

Módulo 17 – Banco de dados não relacional – Atividade 03

Q1) Utilizando o Mongosh, execute os procedimentos seguintes no MongoDB. Envie um arquivo pdf que mostre seus passos de maneira que seja verificável o que você fez e que deu certo. Os procedimentos são:

- I. Pense em um “domínio”, ou seja, um negócio qualquer (e-commerce, banco, eSports, etc.) e crie um database para o mesmo.
- II. Pense numa entidade para esse domínio (clientes, conta bancária, game, etc) e crie uma collection, inserindo no mínimo 10 documentos na mesma.
- III. Mostre como fazer dois tipos de consulta nessa collection:
 - i. Listar todos os documentos existentes.
 - ii. Listar documentos que atendem a um operador de consulta. O operador é arbitrário (exemplo: “contas bancárias onde o saldo é maior que 1000”).
- IV. Mostre como fazer dois tipos de atualização de documento:
 - i. Atualizar um único documento. A atualização é arbitrária (exemplo: “mudar o saldo da conta bancária para 2000”).
 - ii. Atualizar vários documentos de uma só vez.
- V. Exclua um único documento e, logo depois, exclua todos os documentos da collection.

Respostas

Q1)

Item I:

O domínio escolhido foi um banco. Para iniciar o serviço do MondoDB, aplicou-se o comando a seguir:

```
sudo systemctl start mongod
```

Em seguida, para confirmar que o serviço estava ativo, aplicou-se esse comando:

```
sudo systemctl status mongod
```

```
letonio@letonio-Inspiron-15-3567:~$ sudo systemctl start mongod
letonio@letonio-Inspiron-15-3567:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor prese>
   Active: active (running) since Mon 2023-06-05 10:12:44 -03; 2min 8s ago
     Docs: https://docs.mongodb.org/manual
    Main PID: 14595 (mongod)
      Memory: 230.7M
         CPU: 2.529s
    CGroup: /system.slice/mongod.service
            └─14595 /usr/bin/mongod --config /etc/mongod.conf

jun 05 10:12:44 letonio-Inspiron-15-3567 systemd[1]: Started MongoDB Database S>
jun 05 10:12:47 letonio-Inspiron-15-3567 mongod[14595]: {"t":{"$date":"2023-06->
lines 1-12/12 (END)
```

Nota-se que o serviço está ativo. Para interagir com o banco de dados, utiliza-se a CLI (Command Line Interface ou Interface de linha de comando, em português) “mongosh”.

mongosh

```
letonio@letonio-Inspiron-15-3567:~$ mongosh
Current Mongosh Log ID: 647de2d2d6d453b416f16894
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&server
electionTimeoutMS=2000&appName=mongosh+1.9.1
Using MongoDB:      6.0.6
Using Mongosh:      1.9.1
```

Para criar um banco de dados que represente o nosso domínio, aplica-se o comando a seguir:

use letsbankdb

```
test> use letsbankdb
switched to db letsbankdb
letsbankdb> 
```

O nosso banco dados está vazio, por isso, mesmo usando o comando “show dbs”, o database não vai aparecer.

```
letsbankdb> show dbs
admin    132.00 KiB
config  108.00 KiB
local    72.00 KiB
letsbankdb> 
```

Item II:

No MongoDB, uma entidade é representada por uma collection (coleção, em português). Cada collection possui vários documentos, onde cada documento é uma instância de dados no formato JSON. Vamos criar uma collection chamada “clients” que armazena documentos relacionados a cada cliente que faz parte do nosso banco (letsbankdb). A seguir, apresenta-se o modelo de documento que será empregado na collection de “clients”:

```
{
  firstName: "Jose",
  lastName: "Silva",
  cpf: "123.456.789-10",
  email: "josesilva@mail.com",
  location: {
    cep: "55490-123",
    uf: "PE",
    city: "Recife",
    address: "R. Boa viagem",
    number: 12
  },
  account: {
    idAccount: 12345,
    balance: 3000,
    type: "poupança"
  }
}
```

Para criar uma collection e adicionar um documento, podemos usar esse comando:

```
db.clients.insertOne( {
  firstName: "Jose",
  lastName: "Silva",
  cpf: "123.456.789-10",
  email: "josesilva@mail.com",
  location: {
    cep: "55490-123",
    uf: "PE",
    city: "Recife",
    address: "R. Boa viagem",
    number: 12
  },
  account: {
    idAccount: 12345,
    balance: 3000,
    type: "poupança"
  }
});
```

```
letsbankdb> db.clients.insertOne( {
...   firstName: "Jose",
...   lastName: "Silva",
...   cpf: "123.456.789-10",
...   email: "josesilva@mail.com",
...   location: {
...     cep: "55490-123",
...     uf: "PE",
...     city: "Recife",
...     address: "R. Boa viagem",
...     number: 12
...   },
...   account: {
...     idAccount: 12345,
...     balance: 3000,
...     type: "poupança"
...   }
... } );
{
  acknowledged: true,
  insertedId: ObjectId("647df09dd6d453b416f16895")
}
letsbankdb> █
```

Pede-se que seja inserido pelo menos 10 documentos. É possível inserir vários documentos de uma vez através da função `db.letsbanckbd.insertMany()`. A seguir apresenta-se os dados inseridos:

```
db.clients.insertMany([
{
  firstName: "João",
  lastName: "Oliveira",
```

```
    cpf: "222.333.444-55",
    email: "joao.oliveira@mail.com",
    location: {
      cep: "12345-678",
      uf: "SP",
      city: "São Paulo",
      address: "Rua B",
      number: 2
    },
    account: {
      idAccount: 124,
      balance: 8000,
      type: "corrente"
    }
  },
  {
    firstName: "Ana",
    lastName: "Santos",
    cpf: "333.444.555-66",
    email: "ana.santos@mail.com",
    location: {
      cep: "87654-321",
      uf: "RJ",
      city: "Rio de Janeiro",
      address: "Rua C",
      number: 3
    },
    account: {
      idAccount: 125,
      balance: 3000,
      type: "poupança"
    }
  },
  {
    firstName: "Mario",
    lastName: "Santos",
    cpf: "111.333.444-55",
    email: "mario.santos@mail.com",
    location: {
      cep: "12345-678",
      uf: "SP",
      city: "São Paulo",
      address: "Rua tal",
      number: 10
    },
    account: {
      idAccount: 661,
      balance: 7000,
      type: "corrente"
    }
  },
  {
```

```
firstName: "Julia",
lastName: "Ferreira",
cpf: "223.244.555-66",
email: "julia.ferreira@mail.com",
location: {
  cep: "34454-321",
  uf: "PE",
  city: "Caruaru",
  address: "Rua de lá",
  number: 31
},
account: {
  idAccount: 126,
  balance: 3000,
  type: "poupança"
}
},
{
  firstName: "Maria",
  lastName: "Fernandes",
  cpf: "222.333.123-55",
  email: "maria.fe@mail.com",
  location: {
    cep: "12345-678",
    uf: "SP",
    city: "São Paulo",
    address: "Rua mesmo",
    number: 24
  },
  account: {
    idAccount: 1246,
    balance: 6500,
    type: "corrente"
  }
},
{
  firstName: "Natalia",
  lastName: "Silva",
  cpf: "333.404.555-12",
  email: "natalia.silva@mail.com",
  location: {
    cep: "12543-222",
    uf: "RJ",
    city: "Rio de Janeiro",
    address: "Rua barra",
    number: 333
  },
  account: {
    idAccount: 125123,
    balance: 5400,
    type: "poupança"
  }
}
```

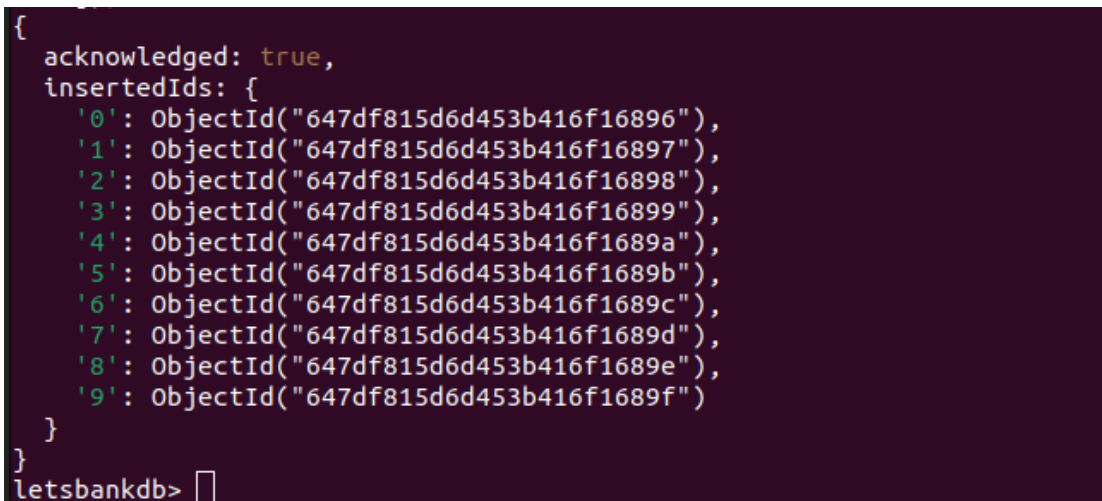
```
},
{
  firstName: "Otávio",
  lastName: "Leite",
  cpf: "245.312.074-55",
  email: "ota.leite@mail.com",
  location: {
    cep: "65655-555",
    uf: "CE",
    city: "Fortaleza",
    address: "Rua Castelão",
    number: 2221
  },
  account: {
    idAccount: 122310,
    balance: 6750,
    type: "corrente"
  }
},
{
  firstName: "Pedro",
  lastName: "Lopes",
  cpf: "102.474.585-66",
  email: "plopes@mail.com",
  location: {
    cep: "12345-321",
    uf: "RJ",
    city: "Rio de Janeiro",
    address: "Rua Chave",
    number: 9
  },
  account: {
    idAccount: 123321,
    balance: 1200,
    type: "poupança"
  }
},
{
  firstName: "Raquel",
  lastName: "Braga",
  cpf: "567.654.443-09",
  email: "raquelbraga@mail.com",
  location: {
    cep: "15543-008",
    uf: "SP",
    city: "São Paulo",
    address: "Rua Liberdade",
    number: 42
  },
  account: {
    idAccount: 1004,
    balance: 8650,
```

```

    type: "corrente"
  },
  {
    firstName: "Tomas",
    lastName: "Solto",
    cpf: "321.123.567-60",
    email: "tsolto@mail.com",
    location: {
      cep: "87654-231",
      uf: "PE",
      city: "Santa Cruz",
      address: "Rua tal tal",
      number: 3
    },
    account: {
      idAccount: 12005,
      balance: 3450,
      type: "poupança"
    }
  }
];

```

Aplicando-se o comando acima no terminal, o resultado é o seguinte:



```

{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("647df815d6d453b416f16896"),
    '1': ObjectId("647df815d6d453b416f16897"),
    '2': ObjectId("647df815d6d453b416f16898"),
    '3': ObjectId("647df815d6d453b416f16899"),
    '4': ObjectId("647df815d6d453b416f1689a"),
    '5': ObjectId("647df815d6d453b416f1689b"),
    '6': ObjectId("647df815d6d453b416f1689c"),
    '7': ObjectId("647df815d6d453b416f1689d"),
    '8': ObjectId("647df815d6d453b416f1689e"),
    '9': ObjectId("647df815d6d453b416f1689f")
  }
}
letsbankdb> 

```

Item III:

Para listar todos os documentos existentes, basta aplicar `db.clients.find()`, sem nenhum parâmetro, pois inserir parâmetros fará uma filtragem. Para a primeira situação, basta executar o comando sem parâmetro algum:

```
db.clients.find();
```

```

letsbankdb> db.clients.find()
[
  {
    _id: ObjectId("647df09dd6d453b416f16895"),
    firstName: 'Jose',
    lastName: 'Silva',
    cpf: '123.456.789-10',
    email: 'josesilva@mail.com',
    location: {
      cep: '55490-123',
      uf: 'PE',
      city: 'Recife',
      address: 'R. Boa viagem',
      number: 12
    },
    account: { idAccount: 12345, balance: 3000, type: 'poupança' }
  },
  {
    _id: ObjectId("647df815d6d453b416f16896"),

```

Note que existe um campo que não foi adicionado por nós, chamado `_id`, que possui um identificador único para cada documento.

Para fazer uma consulta específica, faz-se necessário adicionar um ou mais filtros de pesquisa. Para esse caso, vamos aplicar uma consulta que retorna todas as contas com saldo superior a 7900 reais. Para tal, utilizamos o comando a seguir:

```
db.clients.find({ "account.balance": { $gt: 7900 } });
```

No comando acima, `$gt` indica um query selector que significa valores maiores do que o que foi passado, ou seja, será retornado todos os documentos cuja propriedade `balance` tem valor maior do que (greater than) 7900. Há diversos query selector que podem ser empregados (ver <https://www.mongodb.com/docs/manual/reference/operator/query/#query-selectors>). Outro ponto importante é que nesse caso, como estamos acessando uma chave que é interna a outra chave, faz-se necessário o uso de aspas ("**account.balance**"). Caso essas chaves sejam omitidas, o mongosh emitirá um erro.

O resultado da query retorna 2 documentos:

```

letsbankdb> db.clients.find({ "account.balance": { $gt: 7900 } });
[
  {
    _id: ObjectId("647df815d6d453b416f16896"),
    firstName: 'João',
    lastName: 'Oliveira',
    cpf: '222.333.444-55',
    email: 'joao.oliveira@mail.com',
    location: {
      cep: '12345-678',
      uf: 'SP',
      city: 'São Paulo',
      address: 'Rua B',
      number: 2
    },
    account: { idAccount: 124, balance: 8000, type: 'corrente' }
  },
  {
    _id: ObjectId("647df815d6d453b416f1689e"),
    firstName: 'João',

```



```

},
{
  _id: ObjectId("647df815d6d453b416f1689e"),
  firstName: 'Raquel',
  lastName: 'Braga',
  cpf: '567.654.443-09',
  email: 'raquelbraga@mail.com',
  location: {
    cep: '15543-008',
    uf: 'SP',
    city: 'São Paulo',
    address: 'Rua Liberdade',
    number: 42
  },
  account: { idAccount: 1004, balance: 8650, type: 'corrente' }
}
]
letsbankdb> 

```

Nota-se que o resultado está coerente com o esperado, pois uma conta tem balance 8000 e a outra 8650 (ambos acima de 7900).

Item IV:

Esse item pede que seja feita alguma atualização. Para a primeira situação, pede-se que seja alterado apenas um documento. Escolhemos alterar o saldo da conta que tem CPF: 321.123.567-60. Supondo que ele fez uma retirada de 1000 reais, o seu saldo deixa de ser 3450 e passará a ser 2450. Para isso, executa-se o comando a seguir:

```
db.clients.updateOne({ cpf: "321.123.567-60" }, { $set: { "account.balance": 2450 } });
```

No comando acima, primeiro passa-se um filtro de busca que permite determinar qual o documento que desejamos alterar. Em seguida, explica-se o que desejamos que seja alterado. Nesse caso foi uma mudança de saldo, atualizando-o para 2450. Ao executar esse comando, a resposta obtida no terminal é a seguinte:

```

letsbankdb> db.clients.updateOne({ cpf: "321.123.567-60" },
... { $set: { "account.balance": 2450 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
letsbankdb> 

```

Nota-se que foi encontrado 1 com as características e um documento foi modificado. Para confirmar as alterações, vamos usar o comando de consulta, pesquisando pelo cpf, através do comando a seguir:

```

letsbankdb> db.clients.find({ cpf: "321.123.567-60" });
[
  {
    _id: ObjectId("647df815d6d453b416f1689f"),
    firstName: 'Tomas',
    lastName: 'Solto',
    cpf: '321.123.567-60',
    email: 'tsolto@mail.com',
    location: {
      cep: '87654-231',
      uf: 'PE',
      city: 'Santa Cruz',
      address: 'Rua tal tal',
      number: 3
    },
    account: { idAccount: 12005, balance: 2450, type: 'poupança' }
  }
]
letsbankdb> 

```

O próximo passo é atualizar vários documentos de uma vez. Nesse caso, utiliza-se a função `db.clients.updateMany()`. Aplicaremos a mudança de que todas as contas que possuam saldo (balance) menor que 3000 vão passar a ter saldo igual a 3000. Para tal, aplica-se o comando a seguir:

```

db.clients.updateMany({ "account.balance": { $lt: 3000 } },
  { $set: { "account.balance": 3000 } });

```

```

letsbankdb> db.clients.updateMany({ "account.balance": { $lt: 3000 } },
... {
...   $set: { "account.balance": 3000 }
... });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
letsbankdb> 

```

Nota-se que o update afetou dois documentos.

Item V:

Vamos excluir o documento cujo CPF é "321.123.567-60". Para isso, aplica-se esse comando:

```

db.clients.deleteOne ( {cpf: "321.123.567-60"} );

```

```

letsbankdb> db.clients.deleteOne ( {cpf: "321.123.567-60"} );
{ acknowledged: true, deletedCount: 1 }
letsbankdb> 

```

Aplicando o filtro de busca baseado no CPF, temos que:

db.clients.find({ cpf: "321.123.567-60" });

```
{ acknowledged: true, deletedCount: 1 }  
letsbankdb> db.clients.find({ cpf: "321.123.567-60" });  
letsbankdb> 
```

Nota-se que o resultado foi vazio, confirmando a exclusão.

Por fim, pede-se para que todos os documentos da collection sejam excluídos. Para isso, aplica-se esse comando:

db.clients.deleteMany ({});

```
{ acknowledged: true, deletedCount: 10 }  
letsbankdb> db.clients.deleteMany ({});  
{ acknowledged: true, deletedCount: 10 }  
letsbankdb> 
```

Ao todo, tínhamos 11 documentos, sendo um excluído primeiro e, nesse último comando, houve a exclusão dos outros 10.

Para confirmar que não tem mais documentos na collection, podemos executar a consulta sem passar filtro:

db.clients.find();

```
{ acknowledged: true, deletedCount: 10 }  
letsbankdb> db.clients.find();  
letsbankdb> 
```

Nota-se que não retornou documento algum. Finalizando a atividade.