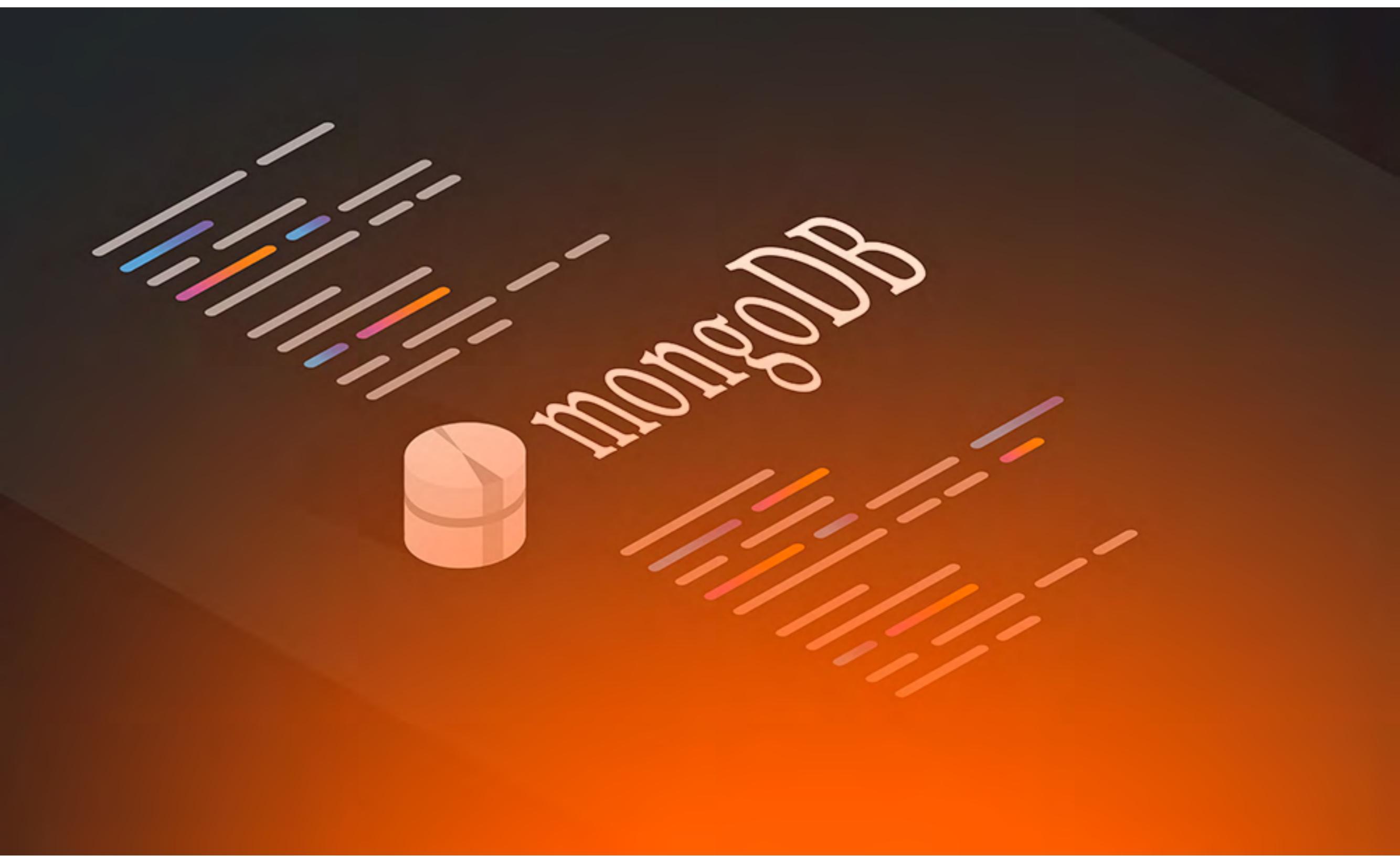


MANIPULAÇÃO DE DADOS COM MONGODB

Unidade 03



Sumário

03	Introdução
04	Objetivos de Aprendizagem
06	3.1 Plataformas de desenvolvimento - 06 3.2 Pré-requisitos para execução e tratamento de erros em operações CRUD - 06
09	3.3 INSERT – Inserção de documentos - 09
10	3.3.1 insertMany() - 10
11	3.3.2 insert() - 11
12	3.4 Tipos de dados - 12
14	3.5 Preparando uma base de testes - 14
18	3.6 Executando insertMany() - 18
23	3.8 Aumentando a base de testes - 23
26	3.9 Executando find() - 26
28	3.10 UPDATE – Alterando os documentos - 28
29	3.11 Executando updateMany() - 29
30	3.12 DELETE – Removendo documentos - 30
31	3.13 Executando deleteMany() - 31
32	3.14 writeResult – Verificando a execução dos métodos de escrita - 32
34	Síntese
35	Fullture Insights

Olá, **Fullturist,**

Seja bem-vindo(a)!

Iremos acompanhá-lo(a) no desenvolvimento desta trilha de aprendizagem, que trará informações a respeito da tecnologia e do ambiente de negócios em bancos de dados.

Pode ser surpreendente notar que este é um tema muito importante em nossa vida atualmente. Praticamente todos os eventos de nossa vida passada, e até mesmo de nossas tendências futuras, estão registrados em um banco de dados. Desde informações de nossa vida civil, como documentos pessoais e registros de propriedades, nossa vida pessoal, por meio das redes sociais, e nosso trajeto ao trabalho são registrados em um banco de dados.

Como você pode perceber, assim como esse tema pode ser vasto em nossa vida cotidiana, a tecnologia oferecida para atender essas diversas situações também é ampla. Entretanto, embora as opções tecnológicas sejam numerosas, esta trilha fornecerá a você todos os conhecimentos para entender o ambiente tecnológico e de negócios disponíveis para a solução da maioria das necessidades. Além disso, conheceremos o que há por trás de toda essa tecnologia e como todas essas informações são armazenadas e recuperadas! Para isso, utilizaremos um dos bancos de dados mais promissores do mercado atualmente: o MongoDB. Vamos aprender a instalar, inserir, recuperar e remover dados, bem como organizar e executar um projeto completo neste banco de dados.

Bom estudo!

Unidade 03

Manipulação de Dados com MONGODB

Olá, Fullturist!

Esta é a unidade 3 da trilha de aprendizagem Banco de Dados. Aqui, o você terá a oportunidade de conhecer e/ou se aprofundar nos métodos que possibilitam o armazenamento e a manipulação dos dados armazenados no MongoDB.

Embora o armazenamento de dados possa ser entendido como uma tecnologia relativamente simples, sabemos que o perfeito gerenciamento, tanto do armazenamento e concessão de acessos, quanto da segurança e manipulação de dados, envolve alta complexidade. E o domínio desta tecnologia também envolve conhecer e compreender como todas estas operações ocorrem em conjunto. Entretanto, para qualquer banco de dados, o aprendizado sempre se inicia pelas operações mais objetivas e principais da ferramenta. Afinal, são estas as operações que mais nos interessam. Desta forma, esta unidade terá foco nas operações de inserção, leitura, alteração e remoção dos dados no MongoDB – operações frequentemente referenciadas como CRUD (Create, Retrieve, Update e Delete).

Vamos lá?

Objetivos de aprendizagem

Ao final do estudo desta unidade, você será capaz de:

- entender que cada plataforma de desenvolvimento codifica de forma diferente a verificação do serviço do banco de dados e a conexão com o banco de dados que contém a coleção que será manipulada;

- conhecer os requisitos que a aplicação deve providenciar antes da execução dos métodos de escrita – CRUD – no banco de dados do MongoDB;
- aprender a incluir documentos no MongoDB, conhecendo quais tipos de dados podem ser usados nesta inclusão;
- compreender como realizar a leitura dos documentos armazenados, e quando esta leitura envolver vários documentos, entendendo que a aplicação deve definir uma área de armazenamento para que o MongoDB disponibilize os documentos recuperados;
- conhecer como alterar os campos dos documentos e como remover os documentos;
- saber preparar uma base de testes, verificando como testar passo a passo os métodos CRUD apresentados nesta unidade;
- conhecer como as informações a respeito do sucesso ou não da execução de escrita no banco são disponibilizadas para que a aplicação possa providenciar tratamento adequado ao resultado demonstrado por estas informações.

3.1 Plataformas de desenvolvimento

Diversas plataformas de desenvolvimento possuem drivers de comunicação com o MongoDB, como Java, Node.js e Python. Entretanto, muito embora a sintaxe dos métodos CRUD em MongoDB seja praticamente padrão, a forma como estes métodos devem ser codificados pode ser ligeiramente diferente entre as várias plataformas.

Esta unidade terá como base a sintaxe da plataforma MongoDB Shell, que é uma plataforma interativa de execução em linha de comando. Em sua atividade de desenvolvimento, você deverá obter informações a respeito de como codificar apropriadamente os métodos CRUD do MongoDB na plataforma de desenvolvimento em que você estiver usando. Mas não se preocupe! O site do MongoDB disponibiliza guias de referência para todas as plataformas que tenham drivers oficiais para este banco de dados e você poderá acessá-los sempre que necessitar.

No link a seguir você encontrará um guia de referência para o desenvolvimento em todas as plataformas oficiais: <https://docs.mongodb.com/drivers/>

Antes de iniciarmos o aprendizado sobre as operações CRUD no MongoDB será necessário também adquirir algum conhecimento sobre pré-requisitos e tratamento de possíveis erros na execução destas operações. Por isso, o próximo tópico aborda estes conceitos.

3.2 Pré-requisitos para execução e tratamento de erros em operações CRUD

É importante ter em mente que toda instrução que envolva manipulação de dados precisa que o ambiente da aplicação esteja preparado para a execução dos comandos pelo banco de dados. Assim, antes de executar qualquer instrução que envolva o MongoDB, a aplicação já deve ter providenciado as seguintes condições:

- o serviço do gerenciador do banco de dados deve estar em execução;
- antes da execução de uma instrução CRUD, a aplicação deve ter uma conexão ativa com o banco de dados que contém a coleção para a qual será executada a instrução.

A forma de garantir estas condições varia conforme a plataforma de desenvolvimento. Como já vimos no tópico anterior, nesta unidade teremos foco na sintaxe dos comandos CRUD MongoDB Shell e em sua atividade de desenvolvimento, por isso, você deverá obter informações a respeito de como providenciar estas condições. Novamente, a melhor fonte de informação sobre isso é o site do fornecedor! Não se preocupe, a forma de codificação nas várias plataformas é apresentada pelo guia de referência de cada método CRUD, com exemplos e explicações para cada plataforma. Neste momento, você deve se concentrar em conhecer e entender os métodos CRUD e como usá-los!

Para que você possa visualizar brevemente as diferenças de codificação entre as plataformas, observe as Figuras 01 e 02, que apresentam como a conexão com o MongoDB e a seleção do banco de dados é feita em Node.js e em Java.

Exemplo em Node.js: Conexão com o MongoDB e seleção do banco de dados

```
const MongoClient = require('mongodb').MongoClient;
const assert = require('assert');

// URL para conexão com o banco de dados
const url = 'mongodb://[$username]:[$password]@[$hostlist]/[$database]?authSource=$[authSource]';

// Execução da conexão com o banco de dados
MongoClient.connect(url, function(err, client) {assert.equal(null, err);client.close();});

// Selecionando o banco de dados "test"
const db = client.db("test");
```

Fonte: Adaptada de MongoDB (2021).

Exemplo em Java: Conexão com o MongoDB e seleção do banco de dados

```
// URL para conexão com o banco de dados  
final String uriString = "mongodb://${username}:${password}@${hostlist}/${database}?authSource=${authSource}";  
  
// Execução da conexão com o banco de dados  
MongoClient mongoClient = MongoClients.create(uriString);  
  
// Selecionando o banco de dados "test"  
MongoDatabase mongoDB = mongoClient.getDatabase("test");
```

Fonte: Adaptada de MongoDB (2021).

Como dissemos, estes exemplos foram extraídos da documentação do MongoDB. Você, com certeza, encontrará todas as informações necessárias para utilizar o MongoDB nas várias plataformas de desenvolvimento utilizadas atualmente.

No link a seguir você encontra um guia de referência com as instruções de conexão com o MongoDB em cada uma das plataformas oficiais:

<https://docs.mongodb.com/guides/server/drivers/>

Outro assunto importante em relação às operações CRUD são as informações sobre o sucesso ou não na execução destas operações e o tratamento de possíveis erros. A comunicação e o tratamento destes erros em MongoDB são especialmente projetados para lidar com o armazenamento em clusters e, por esta razão, é um assunto bastante extenso. Esta unidade está focada nas operações CRUD, por isso não entraremos nos detalhes deste tratamento. Entretanto, uma vez que os métodos de escrita do MongoDB oferecerem o método `writeConcern()` para a comunicação dos erros, é importante conhecer este conceito. Por este motivo abordaremos o funcionamento básico deste método.

Para entender a comunicação e o tratamento de erros no MongoDB, devemos sempre ter em mente uma configuração de armazenamento em cluster. Por esta razão, o MongoDB controla a ocorrência em erros de escrita em dois momentos:

- momento 1: no envio da informação ao servidor;
- momento 2: no momento efetivo da escrita.

Por meio do método `writeConcern()`, as operações de escrita recebem a informação sobre a ocorrência ou não de erros no recebimento da solicitação de escrita pelo servidor (momento 1). Após receber as solicitações de escrita, o servidor as armazena em uma área de armazenamento intermediária, enfileirando as solicitações ainda não executadas. Este banco de dados intermediário é chamado journaling, a partir do qual as escritas são efetivadas no MongoDB (momento 2).

No link a seguir você encontrará um guia de referência para a utilização do método `writeConcern()`: <https://docs.mongodb.com/manual/reference/write-concern/>

O próximo tópico inicia o assunto principal desta unidade: os métodos de manipulação de dados do MongoDB. Iniciaremos pelos métodos que realizam as inserções no banco de dados.

3.3 INSERT – Inserção de documentos

Na unidade 2, você teve um breve contato com a ferramenta MongoDB Compass e a partir dela fizemos a inserção de um documento de forma manual. Nesta unidade você verá como esta inserção é feita em uma instrução que possa ser codificada na construção de uma aplicação.

O MongoDB trabalha com o conceito de objetos e as operações CRUD são concretizadas por meio de métodos. No MongoDB, existem três métodos para a inserção em uma coleção:

- `db.collection.insertOne()`: insere um único documento;
- `db.collection.insertMany()`: insere vários documentos;
- `db.collection.insert()`: insere tanto um único documento, quanto vários documentos.

Estes três métodos também possuem alguns comportamentos que são padrão, entre eles:

- caso ainda não exista, a coleção será criada;
- caso o documento não tenha especificado um identificador, um “id” será criado automaticamente para cada documento inserido;

Dos três métodos mencionados, vamos nos concentrar nos métodos `insertMany()` e `insert()`. Começaremos pelo método `insertMany()`.

3.3.1 `insertMany()`

O método `insertMany()` executa a inserção de vários documentos. A Figura 03 demonstra a sintaxe e um exemplo de utilização deste método.

Sintaxe

```
db.collection.insertMany([< document 1 >, < document 2 >, ... ],  
  { writeConcern:<document>,  
    ordered:<boolean>  
  }  
)
```

Exemplo

```
db.Pedidos.insertMany([ < documento 1 > , < documento 2 >, ... ],)
```

Fonte: Adaptada de MongoDB (2021).

Vamos examinar a sintaxe:

Destacada em vermelho temos a coleção que se deseja inserir os documentos e, como parâmetro, um array de documentos (sinalizado pelos colchetes “[”]), contendo os documentos a serem inseridos na coleção;

Destacada em marrom, a especificação opcional de `writeConcern()`. Quando usado, o documento passado a este método contém os parâmetros que informam como a aplicação deseja o tratar possíveis erros na recepção do `insert` pelo servidor;

Também destacada em marrom, uma variável booleana que indica se a inserção será ou não feita de forma ordenada. Esta especificação é opcional e caso não seja informada, assume o valor “true”.

A Figura 03 também demonstra a construção mais simples para insertMany(). Neste exemplo, em uma única execução, vários documentos serão inseridos na coleção “Pedidos”. O tratamento de erros será padrão e a inserção será de forma ordenada.

Vejamos, então, o método insert().

3.3.2 insert()

O método insert() tem a sintaxe muito semelhante ao insertMany() e executa tanto a inserção de um único documento, quanto a inserção de vários documentos.

Entretanto, uma diferença importante é que insert() pode ser usado em um laço de transação. Como vimos na unidade 1, uma transação pode ser entendida como uma sequência de operações de escrita no banco de dados, que só faz sentido se todas forem bem-sucedidas. Executar apenas uma parte delas resultaria em um banco de dados inconsistente. A Figura 04 demonstra a sintaxe e um exemplo de utilização deste método.

Sintaxe

```
db.collection.insert(<documento ou array de documentos>,
  { writeConcern:<document>,
    ordered:<boolean>
  }
)
```

Exemplo

```
db.Pedidos.insert([ <documento 1> , <documento 2>, ... ],)
```

Fonte: Adaptada de MongoDB (2021).

A sintaxe também é bastante simples:

- Destacada em vermelho, a coleção em que se deseja inserir e como parâmetros, um documento ou um array contendo vários documentos a serem inseridos;
- Destacadas em marrom, as especificações opcionais de writeConcern() e da variável “ordered”, com os mesmos comportamentos já explicados no método insertMany().

O exemplo da Figura 04 demonstra a construção mais simples para insert(). Neste exemplo, em uma única execução, vários documentos serão inseridos na coleção “Pedidos”. O tratamento de erros será padrão e a inserção será de forma ordenada.

Ainda, relacionado à inserção de documentos, outro conhecimento que você também deve ter são os tipos de dados disponíveis no MongoDB. Vejamos os mais usados.

3.4 Tipos de dados

Na unidade 2, vimos que o MongoDB trabalha com documentos padrão JSON (Java Script Object Notation). Mais especificamente, o MongoDB trabalha com uma evolução deste padrão, chamada BSON (Binary JSON).

Além dos tipos de dados do padrão JSON, o BSON oferece tipos adicionais para a compatibilidade com as várias plataformas em que o MongoDB está disponível. Além disso, uma diferença entre os padrões é que o BSON suporta objetos e matrizes aninhados no mesmo documento, podendo criar índices e comparações entre estes objetos aninhados. Assim, além dos tipos de dados já especificados pelo JSON, o BSON disponibiliza, ainda, outros. Observe a relação dos tipos mais usados no Quadro 01.

Tipo	Descrição
String	Qualquer conteúdo entre aspas será considerado uma string
Integer (32 e 64 bits)	Valores numéricos inteiros
Double	Valores numéricos de ponto flutuante
Boolean	Tipo lógico de conteúdo "true" ou "false"
Null	Armazena valores nulos
Object	Objetos ou documentos aninhados
Array	Matrizes ou lista de valores múltiplos com uma única chave
Date	Data ou hora atual
Timestamp	Data e hora atual
Min/Max Keys	Compara um conteúdo dado, com o seu menor e maior valor nos elementos do banco de dados
Symbol	Armazena strings e é utilizado por idiomas que utilizam tipos de símbolos específicos
Binary Data	Refere-se ao atributo criado para relacionar as dependências ao desenvolvimento.
Regular Expression	Armazena expressões regulares
JavaScript code	Armazena códigos em JavaScript

Fonte: Elaborado pela autora (2021).

Encontre, nos links a seguir, um guia de referência para a utilização dos métodos de insert e um guia de referência para o padrão BSON, respectivamente:

[https://docs.mongodb.com/manual/reference insert-methods/](https://docs.mongodb.com/manual/reference	insert-methods/)

<https://bsonspec.org/spec.html>

Com estes conhecimentos, você já pode realizar na prática algumas inserções de documentos em coleções. Que tal praticar com o método `insertMany()`? Para isso, a primeira coisa a fazer é preparar um ambiente de testes. Este ambiente será preparado por você e será utilizado para testar todos os métodos CRUD apresentados nesta unidade. Vamos lá?

3.5 Preparando uma base de testes

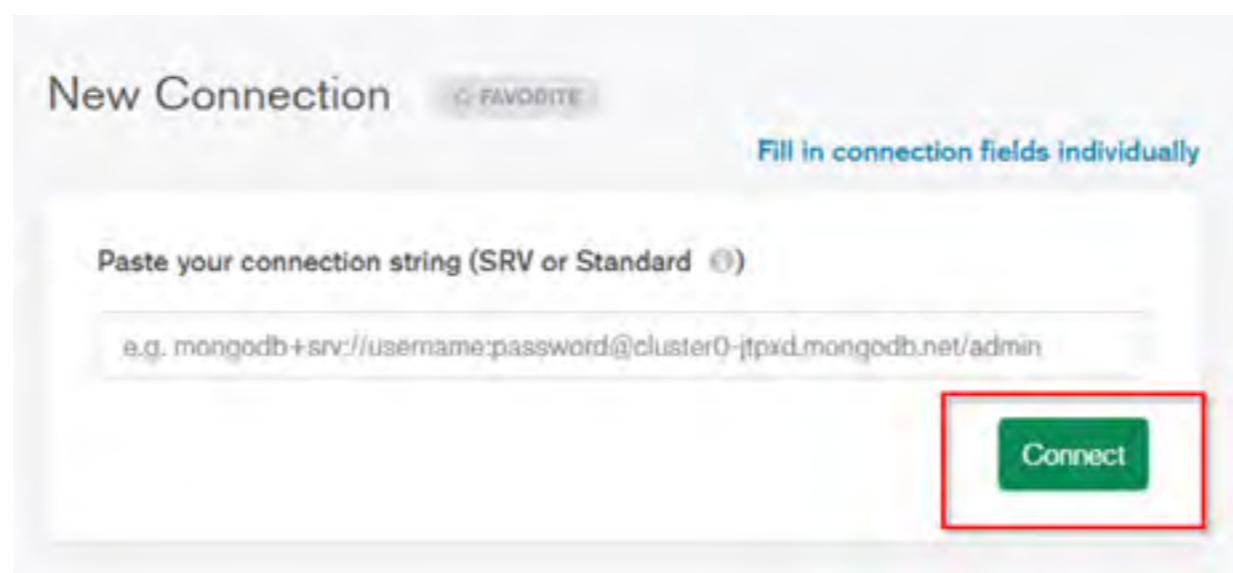
Para que você possa praticar os métodos CRUD apresentados nesta unidade, precisaremos de um volume maior de dados. Para isso, vamos utilizar dois arquivos que contém a programação de partidas e chegadas de voos do aeroporto de Guarulhos – São Paulo. Estes arquivos foram obtidos a partir do Portal Brasileiro de Dados Abertos, que disponibiliza dados públicos de acesso livre. Consulte aqui os arquivos mencionados, os quais serão usados para colocar em prática os comandos CRUD apresentados nesta unidade.

O arquivo original com a relação de voos e operações aéreas pode ser acessado a partir do link a seguir:

<https://dados.gov.br/dataset/voos-e-operacoes-aereas-slots-alocados/resource/64b75480-6cc4-45e7-baf2-c9b5dc9f89bd>

Primeiramente, será necessário criar um banco de dados e, ainda, a coleção que receberá os dados sobre os slots (direito de pousar ou decolar em aeroportos congestionados) do aeroporto de Guarulhos. Você aprendeu estes passos na unidade 2, mas vamos relembrar como fazer isso.

Primeiro, execute o MongoDB Compass e conecte com o MongoDB. A Figura 05 relembra como realizar esta conexão.



Fonte: Adaptada de MongoDB (2021).

Após a conexão com o MongoDB, será apresentada a tela da Figura 06, e você deve selecionar “CREATE DATABASE”.

Database Name	Storage Size	Collections	Indexes
Vendas	36.0KB	1	1
admin	20.0KB	0	1
config	12.0KB	0	2
local	36.0KB	1	1

Fonte: Adaptada de MongoDB (2021).

Na tela da Figura 07, você deve criar o database “FlightSlots” e a coleção “SlotsGRU”.

Create Database

Database Name
FlightSlots

Collection Name
SlotsGRU

Capped Collection
Fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order. ⓘ

Use Custom Collation
Collation allows users to specify language-specific rules for string comparison, such as rules for lowercase and accent marks. ⓘ

Time-Series
Time-series collections efficiently store sequences of measurements over a period of time.

Cancel Create Database

Fonte: Adaptada de MongoDB (2021).

A seguir, a tela da Figura 08 será apresentada, confirmando que a operação foi executada com sucesso, e, então, você deverá selecionar o database “FlightSlots”.

Database Name	Storage Size	Collections	Indexes	
FlightSlots	4.0KB	1	1	
Vendas	38.0KB	1	1	
admin	90.0KB	0	1	
config	34.0KB	0	9	
local	36.0KB	1	1	

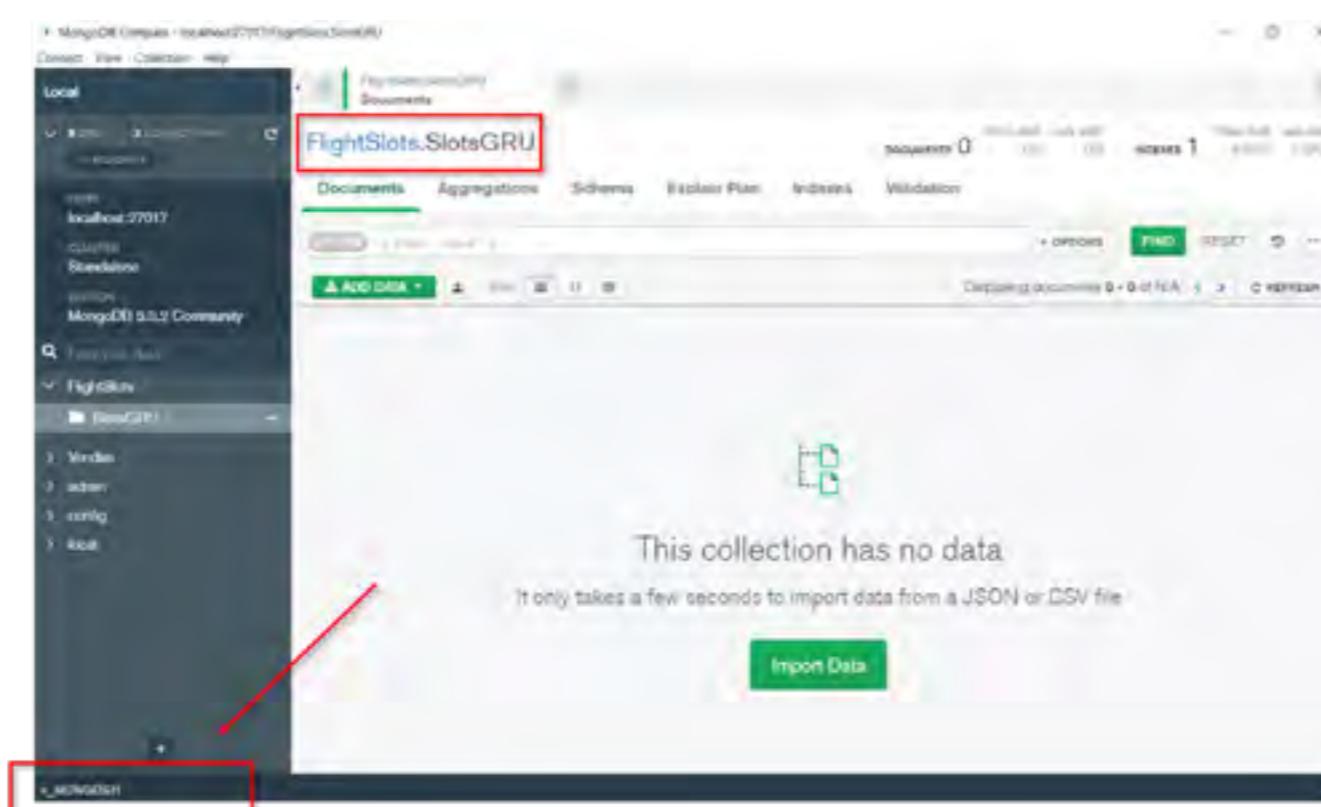
Fonte: Adaptada de MongoDB (2021).

Ao ser apresentada a tela da Figura 09, selecione a coleção “SlotsGRU”, na qual você colocará em prática os conhecimentos sobre o insertMany().

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
SlotsGRU	0	-	0.0 B	1	4.0 KB	

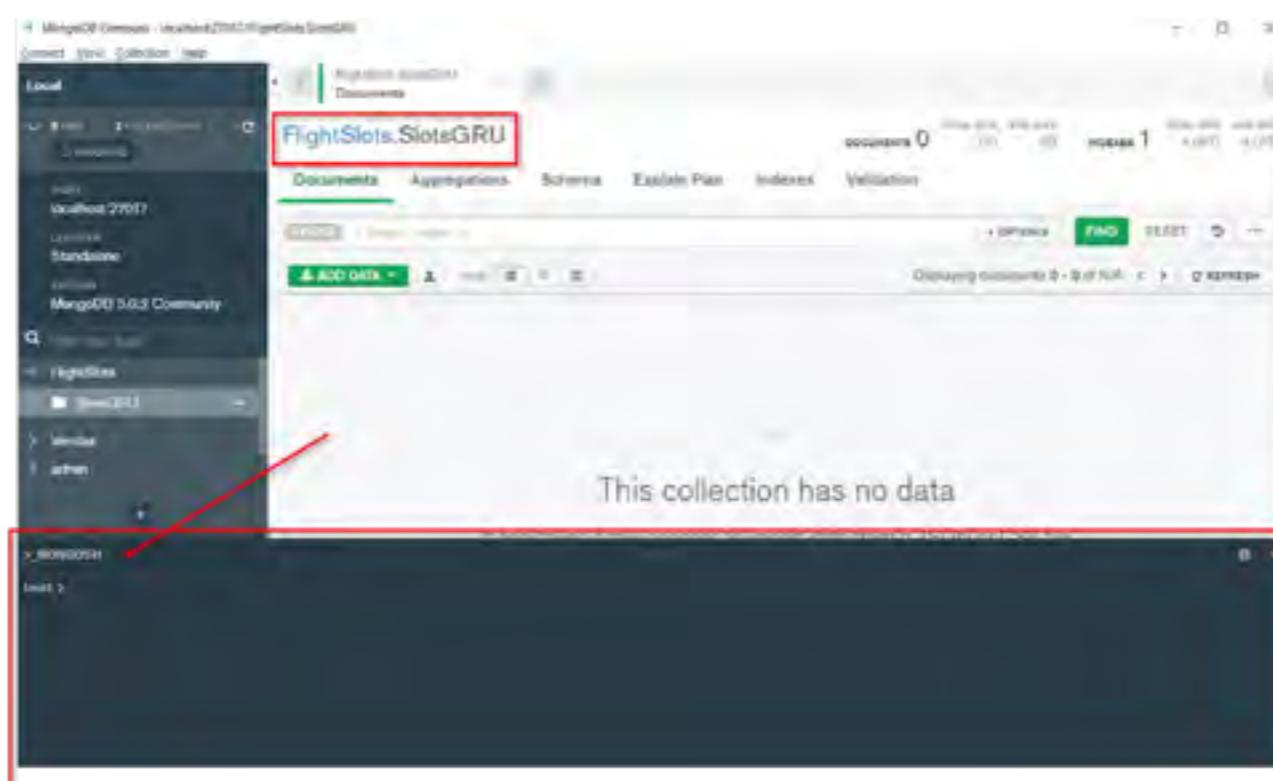
Fonte: Adaptada de MongoDB (2021).

Na unidade 2, você inseriu alguns documentos utilizando o MongoDB Compass. Nesta unidade, você vai utilizar o MongoDB Shell. Observe o ícone em destaque na tela da Figura 10. Acesse o MongoDB Shell açãoando este ícone.



Fonte: Adaptada de MongoDB (2021).

Ao acionar o ícone “MONGOSH”, uma janela de interação com MongoDB Shell será aberta. A tela da Figura 11 apresenta esta área.



Fonte: Adaptada de MongoDB (2021).

Observe novamente a Figura 11. Por padrão, o MongoDB Shell abre em conexão com o database “test”. Isto é sinalizado por “test >” no início da linha de comando. A coleção “SlotsGRU”, na qual você vai trabalhar, faz parte do database “FlightSlots”. Assim, primeiro você deve selecionar este database. Execute na linha de comando do MongoDB Shell:

- Digite o comando “use” [ENTER];
- Ao digitar o comando, o MongoDB Shell abrirá uma janela, sugerindo os databases já existentes no MongoDB;
- Selecione o database “FlightSlots” e a tela da Figura 12 será apresentada confirmando da seleção do database.

```
>_MONGOSH  
  
> use FlightSlots  
< 'switched to db FlightSlots'  
FlightSlots>
```

Fonte: Adaptada de MongoDB (2021).

Aqui, é necessário fazer uma observação muito importante. No MongoDB Shell, por padrão, caso o database ou a coleção não existam, o comando “use” ou a execução dos métodos de insert realizam a criação automática destes objetos. Decorre, no entanto, que erros na digitação nestes nomes podem acarretar erros nos demais acessos.

Você pode achar que estes objetos foram criados, quando, na realidade, podem ter sido criados com outros nomes. Então, mantenha atenção na execução destes comandos.

Até o momento, você apenas criou o database e a coleção para a prática dos comandos CRUD que serão apresentados nesta unidade. Nos próximos passos, você vai colocar em prática o método insertMany() para inserir documentos na coleção “SlotsGRU”.

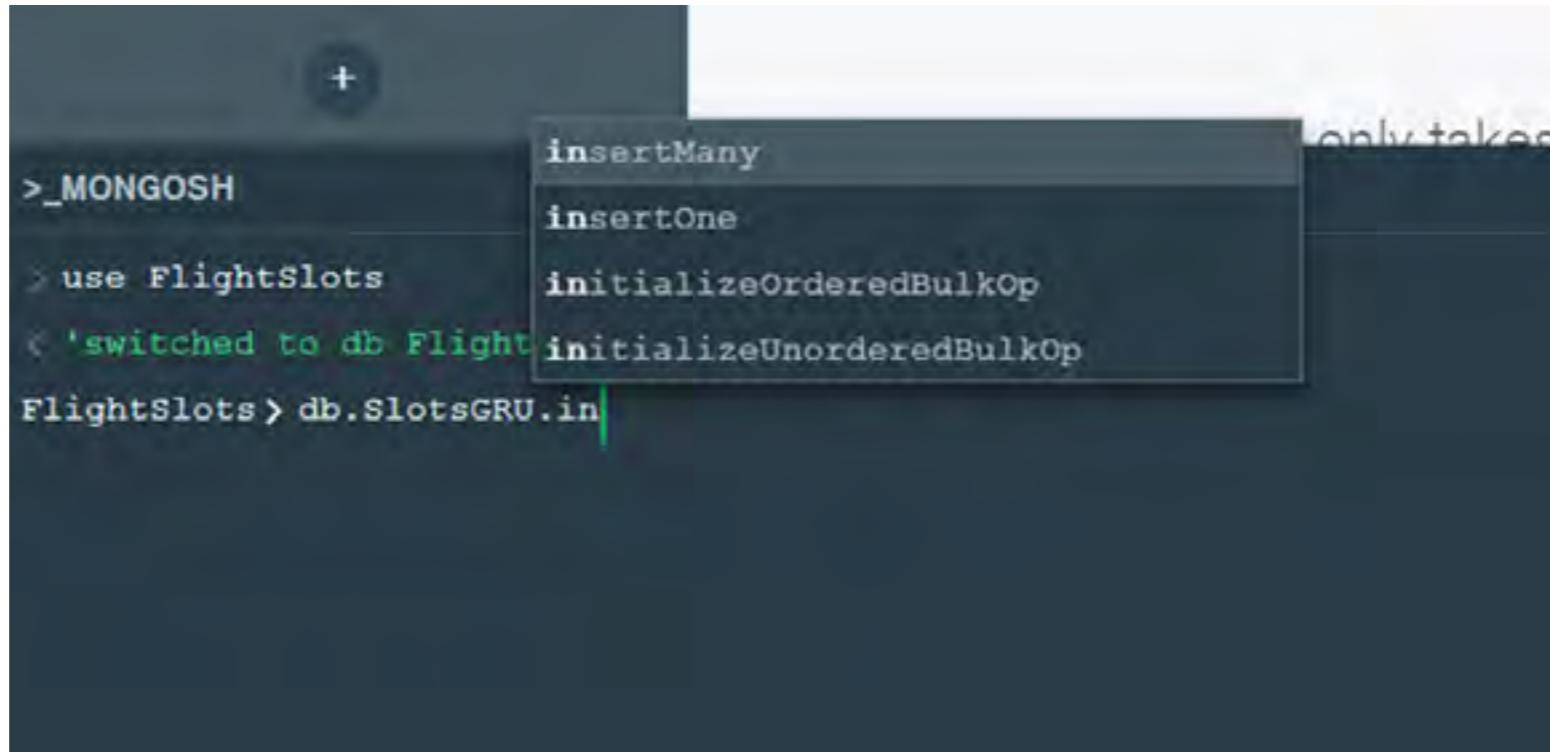
3.6 Executando insertMany()

Vimos que o método insertMany() realiza, ao mesmo tempo, a inserção de vários documentos. Vimos também que estes documentos são passados ao método, por meio de um array de documentos. Assim, para que você possa verificar a execução do método insertMany(), utilizaremos o conteúdo do arquivo “SlotsGRU-insertMany”, disponibilizado como material complementar desta unidade. Este arquivo contém um array de 32 documentos que serão inseridos na coleção “SlotsGRU”. Assim, execute os passos a seguir:

- Acesse e abra o arquivo “SlotsGRU insertMany”;
- Selecione e copie (CRTL-C) todo o seu conteúdo;
- Na linha de comando do MongoDB Shell, você vai digitar o método insertMany() conforme a sintaxe abaixo;

db.SlotsGRU.insertMany(Linhas copiadas do arquivo SlotsGRU-insertMany)

Observe que, ao digitar, o MongoDB Shell vai guiando a construção do método. Primeiro, é aberta uma janela sugerindo as coleções do database, e, logo após, é aberta outra janela, sugerindo os métodos de insert. A Figura 13 apresenta esta dinâmica;



Fonte: Adaptada de MongoDB (2021).

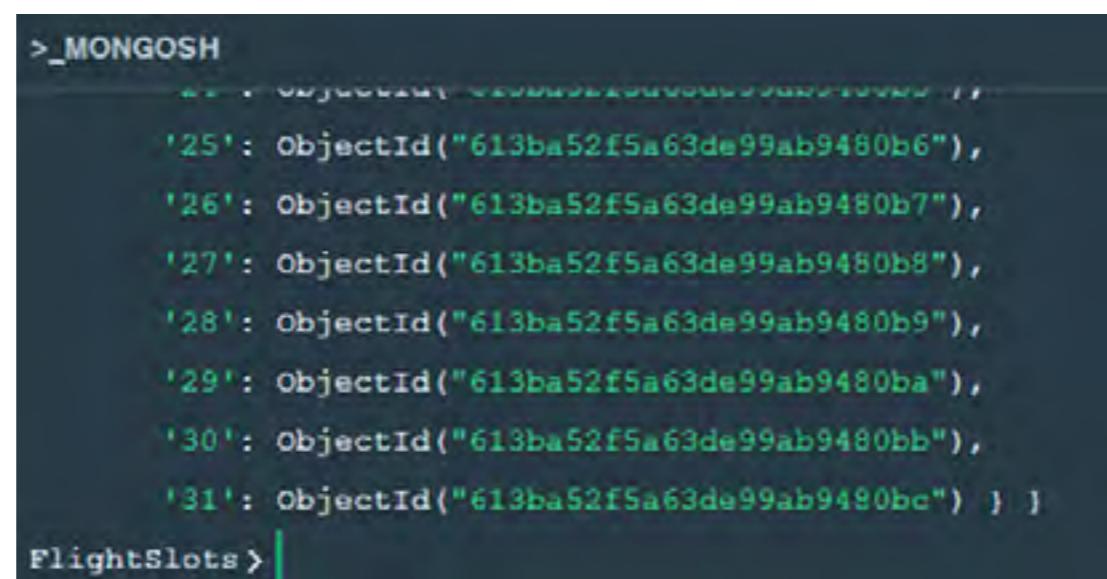
Após selecionar o método insertMany(), cole as linhas copiadas do arquivo “SlotsGRU-insertMany” dentro dos parênteses sugeridos. A Figura 14 apresenta a construção completa do método;

```
>_MONGOSH
> use FlightSlots
< 'switched to db FlightSlots'
FlightSlots> db.SlotsGRU.insertMany({{"Tipo de Operação": "A", "Cód Empresa Aérea": "2F", "Número do Voo": "5600"}}
```

Fonte: Adaptada de MongoDB (2021).

Execute o comando;

Após a execução, você verá a tela da Figura 15.



```
>_MONGOSH
> db.FlightSlots.find({},{_id:1}).limit(10)
{
  "_id": "613ba52f5a63de99ab9480b6",
  "_id": "613ba52f5a63de99ab9480b7",
  "_id": "613ba52f5a63de99ab9480b8",
  "_id": "613ba52f5a63de99ab9480b9",
  "_id": "613ba52f5a63de99ab9480ba",
  "_id": "613ba52f5a63de99ab9480bb",
  "_id": "613ba52f5a63de99ab9480bc"
}
FlightSlots >
```

Fonte: Adaptada de MongoDB (2021).

Como os documentos contidos no arquivo SltosGRU-insertMany.txt não especificam id's para os documentos, ao final da execução, são apresentados para cada documento, os id's gerados automaticamente pelo MongoDB Shell.

Agora que você já conhece os métodos de inserção no MongoDB, no próximo tópico você conhicerá os métodos que possibilitam a leitura e seleção dos documentos de interesse da aplicação.

3.7 FIND - Localizando os documentos de interesse

Os métodos `findOne()` e `find()` são os que executam a seleção de documentos no MongoDB. Entretanto, existem diferenças em relação ao ambiente de execução dos dois métodos. Vejamos as diferenças entre eles:

findOne(): executa a seleção de um único documento. Caso os critérios de seleção determinem a seleção de vários documentos, `findOne()` retornará o primeiro documento encontrado;

find(): seleciona um ou mais documentos ao mesmo tempo. Quando houver a seleção de vários documentos, o método `find()` necessita que a aplicação reserve, de forma prévia à sua execução, uma área de memória – chamada cursor – para que os documentos selecionados sejam depositados. A aplicação então, realizará o acesso aos documentos retornados a partir desta área. Uma vez que o método `findOne()` seleciona um único documento, a definição do cursor não é necessária.

A sintaxe dos dois métodos são muito semelhantes e neste tópico vamos focar no método `find()`. A Figura 16 apresenta a sintaxe deste método.

Sintaxe

```
db.collection.find(<documento definindo os parâmetros de seleção>,
                  <documento que define a projeção>)
```

Exemplo

```
//Especificação do cursor e do método find()
pedidosCursor = db.Pedidos.find(
    // Parâmetros de seleção
    { produto_id: 7239,
      produto_cor: { $eq: "Chumbo" },
      produto_tamanho: { $eq: "39" },
    },
    // Campos a serem retornados
    {cliente_id: 1, cliente_nome: 1} )

//Leitura de cada documento retornado pelo cursor
while (pedidosCursor.hasNext()) {
    print(tojson(pedidosCursor.next()));
}
```

Fonte: Adaptada de MongoDB (2021).

Examinado a sintaxe:

destacados em vermelho, a coleção que será pesquisada e o documento que contém as expressões e operadores que compõem os critérios para a seleção dos documentos desejados. Quando este documento não é informado, o método `find()` retorna todos os documentos da coleção;

destacado em azul, o documento que define a projeção, ou seja, quais campos devem ser retornados à aplicação.

Agora, vamos explorar o exemplo:

em verde, “`pedidos.cursor`” é a área de cursor onde os documentos serão depositados pelo MongoDB e disponibilizados para a aplicação;

destacados em vermelho, a coleção “Pedidos” e o documento que define os critérios de seleção: no caso, produto_id = 7239, produto_cor = “Chumbo” e produto_tamanho = “39”;

destacados em azul, os campos que são de interesse da aplicação. No exemplo, cliente_id e cliente_nome. O identificador 1, para estes campos, indica que estes são os campos de interesse. Ainda, definindo-se o valor zero, todos os campos do documento serão retornados, exceto aqueles definidos por este valor.

Você percebeu que na Figura 16 o exemplo de find() utilizou alguns operadores para a seleção dos documentos? O padrão JSON disponibiliza várias expressões e operadores para a seleção de documentos. Existem vários, e podemos listar alguns de utilização bastante frequente:

- \$eq: exatamente igual (=);
- \$ne: diferente (<> ou !=);
- \$gt: maior do que (>);
- \$lt: menor do que (<);
- \$lte: menor ou igual a (<=);
- \$regex: define uma cadeia de substring a ser pesquisada em uma string.

Neste link, você encontra várias expressões e operadores para a seleção de documentos:

<https://docs.mongodb.com/manual/reference/operator/query/>

Já no link a seguir, você encontra um guia de referência para a utilização dos métodos de find:

<https://docs.mongodb.com/manual/reference/method/db.collection.find/#mongodb-method-db.collection.find>

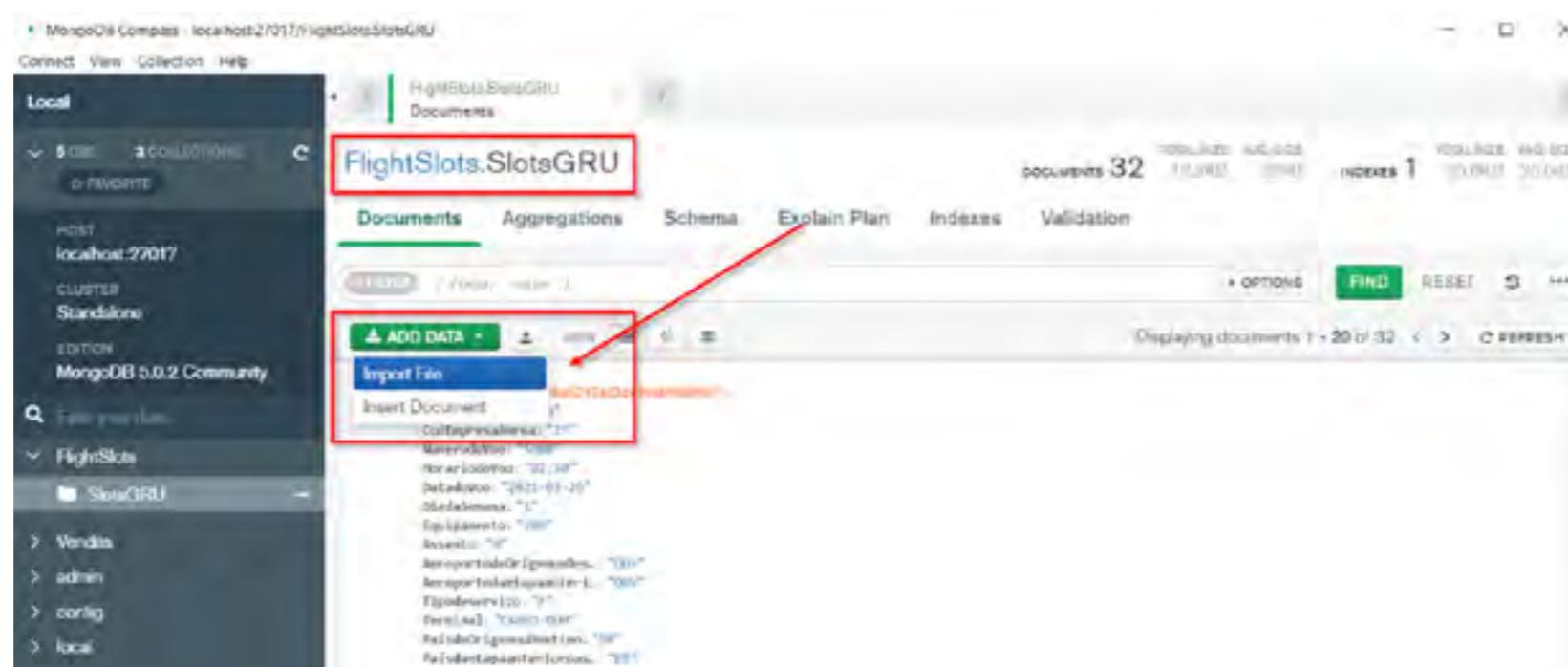
Você também irá colocar em prática o método `find()`! Para que você possa visualizar o quanto o MongoDB é robusto em pesquisas na base de dados, primeiramente vamos importar uma quantidade maior de documentos na coleção SlotsGRU. Nossa próximo passo, será explorar o MongoDB Compass para importar documentos a partir de um arquivo.

3.8 Aumentando a base de testes

Como já dissemos, para que você possa verificar como o MongoDB é potente em pesquisas, antes de colocar em prática o método `find()`, vamos adicionar uma quantidade maior de documentos à coleção SlotsGRU. Entretanto, desta vez vamos explorar uma outra facilidade do MongoDB Compass. Execute os seguintes passos:

- acesse os materiais complementares desta unidade e faça download do arquivo “SlotsGRU.txt”;
- abra o MongoDB Compass;
- selecione o database FlightSlots;
- selecione a coleção SlotsGRU;
- acione o botão “ADD DATA”, conforme apresentado pela Figura 17.

Vamos utilizar a opção “Import file”, observe:



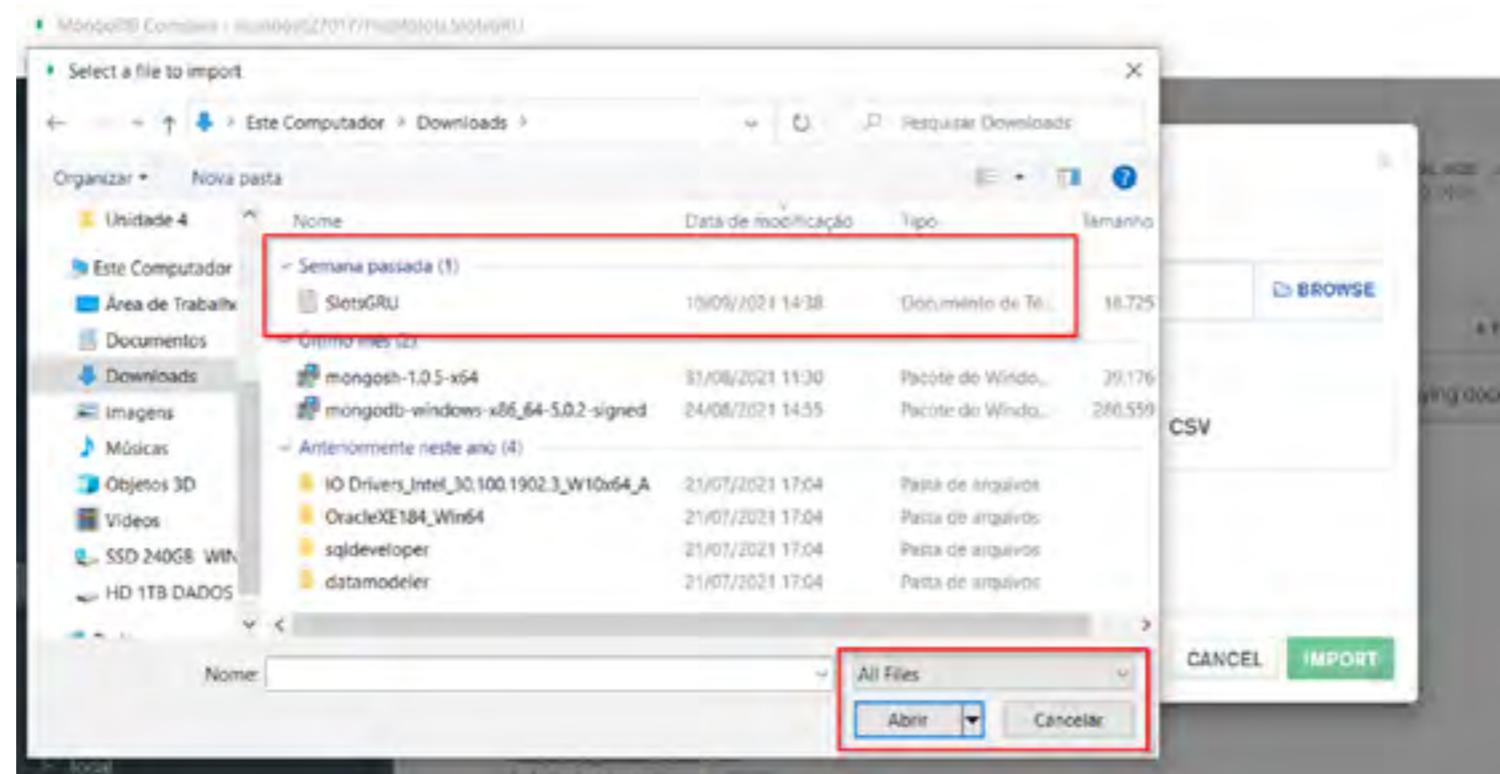
Fonte: Adaptada de MongoDB (2021).

Ao acionar a opção “Import file”, a tela da Figura 18 será apresentada.



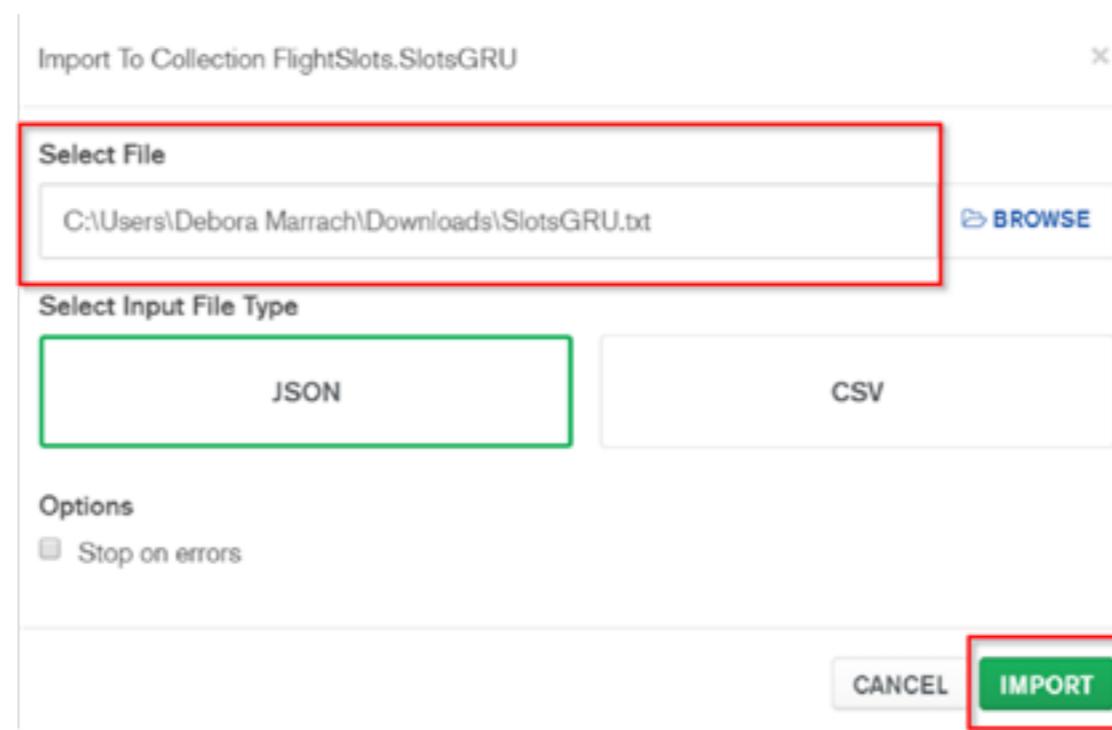
Fonte: Adaptada de MongoDB (2021).

Acione o botão “BROWSE” e selecione a pasta na qual você realizou o download. Como o arquivo SlotsGRU foi criado com o tipo “txt”, selecione também a apresentação de todos os tipos de arquivos da pasta (“All files”), conforme demonstrado pela Figura 19.



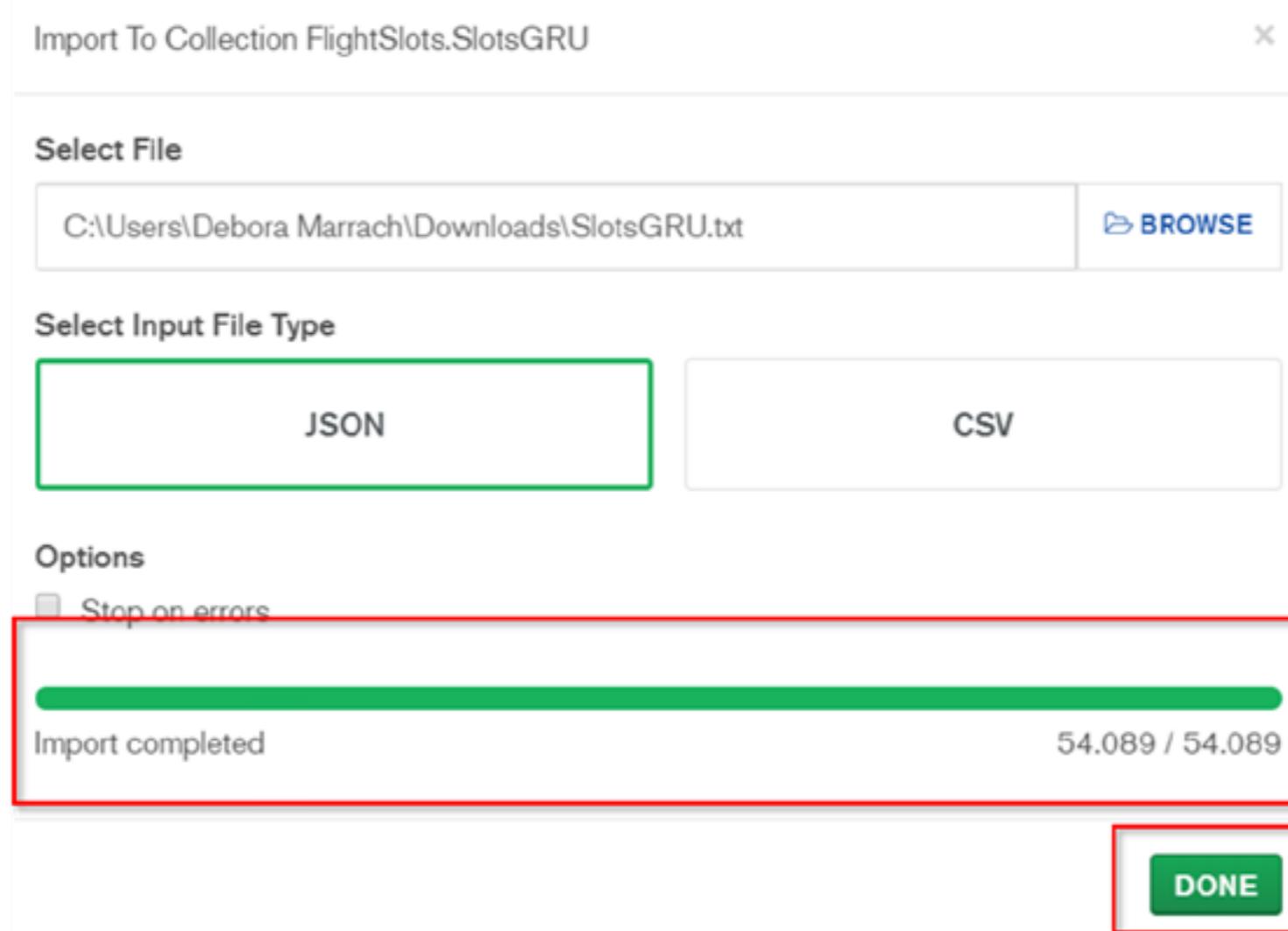
Fonte: Adaptada de MongoDB (2021).

Ao selecionar o arquivo, a tela da Figura 20 será apresentada e você deverá aceitar a importação dos documentos contidos no arquivo SlotsGRU.txt.



Fonte: Adaptada de MongoDB (2021).

A conclusão da importação dos documentos é demonstrada pela Figura 21. Observe que o MongoDB Compass demonstra as quantidades de documentos lidos e importados. Conclua a operação teclando em “DONE”.



Fonte: Adaptada de MongoDB (2021).

A coleção SlotsGRU agora possui um total de 54.121 documentos (54.089 importados e 32 anteriormente incluídos no tópico anterior, quando você executou o método `insertMany()`). Vamos então, executar uma seleção na coleção SlotsGRU usando o MongoDB Shell.

3.9 Executando find()

Antes de executar um `find()` na coleção `SlotsGRU`, examine o conteúdo dos documentos da coleção. Observe atentamente na Figura 22 os campos desta coleção. Vamos executar um `find()` tendo como interesse, por exemplo, os seguintes requisitos:

- Todos os documentos cujo `NumerodoVoo` = “5600”;
- Campos desejados: `NumerodoVoo`, `HorariodoVoo`, `DiadaSemana` e `CodEmpresaAerea`.

The screenshot shows the MongoDB Compass application interface. On the left, there's a sidebar with a tree view of databases and collections. The main area is titled "FlightSlots.SlotsGRU" and shows a table of documents. The table has columns for "_id", "NumerodoVoo", "HorariodoVoo", "DiadaSemana", and "CodEmpresaAerea". One document is expanded to show its full structure, including fields like "TipoOperacao", "CodEmpresalotao", "NumerodoVoo", "HorariodoVoo", "DiadaSemana", "CodEmpresaAerea", "AeroportoOrigem", "AeroportoDestino", "Terminal", "Tiposervico", "PaisdeOrigem", "Linc", and "PAJscoutCapaunter".

Fonte: Adaptada de MongoDB (2021).

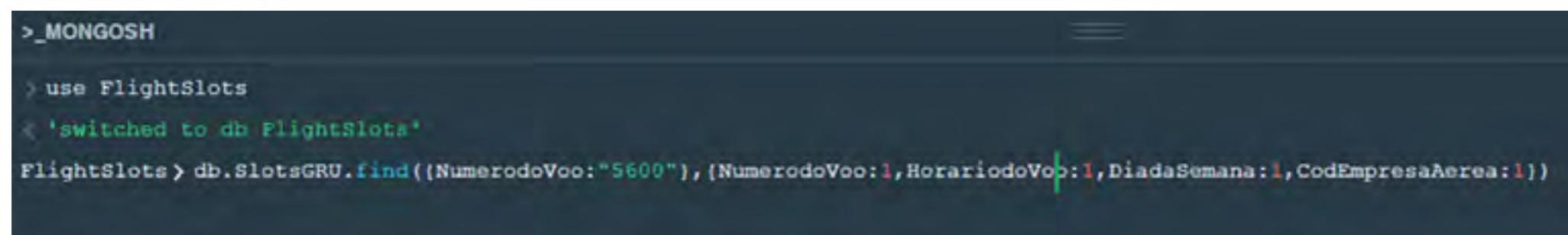
Você lembra da sintaxe do método `find()`? Você consegue construir a sintaxe deste método para que o MongoDB retorne os documentos e os campos de interesse relatados no exemplo?

A sintaxe para que obtenhamos isso é apresentada pela Figura 23.

```
db.SlotsGRU.find( { NumerodoVoo: "5600" }
                  { NumerodoVoo:1, HorariodoVoo:1, DiadaSemana:1, CodEmpresaAerea:1 } )
```

Fonte: Elaborado pelo autor (2021).

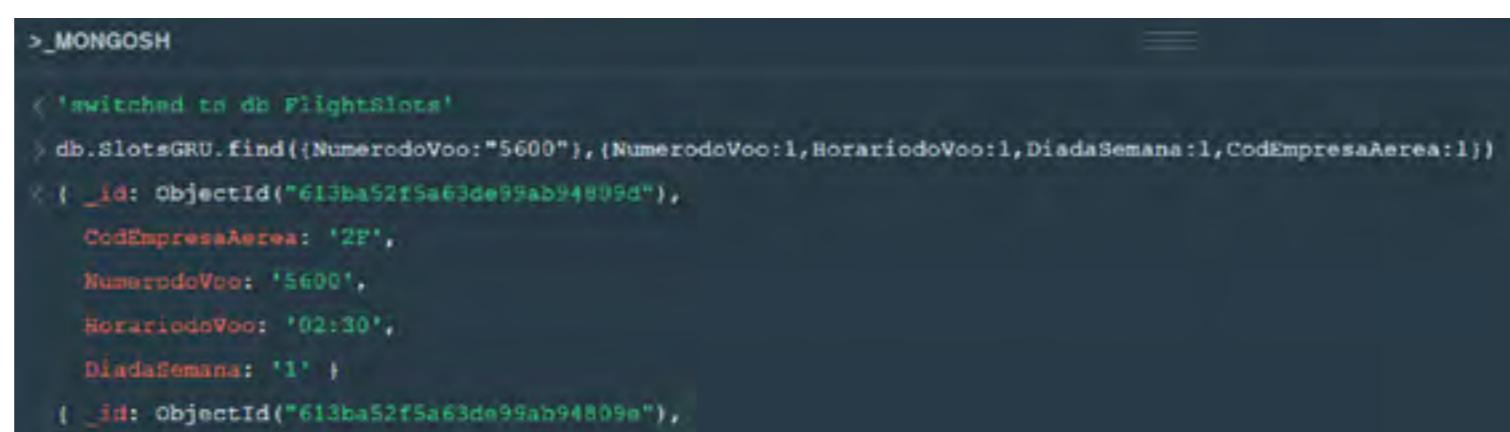
Então, vamos executar este `find()` para a coleção `SlotsGRU`, na janela de execução do MongoDB Shell. A Figura 24 apresenta o método já construído na linha de comando do MongoDB Shell.



```
>_MONGOSH
> use FlightSlots
<'switched to db FlightSlots'
FlightSlots> db.SlotsGRU.find({NumerodoVoo:"5600"}, {NumerodoVoo:1,HorariodoVoo:1,DiadaSemana:1,CodEmpresaAerea:1})
```

Fonte: Adaptada de MongoDB (2021).

Após a execução, para cada documento retornado, você verá os campos selecionados. Por padrão, o MongoDB Shell também apresenta o Id dos documentos. A Figura 25 demonstra os resultados da execução do `find()`.



```
>_MONGOSH
<'switched to db FlightSlots'
> db.SlotsGRU.find({NumerodoVoo:"5600"}, {NumerodoVoo:1,HorariodoVoo:1,DiadaSemana:1,CodEmpresaAerea:1})
< [
  {
    "_id": ObjectId("613ba52f5a63de99ab94809d"),
    "CodEmpresaAerea": "2E",
    "NumerodoVoo": "56001",
    "HorariodoVoo": "02:30",
    "DiadaSemana": "1"
  },
  {
    "_id": ObjectId("613ba52f5a63de99ab94809e"),
    "CodEmpresaAerea": "2E",
    "NumerodoVoo": "56002",
    "HorariodoVoo": "02:30",
    "DiadaSemana": "1"
  }
]
```

Fonte: Adaptada de MongoDB (2021).

Percebeu como o MongoDB varreu toda a base de dados e selecionou os documentos quase que instantaneamente? Navegue pelos resultados apresentados e observe que para `NumerodoVoo = "5600"` há apenas 15 documentos.

Sugerimos que você aproveite a oportunidade e explore melhor o método `find()`, alterando os campos retornados e inclusive aplicando filtros em outros campos de interesse.

Você já sabe inserir e selecionar documentos do MongoDB. No próximo tópico você vai conhecer como alterar um documento já armazenado no banco de dados.

3.10 UPDATE – Alterando os documentos

O MongoDB oferece vários métodos de update para a alteração, tanto de um documento inteiro, quanto de campos em documentos. Entretanto, o método updateMany() atende a grande maioria das situações. Sendo assim, vamos focar neste método. A Figura 26 apresenta a sintaxe mais simples para a utilização deste método.

Sintaxe

```
db.collection.updateMany(<query>, <update>,
  { writeConcern:<document> }
)
```

Exemplo

```
db.pedidos.update(
  // Parâmetros de seleção
  { produto_id: 7239,
    produto_cor: { $eq: "Chumbo" },
    produto_tamanho: { $eq: "39" },
  },
  // Campos a serem alterados
  { $set: { produto_preco: 350 } },
)
```

Fonte: Adaptada de MongoDB (2021).

Explorando a sintaxe:

destacados em vermelho, a coleção e o documento que contém os parâmetros de seleção dos documentos que serão alterados (query). Vale observar que estes filtros são construídos exatamente como no método find();

destacado em azul, o documento que contém os campos a serem alterados e seus novos valores. Este documento pode conter alguns operadores, conforme o que se deseja realizar. Por exemplo, substituir ou adicionar valores aos conteúdos encontrados;

destacada em marrom, a especificação opcional de writeConcern().

Explorando o exemplo:

- destacados em vermelho, a coleção “Pedidos” e o documento que contém os parâmetros de seleção dos documentos a serem alterados. No caso, produto_id = 7239, produto_cor = “Chumbo” e produto_tamanho = “39”;
- destacado em azul, o documento contendo o operador \$set, o qual realiza a substituição do valor anterior pelo novo valor indicado no documento. No caso, o produto_preco terá o novo valor 350;
- todos os pedidos que contiverem o produto do critério de seleção terão seus preços substituídos para 350.

No link a seguir você encontra um guia de referência para a utilização dos métodos de update:

<https://docs.mongodb.com/manual/tutorial/update-documents/>

No próximo tópico, utilizaremos os mesmos critérios de seleção usados na execução prática do método find(), só que alterando alguns destes documentos.

3.11 Executando updateMany()

Ao explorar os resultados retornados ao executar na prática um find() (Tópico 8 – Executando find()), percebemos que a coleção SlotsGRU, para o NumerodoVoo=“5600”, contém:

- documentos de 2 empresas aéreas: As empresas “8I” e “2F”;
- a empresa aérea “8I” opera apenas no DiadaSemana = “2”.

Que tal aplicarmos um updateMany() para alterar este DiadaSemana para 3? A Figura 27 apresenta a construção da sintaxe para updateMany() e o resultado desta alteração.

Após a execução, observe que o MongoDB Shell informa que foram alterados 2 documentos (o que já era esperado), confirmando, assim, que o updateMany() foi executado conforme os critérios propostos.

```
>_MONGOSH
> use FlightSlots
> switched to db FlightSlots
> db.SlotsGRU.updateMany({NumerodoVoo:"5600",CodEmpresaAerea:"8I"},{$set:{DiadaSemana:"3"}})
< { acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0 }
FlightSlots>
```

Fonte: Adaptada de MongoDB (2021).

Agora, falta apenas conhecer como remover os documentos! No próximo tópico você verá como fazer isso. Vamos lá?

3.12 DELETE – Removendo documentos

Da mesma forma que os métodos de update, o MongoDb também oferece vários métodos para a remoção de documentos. Contudo, o método deleteMany() atende a maioria das situações. Sendo assim, este tópico terá foco neste método. A Figura 28 apresenta a sintaxe mais simples para a utilização deste método.

Sintaxe

```
db.collection.deleteMany(<query>
  { writeConcern:<document>}
)
```

Exemplo

```
db.pedidos.deleteMany(
  // Parâmetros de seleção
  { produto_id: 7239,
    produto_cor: { $eq: "Chumbo" },
    produto_tamanho: { $eq: "39" },
  },
)
```

Fonte: Elaborado pela autora (2021).

A sintaxe segue o mesmo padrão dos métodos de escrita do MongoDB:

- destacados em vermelho, a coleção e o documento que contém os critérios de seleção dos documentos a serem alterados (query);
- destacado em marrom, writeConcern(). Quando usado, o documento passado a este método contém os parâmetros que informam como a aplicação deseja o tratar possíveis erros na recepção do comando update pelo servidor;

Explorando o exemplo:

- destacados em vermelho, a coleção “Pedidos” e o documento que define os parâmetros de seleção. No caso, produto_id = 7239, produto_cor = “Chumbo” e produto_tamanho = “39”;
- com esta construção, o MongoDB executará a remoção de todos os documentos que atenderem ao critério de seleção informado.

No link a seguir você encontra um guia de referência para os métodos de update:

<https://docs.mongodb.com/manual/reference/delete-methods/>

Que tal completar seu conhecimento a respeito dos métodos CRUD, colocando em prática o último dos comandos CRUD apresentados nesta unidade: deleteMany()?

3.13 Executando deleteMany()

Vamos evoluir a sua prática usando os mesmos critérios de seleção utilizados para os demais métodos apresentados até o momento. Vamos remover, da coleção SlotsGRU, os slots da empresa aérea “8I”, que tenham o NumerodoVoo=“5600”. Sabemos que existem dois documentos para este critério de seleção. A Figura 29 apresenta a construção da sintaxe e, também, o resultado da remoção com os critérios selecionados.

```
>_MONGOSH  
> use FlightSlots  
< switched to db FlightSlots  
> db.SlotsGRU.deleteMany({NumerodoVoo:"5600",CodEmpresaAerea:"8I"})  
< { acknowledged: true, deletedCount: 2 }>  
FlightSlots >
```

Fonte: Adaptada de MongoDB (2021).

Como já esperávamos, a execução de `deleteMany()` informa que houve a remoção de 2 documentos, confirmando, assim, que o `updateMany()` foi executado conforme o esperado.

Antes de encerrarmos o assunto desta unidade, você precisa conhecer um pouco mais sobre os resultados apresentados na execução dos métodos CRUD. Estes resultados, informados pelo MongoDB, são muito úteis para a construção de aplicações profissionais.

3.14 writeResult – Verificando a execução dos métodos de escrita

As operações de escrita (`insert`, `update`, `remove`) retornam um documento contendo as informações sobre a execução da operação. Este documento pode conter, por exemplo, além de um código relatando o sucesso ou não da operação, a quantidade de linhas inseridas e o identificador gerado pelo MongoDB, caso este não tenha sido informado no documento. A aplicação deve receber estas informações e providenciar o tratamento adequado conforme as informações relatadas.

Estas informações são representadas como propriedades do método `writeResult()` e podemos citar algumas informações muito úteis. O Quadro 02 apresenta algumas destas informações.

Tipo	Descrição
<code>WriteResult.nInserted</code>	Quantidade de documentos inseridos
<code>WriteResult.nMatched</code>	Quantidade de documentos selecionados
<code>WriteResult.nModified</code>	Quantidade de documentos alterados
<code>WriteResult._id</code>	Id de documentos inseridos automaticamente
<code>WriteResult.nRemoved</code>	Quantidade de documentos removidos
<code>WriteResult.writeError</code>	Documento contendo erros encontrados na execução da operação de escrita

Fonte: Elaborado pelo autor (2021).

De posse destas informações, o desenvolvedor pode providenciar o melhor tratamento para o resultado demonstrado.

No link a seguir você encontra um guia de referência para os métodos CRUD MongoDB

<https://docs.mongodb.com/manual/reference/method/WriteResult/#mongodb-method-WriteResult>

No link a seguir, você encontrará um guia de referência para os métodos CRUD MongoDB nas várias plataformas oficiais:

<https://docs.mongodb.com/manual/>

Parabéns, agora você já sabe como inserir, selecionar, alterar e remover documentos em um banco de dados MongoDB. Na próxima unidade, você terá a oportunidade de conhecer como modelar um documento para uma aplicação que se beneficia deste modelo de armazenamento. Até!

Síntese

Nesta unidade, você se aprofundou em como o MongoDB possibilita inserir, ler, alterar e remover os documentos. Verificou que a codificação destes métodos pode variar ligeiramente conforme a plataforma de desenvolvimento e que as execuções exigem que o ambiente da aplicação tenha o serviço do banco de dados ativo e, também, uma conexão com o banco de dados que armazena a coleção a ser trabalhada. Confira a seguir os itens mais importantes:

- O MongoDB possui drivers para as plataformas de desenvolvimento mais utilizadas atualmente. Cada uma destas plataformas possui requisitos próprios de codificação para a preparação do ambiente de execução do método CRUD MongoDB.
- Antes da execução do método CRUD, o desenvolvedor já deve ter providenciado a comunicação com o MongoDB e a conexão com banco de dados que contém a coleção que será manipulada.
- O MongoDB disponibiliza vários métodos para a inserção de documentos. O método mais usado é o método `insertMany()`, onde podemos inserir vários documentos em uma única execução, além de poder ser executado em um laço de transação.
- O MongoDB também disponibiliza vários métodos para a leitura dos documentos. O método `find()` executa, em uma única instrução, a leitura de vários documentos. Estes documentos são disponibilizados em uma área chamada cursor, a qual deve ser definida conforme a plataforma de desenvolvimento em uso. A aplicação realizará o acesso aos documentos retornados a partir desta área do cursor.
- Os métodos de update alteram os documentos armazenados no MongoDB. Dentre eles, o método `updateMany()` executa a alteração de vários documentos em uma única execução.
- O método `deleteMany()` remove, em uma única execução, vários documentos;

Síntese

- A partir do MongoDB Compass é possível acessar a plataforma MongoDB Shell, onde se pode construir os métodos CRUD em linha de comando. O MongoDB Compass também possibilita popular uma coleção através do import de documentos contidos em um arquivo.
- Os métodos writeConcern() e writeResult() fornecem informações sobre execução das operações de escrita no banco (insert, update, delete), sendo que, com estas informações, o desenvolvedor pode codificar o tratamento mais adequado ao resultado relatado por estas informações. A integração entre os serviços do Backend e Frontend pode ser realizada por meio da chamada da Fetch API.

Fullture Insights

- **BSONSPEC.** Especificações BSON. Disponível em: <https://bsonspec.org/spec.html>. Acesso em: 23 set. 2021.
- **MONGODB.** Documentação do banco de dados MongoDB. [S.D.] Disponível em: <https://docs.mongodb.com/guides/>. Acesso em: 23 set. 2021.

FU
L
T
U
RE

www.fullture.com