Programação Orientada e Objetos II

Anhanguera

AVALIE SUA PROFISSÃO

QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:

NOTAS **9 OU 10** SIGNIFICA QUE VOCÊ INDICA

NOTAS **7 OU 8** SIGNIFICA QUE VOCÊ NÃO INDICA

Anhanguera

Anhanguera

A utilização do MongoDB é feita de forma diferente quando comparada aos sistemas de gerenciamento de banco de dados (SGBD) relacionais, pois sua utilização é baseada em uma Interface de Programação de Aplicativos (API, do inglês Application Programming Interface) específica, fornecida pela empresa que desenvolve o MongoDB (PLUGGE; MEMBREY; HAWKINS, 2010).

Para a utilização de banco de dados NoSQL, devem ser consideradas, além de sua estrutura de armazenamento, as formas de acesso (GESSERT et al., 2016). A primeira etapa para a utilização é especificar a conexão com o MongoDB. Nesse momento tanto o banco de dados quanto a collection podem ser selecionados. O Quadro 4.8 apresenta como é feito esse processo de conexão, lembrando que é necessário configurar o eclipse e o projeto via Maven para que as bibliotecas estejam disponíveis – essa classe deve ser criada na pasta src.

No Quadro 4.8, as linhas de 1 a 5 são relacionadas à inclusão das bibliotecas que serão utilizadas por toda a classe. Os atributos entre as linhas 7 a 9 são os tipos necessários para fazer a conexão. O primeiro deles (MongoClient) é utilizado para realizar as conexões com o MongoDB e é responsável por prover o método que faz a seleção do banco de dados que é o próximo atributo (MongoDatabase). Por fim, na linha 9 temos o tipo MongoCollection, que representa as coleções nas quais os dados são armazenados. No construtor, que inicia na linha 10, temos a criação das instâncias. Já na linha 11 é criada uma instância da classe MongoClient, e para isso é passado o endereço do banco (no nosso caso, localhost), e a porta de conexão, que por padrão para o MongoDB é a 27017. Na linha 12, utilizando a instância de MongoClient, é possível selecionar qual base de dados será utilizada com o método getDatabase. Nesse método é passada uma String com o nome do banco a ser utilizado, e ele retorna uma instância de MongoDatabase. Por fim, na linha 13, utilizando a instância de MongoDatabase (database), é possível obter uma instância de MongoCollection com o método getCollection, passando uma String com nome da coleção selecionada ("alunos").

```
1.  import com.mongodb.MongoClient;
2.  import com.mongodb.client.*;
3.  import org.bson.Document;
4.  import static com.mongodb.client.model.Filters.*;
5.  import org.bson.types.ObjectId;

6.  public class ConexaoMongoDB {

7.          private MongoClient mongoClient;
8.          private MongoDatabase database;
9.          private MongoCollection<Document> collection;

10.         public ConexaoMongoDB() {
11.                 mongoClient = new MongoClient
("localhost", 27017);
12.                 database =
mongoClient.getDatabase("baseuniversidade");
13.                 collection =
database.getCollection("alunos");
14.         }
15. }
```

Com a conexão feita, a base de dados selecionada e a coleção definida, é possível executar algumas operações básicas, também conhecidas como CRUD. São elas:

1. Inserção.
2. Seleção.
3. Alteração.
4. Remoção.

A primeira etapa é a inserção. Em um processo utilizando um SGDB relacional, seria necessário definir a estrutura, os tipos de dados e os relacionamentos entre as tabelas. Todavia, utilizando um sistema NoSQL, basta apenas fazer a inserção dos dados, e a estrutura será criada automaticamente, mesmo que depois novos campos sejam criados, pois esses dados podem ter uma estrutura heterogênea (REDMOND; WILSON, 2012). Como exemplo de modelagem de dados, utilizaremos as informações do Quadro 4.9. Nele são definidos dados como o nome do aluno, o curso e o ano de início.

```
{"nome": "Joao Silva",
"curso": "Ciência da Computação",
 "anoInicio": 2018
```

O método insereAluno(), no Quadro 4.10, apresenta como é feita a inserção no MongoDB. Esse método faz parte da classe ConexaoMongoDB do Quadro 4.8. Para esse caso, o método de inserção recebe três parâmetros (nome, curso e ano de ingresso). Para fazer a inserção é preciso criar o registro utilizando a classe Document, que possui um construtor que recebe uma String referenciando a chave e o valor. Esse segundo parâmetro (valor) é um Object (classe superior do Java, sendo que todos os objetos são filhos dessa classe), com isso é possível passar qualquer objeto como valor. Na linha 1 é definido o método que recebe nome, curso e ano de início para ser inserido no banco. A linha 2 apresenta a classe Document, recebendo a relação chave/valor, onde são passados o nome do campo e o valor a ser inserido. Com o método append() são adicionados novos elementos dento da instância doc de Document. Na linha 3, utilizando o método inserOne() da instância de MongoCollection chamada collection, é passada a instância de Document para fazer a inserção. A classe Document pode receber qualquer quantidade de campos, e com ela é possível inserir qualquer combinação de dados.

```
1.  public void insereAluno(String nome, String curso, int
    anoInicio)
    {
2.     Document doc = new Document("nome", nome).append("curso",
                                        curso).append("anoInicio",
    anoInicio);
3.          collection.insertOne(doc);
    }
```

Ao inserir um registro em uma coleção, o próprio SGBD cria um campo extra chamado _id, com tamanho de 12 bytes, que é usado pelo software para gerenciamento. Esse campo possui um valor único, ou seja, seu funcionamento pode ser comparado ao da chave primária em um SGBD relacional.

O próximo passo é a seleção de dados no banco. No caso do MongoDB é possível utilizar um método para buscar as informações. O Quadro 4.11 apresenta como fazer a seleção de todos os dados da coleção selecionada. A linha 1 do Quadro 4.11 apresenta a definição do método. Na linha 2 é definido um objeto chamado “cursor” da classe MongoCursor, que é criado a partir da instância de MongoCollection (collection) utilizando o método find() e o método iterator(). Se não for passado nenhum parâmetro no método find(), são retornados todos os elementos da collection. Na linha 3 foi utilizado o método hasNext() como uma forma de verificar se existem novos registros dentro do objeto da classe MongoCursor. Dentro do while, entre as linhas 4 a 8, é utilizado novamente o objeto cursor para recuperar o Document atual com o método next().

Para cada documento armazenado, é possível acessar seus campos com o método get(). Isso é feito utilizando o objeto chamado “atual” (da classe Document), que busca os itens da collection com o método get() passando o nome da chave como parâmetro. No final do método em finally, o objeto cursor é fechado pelo método close().

```
1.  public void exibeAlunos()
    {
2.          MongoCursor<Document> cursor =
    collection.find().iterator();
            try {
3.              while (cursor.hasNext()) {
4.                  Document atual = cursor.next();
5.
    System.out.println(atual.get("_id"));
6.                  System.out.println(atual.get("nome"));
7.                  System.out.println(atual.get("curso"));
8.
    System.out.println(atual.get("anoInicio"));
9.              }
10.         } finally {
11.             cursor.close ();
12.             }
13. }
```

Fonte: elaborado pelo autor.

Em diversos cenários é necessário fazer uma seleção mais específica de dados, e para isso é necessário passar algum parâmetro no momento de executar o método find() da classe MongoCollection. O Quadro 4.12 apresenta um exemplo de como selecionar um item que deve ser igual a um parâmetro. Repare que no método find(), da linha 2, é utilizado um outro método chamado eq() para verificar se a chave e o valor são encontrados. O primeiro parâmetro é a chave, e o segundo o valor a ser buscado. Esse elemento é um método estático vindo pelo import static com.mongodb.client.model.Filters.*. Note que as inclusões static garantem que você possa utilizar os métodos estáticos sem a necessidade de colocar o caminho completo do método. Entre as linhas 4 a 10 são exibidos os dados do registro, e a linha 13 fecha o objeto da classe MongoCursor.

Anhanguera

```
1.  public void exibeAluno(String nome){
2.     MongoCursor<Document> cursor = collection.
    find(eq("nome", nome)).iterator();
3.        try {
4.          while (cursor.hasNext()) {
5.              Document atual = cursor.next();
6.              System.out.println(atual.get("_id"));
7.              System.out.println(atual.get("nome"));
8.              System.out.println(atual.get("curso"));
9.              System.out.println(atual.get("curso"));
10.             System.out.println(atual.get("anoInicio"));
11.         }
12.       } finally {
13.         cursor.close();
14.       }
15. }
```

No pacote com.mongodb.client.model. Filters existem diversos elementos estáticos que são utilizados para fazer a filtragem de dados. O Quadro 4.13 apresenta alguns desses métodos.

Método	Descrição
`eq(java.lang.String fieldName, TItem value)`	Seleciona os dados que forem iguais ao *value* da chave *fieldname*.
`ne(java.lang.String fieldName, TItem value)`	Seleciona os dados que forem diferentes do *value* da chave *fieldname*.
`lt(java.lang.String fieldName, TItem value)`	Seleciona os dados que forem menores que o *value* da chave *fieldname*.
`lte(java.lang.String fieldName, TItem value)`	Seleciona os dados que forem menores ou iguais ao *value* da chave *fieldname*.
`gte(java.lang.String fieldName, TItem value)`	Seleciona os dados que forem maiores ou iguais ao *value* da chave *fieldname*.
`gt(java.lang.String fieldName, TItem value)`	Seleciona os dados que forem maiores que o *value* da chave *fieldname*.

Para fazer a alteração dos dados é possível utilizar o método updateOne() da classe MongoCollection. O Quadro 4.15 apresenta como fazer essa operação. O método updateOne() na linha 2, precisa de dois parâmetros: o primeiro diz respeito ao registro que se deseja alterar e o segundo, aos dados que serão usados para alterar. Para filtrar o registro a ser alterado (primeiro parâmetro) é usado o método eq(), passando a chave e o valor a ser localizado. Já no segundo parâmetro, um novo objeto Document é instanciado, com o argumento $set, que informa para alterar o campo atual, e o respectivo valor a ser usado para a alteração.

```
1.  public void alteraAluno(String nomeAntigo, String
    nomeNovo)
    {
2       collection.updateOne(eq("nome", nomeAntigo), new
    Document("$set",
    new                 Document("nome", nomeNovo)));
    }
```

O processo para remoção dos dados é feito de maneira semelhante à da atualização, todavia é passado apenas um parâmetro para buscar e selecionar qual registro remover. O Quadro 4.16 apresenta o código necessário para efetuar a remoção. Na linha 1 é apresentado o método que recebe o nome do aluno e na linha 2, o método deleteOne(), que recebe como parâmetro um filtro utilizando o método eq(), comparando o parâmetro nome com a chave do banco de dados.

```
1.  public void removeAluno(String nome) {
2.          collection.deleteOne(eq("nome", nome));
3.  }
```

A alteração ou remoção também pode ser feita utilizando o parâmetro “_id”, que funciona como uma chave primária (valor que não se repete em uma coleção). O Quadro 4.17 apresenta como é feito o processo. Repare que é igual quando utilizado apenas o nome, todavia é passado como parâmetro uma instância da classe ObjectId.

```
1.  public void alteraAluno(ObjectId id, String nomeNovo)
    {
2.      collection.updateOne(eq("_id", id), new
    Document("$set", new Document("nome", nomeNovo)));
    }
3.  public void removeAluno(ObjectId id)
    {
4.      collection.deleteOne(eq("_id", id));
    }
```

Para finalizar esta seção, você vai aprender como usar os métodos para o CRUD, implementados na classe ConexaoMongoDB. Primeiro, lembre-se de abrir uma janela de prompt e carregar o servidor mongod com o comando "C:\Program Files\MongoDB\Server\4.0\bin\mongod.exe". Como um exemplo geral, o Quadro 4.18 apresenta como utilizar a classe de acesso ao MongoDB. A linha 2 faz a conexão com o banco de dados, a linha 32 faz inserção de um aluno, a linha 4 exibe os dados dos alunos (onde é possível ver qual é o “id” de cada objeto), a linha 5 altera um registro e a linha 6 faz a remoção de um registro. Fique atento: quando você inserir um registro, o valor do _id será diferente, e você poderá consultar pelo comando da linha 4.

Anhanguera

```
1.  public static void main(String[] args) {
2.      ConexaoMongoDB c = new ConexaoMongoDB();
3.      c.insereAluno("Pedro Silva", "Ciência da
    Computação", 2016);
4.      c.exibeAlunos();
5.      c.alteraAluno(new ObjectId("5b3ec4603af20f20a8e-
    dcb29"), "Pedro Da Silva");
6.      c.removeAluno(new ObjectId("5b3ec4603af20f20a8e-
    dcb29"));
7.  }
```

Os comandos para manipulação dos dados utilizando MongoDB são mais simples de serem utilizados e não necessitam de uma formalização como os SGDB relacionais. Com eles a forma de acesso se torna mais simples e não são necessários grandes estudos da estrutura da informação.

O trabalho com um banco de dados NoSQL difere de um sistema de gerenciamento de banco de dados relacional no que diz respeito à forma de manipulação dos dados. No NoSQL não é necessário um formalismo como a linguagem SQL, pois é possível fazer o acesso de forma mais simples e fornecido pela API do NoSQL.

Sobre as formas de seleção de dados utilizando o MongoDB, é correto afirmar:

a) Em banco de dados NoSQL não é possível selecionar os dados.

b) O método find(), da classe MongoCollection, recebe um parâmetro que determina o máximo de registro a retornar.

c) A busca de dados é feita pelo método find() da classe MongoBD, e como parâmetro são passados os resultados dos métodos do pacote com.mongodb.client.model.Filters.

d) A busca de dados é feita pelo método find(), da classe MongoCollection, e como parâmetro são passados os resultados dos métodos do pacote com.mongodb.client.model.Filters.

e) O método find() da classe MongoCollection não recebe parâmetros, dessa forma busca sempre retornar todos os dados.

Ao se utilizar a API do MongoDB para manipular os dados é possível fazer alteração, inserção, seleção e remoção dos dados. Essas operações devem ser utilizadas considerando a documentação do projeto. Nesse contexto, avalie as afirmações a seguir:

I. Ao fazer a inserção de dados, deve ser seguida a mesma estrutura definida na configuração inicial, pois os dados devem ter uma estrutura homogênia.

II. O método insertOne() da classe MongoCollection recebe como parâmetro uma instância da classe Document, com isso é possível enviar os dados.

III. Para executar a alteração dos dados utilizando o método updateOne(), da classe MongoCollection, é necessário definir um filtro para informar qual registro alterar.

IV. Ao remover um dado do MongoDB, todos os registros devem receber a atualização em seus índices.

V. O campo id dos registros do MongoDB são úteis para escolher, de forma única, os dados.

Quais das afirmações são corretas?

a) Apenas as afirmações I, II e V.

b) Apenas as afirmações I, II, III e V.

c) Apenas as afirmações II, III e V.

d) Apenas as afirmações II e III.

e) Apenas as afirmações II e IV.