

# MongoDB - Fundamentos, Funcionalidades e Operações CRUD

**UC-13 Desenvolvimento Back-End**

**Curso Técnico em Desenvolvimento de Sistemas - Prof<sup>a</sup> Aline Timm**

**Pelotas- RS**

**Objetivo:** Ao final da aula, o aluno será capaz de entender a arquitetura e os componentes do MongoDB, conectar-se a uma instância e realizar operações de Criar, Ler, Atualizar e Deletar (CRUD) dados de forma proficiente usando o mongosh.

# A Hierarquia do MongoDB

## 1 Cluster

**O que é?** Um Cluster é a camada mais externa. Ele representa um grupo de servidores que trabalham juntos para armazenar seus dados. No mundo real, um cluster de produção geralmente tem múltiplos servidores para garantir alta disponibilidade e escalabilidade.

**No nosso caso:** O comando `atlas deployments setup testing` que vimos no material de apoio cria um cluster simples de nó único, perfeito para desenvolvimento e aprendizado.

## 2 Database

**O que é?** Dentro de um cluster, podemos ter vários Bancos de Dados. Um banco de dados é um contêiner físico para coleções. Cada banco de dados tem seu próprio conjunto de arquivos no sistema de arquivos do servidor.

**Comando para ver os databases:** `show dbs`.

## 3 Collection

**O que é?** Dentro de um banco de dados, temos as Coleções. Uma coleção é um agrupamento de documentos do MongoDB. Ela é o equivalente a uma tabela em um banco de dados relacional (SQL).

 **Dica :** O MongoDB cria uma coleção automaticamente na primeira vez que você insere um documento nela. Você não precisa defini-la previamente.

## 4 Document

**O que é?** Este é o coração do MongoDB. Um Documento é a unidade básica de dados, equivalente a uma linha em SQL. É uma estrutura de dados composta por pares de campo e valor, muito similar a um objeto JSON.

**Analogia:** Um Documento é uma única folha de papel (um registro) dentro de uma pasta (Collection).

**Estrutura:** Os documentos são armazenados em um formato chamado BSON (Binary JSON), que é uma representação binária do JSON que suporta mais tipos de dados.

# Documento e Schema

Exemplo de um Documento em uma collection, por exemplo: 'movies'

```
{  
  "_id": ObjectId("573a1390f29313caabcd42e8"),  
  "plot": "A young man is accidentally sent 30 years into the past...",  
  "genres": [ "Adventure", "Comedy", "Sci-Fi" ],  
  "runtime": 116,  
  "cast": [ "Michael J. Fox", "Christopher Lloyd", "Lea Thompson" ],  
  "title": "Back to the Future",  
  "fullplot": "A young man is accidentally sent 30 years into the past in a time-traveling DeLorean invented by his friend, Dr. Emmett Brown, and must make sure his high-school-age parents fall in love in order to return to the future.",  
  "released": ISODate("1985-07-03T00:00:00Z"),  
  "imdb": {  
    "rating": 8.5,  
    "votes": 779395,  
    "id": 88763  
  }  
}
```

## 5. Schema

**O que é?** Em SQL, o schema é a estrutura rígida e predefinida de uma tabela. No MongoDB, o conceito é diferente. O MongoDB possui um **schema dinâmico**.

**O que isso significa?** Documentos dentro da mesma coleção não precisam ter a mesma estrutura. Um documento de filme pode ter o campo "awards", e outro pode não ter. Isso oferece uma flexibilidade imensa.

 **Dica:** Embora o MongoDB seja flexível, na prática, nossas aplicações geralmente esperam uma estrutura consistente. É por isso que usamos ferramentas como o Mongoose (que vimos na aula anterior) para definir um schema no nível da aplicação, garantindo que os dados que salvamos sigam um padrão.

# Operações CRUD

Agora que temos nosso ambiente pronto e cheio de dados, vamos aprender a fazer as quatro operações essenciais: **Create, Read, Update e Delete.**

## Operações de Leitura (Read): A Arte de Consultar

Operações de leitura recuperam documentos de uma coleção. A principal ferramenta para isso é o método `db.collection.find()`.

Primeiro, vamos mudar para o banco de dados que queremos usar:

```
use sample_mflix
```

Agora nosso contexto é o banco `sample_mflix`. Vamos trabalhar com a coleção `movies`.

### 1. Leitura Básica

**Listar todos os documentos (com limite):** O comando `find()` sem argumentos retorna todos os documentos, o que pode ser lento. Vamos usar `.limit()` para ver apenas 5.

```
db.movies.find().limit(5)
```

**Filtrar por um campo (Variáveis de Consulta):** Para encontrar um documento específico, passamos um objeto de filtro como primeiro argumento para `find()`. Este objeto contém os campos e valores que queremos corresponder.

```
// Encontra filmes com o título "Back to the Future"  
db.movies.find({ title: 'Back to the Future' })
```

Este é o equivalente ao WHERE em SQL.

**Usando `findOne()`:** Se você sabe que haverá apenas um resultado ou só quer o primeiro, `findOne()` é mais conveniente, pois retorna o documento diretamente, e não um cursor.

```
db.movies.findOne({ title: 'Back to the Future' })
```

### 2. Operadores de Consulta (Queries Avançadas)

O poder do MongoDB está nos seus operadores de consulta.

# Operadores de Consulta Avançados

## Operadores de Comparação

- `$eq`: igual a (geralmente omitido, como vimos acima)
- `$gt`: maior que (greater than)
- `$gte`: maior ou igual a (greater than or equal)
- `$lt`: menor que (less than)
- `$lte`: menor ou igual a (less than or equal)
- `$ne`: diferente de (not equal)
- `$in`: o valor está em um array
- `$nin`: o valor não está em um array

## Operadores Lógicos

- `$and`: une cláusulas com um E lógico (comportamento padrão quando se usa vírgulas no filtro).
- `$or`: une cláusulas com um OU lógico.
- `$not`: inverte o efeito de uma expressão.

## Exemplos

**Exemplo:** Encontrar filmes com nota no IMDB maior ou igual a 9.0.

```
// Note a sintaxe para acessar um campo em um documento aninhado: "imdb.rating"  
db.movies.find({ "imdb.rating": { $gte: 9.0 } })
```

**Exemplo:** Encontrar filmes cujo gênero seja "Action" OU "Adventure".

```
db.movies.find({ genres: { $in: ["Action", "Adventure"] } }).limit(5)
```

**Exemplo:** Encontrar filmes que foram lançados em 1985 E têm nota no IMDB maior que 8.0.

```
db.movies.find({ year: 1985, "imdb.rating": { $gt: 8.0 } })
```

**Exemplo:** Encontrar filmes que são do gênero "Comedy" OU foram lançados depois de 2010.

```
db.movies.find({ $or: [ { genres: "Comedy" }, { year: { $gt: 2010 } } ] }).limit(5)
```

## 3. Projeções: Escolhendo quais campos retornar

Muitas vezes, não queremos o documento inteiro. Podemos passar um segundo objeto para `find()` para especificar os campos que queremos. `1` significa incluir e `0` significa excluir.

```
// Buscar filmes de 1999, retornando apenas o título e o ano. O _id é retornado por padrão.  
db.movies.find({ year: 1999 }, { title: 1, year: 1 })
```

```
// Para omitir o _id, você precisa excluí-lo explicitamente  
db.movies.find({ year: 1999 }, { title: 1, year: 1, _id: 0 })
```

# Operações de Criação (Create): Inserindo Dados

Operações de criação adicionam novos documentos a uma coleção. Se a coleção não existir, ela será criada.

## insertOne()

Adiciona um único documento à coleção.

```
db.movies.insertOne({  
  title: "Meu Filme Fantástico",  
  year: 2025,  
  genres: ["Awesome", "Sci-Fi"],  
  cast: ["Aluno Destaque"],  
  plot: "Um estudante de desenvolvimento back-end aprende MongoDB e cria aplicações incríveis.",  
  imdb: { rating: 10, votes: 9999 }  
})
```

O resultado mostrará `acknowledged: true` e o `insertedId` do novo documento.

## insertMany()

Adiciona múltiplos documentos de uma vez a partir de um array.

```
db.movies.insertMany([  
  {  
    title: "Meu Filme 2: A Vingança",  
    year: 2026,  
    genres: ["Action", "Thriller"]  
  },  
  {  
    title: "Meu Filme 3: O Retorno",  
    year: 2027,  
    genres: ["Drama"]  
  }  
])
```

 **Dica:** Todas as operações de escrita no MongoDB (inserir, atualizar, deletar) são atômicas no nível de um único documento. Isso significa que a operação em um documento ou é concluída com sucesso, ou falha completamente, nunca deixando o documento em um estado inconsistente.

# Operações de Atualização e Deleção (Update & Delete)

## 1. Operações de Atualização (Update)

Modificam documentos existentes em uma coleção.

**updateOne():** Modifica o primeiro documento que corresponde ao filtro.

**Importante:** Precisamos usar operadores de atualização como `$set` para modificar campos específicos. Se passarmos um documento sem um operador, ele substituirá o documento inteiro!

```
// Atualiza o filme "Meu Filme Fantástico" para adicionar um prêmio  
db.movies.updateOne(  
  { title: "Meu Filme Fantástico" }, // O filtro para encontrar o documento  
  { $set: { awards: { wins: 1, text: "Melhor Filme de Aula" } } } // A modificação  
)
```

**updateMany():** Modifica todos os documentos que correspondem ao filtro.

```
// Adiciona o gênero "Classic" a todos os filmes lançados antes de 1970  
db.movies.updateMany(  
  { year: { $lt: 1970 } },  
  { $push: { genres: "Classic" } } // $push adiciona um item a um array  
)
```

**replaceOne():** Substitui completamente o primeiro documento que corresponde ao filtro.

```
db.movies.replaceOne(  
  { title: "Meu Filme 2: A Vingança" },  
  {  
    title: "Meu Filme 2: A Redenção",  
    year: 2026,  
    plot: "Uma nova história."  
  })
```

## 2. Operações de Deleção (Delete)

Removem documentos de uma coleção.

**deleteOne():** Remove o primeiro documento que corresponde ao filtro.

```
// Deleta o filme que acabamos de criar  
db.movies.deleteOne({ title: "Meu Filme 3: O Retorno" })
```

**deleteMany():** Remove todos os documentos que correspondem ao filtro.

```
// CUIDADO: Deleta todos os filmes criados nesta aula  
db.movies.deleteMany({ year: { $gte: 2025 } })
```