informações dos clientes, menos as de empréstimo:

```
db.clientes.find({},{emprestimo_tipo:0, emprestimo_disponivel:0})
```

Em resumo, quando quer exibir apenas alguns campos em uma busca, basta adicionar depois do filtro um documento informando o nome do campo e valor 0 ou 1:

- 0 para não exibir
- 1 para exibir.

O padrão é 1 para todos os campos. Se você colocar apenas 1 campo como 1, ele modifica toda a regra e exibe apenas ele, se você coloca apenas um campo como 0 ele apenas altera a regra para aquele campo.

Cuidado!

Observe a query e o erro retornado no MongoDB.

```
dbBank> db.clientes.find({emprestimo_disponivel:true},{nome:0, sobrenome:1})
MongoServerError: Cannot do inclusion on field sobrenome in exclusion projection
```

O erro ocorre porque na query está pedindo a projeção do campo "sobrenome" (que está definido para inclusão) enquanto também está excluindo o campo "nome". No MongoDB, a projeção de campos deve ser consistente, **o que significa que você só pode incluir ou excluir campos em uma única projeção.**

Portanto, para corrigir esse erro, tem que excluir o campo "nome" da projeção, não pode incluir o campo "sobrenome" simultaneamente. Teria que, simplesmente remover a exclusão do campo "nome" da consulta, ou remover a inclusão do campo "sobrenome".

$set

Esse a gente já sabe, mas custa nada lembrar que se quer adicionar um novo campo, usa o $set junto com o update().

O Cliente Jucelio Gomes foi contemplado com um empréstimo, então, primeiro vamos mudar o valor do campo **emprestimo_disponivel** para true.

```
dbBank> db.clientes.updateOne({nome:"Jucelio"}, {$set:{emprestimo_disponivel:true}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

$set

Sabemos que o $set também é usado para adicionar um campo, então, já que ele tem empréstimo vamos adicionar a lista de opções que ele tem direito:

emprestimo_tipo: [“FGTS”, “Garantia”]

```
dbBank> db.clientes.updateOne({nome:"Jucelio"}, {$set:{emprestimo_tipo:["FGTS", "Garantia"]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

$set

Agora, o banco mudou a seguinte regra, quem é de fora de SP vai ficar na agência 0002.

Mude isso, lembrando que a agencia é um campo com valor STRING. Tente praticar usando o **typeof** para verificar o tipo de dado desse campo.

Mas antes, percebeu que o estado da cidade Rio de Janeiro está errado? Corrija primeiro.

```
dbBank> db.clientes.updateMany({"endereço.cidade":"Rio de Janeiro"},{$set:{"endereço.estado":"RJ"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

$set

Agora sim, vamos mudar o valor do campo buscando pelo estado.

Agora, o banco mudou a seguinte regra, quem é de fora de SP vai ficar na agência 0002.

```
dbBank> db.clientes.updateMany({"endereço.estado":"RJ"},{$set:{agencia:"0002"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

O que temos até então!

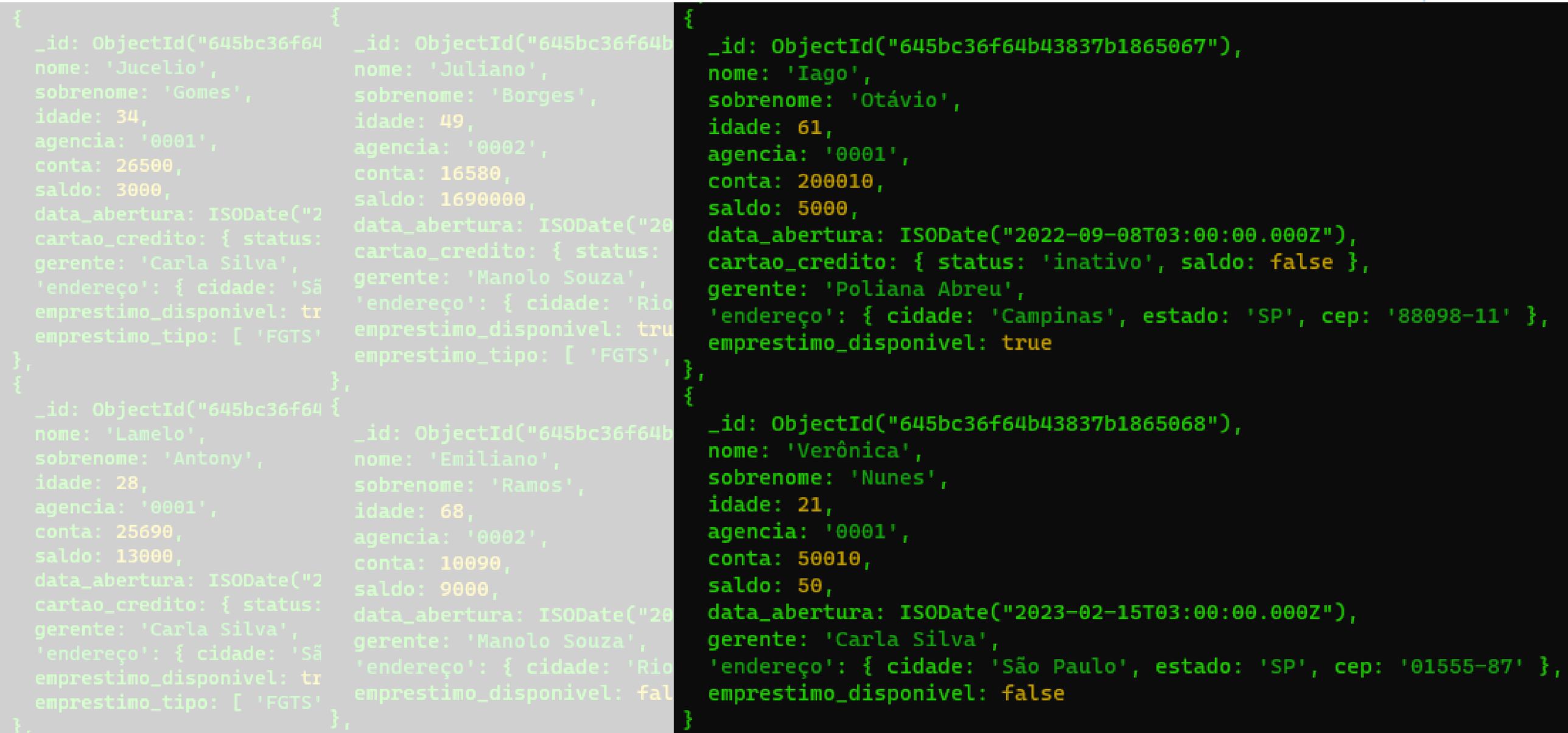
```
{
  _id: ObjectId("645bc36f64b43837b1865063"),
  nome: 'Jucelio',
  sobrenome: 'Gomes',
  idade: 34,
  agencia: '0001',
  conta: 26500,
  saldo: 3000,
  data_abertura: ISODate("2022-02-10T03:00:00.000Z"),
  cartao_credito: { status: 'ativo', saldo: 2000 },
  gerente: 'Carla Silva',
  'endereço': { cidade: 'São Paulo', estado: 'SP', cep: '01001-50' },
  emprestimo_disponivel: true,
  emprestimo_tipo: [ 'FGTS', 'Garantia' ]
},
{
  _id: ObjectId("645bc36f64b43837b1865064"),
  nome: 'Lamelo',
  sobrenome: 'Antony',
  idade: 28,
  agencia: '0001',
  conta: 25690,
  saldo: 13000,
  data_abertura: ISODate("2021-02-15T03:00:00.000Z"),
  cartao_credito: { status: 'ativo', saldo: 50000 },
  gerente: 'Carla Silva',
  'endereço': { cidade: 'São Paulo', estado: 'SP', cep: '01002-10' },
  emprestimo_disponivel: true,
  emprestimo_tipo: [ 'FGTS', 'Pessoal', 'Garantia', 'Consignado' ]
},
```

O que temos até então!

```
{
  _id: ObjectId("645bc36f64
  nome: 'Jucelio',
  sobrenome: 'Gomes',
  idade: 34,
  agencia: '0001',
  conta: 26500,
  saldo: 3000,
  data_abertura: ISODate("2
  cartao_credito: { status:
  gerente: 'Carla Silva',
  'endereço': { cidade: 'Sã
  emprestimo_disponivel: tr
  emprestimo_tipo: [ 'FGTS'
},
{
  _id: ObjectId("645bc36f64
  nome: 'Lamelo',
  sobrenome: 'Antony',
  idade: 28,
  agencia: '0001',
  conta: 25690,
  saldo: 13000,
  data_abertura: ISODate("2
  cartao_credito: { status:
  gerente: 'Carla Silva',
  'endereço': { cidade: 'Sã
  emprestimo_disponivel: tr
  emprestimo_tipo: [ 'FGTS'
},
```

```
{
  _id: ObjectId("645bc36f64b43837b1865065"),
  nome: 'Juliano',
  sobrenome: 'Borges',
  idade: 49,
  agencia: '0002',
  conta: 16580,
  saldo: 1690000,
  data_abertura: ISODate("2019-02-10T02:00:00.000Z"),
  cartao_credito: { status: 'ativo', saldo: 10000 },
  gerente: 'Manolo Souza',
  'endereço': { cidade: 'Rio de Janeiro', estado: 'RJ', cep: '05100-70' },
  emprestimo_disponivel: true,
  emprestimo_tipo: [ 'FGTS', 'Pessoal', 'Garantia' ]
},
{
  _id: ObjectId("645bc36f64b43837b1865066"),
  nome: 'Emiliano',
  sobrenome: 'Ramos',
  idade: 68,
  agencia: '0002',
  conta: 10090,
  saldo: 9000,
  data_abertura: ISODate("2020-07-05T03:00:00.000Z"),
  gerente: 'Manolo Souza',
  'endereço': { cidade: 'Rio de Janeiro', estado: 'RJ', cep: '05890-06' },
  emprestimo_disponivel: false
},
```

O que temos até então!



Removendo campo

Tem um cliente que ele está com o status do cartão como “inativo”, se isso acontecer é para o campo **cartao_credito** ser removido do documento.

Primeiro, procure com o **find()** o usuário que tem o status diferente de “ativo”, mas mostre apenas, aqueles que o campo **cartao_credito** existe.

```
dbBank> db.clientes.find({$and:[{"cartao_credito.status":{$ne:"ativo"}},{cartao_credito:{$exists:true}}]})
[
  {
    _id: ObjectId("645a41fb9acacf22df1ed814"),
    nome: 'Iago',
    sobrenome: 'Otávio',
    idade: 61,
    agencia: '0001',
    conta: 200010,
    saldo: 5000,
    data_abertura: ISODate("2022-09-08T03:00:00.000Z"),
    cartao_credito: { status: 'inativo', saldo: false },
    gerente: 'Poliana Abreu',
    'endereço': { cidade: 'Campinas', estado: 'SP', cep: '88098-11' },
    emprestimo_disponivel: true
  }
]
```

$unset

Para excluir um campo usamos o $unset.

Agora vamos usar a query anterior no filtro do updateOne(), mas usar o $unset para remover esse campo desse documento.

```
updateOne({$and:[{"cartao_credito.status":{$ne:"ativo"}},{cartao_credito:{$exists:true}}]},{$unset:{cartao_credito:""}})

 acknowledged: true,
 insertedId: null,
 matchedCount: 1,
 modifiedCount: 1,
 upsertedCount: 0

find({$and:[{"cartao_credito.status":{$ne:"ativo"}},{cartao_credito:{$exists:true}}]})
```

O valor do campo é uma aspas vazias mesmo, tanto faz o valor que contenha e o tipo.

Query:

updateOne({filtro}, {$unset:{nome_campo:””}})

$setOnInsert

Este operador é usado quando um novo documento é inserido com a ajuda da operação de atualização, definindo o valor de um campo **upsert:true**, então o $setOnInsert atribui o valor especificado aos campos no documento.

O operador $setOnInsert também pode trabalhar com documentos embutidos / aninhados. Você pode usar esse operador em métodos como update(), findAndUpdate().

$set → Atualiza um campo independentemente de ser um documento novo ou existente.
$setOnInsert → Define um campo somente se o documento for inserido (ou seja, se ele não existia antes).

$setOnInsert

Qual a finalidade dele? Você quer atualizar algo, se ele encontrar, ele não cria, se não encontrar então ele cria um novo documento.

EX: O que vai acontecer com esse comando?

```
e({nome:"Iago"},{$setOnInsert:{teste:"teste", outro:"teste2"}}, {upsert:true})
```

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

$setOnInsert

O comando anterior não muda nada. Agora observe o que vai acontecer agora.

```
dbBank> db.clientes.updateOne({nome:"Outro"},{$setOnInsert:{teste:"teste", outro:"teste2"}}, {upsert:true})
{
  acknowledged: true,
  insertedId: ObjectId("645a4b7f9cbbd288a1eebd11"),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

Ele não encontrou o nome:”outro” então adicionou um novo documento com essas informações.

```
dbBank> db.clientes.find({outro:{$exists:true}})
[
  {
    _id: ObjectId("645a4b7f9cbbd288a1eebd11"),
    nome: 'Outro',
    outro: 'teste2',
    teste: 'teste'
  }
]
```

Exclua esse documento!

findOneAndDelete()

Não existe mais!

```
dbBank> db.clientes.findOneAndDelete({outro:{$exists:true}})
{
  _id: ObjectId("645a4b7f9cbbd288a1eebd11"),
  nome: 'Outro',
  outro: 'teste2',
  teste: 'teste'
}
dbBank> db.clientes.find({outro:{$exists:true}})

dbBank>
```

$inc

Este operador é responsável por incrementar ou decrementar um número ao campo (para decrementar só usar números negativos). Obvio, só vale para dados numéricos.

Digamos que todos os cliente que tenham cartão de credito recebam um aumento de 100 reais de uma promoção do banco.. Para isso faremos um incremento no saldo:

```
dbBank> db.clientes.updateMany({cartao_credito:{$exists:true}},{$inc:{saldo:100}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

Query:

updateMany({filtro}, {$inc:{campo : valor_incremento}})

$min e $max

Atualiza o campo apenas se o valor especificado for **menor(min) ou maior(max)** que o valor do campo existente.

$min irá comparar os valores de diferentes tipos de dados de acordo com a ordem de comparação BSON. Se o campo fornecido não existir, este operador criará o campo e definirá o valor desse campo.

Atualizar o valor do saldo do cliente Iago, para no mínimo R$100,00.

```
dbBank> db.clientes.updateOne({nome:"Iago"},{$min:{saldo:100}})
```

A query acima funciona, mas a de baixo não ira mais atualizar nada, sabe porque?

```
dbBank> db.clientes.updateOne({nome:"Iago"},{$min:{saldo:150}})
```

$mul

Atualiza o campo baseado em uma operação de multiplicação.

Os clientes que tenham mais do que 60 anos irão receber 10% de bônus no saldo de sua conta. Torne possível!

```
dbBank> db.clientes.updateMany({idade:{$gt:60}},{$mul:{saldo:1.1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

$rename

Usado para renomear um nome de campo.

Por exemplo, o campo endereço foi cadastrado com Ç, vamos mudar o nome para **endereco** (sem o Ç). Façamos isso para todos os documentos.

```
dbBank> db.clientes.updateMany({},{$rename:{"endereço":"endereco"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 6,
  modifiedCount: 6,
  upsertedCount: 0
}
```

Query:

updateMany({filtro}, {$rename:{*nome_antigo*: *novo_nome*}})

$rename

Volte no dbGerminare e vamos ver a Juliana:

```
dbGerminare> db.alunos.find({nome:"Juliana"})
[
  {
    _id: ObjectId('67a24ece5fa9037b8ccb0ce4'),
    nome: 'Juliana',
    sobrenome: 'Lima',
    idade: 16,
    matricula: 1019,
    media: 6,
    data_matricula: ISODate('2024-01-24T03:00:00.000Z'),
    estagio: true,
    escola: {
      nome: 'Germinare Tech',
      diretor: 'João Pilla',
      qnd_professores: 7
    }
```

$rename

Primeiro apenas nesse documento, seria possível mudar tento o nome do campo **escola** para **escolas** e o **qnd_professores** para **qnt_professores** em uma única query? Apenas para a Juliana primeiro.

```
dbGerminare> db.alunos.updateOne({nome:"Juliana"}, {$rename:{"escola.qnd_professores":"escolas.qnt_professores"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Atualizou algo, eba!

$rename

Deu certo?

Não, foi criado um novo campo e subcampo, mas ao mesmo tempo o qnd_professores não existe mais em escola. CUIDADO!

```
dbGerminare> db.alunos.find({nome:"Juliana"})
[
  {
    _id: ObjectId('67a24ece5fa9037b8ccb0ce4'),
    nome: 'Juliana',
    sobrenome: 'Lima',
    idade: 16,
    matricula: 1019,
    media: 6,
    data_matricula: ISODate('2024-01-24T03:00:00.000Z'),
    estagio: true,
    escola: { nome: 'Germinare Tech', diretor: 'João Pilla' },
    telefone: 11911223344,
    disciplinas: [
      'Lógica de Programação',
      'Análise de Dados',
      'Desenvolvimento II'
    ],
    escolas: { qnt_professores: 7 }
  }
]
```

$rename

O recomendado é atualizar os campos separadamente, sempre renomeando primeiro o subcampo, depois renomeando o campo.

```
dateOne({nome:"Juliana"}, {$rename:{"escola.nome":"escola.nomes"}})
```

```
dateOne({nome:"Juliana"}, {$rename:{"escola":"escolas"}})
```

```
    ],
    escolas: { diretor: 'João Pilla', nomes: 'Germinare Tech' }
  }
]
```

$rename

Mas é possível atualizar 2 campos ao mesmo tempo com **$rename** sim:

```
dbGerminare> {
...       $rename: {
...          "antigoCampo1": "novoCampo1",
...          "antigoCampo2": "novoCampo2"
...       }
```

Trabalhando com arrays

Já vimos que temos como fazer consultas dentro de arrays, mas também temos operadores para realizar modificações em arrays. Bem parecido com os comandos de Javascript e Python. Vamos aprender alguns como:

$push, $pull, $pop, $pullAll....

$pull

Este operador é responsável por remover OS ITENS em um array.

Vamos remover o empréstimo **Consignado** para o Lamelo Antony.

```
dbBank> db.clientes.updateOne({nome:"Lamelo"},{$pull:{emprestimo_tipo:"Consignado"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

$push

Este operador é responsável por adicionar UM novo item a um array.

Agora vamos adicionar o empréstimo **Pessoal** para o Jucelio.

```
dbBank> db.clientes.updateOne({nome:"Jucelio"},{$push:{emprestimo_tipo:"Pessoal"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

$push e $pull - cuidados

Se você usar o $push para adicionar um item que já existe, ele vai ficar duplicado, por exemplo. Adicionar empréstimo Pessoal para o Lamelo....

```
dbBank> db.clientes.updateOne({nome:"Lamelo"},{$push:{emprestimo_tipo:"Pessoal"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId("645a41fb9acacf22df1ed811"),
  nome: 'Lamelo',
  sobrenome: 'Antony',
  idade: 28,
  agencia: '0001',
  conta: 25690,
  saldo: 13100,
  data_abertura: ISODate("2021-02-15T03:00:00.000Z"),
  cartao_credito: { status: 'ativo', saldo: 50000 },
  gerente: 'Carla Silva',
  'endereço': { cidade: 'São Paulo', estado: 'SP', cep: '01002-10' },
  emprestimo_disponivel: true,
  emprestimo_tipo: [ 'FGTS', 'Pessoal', 'Garantia', 'Pessoal' ]
},
```

$push e $pull - cuidados

Ah, vamos corrigir usando o $pull, vai remover apenas o último Pessoal. SQN!

```
dbBank> db.clientes.updateOne({nome:"Lamelo"},{$pull:{emprestimo_tipo:"Pessoal"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId("645a41fb9acacf22df1ed811"),
  nome: 'Lamelo',
  sobrenome: 'Antony',
  idade: 28,
  agencia: '0001',
  conta: 25690,
  saldo: 13100,
  data_abertura: ISODate("2021-02-15T03:00:00.000Z"),
  cartao_credito: { status: 'ativo', saldo: 50000 },
  gerente: 'Carla Silva',
  'endereço': { cidade: 'São Paulo', estado: 'SP', cep: '01002-10' },
  emprestimo_disponivel: true,
  emprestimo_tipo: [ 'FGTS', 'Garantia' ]
},
```

$addToSet

Adiciona elementos a uma matriz **somente** se eles ainda não existirem no conjunto.

Se tentar adicionar o emprestimo “FGTS” para o Jucelio, usando o $addToSet:

```
dbBank> db.clientes.updateOne({nome:"Jucelio"},{$addToSet:{emprestimo_tipo:"FGTS"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
dbBank> db.clientes.updateOne({nome:"Jucelio"},{$push:{emprestimo_tipo:"FGTS"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Viu a diferença? O de cima não foi consolidado, o de baixo foi.

$pop

Remove o **primeiro** OU o **último** item de uma matriz. Isso mesmo, um ou outro, vamos remover

Vamos remover a última posição do array de **emprestimo_tipo** do cliente Jucelio, já que está repetida.

```
dbBank> db.clientes.updateOne({nome:"Jucelio"},{$pop:{emprestimo_tipo:1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Query:

Primeiro -> updateOne({filtro}**,{$pop:{**emprestimo_tipo: **-1}})**

Ultimo -> updateOne({filtro}**,{$pop:{**emprestimo_tipo: **1}})**

$addToSet

Juliano tem os seguintes empréstimo:

```
dbBank> db.clientes.find({nome:"Juliano"})
[
  {
    _id: ObjectId("645bc36f64b43837b1865065"),
    nome: 'Juliano',
    sobrenome: 'Borges',
    idade: 49,
    agencia: '0002',
    conta: 16580,
    saldo: 1690100,
    data_abertura: ISODate("2019-02-10T02:00:00.000Z"),
    cartao_credito: { status: 'ativo', saldo: 10000 },
    gerente: 'Manolo Souza',
    emprestimo_disponivel: true,
    emprestimo_tipo: [ 'FGTS', 'Pessoal', 'Garantia' ],
    endereco: { cidade: 'Rio de Janeiro', estado: 'RJ', cep: '05100-70' }
  }
]
```

Se tentar adicionar agora o empréstimo Consignado para ele? Usando o $addToSet, vai funcionar?

```
dbBank> db.clientes.updateOne({nome:"Juliano"},{$addToSet:{emprestimo_tipo:"Consignado"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

$pullAll

o operador $pullAll remove todas as instâncias dos valores especificados de uma matriz existente. Ao contrário do operador $pull que remove elementos especificando uma consulta, $pullAll remove elementos que correspondem aos valores listados.

```
dbBank> db.clientes.insertOne( { _id: 1, scores: [ 0, 2, 5, 5, 1, 0 ] } )
{ acknowledged: true, insertedId: 1 }
dbBank> db.clientes.find({_id:1}
... )
[ { _id: 1, scores: [ 0, 2, 5, 5, 1, 0 ] } ]
dbBank> db.clientes.updateOne( { _id: 1 }, { $pullAll: { scores: [ 0, 5 ] } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.clientes.find({ _id: 1 })
[ { _id: 1, scores: [ 2, 1 ] } ]
```

Você pode até usar o $pull com o operador $in, vai ser quase igual.

O que acontece?

Queremos que o Lamelo tenha todos os empréstimos, então, usando o $push, adicione os que faltam para ele. O que aconteceu?

```
dbBank> db.clientes.updateOne({nome:"Lamelo"},{$push:{emprestimo_tipo:["Consignado","Pessoal"]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.clientes.find({nome:"Lamelo"})
[
  {
    _id: ObjectId("6464d2f6c2d23bb5609bf29c"),
    nome: 'Lamelo',
    sobrenome: 'Antony',
    idade: 28,
    agencia: '0001',
    conta: 25690,
    saldo: 13100,
    data_abertura: ISODate("2021-02-15T03:00:00.000Z"),
    cartao_credito: { status: 'ativo', saldo: 50000 },
    gerente: 'Carla Silva',
    emprestimo_disponivel: true,
    emprestimo_tipo: [ 'FGTS', 'Garantia', [ 'Consignado', 'Pessoal' ] ],
    endereco: { cidade: 'São Paulo', estado: 'SP', cep: '01002-10' }
  }
]
```

Delete o array do array

Delete o que foi adicionado anteriormente [“Consignado”, “Pessoal”]

Mude o $push pelo $pull, apenas isso!

$each

Pode ser usado em conjunto com os operadores $push e $addToSet para anexar vários itens para atualizações de um array.

Vamos adicionar todos os tipos de empréstimos para o cliente Lamelo, ele tem apenas FGTS e Garantia, as opções disponíveis são: [“FGTS”, “Consignado”, “Pessoal”, “Garantia”]. OBS: Mas não pode inserir repetidos...

```
dbBank> db.clientes.updateOne({nome:"Lamelo"},{$addToSet:{emprestimo_tipo:{$each:["FGTS","Consignado","Pessoal","Garantia"]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Query:

updateOne({filtro**},{$addToSet:{**campo: { **$each**:[valor1, valor2] } **}})**

Nova collection para ordenação!

Vamos criar essa nova collection chamada lista para aprender os últimos operadores de arrays.

```
dbBank> db.lista.insertOne({_id:1, vetor_num:[5, 8, 11, 15, 9], vetor_nome:["Maria", "Carlos", "Victor", "Hugo", "Yuri"]})
{ acknowledged: true, insertedId: 1 }
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [ 5, 8, 11, 15, 9 ],
    vetor_nome: [ 'Maria', 'Carlos', 'Victor', 'Hugo', 'Yuri' ]
  }
]
```

Inserindo com o $push e $each

O que acontece se a gente inserir os valores [10, 3] usando a sintaxe normal?

Vai inserir no final!

```
dbBank> db.lista.updateOne({_id:1},{$push:{vetor_num:{$each:[10,3]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [
      5,  8, 11, 15,
      9, 10,  3
    ],
    vetor_nome: [ 'Maria', 'Carlos', 'Victor', 'Hugor', 'Yuri' ]
  }
]
```

Delete esses valores usando o $pullAll

O que acontece se a gente inserir os valores [10, 3] usando a sintaxe normal?

```
dbBank> db.lista.updateOne({_id:1},{$pullAll:{vetor_num:[10,3]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [ 5, 8, 11, 15, 9 ],
    vetor_nome: [ 'Maria', 'Carlos', 'Victor', 'Hugor', 'Yuri' ]
  }
]
```

$position

O modificador $position especifica o local do array em que o operador $push insere elementos. Sem o modificador $position, o operador $push insere elementos no final do array. Para mais de um valor, pode usar o $each.

O comando abaixo irá adicionar os valores **10 e 3** no começo do campo **vetor_num:**

```
dbBank> db.lista.updateOne({_id:1},{$push:{vetor_num:{$each:[10,3], $position:0}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [
      10,  3, 5, 8,
      11, 15, 9
    ],
    vetor_nome: [ 'Maria', 'Carlos', 'Victor', 'Hugo', 'Yuri' ]
  }
]
```

$position

Adicione os nomes “Gabriela” e “Anderson” entre os nomes “Carlos” e “Victor”.

```
dbBank> db.lista.updateOne({_id:1},{$push:{vetor_nome:{$each:["Gabriela","Anderson"], $position:2}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [
      10,  3, 5, 8,
      11, 15, 9
    ],
    vetor_nome: [
      'Maria',    'Carlos',
      'Gabriela', 'Anderson',
      'Victor',   'Hugo',
      'Yuri'
    ]
  }
]
```

$slice

O $slice limita o número de elementos do array durante uma operação $push. Deve aparecer com o modificador $each. Você pode passar um array vazio [] para o modificador $each de forma que apenas o modificador $slice tenha efeito.

```
dbBank> db.lista.updateOne({_id:1},{$push:{vetor_num:{$each:[100,200], $slice:-5}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [ 11, 15, 9, 100, 200 ],
    vetor_nome: [
      'Maria',    'Carlos',
      'Gabriela', 'Anderson',
      'Victor',   'Hugo',
      'Yuri'
    ]
  }
]
```

Se adicionar como valor do $slice um número **negativo**, vai pegar os últimos elementos do array, se passar um número **positivo**, vai pegar os primeiros elementos do array.

$slice

Para deixar apenas os 3 primeiros nomes no campo **vetor_nome:**

```
dbBank> db.lista.updateOne({_id:1},{$push:{vetor_nome:{$each:[], $slice:3}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [ 11, 15, 9, 100, 200 ],
    vetor_nome: [ 'Maria', 'Carlos', 'Gabriela' ]
  }
]
```

Você pode usar o $position e o $slice juntos!

$sort

O modificador $sort ordena os elementos de um array durante uma operação $push. Para usar o modificador $sort, ele deve aparecer com o modificador $each. Você pode passar um array vazio [] para o modificador $each de forma que apenas o modificador $sort tenha efeito.

Vamos inserir os nomes “Zacarias”, “Andre” e “Bruno” em **vetor_nome**, mas mostrar esse array ordenado.

$sort

Vamos inserir os nomes “Zacarias”, “Andre” e “Bruno” em **vetor_nome**, mas mostrar esse array ordenado.

```
dbBank> db.lista.updateOne({_id:1},{$push:{vetor_nome:{$each:["Zacarias", "Andre", "Bruno"],$sort:1}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
dbBank> db.lista.find()
[
  {
    _id: 1,
    vetor_num: [ 11, 15, 9, 100, 200 ],
    vetor_nome: [ 'Andre', 'Bruno', 'Carlos', 'Gabriela', 'Maria', 'Zacarias' ]
  }
]
```

$sort

Ordene em todos os clientes para que os empréstimos apareçam sempre na ordem alfabética:

Cuidado!

Se você usou o filtro vazio, com a query abaixo o que aconteceu?

```
dbBank> db.clientes.updateMany({},{$push:{emprestimo_tipo:{$each:[], $sort:1}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 6,
  modifiedCount: 6,
  upsertedCount: 0
}
```

Corrigindo!

Ele criou o campo vazio nos que não tinham o campo, então para isso, corrija com o comando abaixo:

```
dbBank> db.clientes.updateMany({emprestimo_tipo:[]},{$unset:{emprestimo_tipo:""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

$sort

Ordene em todos os clientes para que os empréstimos apareçam sempre na ordem alfabética:

```
dbBank> db.clientes.updateMany({emprestimo_tipo:{$exists:true}},{$push:{emprestimo_tipo:{$each:[], $sort:1}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

$sort *vs* .sort()

O operador $sort é diferente da função **.sort()** que ordena a apresentação dos documentos em uma busca com o **find().**

Ordenando a saída por ordem alfabética (a -> z):

```
dbBank> db.clientes.find().sort({nome:1})
```

Ordenando a saída pelo saldo (menor -> maior):

```
dbBank> db.clientes.find().sort({saldo:1})
```

Ordenando a saída pela data de abertura da conta (mais recendo -> mais antigo):

```
dbBank> db.clientes.find().sort({data_abertura:-1})
```

Desafios:

O banco vai aumentar o limite do cartão de crédito de alguns clientes em 65%, mas como decidir esses clientes? Serão todos os clientes que tenham mais do que 10.000,00 de saldo, que tenham pelo menos 2 anos de conta no banco e que tenham cartão de crédito (claro), todos esses devem ter o reajuste no limite do cartão.

Desafios:

Os clientes com saldo acima de R$ 10.000,00, não terão mais acesso ao empréstimo "FGTS“, mas obrigatoriamente devem ter o empréstimo “Pessoal”, ou seja, o campo emprestimo_tipo deve ser inserido se ainda não existir, mas não pode ser duplicado.

Desafios:

Todos os clientes que tem **emprestimo_disponivel** com valor verdadeiro devem possuir, no mínimo, acesso ao empréstimo FGTS. Isso já acontece? Corrija sem permitir duplicada de empréstimos.

Depois que fizer essa alteração, verifique quantos clientes possuem o empréstimo por Garantia **e são do gerente que tenha o nome** Manolo, **independente do sobrenome.**

Desafios:

Todos os clientes que tem **emprestimo_disponivel** com valor verdadeiro devem possuir, no mínimo, acesso ao empréstimo FGTS. Isso já acontece? Corrija sem permitir duplicada de empréstimos.

Mostre os campos, emprestimo_disponivel**,** emprestimo_tipo **e** nome **de todos os clientes por ordem alfabética.**

Desafios:

Depois que fizer essa alteração, verifique quantos clientes possuem o empréstimo por Garantia **e são do gerente que tenha o nome** Manolo, **independente do sobrenome.**

Quem são?

Desafios:

O Jucélio mudou de endereço, para uma nova cidade e estado (Teresópolis, RJ). Atualize não apenas esses campos, mas também a agencia (verifique a agencia de quem é de RJ) desse cliente. Na mesma query, insira um novo campo **data_ultima_atualizacao** com a data e hora atuais.

Desafios:

O que essa query abaixo vai fazer?

```
dbBank> db.clientes.updateOne( {'cartao_credito': { $exists: false } }, { $push: { cartao_credito: { status: 'ativo',
 saldo: 3000 } } } )
```

1. **Vai da erro?**
2. **Vai criar um novo campo, que vai ser um documento como valor? O documento vai ter 2 campos {status:** e **saldo:}**
3. **Vai criar um novo campo, que vai ser um array como valor? O array vai ter 2 valores [“ativo”, 3000]**
4. **Não vai da erro, mas também não vai fazer nada!**

Desafios:

Atualize o saldo de "Lamelo Antony" para ser igual ao saldo do cliente com o menor saldo. Use uma consulta para encontrar o cliente com o menor saldo e, em seguida, atualize o saldo de "Lamelo Antony" com esse valor.

Como fazer uma query para saber o cliente com menor saldo?

Desafios:

Para clientes que possuem mais de 3 tipos de empréstimos, remover o empréstimo de menor prioridade

- Considere a ordem de prioridade: **Consignado** < **FGTS** < **Garantia** < **Pessoal**.
  - *Percebeu a ordem de prioridade?*

Desafios:

Se o saldo de "Juliano Borges" for maior que a média de saldo de todos os clientes, adicione 10% ao saldo atual. Caso contrário, subtraia 5% do saldo atual. Utilize o operador de agregação e o operador de atualização apropriados para resolver este desafio.

$Operadores aprendidos

Operador	Função	EX
$set	Atualizar ou adicionar um campo	{{**$set**: {campo: valor} }
$unset	Usado para remover um campo	{{**$unset**: {campo: “” } }
$setOnInsert	Adicionar um novo documento	{{**$setOnInsert**: <novo_doc>}, {**$upsert**: true} }
$inc	Atualiza um campo incrementando algo em um valor numérico	{**$inc**: {campo: **valor_incremento**} }
$min $max	Atualiza valores que tenham um target máximo ou mínimo	{**$min**: {campo: **valor_min**} }
$mul	Multiplica o valor de um campo numérico	{**$mul**: {vampos: **valor_para_multiplicas** } }
$rename	Renomeia o nome de um campo	{**$rename**: {nome_antigo : **novo_nome**} }

$Operadores para arrays

Operador	Função	EX
$pull	Remover os itens (uma correspondência)	{**$pull**: {campo: *elemento*} }
$push	Adicionar um elemento em um array	{**$push**: {campo: *elemento*} }
$addToSet	Adiciona um elemento apenas se ele já não existir.	{**$addToSet**: {campo: *elemento*} }
$pop	Remove ou o primeiro ou último elemento de um array	{**$pop**: {campo: *1*} } ou {**$pop**: {campo: *-1*} }
$pullAll	Remove mais de um elemento em um array	{**$pullAll**: {campo: [*elemento1, elemento2, ...]*} }
$each	Para adicionar vários elementos em um array	{campo: {**$each**: [*elemento1, elemento2, ...*] } }
$position	Especifica uma posição para inserir um elemento	{**$push**: {campo: *elemento,* **$position: *index***} }
$slice	Limite o número de elementos de um array	{**$push**: {campo: *elemento,* **$slice: *qnt***} }
$sort	Ordena um array	{**$push**: {campo: *elemento,* **$sort: *1 ou -1***} }