Mongo e Java

Fernando Anselmo
http://fernandoanselmo.orgfree.com/wordpress/

Versão 1.0 em 19/03/2016

Resumo

Atualmente muito se tem comentado sobre bancos de dados não relacionais. O conhecimento deste novo paradigma pode abrir portas e deve ser considerado um fator de extrema importância para garantir uma boa empregabilidade. É sempre importante estar atento a novas tecnologias e como elas resolvem problemas provenientes das limitações existentes. Neste tutorial será visto o que vem a ser o banco NoSQL MongoDB [1] e como é sua utilização com a linguagem de programação Java [2] uma das mais usadas pelo mercado.

1 Parte inicial

MongoDB (de "humongous" - monstruoso) é um Sistema de Banco de dados não relacional, Orientado a Objetos e de fonte aberto. É parte da família de sistemas de Banco de Dados denominados **NoSQL**, ou seja, em vez de armazenar dados em tabelas - como é feito em um banco de dados relacional - armazena seus dados em uma estrutura como JSON, ou seja, documentos com esquemas dinâmicos. Este formato é conhecido como **JSON Binário** ou simplesmente BSON.

O objetivo principal deste banco é promover uma integração mais fácil e rápida com os dados. Resumidamente, o MongoDB possui as seguintes características:

- Escrito em linguagem de programação C++
- Gerencia coleções de documentos BSON formato de intercâmbio de dados usado principalmente como um formato de armazenamento de dados e transferência de rede no banco de dados MongoDB.
- BSON é uma forma binária para a representação de estruturas de dados simples e matrizes associativas (chamados de objetos ou documentos no MongoDB)

1.1 Levantar o Servidor

Como passo inicial é necessário instalar e levantar o servidor, para fins deste tutorial será utilizado o Sistema Operacional Ubuntu [4]:

1. Adicionar a chave de suporte ao pacote: `$ sudo apt-key adv -keyserver hkp://keyserver.ubuntu.com:80 -recv EA312927`

2. Adicionar a lista de arquivo: `$ echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list`
3. Atualizar o repositório de arquivos: `$ sudo apt-get update`
4. Instalar: `$ sudo apt-get install -y mongodb-org`
5. Levantar o servidor: `$ sudo service mongod start`
6. Verificar se está tudo bem: `$ sudo service mongod status`
7. Se quiser parar o servidor: `$ sudo service mongod stop`
8. Acessar a console de comandos: `$ mongod`
9. Para sair da console: `> exit`

1.2 Shell - a console de comandos

O Mongo Shell, também conhecida como Console de Comandos, utiliza uma interatividade entre comandos JavaScript e o MongoDB. Aqui é possível realizar operações administrativas como consultas ou manutenções de dados.

Mostrar as bases de dados existentes:

```
> show dbs
```

Criar (ou mudar) a base de dados para a atual:

```
> use nome_base
```

Mostrar as coleções existentes na base de dados atual:

```
> show collections
```

Inserir (ou alterar caso o objeto tenha sido chamado anteriormente) um documento em uma coleção (se a coleção não existe será criada) na base de dados corrente (db é uma variável interna apontada para a base de dados atual)

```
> db.nome_colecao.save(campo1:valor1, ..., campoN:valorN)
```

Listar os documentos de uma coleção existente na base de dados atual:

```
> db.nome_colecao.find()
```

Eliminar documento(s) de uma coleção existente na base de dados atual:

```
> db.nome_colecao.remove(campo:valor)
```

Apagar uma coleção existente na base de dados atual:

```
> db.nome_colecao.drop()
```

Apagar a base de dados atual:

```
> db.dropDatabase()
```

Para conhecer mais comandos do Mongo Shell, acessar o seguinte endereço:

`https://docs.mongodb.org/manual/mongo/`

1.3 Baixar o Driver Java

Para proceder a conexão com Java, é necessário baixar um driver JDBC (Java Database Connection). Existem vários drivers construídos, porém o driver oficialmente suportado pelo MongoDB se encontra no endereço:

`http://mongodb.github.io/mongo-java-driver`

Na página existe o caminho para localizar o driver no repositório **Maven**. Este é um arquivo compactado em formato JAR.

1.4 Testar a Conexão com o Java

Para testar a conexão entre o MongoDB e o Java. Devem ser realizadas as seguintes tarefas: iniciar o editor de Java, criar um projeto, disponibilizar o arquivo JAR contendo o driver para que o editor possa reconhecê-lo.

Para proceder sua disponibilização para o editor BlueJ [3] no menu principal acesse a opção "tools|preferences", acessar a opção "Libraries", pressionar o botão "Add" e adicionar o arquivo JAR. Sair e entrar novamente no editor.

Para proceder um teste completo se realmente está tudo funcionando, copie e execute a seguinte classe:

```
1  import java.net.UnknownHostException;
2  import com.mongodb.DB;
3  import com.mongodb.MongoClient;
4  import com.mongodb.MongoException;
5  import com.mongodb.client.MongoDatabase;
6  import com.mongodb.client.MongoCollection;
7  import com.mongodb.client.MongoCursor;
8  import org.bson.Document;
9
10 public class TesteMongoDB {
11
12   private MongoCollection<Document> col;
13   private MongoDatabase db;
14   private MongoClient mongo;
15
16   public static void main(String[] args) {
17     new TesteMongoDB().executar();
18   }
19
20   protected boolean conectar() {
21     try {
22       mongo = new MongoClient("localhost", 27017);
23       db = mongo.getDatabase("escola");
24       col = db.getCollection("aluno");
25     } catch (Exception e) {
26       return false;
27     }
28     return true;
29   }
30
31   protected boolean desconectar() {
32     try {
33       mongo.close();
34     } catch (Exception e) {
```

```
35        return false;
36      }
37      return true;
38    }
39
40    public void executar() {
41      if (conectar()) {
42        // Inserir os alunos
43        Document doc = new Document("nome", "Mario da Silva")
44         .append("nota", (int)(Math.random()*10));
45        col.insertOne(doc);
46        doc = new Document("nome", "Aline Moraes")
47         .append("nota", (int)(Math.random()*10));
48        col.insertOne(doc);
49        doc = new Document("nome", "Soraya Gomes")
50         .append("nota", (int)(Math.random()*10));
51        col.insertOne(doc);
52
53        // Listar os Alunos
54        MongoCursor<Document> cursor = col.find().iterator();
55        while (cursor.hasNext()) {
56          doc = cursor.next();
57          System.out.println(doc.get("nome") + ": " + doc.get("nota"));
58        }
59        cursor.close();
60        desconectar();
61      }
62    }
63  }
```

Esta classe adiciona três registros ao banco de dados contendo o nome do aluno e sua nota que é gerada de forma randômica e em seguida procede uma consulta para verificar se os registros foram realmente inseridos. A conexão e a desconexão ao MongoDB foi colocada em métodos separados.

2 Programação Java usando o MongoDB

Nesta seção será visto como via linguagem Java é possível gerenciar os objetos do MongoDB.

Todos os comandos dos exemplos a seguir foram escritos a partir dos objetos existentes no código visto anteriormente, assim sendo, o método executar deve ser reescrito para a seguinte codificação:

```
1  public void executar() {
2    if (conectar()) {
3      \\ inserir aqui o codigo dos exemplos
4      desconectar();
5    }
6  }
```

2.1 Informações dos Objetos

Para obter informações dos os objetos do MongoDB através do Java, é possível utilizar diversas ações.

Listar as bases de dados existentes:

```
for (String s: mongo.listDatabaseNames())
  System.out.println(s);
```

Criar um novo objeto de base de dados pelo seu nome:

```
MongoDatabase db2 = mongo.getDatabase("escola");
```

Verificar quais são as coleções existentes em uma determinada base de dados:

```
for (String s: db.listCollectionNames())
  System.out.println(s);
```

Criar um novo objeto de coleção pelo seu nome e através deste obter a quantidade de registros existentes:

```
MongoCollection<Document> col2 = db.getCollection("aluno");
System.out.println("Total de Documentos:" + col2.count());
```

Obter, em formato JSON, as coleções de uma determinada base de dados (adicionar um import para a classe `com.mongodb.client.ListCollectionsIterable`):

```
ListCollectionsIterable<Document> it = db.listCollections();
MongoCursor<Document> cursor = it.iterator();
while (cursor.hasNext()) {
  System.out.println(cursor.next().toJson());
}
cursor.close();
```

Criar um indice para uma coleção, o parâmetro com valor igual a 1 informa que deve ser ordenado de forma ascendente, para descendente utilizar o valor -1:

```
col.createIndex(new Document("nota", 1));
```

Obter, em formato JSON, os índices de uma determinada coleção (adicionar um import para a classe `com.mongodb.client.ListIndexesIterable`):

```
ListIndexesIterable<Document> it = col.listIndexes();
MongoCursor<Document> cursor = it.iterator();
while (cursor.hasNext()) {
  System.out.println(cursor.next().toJson());
}
cursor.close();
```

Eliminar um indice de uma coleção:

```
col.dropIndex(new Document("nota", 1));
```

Obter, em formato JSON, os registros de uma determinada coleção:

```
MongoCursor<Document> cursor = col.find().iterator();
while (cursor.hasNext()) {
  System.out.println(cursor.next().toJson());
}
cursor.close();
```

2.2 Filtrar Coleções

Considere o último trecho de código visto para os próximos exemplos.

Limitar a quantidade de registros retornados (por exemplo 3 registros):

```
MongoCursor<Document> cursor = col.find().limit(3).iterator();
```

Trazer os alunos que obtiveram nota 10:

```
MongoCursor<Document> cursor = col.find(new Document("nota", 10)).iterator();
```

Importar a classe `com.mongodb.client.model.Filters`, e com ela é possível realizar a mesma ação:

```
MongoCursor<Document> cursor = col.find(Filters.eq("nota", 10)).iterator();
```

Ainda com esta classe é possível realizar as seguintes ações:

Filters.ne obter registros não iguais a um determinado valor

Filters.gt obter registros maiores que um determinado valor

Filters.gte obter registros maiores ou iguais a um determinado valor

Filters.lt obter os registros menores que um determinado valor

Filters.lte obter os registros menores ou iguais a um determinado valor

Também é possível utilizar as variáveis: $eq (igual), $ne (não igual), $gt (maior), $gte (maior ou igual), $lt (menor) ou $lte (menor ou igual). Para obter todos os documentos da coleção com a nota é maior que 6:

```
MongoCursor<Document> cursor = db.find(
 new Document("nota", new Document("$gt",6))).iterator();
```

Parece mais complicado, porém é possível criar separadamente um objeto Documento e a partir dele compor combinações. Para obter todos os documentos cujas notas são maiores que 3 e menores que 9:

```
Document doc = new Document();
doc.append("nota", new Document("$gt", 3).append("$lt", 9));
MongoCursor<Document> cursor = col.find(doc).iterator();
```

Para realizar a mesma consulta com a utilização dos filtros:

```
MongoCursor<Document> cursor = col.find(
 Filters.and(Filters.gt("nota", 3), Filters.lt("nota", 9))).iterator();
```

2.3 Ordenações

Importar a classe `com.mongodb.client.model.Sorters`, e podemos utilizar as variáveis "ascending" e "descending" para obter ordenações:

```
MongoCursor<Document> cursor =
 col.find().sort(Sorts.ascending("nota")).iterator();
```

2.4 Modificar Coleções

Uma vez identificado o(s) documento(s) desejado(s) é possível proceder:

- Alterações. Utilizar os métodos updateOne ou updateMany.
- Eliminações. Utilizar os métodos deleteOne ou deleteMany.

Também é possível enviar o comando de consulta e alteração unidos, por exemplo para modificar a nota do aluno "Mario da Silva" para 5:

```
1 col.updateOne(new Document("nome","Mario da Silva"),
2  new Document("$set", new Document("nota", 5)));
```

Para eliminar o aluno "Mario da Silva":

```
1 col.deleteMany(new Document("nome","Mario da Silva"));
```

2.5 Eliminar os Objetos

Para eliminar a coleção "aluno":

```
1 col.drop();
```

Para eliminar a base de dados "escola":

```
1 db.drop();
```

3 Conclusão

Como visto o banco de dados MongoDB pode ser facilmente utilizado com aplicações em linguagem Java com o poder de substituir os bancos de dados relacionais, sendo que a grande motivação para NoSQL é resolver o problema de escalabildade dos bancos tradicionais.

Sou um entusiasta do mundo Open Source e os bancos NoSQL está bastante ligado, basta observar bancos como Hadoop, CouchDB ou Cassandra. Veja outros artigos que publico sobre tecnologia acessando meu Blog Oficial [5].

Referências

[1] Página do Banco MongoDB
`https://www.mongodb.org/`

[2] Página do Oracle Java
`http://www.oracle.com/technetwork/java/`

[3] Editor BlueJ para códigos Java
`http://www.bluej.org/`

[4] Instalação MongoDB em outros sistemas operacionais
`https://docs.mongodb.org/manual/installation/`

[5] Fernando Anselmo - Blog Oficial de Tecnologia
`http://www.fernandoanselmo.blogspot.com.br/`