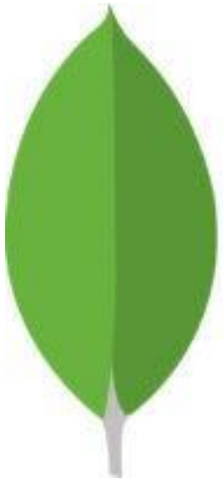
AULA 2



mongoDB

ELISA DE FÁTIMA ANDRADE SOARES
(efas@cin.ufpe.br)

Resumo da Aula 2

- Trabalhar com **agregações**
- Utilizar os operadores **$where**,**$text**, **$expr** e **$cond**
- Realizar atualizações em documentos
- Remover coleções e dbs através do shell
- Remover documentos.

Trabalhando com agregações

- Um exemplo de agregação que usamos até agora foi o método de ponteiro **count()**
  - Podemos também usar o count da mesma maneira que usamos o find:
    - db.movieDetails.count({"director": "George Lucas"})
- Assim como o count, podemos realizar algo como um **SELECT DISTINCT**
    - db.movieDetails.distinct("year")
    - O método distinct retorna um **array**
    - Para saber a quantidade de valores distintos, podemos consultar o tamanho do array:
        - db.movieDetails.distinct("year").length

Trabalhando com agregações

- **db.collection.aggregate()**
  - Uma das maneiras de realizar agregações de dados no mongoDB é através do pipeline de agregação
- Esse pipeline é um array de etapas (stages) sequenciais
  - Todos os documentos da coleção entram no pipeline
  - Estágios podem se repetir
  - Alguns estágios podem **gerar documentos** ou **filtrar documentos**, ou seja, a quantidade de documentos que entra não necessariamente será a que é agregada
    - Para fins de performance, é interessante **filtrar primeiro**

match

- Filtra os documentos para exibir somente os documentos que correspondem à(s) condição(ões) especificada(s).
- Exemplo: { $match: { <query> } }

```
db.movieDetails.aggregate([
                    {$match: {year:
{$gte: 2010}}},
                    {$sort:
{"media_imdb": -1}}
])
```

group

- Agrupa os documentos pelo **_id** e**,** esse **_id** pode ser especificado através de campo existente no documento.
- Os documentos de saída também podem conter campos a ser computado, que retorna os valores de alguma expressão do acumulador .
- Exemplo:

```
db.movieDetails.aggregate([
                        {$group: {_id: "$year", Indicacoes:
{$sum: "$awards.nominations"}}},
])
```

Trabalhando com agregações

- **db.collection.aggregate()**

```
db.movieDetails.aggregate(
  [
    {$match: {}},
    {$group: {_id: "$year", total: {$sum: "$runtime"}}}
  ])
```

- Outros estágios podem incluir:
  - **$sort**
  - **$project**

Trabalhando com agregações

- db.collection.aggregate()
- **Acumuladores:**
  - **$avg:** média
  - **$first:** primeiro valor
  - **$last:** último valor (só funcionam se os documentos estiverem ordenados)
  - **$max:** máximo
  - **$min:** mínmo
  - **$sum:** soma

  Mais em:

   https://docs.mongodb.com/manual/reference/operator/aggregation/group/#pipe._S_group

limit/sort

- **Limit**
- Define a quantidade de registros que aparecerão na tela.
- Exemplo: { $limit: <positive integer> }
  - db.movieDetails.find( ).limit(2)

- **Sort**
- Classifica todos os documentos de entrada e os retorna ao pipeline na ordem classificada. Exemplo: { $sort: { “campo”: -1 ou 1 ... } }
- **1** para ordem crescente , **- 1** para ordem decrescente
  - db.movieDetails.find().sort({"runtime":-1})

Trabalhando com agregações

Atividade Prática! **10 Min?**

- Qual o diretor com o maior número de prêmios vencidos (awards.wins)?
- Considerando os filmes de 2010 para cá, qual a combinação de gêneros tem a maior nota média no imdb?
- Quais são os anos cujos filmes têm, em média, tempo de duração maior que 110 minutos?

$where

- Fornece maior flexibilidade, mas requer que o banco de dados processe uma expressão ou função JavaScript para cada documento na coleção.
- Faz referência ao documento na expressão ou função JavaScript usando **this** ou **obj.**
- Exemplo:

```
db.movieDetails.find( { $where:
function() {
   return ((this.runtime) == "540")
} } );
```

$text

- **$text** permite realizar buscas em campos textuais
  - Para fazê-lo, é necessário especificar um índice textual (text index) no campo:
    - db.movieDetails.createIndex({title: "text"})
  - Só pode haver um text **index** por **coleção**

  - Após especificado o índice, você pode fazer o find:
    - db.movieDetails.find({$text: {$search: "Star Wars"}})
    - db.movieDetails.find({$text: {$search: "\"Star Wars\""}})
    - db.movieDetails.find({$text: {$search: "Star -Trek"}})

$expr e $cond

- **$expr:** permite que se compare os valores de mais de um campo do mesmo documento

```
db.movieDetails.find({
    $expr: {
        $gt: ["$tomato.userMeter", "$tomato.meter"]
    }
})
```

- Você pode combinar também com um operador condicional **$cond**

```
db.movieDetails.find({
    $expr: {
        $gt: [{
            $cond: {
                if: {$gte: ["$tomato.meter", "$tomato.userMeter"]},
                then: "$tomato.meter",
                else: "$tomato.userMeter"
            }
        }, 95]
    }
})
```

Trabalhando com agregações

Atividade Prática! **10 Min?**

- Queremos saber quantos filmes há na coleção com nota acima de 8.5
- Como temos várias notas diferentes na coleção, iremos utilizar a seguinte lógica:
  - Se imdb.votes for maior que tomato.userReviews, consideramos o imdb.meter
  - Caso contrário utilizamos o tomato.userMeter/10
  - Você pode utilizar o operador {$divide: [x, y]}, que retorna o valor de x/y

Substituindo valores dos campos

- **updateOne()**
- Atualiza o primeiro documento que satisfaça os critérios estabelecidos
  - db.movieDetails.updateOne({"title": "Aquarius"}, {$set: {"director": “Elisa Andrade"}})
- **updateMany()**
  - Atualiza todos os documentos que satisfaçam os filtros
- **Upserts**
- O mongoDB permite adicionar novos documentos através de updates
- É só utilizar a opção {upsert: true}
  - db.movieDetails.updateOne({query}, {update}, {upsert: true})

Substituindo valores dos campos

- **Um Exemplo utilizando o updateOne()**

```
db.movieDetails.updateOne({
title:"The Martian"
},{
$set:{
"awards":{
"wins":8,
"nominations": 14,
"text": "Nomimated for 3 Golden Globes.
Another 8 wins & 14 nominations."

        }

    }
})
```

Substituindo valores dos campos

- **Updates**
- **Operadores de campos**
- **$currentDate –** substitui o valor do campo para a data corrente
- **$inc –** incrementa o valor do campo por uma quantidade especificada
- **$min –** faz o update se o valor dado for menor que o existente
- **$max –** faz o update se o valor dado for maior que o existente
- **$mul –** multiplica o valor do campo por um valor especificado
- **$rename –** renomeia o campo
- **$setOnInsert –** somente atualiza se o update for um upsert
- **$unset –** remove o campo do documento

Updates

- **Exemplos:**
- **$inc**
  - db.movieDetails.updateOne({"title": “The Martian"}, {$inc: {"year": 2}})
- **$min**
  - db.movieDetails.updateOne({"title": “The Martian"}, {$min: {"year": 2016}})
- **$unset**
  - db.movieDetails.updateOne({"title": “The Martian"}, {$unset: {"imdb.votes": 0}})
- Como o **unset** apenas remove o campo, você pode colocar qualquer valor

Updates

Atividade Prática!

15 Min?

- O metacritic do primeiro Star Wars está desatualizado... faça o update desse campo, alterando-o para o valor atual:

Star Wars: Episode IV - A New Hope MUST-SEE **90**

Movie, 1977

A long time ago in a galaxy far, far away... Princess Leia (Carrie Fisher) is held hostage by the evil Imperial forces in their effort to quell the rebellion against the Galactic Empire. Luke...

- Além disso, atualize o número de votos no imdb (imdb.votes) para o maior entre o valor atual e **1.121.505.**
- Por fim, atualize o campo director do filme de título “The Shawshank Redemption” para **Frank Darabont.** Se o documento não existir, a query deverá inseri-lo na coleção.

Atualizações em arrays

- **$push**
- Permite adicionar um valor ao final do array
  - db.movieDetails.updateMany({filtro}, {$push: {genres: “Drama”}})
- **$pop**
- Remove o primeiro ou o último item do array
  - db.movieDetails.updateMany({filtro}, {$pop: {genres: -1}})
    - Remove o primeiro elemento
  - db.movieDetails.updateMany({filtro}, {$push: {genres: 1}})
    - Remove o último elemento

Atualizações em arrays

- **$pull**
- Permite remover todas as instâncias de um valor em um array
  - db.movieDetails.updateMany({filtro}, {$pull: {actors: "Hayden Christensen"}})

- **$pullAll**
  - Similar, mas, no lugar de especificar uma query, você deve prover uma lista de valores a serem removidos

- Mais detalhes em: https://docs.mongodb.com/manual/reference/operator/update-array/

Atualizações em arrays

Atividade Prática! **10 Min?**

- Adicione o gênero “Sci-fi” ao final do array “genres” para os filmes das duas trilogias de Star Wars. Utilize $text na sua solução.

Removendo documentos de uma coleção

- db.collection.**deleteOne**({filtro})
- db.collection.**deleteMany**({filtro})

Removendo dbs

- **Removendo dbs e coleções**
  - db.collection.drop()
  - db.dropDatabase()

OBRIGADA 

elisaandrade21

efas@cin.ufpe.br