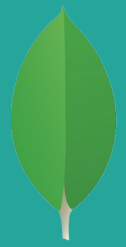


mongoDB®

MongoDB

Prof. Victor Farias

V 1.2

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
2. Criar diretório de dados

   $ mkdir datadir
3. Executar mongod

   $.\mongod.exe --dbpath ./datadir
4. Temos o mongo rodando e escutando na porta 27017

MongoShell

- Mongo dispõe de um cliente em linha de comando
  ```
  $ .\mongo.exe
  ```
- 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

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”
  > let criterio = {nome:"jo"}
  > let 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

  > let 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 disciplinas
  - 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 oferta de disciplina deve conter id da disciplina matriculada
- O modo de implementar a relação muda conforme o tipo da relação

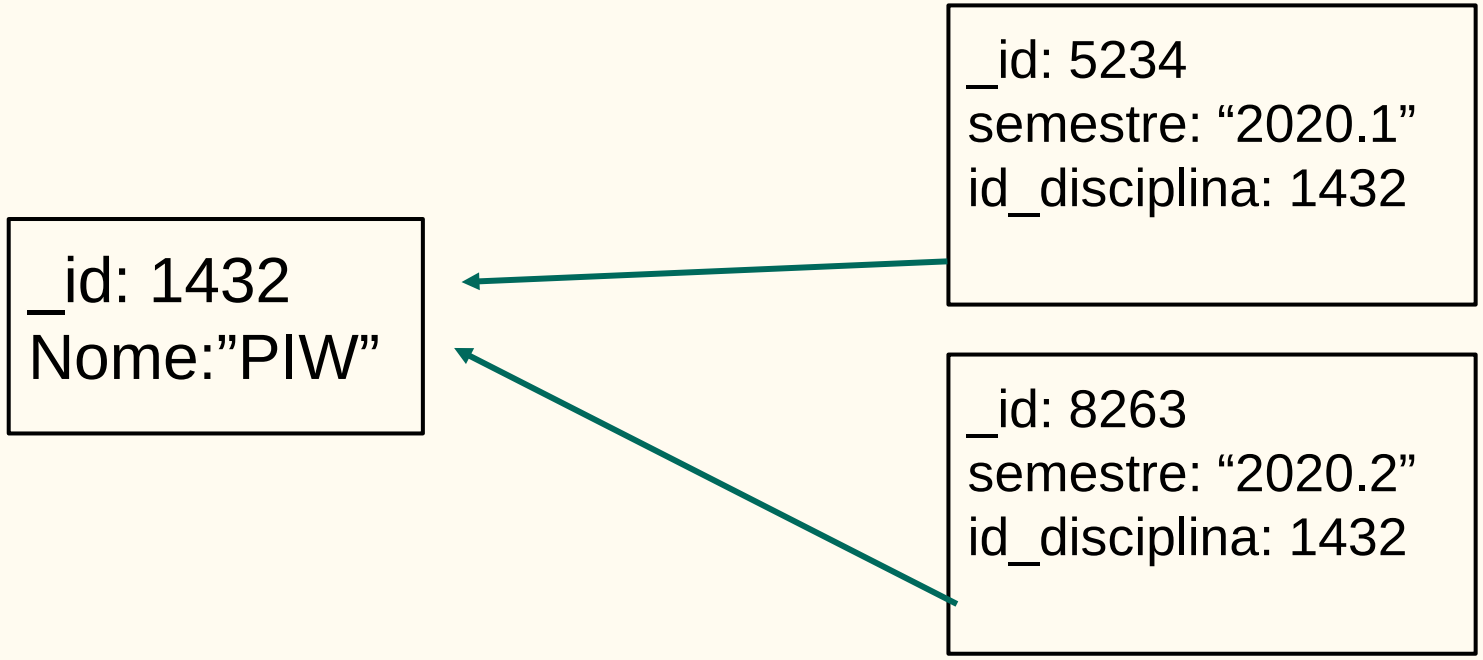
Junção

- Relações podem ser classificados quanto a
  - Multiplicidade
    - 1x1: Aluno - Endereço
      - Aluno 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: Oferta - Disciplina:
      - Cada disciplina contém uma lista de referências de ofertas e/ou ofertas contém uma referência para disciplina (mais comum)
    - NxM: Aluno - Ofertas
      - Surge entidade intermediária (ex: matrícula) que contém a referência dos dois lados

Junção 1xN

- Inserindo dados

```
> db.disciplinas.insert({nome:"PIW"})
> let piw = db.disciplinas.findOne({nome:"PIW"})
>  db.ofertas.insert({semestre:"2020.1", id_disciplina:piw._id})
```



Junção 1xN

- Executando a junção manualmente
- Recuperar todas as ofertas de uma dada disciplina

  > db.ofertas.find({id_disciplina:lms._id})

- Recuperar a disciplina de uma dada oferta

  > db.disciplinas.findOne({_id: oferta.id_disciplina})

Junção - NxM

- Ex: Registrar que aluno “victor” está matriculado na oferta da disciplina “PIW” de 2020.1

```
> let victor = db.alunos.findOne({matricula: 123})
> let oferta_piw = db.ofertas.findOne({id_disciplina: piw._id, semestre:"2020.1"})
> db.matriculas.insert({id_aluno: victor._id, id_oferta: oferta_piw._id})
```

Junção - NxM

- Ex: Recuperar as ofertas de disciplinas de victor

```
> let matriculas = db.matriculas.find({id_aluno: victor._id}).toArray()
> let ids_ofertas = []
> for(matricula of matriculas){ids_oferta.push(matricula.id_oferta)} > db.ofertas.find({_id: {"$in": ids_ofertas}})
```

Perguntas?

Prof. Victor Farias