Você será capaz de:

- Utilizar os operadores de comparação
    - $lt (*less than* , menor que, <)
    - $lte (*less than or equal* , menor ou igual a, <=)
    - $gt (*greater than* , maior que, >)
    - $gte (*greater than or equal* , maior ou igual a, >=)
    - $eq (*equal* , igual a, =)
    - $ne (*not equal* , diferente de, !=, <>)
    - $in (*in* , dentro de)
    - $nin (*not in* , não está dentro de)
- Utilizar os operadores lógicos
    - $and (*and* , se todas as condições forem verdadeiras retorna *true*)
    - $or (*or* , se apenas uma condição for verdadeira retorna *true*)
- Compor filtros avançados com
    - $not (*not* , inverte o resultado da expressão)
    - $nor (*not or* , semelhante ao or , porém trabalha com a condição *false*)
- Utilizar o operador
    - $exists (*exists* , verifica a existência de um atributo)
- Utilizar o método
    - sort() (*sort* , ordenar)
- Remover documentos

Operadores de Comparação

Os operadores de comparação servem para que você execute consultas comparando os valores de atributos dos documentos de uma coleção.
Esses operadores são utilizados como parte do filtro de alguns métodos para leitura de documentos do MongoDB . Por exemplo, o find() e o count() , que vimos no dia anterior, ou o update() e o distinct() , que veremos mais adiante, aceitam filtros de comparação.
Vale lembrar que, para comparações de BSON types diferentes, você deve entender a ordem de comparação .
Os operadores seguem uma sintaxe padrão que é composta por um subdocumento, como no exemplo abaixo.

Copiar

```
{ <campo>: { <operador>: <valor> } }
```

Além disso, os operadores são identificados pelo prefixo $.

Operador $lt

O operador $lt seleciona os documentos em que o valor do atributo filtrado é menor do que (<) o valor especificado.
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $lt: 20 } })
```

Essa consulta selecionará todos os documentos na coleção inventory cujo valor do atributo qty é menor do que 20 .

Operador $lte

O operador $lte seleciona os documentos em que o valor do atributo filtrado é menor ou igual (<=) ao valor especificado.
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $lte: 20 } })
```

Essa query selecionará todos os documentos na coleção inventory cujo valor do atributo qty é menor ou igual a 20 .

Operador $gt

O operador $gt seleciona os documentos em que o valor do atributo filtrado é maior do que (>) o valor especificado.
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $gt: 20 } })
```

Essa query selecionará todos os documentos na coleção inventory cujo valor do atributo qty é maior do que 20 .

Operador $gte

O operador $gte seleciona os documentos em que o valor do atributo filtrado é maior ou igual (>=) ao valor especificado.
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $gte: 20 } })
```

Essa query selecionará todos os documentos na coleção inventory cujo valor do atributo qty é maior ou igual a 20 .

Operador $eq

O operador $eq seleciona os documentos em que o valor do atributo filtrado é igual ao valor especificado. Esse operador é equivalente ao filtro { campo: <valor> } e não tem nenhuma diferença de performance.
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $eq: 20 } })
```

A operação acima é exatamente equivalente a:

Copiar
```
db.inventory.find({ qty: 20 })
```

Durante a aula você verá mais exemplos que mostrarão que o $eq é apenas uma maneira de explicitar o operador.

Operador $ne

Esse operador é o contrário do anterior. Ao utilizar o $ne , o MongoDB seleciona os documentos em que o valor do atributo filtrado não é igual ao valor especificado.

Copiar
```
db.inventory.find({ qty: { $ne: 20 } })
```

A query acima retorna os documentos da coleção inventory cujo valor do atributo qty é diferente de 20 , incluindo os documentos em que o atributo qty não existe.

Operador $in

A consulta abaixo retorna todos os documentos da coleção inventory em que o valor do atributo qty é 5 ou 15 . E embora você também possa executar essa consulta utilizando o operador $or , que você verá mais à frente no conteúdo, escolha o operador $in para executar comparações de igualdade com mais de um valor no mesmo atributo.

Copiar
```
db.inventory.find({ qty: { $in: [ 5, 15 ] } })
```

Operador $nin

O operador $nin seleciona os documentos em que o valor do atributo filtrado não é igual ao especificado no array , ou o campo não existe.

Copiar
```
db.inventory.find( { qty: { $nin: [ 5, 15 ] } } )
```

Essa consulta seleciona todos os documentos da coleção inventory em que o valor do atributo qty é diferente de 5 e 15 . Esse resultado também inclui os documentos em que o atributo qty não existe.

Vamos praticar !

Agora que aprendemos sobre os operadores de comparação, vamos sedimentar esses conhecimentos com alguns exercícios de fixação. Para isso, vamos criar um novo banco de dados chamado business com uma coleção chamada restaurants :

1. Clique neste link ;
2. Copie todo o conteúdo do link e depois abra o MongoDB Shell ;
3. Utilize o comando use business para criar e utilizar este banco de dados;
4. Cole todo o conteúdo no terminal do MongoDB Shell e confirme com *ENTER* ou baixe o arquivo e o execute usando o comando mongo exercise-filter-operators.js .

Para confirmar que está tudo funcionando, rode o seguinte comando:

Copiar
```
db.restaurants.count()
```

O valor retornado deve ser 60 , que é a quantidade de documentos nesta coleção. Agora utilize os operadores de comparação para resolver os desafios de 1 a 5.

1. Selecione e faça a contagem dos restaurantes presentes nos bairros Queens , Staten Island e Bronx . (utilizando o atributo borough)
2. Selecione e faça a contagem dos restaurantes que não possuem culinária do tipo American . (utilizando o atributo cuisine)
3. Selecione e faça a contagem dos restaurantes que possuem avaliação maior ou igual a 8 . (utilizando o atributo rating)
4. Selecione e faça a contagem dos restaurantes que possuem avaliação menor que 4 .
5. Selecione e faça a contagem dos restaurantes que não possuem as avaliações 5 , 6 e 7 .

Operadores Lógicos

Assim como os operadores de comparação, os operadores lógicos também podem ser utilizados nos mesmos métodos para leitura e atualização de documentos do MongoDB . Eles também ajudam a elaborar consultas mais complexas, juntando cláusulas para retornar documentos que satisfaçam os filtros.

Operador $not

Sintaxe:

Copiar
```
{ campo: { $not: { <operador ou expressão> } } }
```

O operador $not executa uma operação lógica de NEGAÇÃO no < operador ou expressão > especificado e seleciona os documentos que não correspondam ao < operador ou expressão > . Isso também inclui os documentos que não contêm o atributo .
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ price: { $not: { $gt: 1.99 } } })
```

Essa consulta seleciona todos os documentos na coleção inventory em que o valor do atributo price é menor ou igual a 1.99 (em outras palavras, não é maior que 1.99), ou em que o atributo price não exista.
É importante destacar que a expressão { $not: { $gt: 1.99 } } retorna um resultado diferente do operador $lte . Ao utilizar { $lte: 1.99 } , os documentos retornados serão somente aqueles em que o campo price existe e cujo valor é menor ou igual a 1.99 .

Operador $or

O operador $or executa a operação lógica OU em um array de uma ou mais expressões e seleciona os documentos que satisfaçam ao menos uma das expressões.
Sintaxe:

Copiar
```
{ $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }
```

Considere o exemplo a seguir:

Copiar
```
db.inventory.find({ $or: [{ qty: { $lt: 20 } }, { price: 10 }] })
```

Essa consulta seleciona todos os documentos da coleção inventory em que o valor do atributo qty é menor do que 20 ou o valor do atributo price é igual a 10 .

Operador $nor

O operador $nor também executa uma operação lógica de NEGAÇÃO , porém, em um array de uma ou mais expressões, e seleciona os documentos em que todas essas expressões falhem , ou seja, seleciona os documentos em que todas as expressões desse array sejam falsas.
Sintaxe:

Copiar
```
{ $nor: [ { <expressão1> }, { <expressão2> }, ...  { <expressãoN> } ] }
```

Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ $nor: [{ price: 1.99 }, { sale: true }] })
```

Essa query retorna todos os documentos da coleção inventory que:

- Contêm o atributo price com o valor diferente de 1.99 e o atributo sale com o valor diferente de true ;
- Ou contêm o atributo price com valor diferente de 1.99 e não contêm o atributo sale ;
- Ou não contêm o atributo price e contêm o atributo sale com valor diferente de true ;
- Ou não contêm o atributo price e nem o atributo sale .

Pode parecer complexo, mas você fará mais exercícios para praticar esse operador.

Operador $and

O operador $and executa a operação lógica E num array de uma ou mais expressões e seleciona os documentos que satisfaçam todas as expressões no array. O operador $and usa o que chamamos de avaliação em curto-circuito (*short-circuit evaluation*). Se alguma expressão for avaliada como falsa , o MongoDB não avaliará as expressões restantes, pois o resultado final sempre será falso independentemente do resultado delas.
Sintaxe:

Copiar
```
{ $and: [{ <expressão1> }, { <expressão2> } , ... , { <expressãoN> }] }
```

Múltiplas expressões especificando o mesmo atributo

Considere o exemplo abaixo:

Copiar
```
db.inventory.find({
    $and: [
        { price: { $ne: 1.99 } },
        { price: { $exists: true } }
    ]
})
```

Essa consulta seleciona todos os documentos da coleção inventory em que o valor do atributo price é diferente de 1.99 e o atributo price existe.

Múltiplas expressões especificando o mesmo operador

Considere o exemplo abaixo:

Copiar
```
db.inventory.find({
    $and: [
        { price: { $gt: 0.99, $lt: 1.99 } },
```

```
    {
        $or: [
            { sale : true },
            { qty : { $lt : 20 } }
        ]
    }
]
})
```

Essa consulta seleciona todos os documentos da coleção inventory em que o valor do campo price é maior que 0.99 e menor que 1.99 , E o valor do atributo sale é igual a true , OU o valor do atributo qty é menor do que 20 . Ou seja, essa expressão é equivalente a (price > 0.99 E price < 1.99) (onde o E está implícito na vírgula aqui { $gt: 0.99, $lt: 1.99 }) E (sale = true OU qty < 20) .

Vamos praticar !

Faça os desafios de 1 a 5 abaixo, sobre os operadores lógicos utilizando a coleção restaurants criada no tópico anterior.

1. Selecione e faça a contagem dos restaurantes que não possuem avaliação menor ou igual a 5 , essa consulta também deve retornar restaurantes que não possuem o campo avaliação.
2. Selecione e faça a contagem dos restaurantes em que a avaliação seja maior ou igual a 6 , ou restaurantes localizados no bairro Brooklyn .
3. Selecione e faça a contagem dos restaurantes localizados nos bairros Queens , Staten Island e Broklyn e possuem avaliação maior que 4 .
4. Selecione e faça a contagem dos restaurantes onde nem o campo avaliação seja igual a 1 , nem o campo culinária seja do tipo American .
5. Selecione e faça a contagem dos resturantes em que a avaliação seja maior que 6 ou menor que 10 , E esteja localizado no bairro Brooklyn , OU não possuem culinária do tipo Delicatessen .

Operador $exists

Sintaxe :

Copiar
```
{ campo: { $exists: <boolean> } }
```

Quando o <boolean> é verdadeiro (true), o operador $exists encontra os documentos que contêm o atributo , incluindo os documentos em que o valor do atributo é igual a null . Se o <boolean> é falso (false), a consulta retorna somente os documentos que não contêm o atributo.
Veja o exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $exists: true } })
```

Essa consulta retorna todos os documentos da coleção inventory em que o atributo qty existe.
Você também pode combinar operadores, como no exemplo abaixo:

Copiar
```
db.inventory.find({ qty: { $exists: true, $nin: [ 5, 15 ] } })
```

Essa consulta seleciona todos os documentos da coleção inventory em que o atributo qty existe E seu valor é diferente de 5 e 15 .

Método sort()

Sintaxe :

Copiar
```
db.colecao.find().sort({ "campo": "1 ou -1"})
```

Quando existe a necessidade de ordenar os documentos por algum atributo, o método sort() se mostra muito útil. Usando um valor positivo (1) como valor do atributo, os documentos da consultas são ordenados de forma crescente ou alfabética (também ordena por campos com strings). Em complemento, usando um valor negativo (-1), os documentos de saída estarão em ordem decrescente ou contra alfabética.
Ele pode ser combinado com o método find() :

Copiar
```
db.example.find({}, { value, name }).sort({ value: -1 }, { name: 1 })
```

O sort() só pode ser usado se tiver algum resultado de busca antes:

Copiar
```
db.colecao.find().sort({ nomeDoAtributo: 1 }) // certo
db.colecao.sort({ nomeDoAtributo: 1 }) // errado
```

Vamos ver um exemplo?

Copiar
```
db.example.insertMany([
    { "name": "Mandioquinha Frita", "price": 14 },
    { "name": "Litrão", "price": 8 },
    { "name": "Torresmo", "price": 16 }
])
```

Copiar
```
db.example.find().sort({ "price": 1 }).pretty()
```

Copiar
```
// Resultado esperado:
{
        "_id" : ObjectId("5f7dd0582e2738debae74d6c"),
        "name" : "Litrão",
        "price" : 8
}
{
        "_id" : ObjectId("5f7dd0582e2738debae74d6b"),
        "name" : "Mandioquinha Frita",
        "price" : 14
}
{
        "_id" : ObjectId("5f7dd0582e2738debae74d6d"),
        "name" : "Torresmo",
        "price" : 16
}
```

Copiar
```
db.example.find().sort({ "price": -1 }, { "name" : 1 }).pretty()
```

Copiar
```
// Resultado esperado:
{
        "_id" : ObjectId("5f7dd0582e2738debae74d6d"),
        "name" : "Torresmo",
        "price" : 16
}
{
        "_id" : ObjectId("5f7dd0582e2738debae74d6b"),
        "name" : "Mandioquinha Frita",
        "price" : 14
}
{
        "_id" : ObjectId("5f7dd0582e2738debae74d6c"),
```

```
    "name" : "Litrão",
    "price" : 8
}
```

Vamos praticar !

Faça os desafios 1 e 2 abaixo sobre o sort() utilizando a coleção restaurants criada anteriormente.

1. Ordene alfabeticamente os restaurantes pelo nome (atributo name).
2. Ordene os restaurantes de forma descrescente baseado nas avaliações.

Removendo documentos

Para remover documentos de uma coleção temos dois métodos que são utilizados para níveis de remoção diferentes: o deleteOne() e o deleteMany() . Os dois métodos aceitam um documento como parâmetro, que pode conter um filtro simples ou até mesmo um conjunto de expressões para atender aos critérios de seleção.

deleteOne()

Esse método remove apenas um documento, que deve satisfazer o critério de seleção, mesmo que muitos outros documentos também se enquadrem no critério de seleção. Se nenhum valor for passado como parâmetro, a operação removerá o primeiro documento da coleção.
O exemplo abaixo remove o primeiro documento da coleção inventory em que o atributo status é igual a D :

Copiar
```
db.inventory.deleteOne({ status: "D" })
```

deleteMany()

Esse método remove todos os documentos que satisfaçam o critério de seleção.
O exemplo abaixo remove todos os documentos da coleção inventory em que o atributo status é igual a A :

Copiar
```
db.inventory.deleteMany({ status : "A" })
```

Para remover todos os documentos da coleção, basta não passar nenhum parâmetro para o método deleteMany() :

Copiar
```
db.inventory.deleteMany({})
```

Vamos praticar !

Faça os desafios 1 e 2 abaixo, sobre remoção de documentos utilizando a coleção restaurants criada anteriormente.

1. Remova o primeiro restaurante que possua culinária do tipo Ice Cream, Gelato, Yogurt, Ices .
2. Remova todos os restaurantes que possuem culinária do tipo American .

Exercícios

back-end

Antes de começar: versionando seu código

Para versionar seu código, utilize o seu repositório de exercícios. 😉
Abaixo você vai ver exemplos de como organizar os exercícios do dia em uma branch, com arquivos e commits específicos para cada exercício. Você deve seguir este padrão para realizar os exercícios a seguir.

1. Abra a pasta de exercícios:
2. Copiar
3. `$ cd ~/trybe-exercicios`
4. Certifique-se de que está na branch main e ela está sincronizada com a remota. Caso você tenha arquivos modificados e não comitados, deverá fazer um commit ou checkout dos arquivos antes deste passo.
5. Copiar

`$ git checkout main`

6. `$ git pull`
7. A partir da main, crie uma branch com o nome exercicios/23.2 *(bloco 23, dia 2)*
8. Copiar
9. `$ git checkout -b exercicios/23.2`
10. Caso seja o primeiro dia deste módulo, crie um diretório para ele e o acesse na sequência:
11. Copiar

`$ mkdir back-end`

12. `$ cd back-end`
13. Caso seja o primeiro dia do bloco, crie um diretório para ele e o acesse na sequência:

14. Copiar

```
$ mkdir bloco-23-introducao-ao-mongodb
```

15. `$ cd bloco-23-introducao-ao-mongodb`
16. Crie um diretório para o dia e o acesse na sequência:
17. Copiar

```
$ mkdir dia-2-filter-operators
```

18. `$ cd dia-2-filter-operators`
19. Os arquivos referentes aos exercícios deste dia deverão ficar dentro do diretório ~/trybe-exercicios/back-end/block-23-introducao-ao-mongodb/dia-2-filter-operators. Lembre-se de fazer commits pequenos e com mensagens bem descritivas, preferencialmente a cada exercício resolvido.

    Verifique os arquivos alterados/adicionados:
20. Copiar

```
$ git status
On branch exercicios/23.2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
```

21. `  modified:   exercicio-1`

    Adicione os arquivos que farão parte daquele commit:
22. Copiar

```
# Se quiser adicionar os arquivos individualmente
$ git add caminhoParaArquivo

# Se quiser adicionar todos os arquivos de uma vez, porém, atente-se para não adicionar arquivos indesejados acidentalmente
```

23. `$ git add --all`

    Faça o commit com uma mensagem descritiva das alterações:
24. Copiar

25. `$ git commit -m "Mensagem descrevendo alterações"`
26. Você pode visualizar o log de todos os commits já feitos naquela branch com `git log`.
27. Copiar

```
$ git log
commit 100c5ca0d64e2b8649f48edf3be13588a77b8fa4 (HEAD -> exercicios/23.2)
Author: Tryber Bot <tryberbot@betrybe.com>
Date:   Fry Sep 27 17:48:01 2019 -0300

    Exercicio 2 - mudando o evento de click para mouseover, tirei o alert e coloquei pra quando clicar aparecer uma imagem do lado direito da tela

commit c0701d91274c2ac8a29b9a7fbe4302accacf3c78
Author: Tryber Bot <tryberbot@betrybe.com>
Date:   Fry Sep 27 16:47:21 2019 -0300

    Exercicio 2 - adicionando um alert, usando função e o evento click

commit 6835287c44e9ac9cdd459003a7a6b1b1a7700157
Author: Tryber Bot <tryberbot@betrybe.com>
Date:   Fry Sep 27 15:46:32 2019 -0300
```

28. `  Resolvendo o exercício 1 usando eventListener`
29. Agora que temos as alterações salvas no repositório local precisamos enviá-las para o repositório remoto. No primeiro envio, a branch `exercicios/23.2` não vai existir no repositório remoto, então precisamos configurar o `remote` utilizando a opção `--set-upstream` (ou `-u`, que é a forma abreviada).
30. Copiar
31. `$ git push -u origin exercicios/23.2`
32. Após realizar o passo 9, podemos abrir a Pull Request a partir do link que aparecerá na mensagem do push no terminal, ou na página do seu repositório de exercícios no GitHub através de um botão que aparecerá na interface. Escolha a forma que preferir e abra a Pull Request. De agora em diante, você repetirá o fluxo a partir do passo 7 para cada exercício adicionado, porém como já definimos a branch remota com `-u` anteriormente, agora podemos simplificar os comandos para:
33. Copiar

```
# Quando quiser enviar para o repositório remoto
$ git push
```

```
# Caso você queria sincronizar com o remoto, poderá utilizar apenas
$ git pull
```

34.
35. Quando terminar os exercícios, seus códigos devem estar todos commitados na branch exercicios/23.2, e disponíveis no repositório remoto do GitHub. Pra finalizar, compartilhe o link da Pull Request no canal de Code Review para a monitoria e/ou colegas revisarem. Faça review você também, lembre-se que é muito importante para o seu desenvolvimento ler o código de outras pessoas. 👊

Agora, a prática!

Antes de iniciar os exercícios

Para os exercícios a seguir, utilizaremos um banco de dados de super-heróis. Faça o download do arquivo JSON aqui .
Após fazer o download do arquivo, importe-o para o MongoDB através da ferramenta mongoimport . No seu terminal, utilize o seguinte comando (lembre-se de substituir <caminho do arquivo> pelo caminho do arquivo na sua máquina):

Copiar
```
mongoimport --db class --collection superheroes <caminho_do_arquivo>
```

Pronto! Você já tem uma base de dados com 734 super-heróis. Para conferir, conecte-se à instância do seu banco de dados utilizando o Mongo Shell e execute os seguintes comandos:

```
use class;
```

```
db.superheroes.count();
```

Os documentos têm a seguinte estrutura:

Copiar
```
{
    "_id" : ObjectId("5e4ed2b2866be74b5b26ebf1"),
    "name" : "Abin Sur",
    "alignment" : "good",
    "gender" : "Male",
    "race" : "Ungaran",
    "aspects" : {
        "eyeColor" : "blue",
        "hairColor" : "No Hair",
```

```
        "skinColor" : "red",
        "height" : 185,
        "weight" : 40.82
    },
    "publisher" : "DC Comics"
}
```

Exercícios

O MongoDB possui diversas ferramentas. Por exemplo, o mongo , o mongosh , o Compass e outras ferramentas de terceiros. Você pode utilizar aquela que achar melhor para executar as consultas (*queries*), o importante é realizá-las.
Exercício 1: Inspecione um documento para que você se familiarize com a estrutura. Entenda os atributos e os níveis existentes.
Exercício 2: Selecione todos os super-heróis com menos de 1.80m de altura. Lembre-se de que essa informação está em centímetros.
Exercício 3: Retorne o total de super-heróis menores que 1.80m.
Exercício 4: Retorne o total de super-heróis com até 1.80m.
Exercício 5: Selecione um super-herói com 2.00m ou mais de altura.
Exercício 6: Retorne o total de super-heróis com 2.00m ou mais.
Exercício 7: Selecione todos os super-heróis que têm olhos verdes.
Exercício 8: Retorne o total de super-heróis com olhos azuis.
Exercício 9: Utilizando o operador $in , selecione todos os super-heróis com cabelos pretos ou carecas ("No Hair").
Exercício 10: Retorne o total de super-heróis com cabelos pretos ou carecas ("No Hair").
Exercício 11: Retorne o total de super-heróis que não tenham cabelos pretos ou não sejam carecas.
Exercício 12: Utilizando o operador $not , retorne o total de super-heróis que não tenham mais de 1.80m de altura.
Exercício 13: Selecione todos os super-heróis que não sejam humanos nem sejam maiores do que 1.80m .
Exercício 14: Selecione todos os super-heróis com 1.80m ou 2.00m de altura e que sejam publicados pela Marvel Comics .
Exercício 15: Selecione todos os super-heróis que pesem entre 80kg e 100kg , sejam Humanos ou Mutantes e não sejam publicados pela DC Comics .
Exercício 16: Retorne o total de documentos que não contêm o campo race .
Exercício 17: Retorne o total de documentos que contêm o campo hairColor .
Exercício 18: Remova apenas um documento publicado pela Sony Pictures .
Exercício 19: Remova todos os documentos publicados pelo George Lucas .