



IBD-016 – BANCO DE DADOS - NÃO RELACIONAL


Prof. Me. Anderson Vanin



Case

Vamos explorar um case de modelagem em MongoDB para um artigo comparando-o inicialmente como seria a construção de tabelas e por fim as consultas que deveriam ser realizadas para que a tela proposta de um Blog seja exibida com todas as informações necessárias.

O Blog



Tutorial MongoDB para iniciantes

luiztools.com.br/post/tutorial-mongodb-para-iniciantes-em-nosql/

Luiz Tools

Inicial Guias de Estudo Meus Cursos Meus Livros Sobre

WEB

Tutorial MongoDB para iniciantes em NoSQL

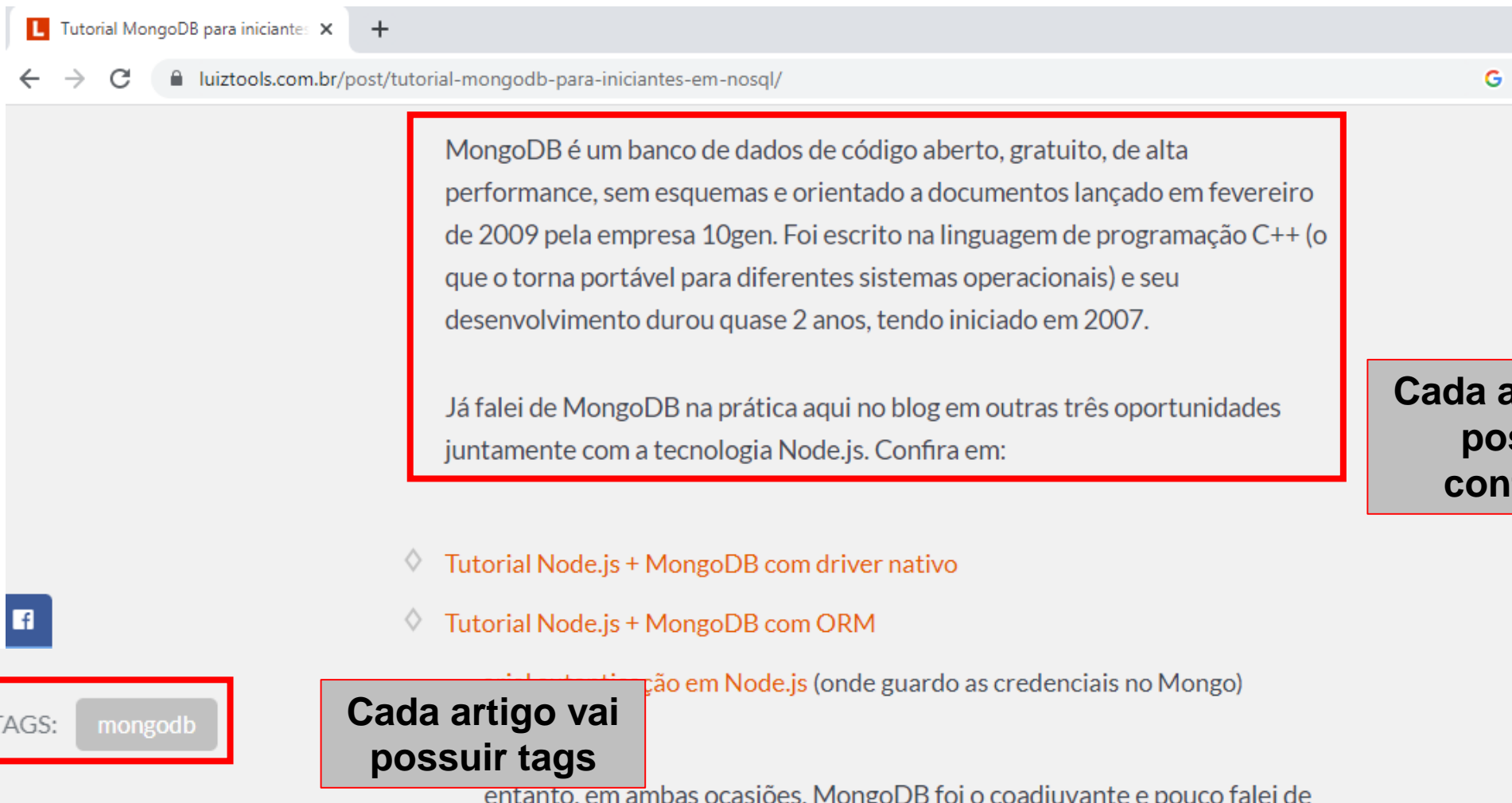
Cada artigo do blog deve possuir uma URL única

Cada artigo possui um título

Cada artigo possui o Autor e a Data

Escrito por **Luiz Duarte** em 01/02/2023

O Blog



Tutorial MongoDB para iniciantes x +

← → ↺ 🔒 luiztools.com.br/post/tutorial-mongodb-para-iniciantes-em-nosql/ 🔍

MongoDB é um banco de dados de código aberto, gratuito, de alta performance, sem esquemas e orientado a documentos lançado em fevereiro de 2009 pela empresa 10gen. Foi escrito na linguagem de programação C++ (o que o torna portátil para diferentes sistemas operacionais) e seu desenvolvimento durou quase 2 anos, tendo iniciado em 2007.

Já falei de MongoDB na prática aqui no blog em outras três oportunidades juntamente com a tecnologia Node.js. Confira em:

- ◇ Tutorial Node.js + MongoDB com driver nativo
- ◇ Tutorial Node.js + MongoDB com ORM

Tags: mongodb

Cada artigo vai possuir conteúdo

Cada artigo vai possuir tags

entanto, em ambas ocasiões, MongoDB foi o coadjuvante e pouco falei de

O Blog



My name is... , 03/11/2017 às 10:59

Show de bola mas vc não explicou pq tela de detalhes de produtos é melhor usando MongoDB do que bancos relacionais.

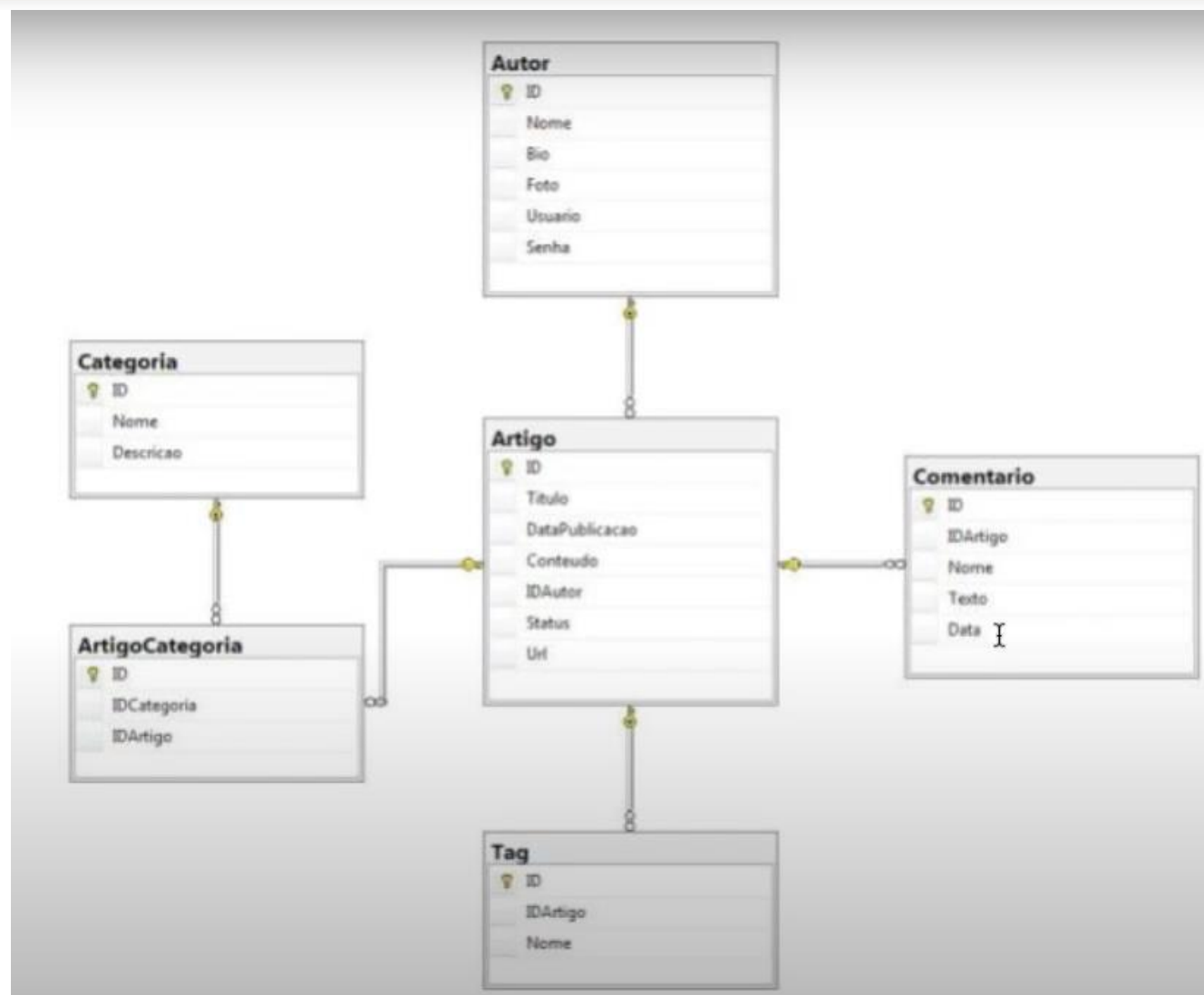
Responder

Cada artigo vai possuir comentários com o nome da pessoa, data e o comentário

Blog Relacional

- Artigo: FK com Autor
- Autor
- Categoria
- ArtigoCategoria: tabela-meio, FK com Categoria e Artigo
- Tag: FK com Artigo
- Comentário: FK com Artigo

Blog Relacional



Tela de um artigo no blog

Qual ou quais consultas seriam necessárias para exibir todas as informações na tela?

- `SELECT Artigo.*,Autor.* JOIN Autor WHERE Url = 'xxx'`
- `SELECT Categoria.* JOIN ArtigoCategoria WHERE IDArtigo = x`
- `SELECT Tag.* WHERE IDArtigo = x`
- `SELECT Comentario.* WHERE IDArtigo = x`

MongoDB

```
use blog
```

```
db.artigos.insert({  
  titulo: "Teste",  
  data: new Date(),  
  texto: "TesteTeste",  
  url: "http://www",  
  status: 1  
})
```

```
db.autores.insert({  
  nome: "Luiz",  
  bio: "história do autor",  
  imagem: "imagem.jpg"  
})
```

```
> db.autores.find().pretty()  
{  
  "_id" : ObjectId("645246dabd0c12adce0fcfc0"),  
  "nome" : "Luiz",  
  "bio" : "história do autor",  
  "imagem" : "imagem.jpg"  
}  
> db.artigos.find().pretty()  
{  
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),  
  "titulo" : "Teste",  
  "data" : ISODate("2023-05-03T11:33:15.896Z"),  
  "texto" : "TesteTeste",  
  "url" : "http://www",  
  "status" : 1  
}
```

MongoDB

```
db.artigos.updateOne({
  _id: ObjectId("6452467bbd0c12adce0fcfbf"),
  {
    $set: {
      autor: ObjectId("645246dabd0c12adce0fcfc0")
    }
  }
})
```

Aqui estamos muito próximo do que é um BD Relacional e neste caso precisaríamos fazer consultas em vários *documents*. O ideal seria colocar todas as informações em um único *document* para realizarmos uma única consulta.

```
> db.artigos.find().pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : ObjectId("645246dabd0c12adce0fcfc0")
}
> db.autores.find().pretty()
{
  "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
  "nome" : "Luiz",
  "bio" : "história do autor",
  "imagem" : "imagem.jpg"
}
```

MongoDB

```
db.artigos.updateOne(  
  {  
    _id: ObjectId("6452467bbd0c12adce0fcfbf")  
  },  
  {  
    $set: {  
      autor: {  
        _id: ObjectId("645246dabd0c12adce0fcfc0"),  
        nome: "Luiz"  
      }  
    }  
  }  
)
```

Agora dentro do document de artigos temos também um sub document de autor.

Aqui reunimos somente as informações necessárias para serem apresentadas na tela quando acessarmos uma Url de um artigo. De autores precisaremos do Id do autor e de seu Nome

```
> db.artigos.find().pretty()  
{  
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),  
  "titulo" : "Teste",  
  "data" : ISODate("2023-05-03T11:33:15.896Z"),  
  "texto" : "TesteTeste",  
  "url" : "http://www",  
  "status" : 1,  
  "autor" : {  
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),  
    "nome" : "Luiz"  
  }  
}
```

MongoDB

```
db.categorias.insert([
  {
    nome: "categoria1",
    descricao: "descrição cat 1"
  },
  {
    nome: "categoria2",
    descricao: "descrição cat 2"
  }
])
```

```
> db.categorias.find().pretty()
{
  "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
  "nome" : "categoria1",
  "descricao" : "descrição cat 1"
}
{
  "_id" : ObjectId("645251a8bd0c12adce0fcfc2"),
  "nome" : "categoria2",
  "descricao" : "descrição cat 2"
}
```

No document de artigos, precisamos
criar um outro sub document que será
Array de categorias

MongoDB

```
db.artigos.updateOne(  
  {  
    _id: ObjectId("6452467bbd0c12adce0fcfbf")  
  },  
  {  
    $set:  
    {  
      categorias:[  
        {  
          _id: ObjectId("645251a8bd0c12adce0fcfc1"),  
          nome: "categoria1"  
        }  
      ]  
    }  
  }  
)
```

MongoDB

```
> db.artigos.find().pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : {
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
    "nome" : "Luiz"
  },
  "categorias" : [
    {
      "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
      "nome" : "categoria1"
    }
  ]
}
```

MongoDB

```
db.artigos.updateOne(  
  {  
    _id: ObjectId("6452467bbd0c12adce0fcfbf")  
  },  
  {  
    $set:  
    {  
      tags:  
        ["tag1", "tag2"]  
    }  
  }  
)
```

Para as tags, o processo é semelhante, mas para as tags teremos somente um array de strings. Importante lembrar que na regra de negócios do blog, deveremos ter tags únicas para cada artigo!

MongoDB

```
> db.artigos.find().pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : {
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
    "nome" : "Luiz"
  },
  "categorias" : [
    {
      "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
      "nome" : "categoria1"
    }
  ],
  "tags" : [
    "tag1",
    "tag2"
  ]
}
```


MongoDB

```
db.artigos.updateOne(  
  {  
    _id: ObjectId("6452467bbd0c12adce0fcfbf")  
  },  
  {  
    $set:  
    {  
      comentarios:  
      ]  
    }  
  }  
)
```

Para os comentários, vamos setar um campo de comentários que será um array vazio, imaginando que ainda não existam os comentários. Assim cada comentário será um valor desse array.

MongoDB

```
> db.artigos.find().pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : {
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
    "nome" : "Luiz"
  },
  "categorias" : [
    {
      "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
      "nome" : "categoria1"
    }
  ],
  "tags" : [
    "tag1",
    "tag2"
  ],
  "comentarios" : [ ]
}
```

MongoDB

Para inserirmos novos valores em campos multivalorados, o MongoDB fornece alguns operadores que também são utilizados em atualizações de documentos.

\$set → insere campos

\$unset → remove campos existentes

\$push → adiciona elementos em um campo multivalorado

\$pull → remove elementos em um campo multivalorado

\$in → usado na pesquisa para encontrar valores (e/ou)

\$all → usado na pesquisa exata de valores (e)

MongoDB

```
db.artigos.updateOne(  
  {  
    _id: ObjectId("6452467bbd0c12adce0fcfbf")  
  },  
  {  
    $push:  
    {  
      comentarios:{  
        nome: "Fulano da Silva",  
        texto: "Fiz um comentário sobre o artigo",  
        data: new Date()  
      }  
    }  
  }  
)
```

Esta instrução utilizando o operador *\$push*, serve para adicionar um novo valor dentro de um array, que no nosso caso possui a chave comentários. Isso faz com que seja adicionado um sub documento dentro do array de comentários.

MongoDB

```
> db.artigos.find().pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : {
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
    "nome" : "Luiz"
  },
  "categorias" : [
    {
      "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
      "nome" : "categoria1"
    }
  ],
  "tags" : [
    "tag1",
    "tag2"
  ],
  "comentarios" : [
    {
      "nome" : "Fulano da Silva",
      "texto" : "Fiz um comentário sobre o artigo",
      "data" : ISODate("2023-05-03T12:45:02.386Z")
    }
  ]
}
```

MongoDB

```
db.artigos.find({url:"http://www"}).pretty()
```

```
> db.artigos.find({url:"http://www"}).pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : {
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
    "nome" : "Luiz"
  },
  "categorias" : [
    {
      "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
      "nome" : "categoria1"
    }
  ],
  "tags" : [
    "tag1",
    "tag2"
  ],
  "comentarios" : [
    {
      "nome" : "Fulano da Silva",
      "texto" : "Fiz um comentário sobre o artigo",
      "data" : ISODate("2023-05-03T12:45:02.386Z")
    }
  ]
}
```

Agora que o BD está modelado para apenas um document, podemos fazer uma consulta de um determinado artigo apenas com base, por exemplo, na sua Url.

MongoDB

```
db.artigos.find({"autor.nome":"Luiz"}).pretty()
```

```
> db.artigos.find({"autor.nome":"Luiz"}).pretty()
{
  "_id" : ObjectId("6452467bbd0c12adce0fcfbf"),
  "titulo" : "Teste",
  "data" : ISODate("2023-05-03T11:33:15.896Z"),
  "texto" : "TesteTeste",
  "url" : "http://www",
  "status" : 1,
  "autor" : {
    "_id" : ObjectId("645246dabd0c12adce0fcfc0"),
    "nome" : "Luiz"
  },
  "categorias" : [
    {
      "_id" : ObjectId("645251a8bd0c12adce0fcfc1"),
      "nome" : "categoria1"
    }
  ],
  "tags" : [
    "tag1",
    "tag2"
  ],
  "comentarios" : [
    {
      "nome" : "Fulano da Silva",
      "texto" : "Fiz um comentário sobre o artigo",
      "data" : ISODate("2023-05-03T12:45:02.386Z")
    }
  ]
}
```

MongoDB

Pesquisando dentro de um campo multivalorado

```
db.artigos.find({tags:"tag3"}).pretty()
```

Retorna artigos que tenham no mínimo a tag3

```
db.artigos.find(  
  {  
    tags:{  
      $in:[  
        "tag1",  
        "tag2"  
      ]  
    }  
  }  
)
```

Retorna artigos que tenham as tags tag1 **e/ou** tag2

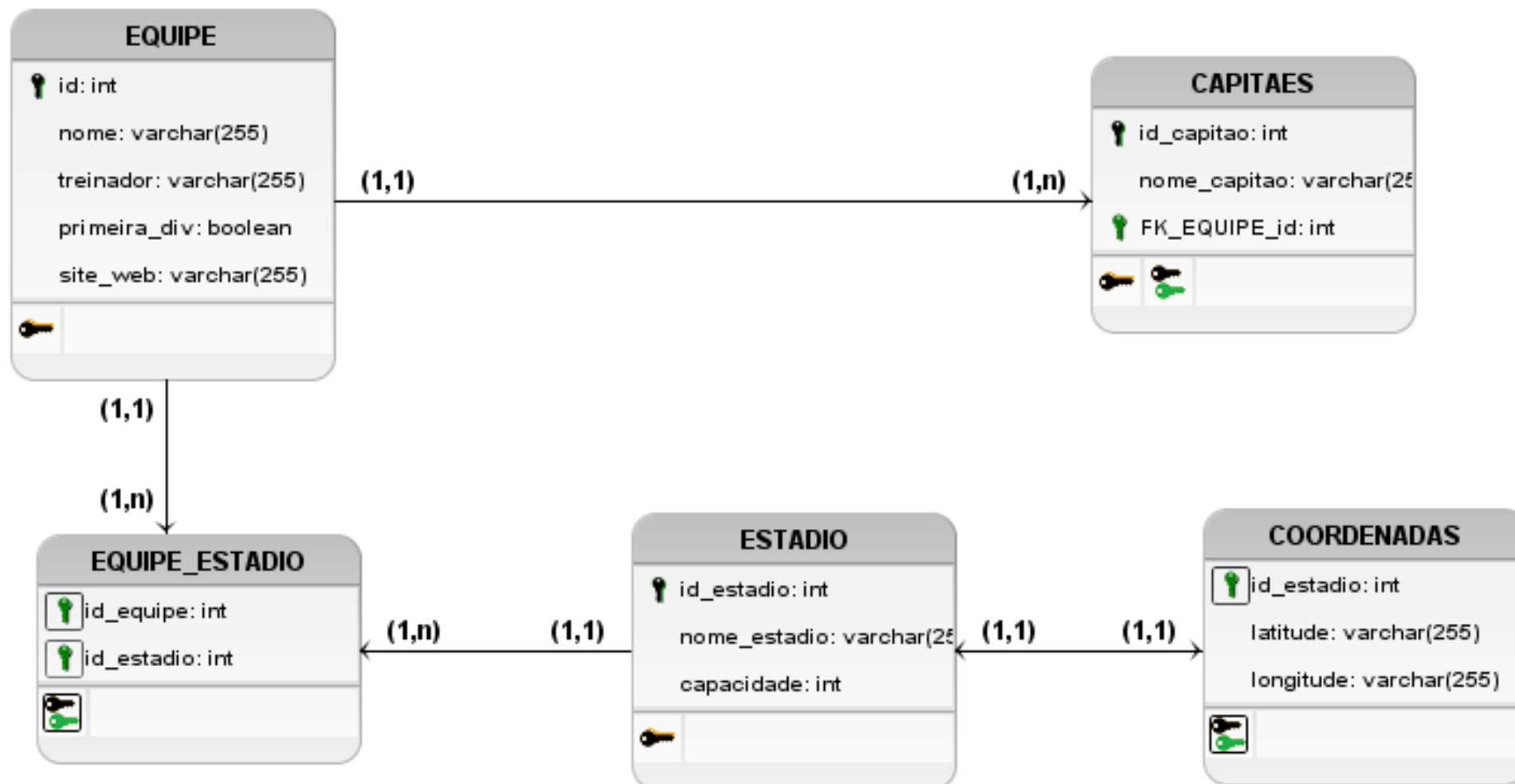
```
db.artigos.find(  
  {  
    tags:{  
      $all:[  
        "tag1",  
        "tag3"  
      ]  
    }  
  }  
)
```

Retorna artigos que tenham as tags tag1 **e** tag3

Exercício

Neste exercício vamos ver como seria uma aplicação ou um sistema de gerenciamento de dados para "**Clubes de Futebol**" feito com MongoDB e feito com banco de dados relacional, para ver as diferenças. Vamos supor que nosso aplicativo queira trabalhar com determinados dados de times de futebol como o nome do clube, quem é o técnico, quem ou quais são os capitães dos times, se o time é da primeira divisão ou não, o site do clube e dados relacionados ao estádio de futebol em que o clube joga; como o nome, a capacidade e as suas coordenadas geográficas. A tudo isto existem sempre algumas restrições quanto aos dados, tais como que um clube não pode ter ou ter mais do que um estádio de futebol; Pode ser que eles não tenham um site ou que houvesse vários capitães no clube. Conhecendo as características que nossa aplicação terá, podemos capturar em um diagrama de classes as classes que nosso sistema deve ter:

Exercício - DER



Exercício

1. Esboce uma tela para computador na qual devem ser apresentadas as informações sobre um time de futebol, seu(s) treinador(es), seu estádio e localização geográfica.
2. Elabore uma planilha no Excel contendo os dados dos 20 principais estádios de futebol do Brasil contendo: id, nome, capacidade, latitude e longitude.
 - Estádios: <https://www.cnnbrasil.com.br/esportes/maiores-estadios-do-brasil/>
 - Localização Geográfica: <https://latitude.to/>
 - Converta a planilha para um arquivo do tipo CSV para ser importado no Google Colab utilizando Python.
3. Elabore uma planilha no Excel contendo os dados de 20 equipes de futebol com seus respectivos nomes, treinadores, capitães, se pertence ou não a 1ª divisão e seu site na web.
 - Converta a planilha para um arquivo do tipo CSV para ser importado no Google Colab utilizando Python.

Exercício

4. Com base no DER apresentado, crie uma estrutura baseada no MongoDB (modelo baseado em documentos) na qual em uma única consulta possa ser obtidas as informações a serem carregadas na tela do item 1.