mongoDB®

MongoDB

—

Prof. Victor Farias

V 1.1

Livro Texto cap. 6

Introdução

MongoDB

- NoSQL
- Orientado a documentos
  - JSON
- Comandos de administração em JavaScript
- Baixa impedância com tecnologias WEB JS

Instalação

Instalação - Computadores do Lab

1. Baixar e descompactar binários do mongo

$ curl -O https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-3.4.4.tgz

$ tar -zxvf mongodb-linux-x86_64-3.4.4.tgz

2. Criar diretório de dados

$ mkdir datadir

3. Executar mongod

$ cd mongodb-linux-x86_64-3.4.4/bin/

$ ./mongod --dbpath ~/datadir

Temos o mongo rodando e escutando na porta 27017

MongoShell

Mongo dispõe de um cliente em linha de comando

$ ./mongo (em ~/mongodb-linux-x86_64-3.4.4/bin)

Esse cliente oferece comandos para manipulação de dados e gerenciamento do banco

Bancos e Coleções

MongoDB

- Cada instância do MongoDB tem uma conjunto de bancos
- Cada banco é uma conjunto de coleções
- Cada coleção é um conjunto de documentos JSON

Bancos - **MongoShell**

Listar todos bancos

> **show dbs**

Criar banco (também troca para banco existente)

> **use sistemamatricula**

Variável que contém referência do banco atual

> **db**

Coleções - MongoShell

Criar documento

```
> var aluno = {nome:"aluno"};
```

Adicionar documento em coleção **alunos**

```
> db.alunos.insert(aluno);
```

Mostrar todas coleções

```
> show collections
```

Destruir banco

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

Recuperando Documentos

Método find() no permite recuperar documentos de uma coleção

> db.alunos.find()

Mongo cria identificador único para cada documento

find() retorna um cursor

> var cursor = db.test.find()
> cursor.next()
> cursor.next()
> cursor.next()

Recuperando Documentos

Retornar apenas o um objeto

> **db.alunos.findOne()**

Quantidade de objetos em coleção

> **db.contatos.count()**

Busca com Critério

Operadores

- É possível criar critérios de consulta a partir de operadores
- MongoDB dispõe vários tipos de operadores:
  - Comparação
  - Lógicos
  - Expressão regular
  - Geoespacial

Busca com Critérios

Criando critério para busca por **igualdade**
Ex: Buscamos documentos que tem nome igual a “jo”

```
> var criterio = {nome:"jo"}

> var cursor = db.alunos.find(criterio);

> cursor
```

```
{ "_id" : ObjectId("5918f3f81e057cd74b129884"),
"nome" : "jo", "matricula" : "123" }
```

Operadores de Comparação

- Operadores de comparação
  - $eq - igualdade
  - $gt - maior que
  - $gte - maior ou igua que
  - $lt - menor que
  - $lte - menor ou igual que
  - $ne - diferente
  - $in - casa com algum elemento de uma lista
  - $nin - casa com nenhum elemento de uma lista

Operadores de Comparação

Ex: Alunos com IRA igual a 5000

```
> var criterio = { ira: {"$eq":5000}}

> db.alunos.find(criterio)

{ "_id" : ObjectId("591908a61e057cd74b129887"),
"nome" : "jo", "matricula" : "123", "ira" : 5000 }
```

Mesmo que:

```
> var criterio = {ira:5000}
```

Operadores de Comparação

Ex: Alunos com IRA menor ou a igual a 5000

```
> var criterio = {ira:{"$lte":5000}}

> db.alunos.find(criterio)

{ "_id" : ObjectId("591908a61e057cd74b129887"),
"nome" : "jo", "matricula" : "123", "ira" : 5000 }

{ "_id" : ObjectId("591908bb1e057cd74b129888"),
"nome" : "sa", "matricula" : "434", "ira" : 3000 }

{ "_id" : ObjectId("591908c91e057cd74b129889"),
"nome" : "de", "matricula" : "853", "ira" : 1234 }
```

Operadores de Comparação

Ex: Alunos com nome “jo” ou “sa”

```
> var criterio = {nome:{"$in":["jo","sa"]}}

> db.alunos.find(criterio)

{ "_id" : ObjectId("591908a61e057cd74b129887"),
"nome" : "jo", "matricula" : "123", "ira" : 5000 }

{ "_id" : ObjectId("591908bb1e057cd74b129888"),
"nome" : "sa", "matricula" : "434", "ira" : 3000 }
```

Operador Lógicos

- Operadores Lógicos
  - $or - OU
  - $and - E
  - $not - Não
  - $nor - Não OU

Operadores Lógicos

Ex: alunos cujo nome seja “jo” ou matrícula seja “453”

```
> var criterio = {"$or":[
                        {nome:"jo"},
                        {matricula:"453"}
                    ]}
> db.alunos.find(criterio)
{ "_id" : ObjectId("5918f3f81e057cd74b129884"), "nome" : "jo",
"matricula" : "123" }
{ "_id" : ObjectId("5918f4051e057cd74b129885"), "nome" : "sa",
"matricula" : "453" }
```

Operadores Lógicos

Ex: Alunos cujo nome é “jo” E matrícula é “123”

```
> var criterio =
{"$and":[{nome:"jo"},{matricula:"123"}]}

> db.alunos.find(criterio)
{ "_id" : ObjectId("5918f3f81e057cd74b129884"),
"nome" : "jo", "matricula" : "123" }
```

Operadores Lógicos

Ex: Alunos cujo nome é “jo” E IRA está entre 3000 e 7000 (incluso)

```
> var criterio =
{"$and":[
        {nome:"jo"},
        {"$and":[
            {ira:{"$lte":7000}},
            {ira:{"$gte":4000}}
            ]
        }
    ]
}
```

Removendo Documentos

Remoção de documentos

- Função **remove(criterio)**
  - **remove(criterio)** recebe critério como argumento igual a find()
  - Remove todos documentos que atendem ao critério

Remoção de documentos

Ex: Remover todos alunos com nome “jo”

```
> db.alunos.remove({nome:"jo"})
WriteResult({ "nRemoved" : 4 })
```

Remoção de documentos

Ex: Remover todos documentos da coleção

```
> db.alunos.remove()
```

Atualizando Documentos

Atualização de Documentos

- Função **update(criterio, atualizacao)**
  - **criterio** é um critério como usado anteriormente
    - modifica apenas um documento (por padrão) que atende ao critério
  - **atualizacao** indica como documento vai ser modificado
    - Document replacement ou Field update

Document Replacement

Ex: Atualizar no ira de “jo” para 9000

```
> var criterio = {nome:"jo"}
> var aluno = db.alunos.findOne(criterio)
> aluno.ira = 9000
9000
> db.alunos.update(criterio, aluno)
WriteResult({ "nMatched" : 1, "nUpserted" : 0,
"nModified" : 1 })
```

Field Update

Ex: Atualizar no ira de “jo” para 8000

```
> var criterio = {nome:"jo"}
> db.alunos.update( criterio,
                    { "$set":
                        { "ira":8000 }
                    }
                  )
WriteResult({ "nMatched" : 1, "nUpserted" : 0,
"nModified" : 1 })
```

Junção

Junção

- Documentos podem se relacionar
  - Aluno pode estar matriculado em várias matrículas
  - Vários atletas jogam por um time de futebol
- Em casos simples de composição
  - É possível aninhar um documento em outro
    - Pode gerar inconsistência
- No caso geral, precisamos fazer junções
- No MongoDB, a junções são feitas na aplicação
  - Temos que chamar find() várias vezes

Junção

- Junção é implementada através da referência de documentos pelo id
- Ex: Aluno ao se relacionar com disciplina deve conter id da disciplina matriculada
- O modo de implementar a relação muda conforme o tipo da relação

Junção

- Relações podem ser classificados quanto a
  - **Direção**
    - unidirecional: aluno -> disciplinas
      - Apenas aluno deve conter id das disciplinas matriculadas
    - bidirecional: aluno <-> disciplinas
      - Aluno deve conter id das disciplinas matriculadas e a disciplina deve conter id dos alunos matriculados

Lembrete! Em aluno -> disciplinas é possível obter todas as disciplinas de um aluno mas não todos alunos de uma disciplina (embora na prática seja possível)

Junção

- Relações podem ser classificados quanto a
  - **Multiplicidade**
    - 1x1: Professor - Endereço
      - Pessoa contém **uma** referência para endereço e/ou endereço contém **uma** referência para pessoa
      - Também pode-se aninhar documentos (colocar objeto endereço dentro de uma pessoa)
    - 1xN: Professor - Disciplina:
      - Professor contém uma **lista** de referências de disciplinas e/ou disciplinas contém **uma** referência para professor (mais comum)
    - NxN: Aluno - Disciplinas
      - Aluno contém uma **lista** de referências para disciplinas e/ou disciplinas contém um **lista** de referências para aluno
      - Caso relação tenha atributo, surge entidade intermediária (ex: matrícula)

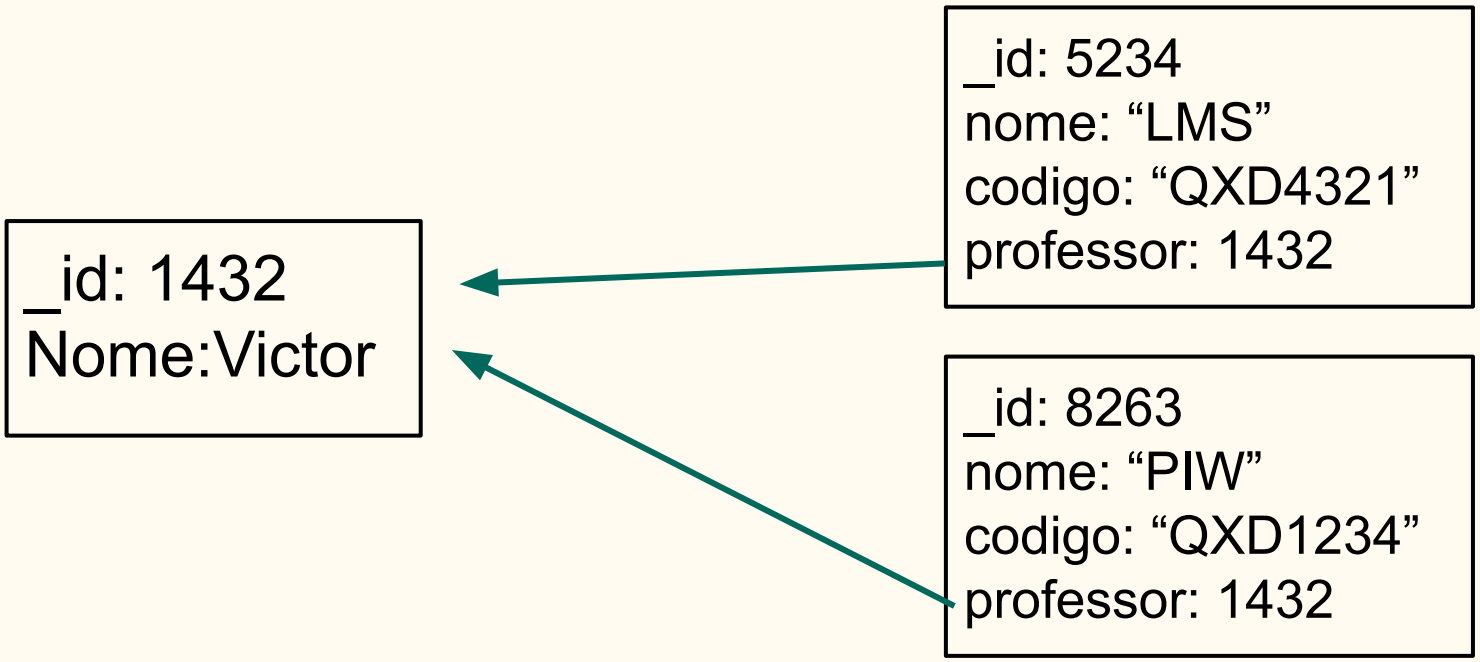
Junção 1xN

Inserindo dados

> db.professores.insert({"nome":"Victor"})

> victor=db.professores.findOne({"nome":"Victor"})

> db.disciplinas.insert({nome:"LMS", codigo:"QXD4321", professor:victor._id})



Junção 1xN

Executando a junção manualmente

- Recuperar todas as disciplinas de um professor

```
> db.disciplinas.find({professor:victor._id})
```

- Recuperar o prof de uma dada disciplina

```
> db.professores.findOne({_id:lms.professor})
```

Junção 1xN

Executando a junção usando lookup

- Recuperar a coleção de todas as disciplinas com todos professores

```
> db.disciplinas.aggregate([{
    $lookup:
        {from:"professor",
         localField:"professor",
         foreignField:"_id",
         as:"prof"}
}])
```

Junção - NxN - Unidirecional

Ex: Registrar que aluno “jo” está matriculado na disciplina “PIW” mas não registramos que “jo” na disciplina “PIW”

```
> var disciplina = {nome:"PIW", codigo:"QXD001"}
> db.disciplinas.insert(disciplina)
> disciplina = db.disciplinas.findOne(disciplina)
> var aluno = db.alunos.findOne({nome:"jo"})
> aluno.matriculado = []
> aluno.matriculado.push(disciplina._id)
> db.alunos.update({_id:aluno._id},aluno)
```

Junção - NxN - Unidirecional

Ex: Recuperar todas as disciplinas de “jo”

```
> var jo = db.alunos.findOne({nome:"jo"})
> db.disciplinas.find({_id:{"$in":jo.matriculado}})

{ "_id" : ObjectId("5919b9680afee2f96e6848b7"),
"nome" : "PIW", "codigo" : "QXD001" }

{ "_id" : ObjectId("5919bb610afee2f96e6848b8"), "nome"
: "POO", "codigo" : "QXD002" }
```

Junção - NxN - Unidirecional

Ex: Recuperar todos alunos de “PIW”

```
> var piw = db.disciplinas.findOne({nome:"PIW"})
> db.alunos.find({matriculado:piw._id})
{ "_id" : ObjectId("591908bb1e057cd74b129888"), "nome" :
"sa", "matricula" : "434", "ira" : 3000, "matriculado" : [
ObjectId("5919b9680afee2f96e6848b7") ] }
{ "_id" : ObjectId("5919af970afee2f96e6848b4"), "nome" :
"jo", "matricula" : "2345", "ira" : 8000, "matriculado" : [
ObjectId("5919b9680afee2f96e6848b7"),
ObjectId("5919bb610afee2f96e6848b8") ] }
```

Perguntas?

Prof. Victor Farias