PÓS-GRADUAÇÃO EM ENGENHARIA DE DADOS

IEC PUC MINAS

Disciplina:

BANCOS DE DADOS NoSQL

Professor: Augusto Zadra

2.5 MongoDB CRUD

INTRODUÇÃO

- CRUD é a composição da primeira letra de 4 transações básicas que um sistema faz quando trabalha com banco de dados:
  - ✓ C (Create): criar um novo registro.
  - ✓ R (Read): exibir as informações de um registro.
  - ✓ U (Update): atualizar os dados do registro.
  - ✓ D (Delete): apagar um registro.

PUC Minas

Bancos de dados NoSQL

mongoDB

Create

- Operações de **Create** irão inserir o novo documento na coleção.
- Se uma coleção não existir, o MongoDB criará uma nova coleção e inserirá um documento nela. O MongoDB fornece os seguintes métodos para inserir um documento no banco de dados:

  - ✓ db.collection.insertOne();
  - ✓ db.collection.insertMany();

PUC Minas

Bancos de dados NoSQL

mongoDB

Create

- **db.collection.insertOne():** função que adiciona um documento a uma coleção.

```
> db.MegaSena.NumerosSorteados.insertOne({
    "Numero Sorteio" : NumberInt(1),
    "Bola1"          : NumberInt(2),
    "Bola2"          : NumberInt(3),
    "Bola3"          : NumberInt(5),
    "Bola4"          : NumberInt(6),
    "Bola5"          : NumberInt(23),
    "Bola6"          : NumberInt(35)
 });
```

- **db.collection.insertMany():** função que pode inserir vários documentos na coleção ao mesmo tempo. Precisamos passar um array para inserção.

```
> db.MegaSena.NumerosSorteados.insertMany([
{"NumeroSorteio" :NumberInt(2), "Bola1"  : NumberInt(09),  "Bola2"  : NumberInt(15) , "Bola3"  : NumberInt(30) , "Bola4"  : NumberInt(33) , "Bola5"  : NumberInt(35) , "Bola6"  : NumberInt(59) , "inserted" : Date()},
{"NumeroSorteio" :NumberInt(3), "Bola1"  : NumberInt(16),  "Bola2"  : NumberInt(23) , "Bola3"  : NumberInt(23) , "Bola4"  : NumberInt(29) , "Bola5"  : NumberInt(37) , "Bola6"  : NumberInt(52) , "inserted" : Date()},
{"NumeroSorteio" :NumberInt(4), "Bola1"  : NumberInt(02),  "Bola2"  : NumberInt(08) , "Bola3"  : NumberInt(14) , "Bola4"  : NumberInt(38) , "Bola5"  : NumberInt(44) , "Bola6"  : NumberInt(55) , "inserted" : Date()},
{"NumeroSorteio" :NumberInt(5), "Bola1"  : NumberInt(07),  "Bola2"  : NumberInt(14) , "Bola3"  : NumberInt(29) , "Bola4"  : NumberInt(45) , "Bola5"  : NumberInt(49) , "Bola6"  : NumberInt(60) , "inserted" : Date()},
{"NumeroSorteio" :NumberInt(6), "Bola1"  : NumberInt(11),  "Bola2"  : NumberInt(20) , "Bola3"  : NumberInt(32) , "Bola4"  : NumberInt(49) , "Bola5"  : NumberInt(50) , "Bola6"  : NumberInt(59) , "inserted" : Date()}
]);
```

PUC Minas

Bancos de dados NoSQL

mongoDB

Read

- Operações de leitura recuperam documentos ou dados de documentos na coleção.
- Para recuperar todos os documentos de uma determinada coleção, passe um documento vazio como filtro.

```
> db.MegaSena.NumerosSorteados.find({});
                    OU
> db.MegaSena.NumerosSorteados.find();
```

- Para uma visualização indentada utilize:

```
> db.MegaSena.NumerosSorteados.find().pretty();
```

PUC Minas

Bancos de dados NoSQL

mongoDB

filtros

- Conhecendo a estrutura do documento retornada pelo comando find(), poderemos aplicar filtros.
- Para isto precisaremos passar a chave de busca como parâmetro.
- Podemos aplicar parâmetros condicionais ao recuperar dados de coleções, como IN, AND e OR com condições de menor que e maior que.

```
> db.MegaSena.NumerosSorteados.find( {"Bola1": 7}).pretty();
```

- E aplicando condições lógicas podemos ampliar o filtro:

```
> db.MegaSena.NumerosSorteados.find( {"Bola1": { $in : [9,7,11,12,15] }}).pretty();
```

PUC Minas

Bancos de dados NoSQL

mongoDB

filtros

- Aplicando a condição AND nos campos do mesmo documento utilizamos:

```
> db.MegaSena.NumerosSorteados.find( {
                                      "Bola1": { $in : [9,7,11,12,15] },
                                      "Bola2": {$in: [20,14]}
                                      }).pretty();
```

- E podemos aplicar filtros de comparação entre eles o less than (menor que):

```
> db.MegaSena.NumerosSorteados.find( {
                                      "Bola1": { $in : [9,7,11,12,15] },
                                      "Bola2": {$in: [20,14]},
                                      "Bola3": { $lt:20 }
                                      }).pretty();
```

PUC Minas

Bancos de dados NoSQL

mongoDB

filtros

- **$lte**: Less Than equal, **$gt** Greater than e **$gte** Greater than equal também são critérios de restrição que podem ser aplicados às consultas:

```
> db.MegaSena.NumerosSorteados.find( {
                                    "NumeroSorteio" : {$gte: 1, $lte: 30 },
                                     "Bola1": { $in : [9,7,11,12,15] },
                                          $or :[
                                                { "Bola2": {$in : [8,14,15] } },
                                                { "Bola5": { $in : [9,7,11,12,15] } }

                                              ],
                                     "Bola3" : { $gt : 29}
                                        }).pretty();
```

- **$ne** e **$nin** são critérios de negação para as cláusulas já conhecidas.

```
> db.MegaSena.NumerosSorteados.find( { "NumeroSorteio" : {$ne: 1 },
                                       $and : [{"Bola1":
                                                      { $nin : [9,7,11,12,15] }
                                                }]
                                    }).pretty();
```

PUC Minas

Bancos de dados NoSQL

mongoDB

filtros

- É possível e bastante aplicável, utilizarmos consultas de identificação de padrões em nossos **documents** e **collections.**
- **$regexp** fornece recursos de expressão regular para strings de correspondência de padrões em consultas.

```
> db.usuario.find({"message": {$regex: /ganhou/ }}).pretty();
```

```
> db.usuario.find({"nome": {$regex: /ad/ }}).pretty();
```

```
> db.usuario.find({"nome": {$regex: /ad/i }}).pretty();
```

PUC Minas

Bancos de dados NoSQL

mongoDB

Filtros

Cláusulas $where

- Para consultas que não podem ser feitas de outra forma, existem cláusulas **$where**, que permitem executar *JavaScript* customizado como parte de sua consulta.
- Isso permite que você faça **quase** qualquer coisa dentro de uma consulta.
- Porém é recomendado que seu uso seja altamente restrito ou eliminado.
- Os usuários finais nunca devem ter esta permissão de execução.

```
> db.MegaSena.NumerosSorteados.find({"$where" : function () {
                                for (var current in this) {
                                    for (var other in this) {
                                        if (current != other && this[current] == this[other]) {
                                            return true;
                                        }
                                    }
                                }
                                return false;
                                }});
```

PUC Minas

Bancos de dados NoSQL

mongoDB

Opções de consulta

- As opções de consulta mais comuns são limitar o número de resultados retornados, pular vários resultados e classificar.
- Todas essas opções devem ser adicionadas antes que uma consulta seja enviada ao banco de dados.
- Para limitar os registros iremos utilizar a função **limit**, e aí melhoraremos a linguagem, você encadeará a função **limit** com a função **find:**

```
> db.usuario.find({"nome": {$regex: /ad/i }}).pretty().limit(1);
```

PUC Minas

Bancos de dados NoSQL

mongoDB

Opções de consulta

- Pular funciona de maneira semelhante para limitar e é interessante quando sabemos a posição que queremos e conhecemos a ordem de inserção deles.

```
> db.usuario.find({"nome": {$regex: /ad/i }}).pretty().skip(1);
```

- A ordenação é feita utilizando-se o método **sort** que recebe dois parâmetros onde as chaves são nomes de chaves e os valores são as direções de classificação.
- A direção de classificação pode ser 1 (crescente) ou -1 (decrescente).
- Se várias chaves forem fornecidas, os resultados serão classificados nessa ordem.

PUC Minas

Bancos de dados NoSQL

mongoDB

Opções de consulta

- Pular funciona de maneira semelhante para limitar e é interessante quando sabemos a posição que queremos e conhecemos a ordem de inserção deles.

```
> db.usuario.find({"nome": {$regex: /ad/i }}).pretty().skip(1);
```

- A ordenação é feita utilizando-se o método **sort** que recebe dois parâmetros onde as chaves são nomes de chaves e os valores são as direções de classificação.
- A direção de classificação pode ser 1 (crescente) ou -1 (decrescente).
- Se várias chaves forem fornecidas, os resultados serão classificados nessa ordem.

PUC Minas

Bancos de dados NoSQL

mongoDB

Opções de consulta

- Desta forma pode-se aplicar um conjunto de critérios para que obtenhamos os documentos com os critérios desejados:

```
> db.restaurants.find(
                    {"cuisine" : "American"}
                    ).pretty().sort(
                                        {"name" : 1, "borough" :-1});
```

Bancos de dados NoSQL

mongoDB

Update

- A atualização de documentos possui também algumas particularidades.
- Os métodos mais importantes são

✓db.collection.updateOne (<filter>, <update>, <options>);

✓db.collection.updateMany (<filter>, <update>, <options>);

✓db.collection.replaceOne (<filter>, <update>, <options>);

Nunca se esqueça de passer os parâmtreos de filtros para atualização dos registros.

Bancos de dados NoSQL

mongoDB

Update

- A atualização de documentos possui também algumas particularidades.
- Existem as opções Upser e Multi que quando apontadas mudam o comportamento dos métodos.
- **Upsert:** se definido como true, cria um novo documento quando nenhum documento corresponde aos critérios de consulta.
- **Multi:** se definido como true, atualiza vários documentos que atendem aos critérios de consulta. Se definido como false, atualiza um documento.

Bancos de dados NoSQL

mongoDB

Update

- Devemos ficar atentos à passagem de parâmetros para que não haja criação de registros desnecessariamente.

```
> db.usuario.update( {} , {$set: { "GostaDoCampeao": [] }},
                           {upsert: false, multi: true} );
```

- Se eu precisar mudar o tipo de dados devo alterar o comando:

```
> db.usuario.update( {} , {$set: { "GostaDoCampeao": "" }},
                           {upsert: false, multi: true} );
```

- E posso definir como **null** quando ainda não sei qual será o tipo de dados que vou utilizar.

```
> db.usuario.update( {} , {$set: { "GostaDoCampeao": null }},
                           {upsert: false, multi: true} );
```

Bancos de dados NoSQL

mongoDB

Update

- A atualização de documentos possui também algumas particularidades.
- Existem as opções Upser e Multi que quando apontadas mudam o comportamento dos métodos.
- **Upsert:** se definido como true, cria um novo documento quando nenhum documento corresponde aos critérios de consulta.
- **Multi:** se definido como true, atualiza vários documentos que atendem aos critérios de consulta. Se definido como false, atualiza um documento.

Bancos de dados NoSQL

mongoDB

Update

- Outras funções para atualização de documentos:

  - ✓ db.collection.findOneAndReplace();
  - ✓ db.collection.findOneAndUpdate();
  - ✓ db.collection.findAndModify();
  - ✓ db.collection.save();
  - ✓ db.collection.bulkWrite();

Bancos de dados NoSQL

mongoDB

Delete

- Para fecharmos as operações de CRUD, vamos falar dos métodos de deleção.
- É possível excluir os documentos a partir dos métodos:
  - ✓ db.collection.deleteOne();
  - ✓ db.collection.deleteMany();

Nunca se esqueça de passer os parâmtreos de filtros para atualização dos registros

Bancos de dados NoSQL

mongoDB

Delete

- Outras funções para deleção de documentos:

✓ db.collection.findOneAndDelete()

✓ db.collection.findAndModify()

✓ db.collection.bulkWrite()

PUC Minas

Now, Let's go to pratice!
On MongoDb!
But before...

Download das ferramentas:

- VS Code
- Bancos de dados de Exemplo Mongo – Atlas Education
- Mongo database tools – Ferramentas de restore

REFERÊNCIAS BIBLIOGRÁFICAS

SANTOS, Hudson Leandro, 2020. Vamos falar um pouco do WiredTiger?. Disponível em: <https://hudsondosantos.com/2020/08/18/vamos-falar-um-pouco-do-wiredtiger/>. Acesso em: 25 Nov. 2021.

PÚBLIO, Angelo. CRUD: o que é este conceito no Desenvolvimento de SistemasDisponível em: < https://angelopublio.com.br/blog/crud > Acesso em: 06 Dez. 2021.

LÓSCIO, B. F.;; OLIVEIRA, H. R. de;; PONTES, J. C. de S. NoSQL no desenvolvimento de aplicações web colaborativas. In: Simpósio Brasileiro de Sistemas Colaborativos – SBSC, 8., 2011, Paraty (RJ). Anais... Paraty: SBC, 2011.

VIEIRA, M. R. et al. Bancos de Dados NoSQL: conceitos, ferramentas, linguagens e estudos de casos no contexto de Big Data. In: Simpósio Brasileiro de Bancos de Dados, 27., 2012, São Paulo. Anais... São Paulo: SBC, 2012.