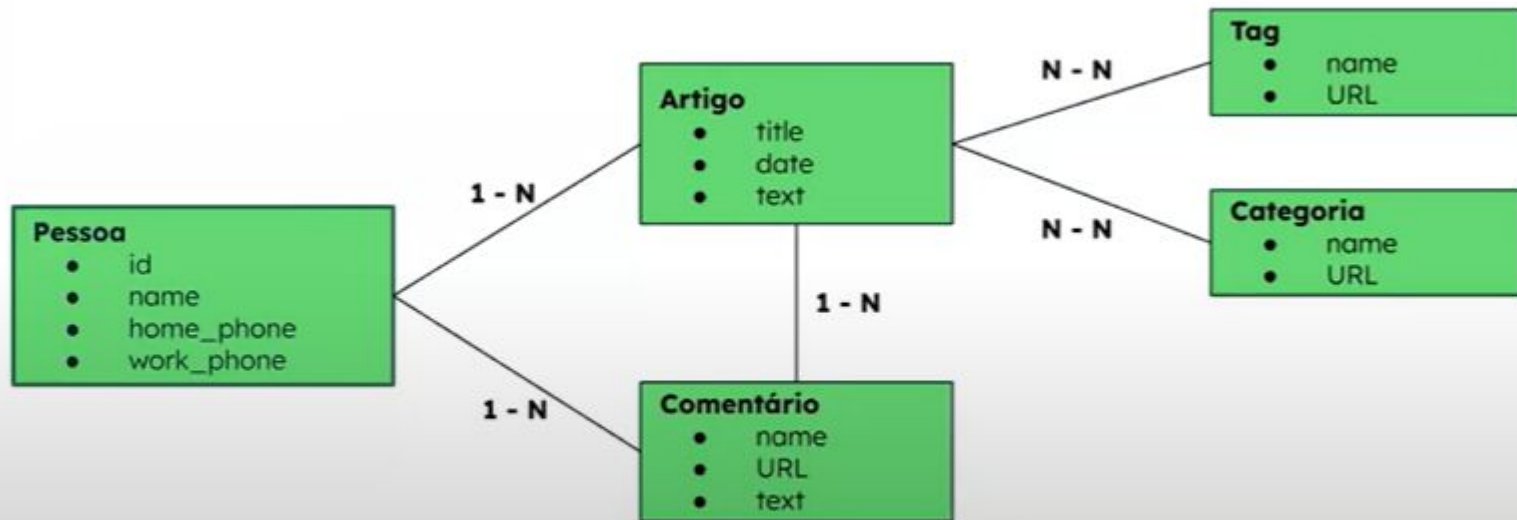


# Modelagem com MongoDB



Profa. Kelly Lais Wiggers

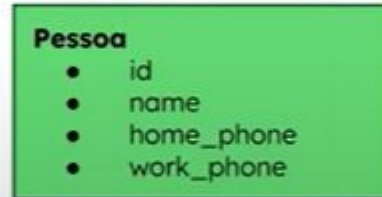
# BD relacional



# A partir de 2010....big data



ou



N - N



# E então, usar 1 ou 2 coleções?



ou



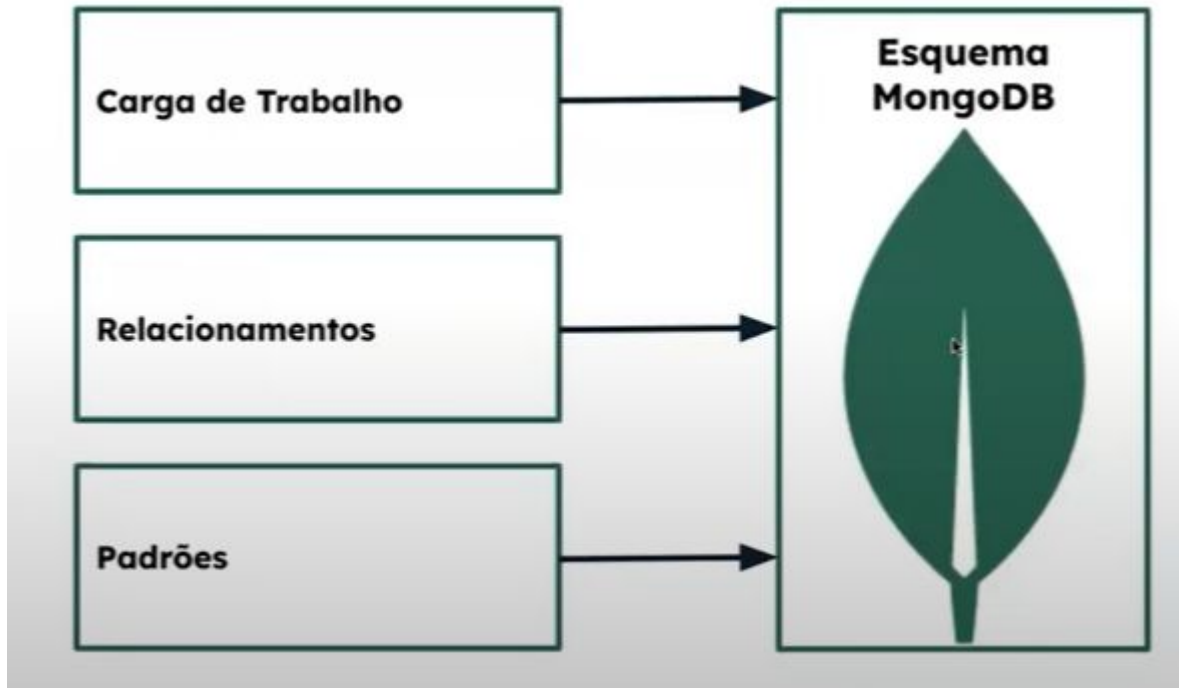
N - N



# **Mas antes de pensar em coleções**

- Saber se irei realizar mais escritas ou leituras de dados, ou mesmo escrita e leitura em paralelo
  - carga de trabalho

# Técnicas de modelagem



# Carga de trabalho

Tipo	Operação	Informação	Frequência	Criticidade
escrita	postar novos artigos	autor, texto	10 por dia	alta
escrita	comentar	leitor	10K por dia (1000 / artigo)	média
leitura	lendo um artigo	id, texto, comentário	10 milhões por dia	média
leitura	consultas analíticas	artigos, comentários, cliques, ...	10 por hora	baixo

# RELACIONAL



# MONGODB





# RELACIONAL



# MONGODB



O agrupamento da coleção vai depender da forma que você organizar os dados

# RELACIONAL

# MONGODB

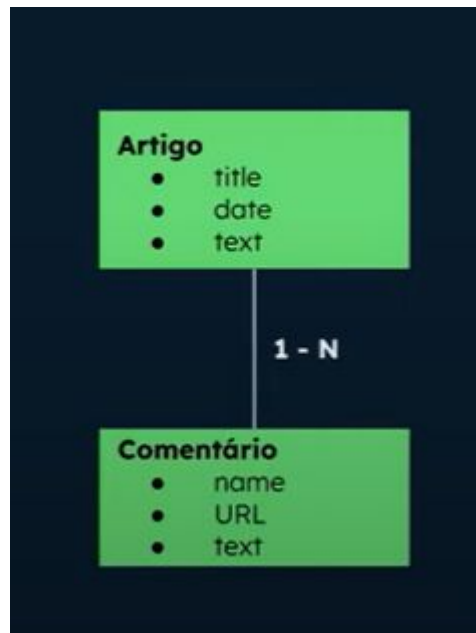


O que é usado junto na aplicação é armazenado junto no banco de dados



# **Diferenças entre relacionar x embarcar**

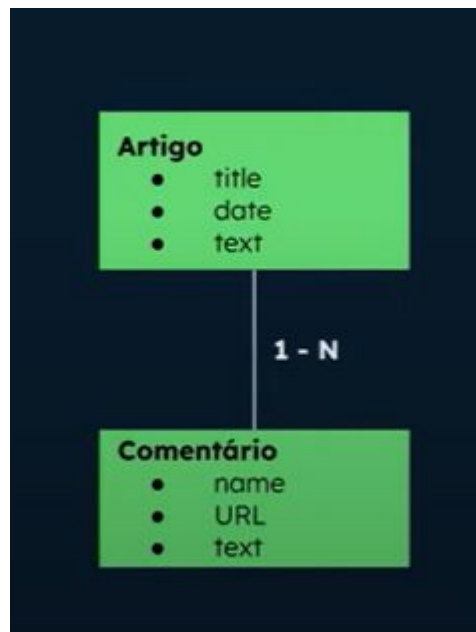
# Relacional



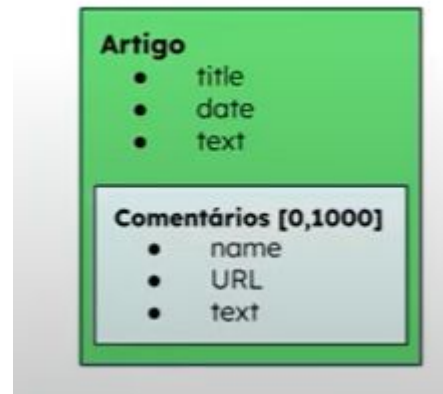
# Uma referência vincula 2 coleções em MongoDB



# Relacional



# Embarcar usa 1 coleção no MongoDB



# Quando referenciar?

1. Quando o lado “muitos” é um número enorme
2. Para integridade em operações de escrita nos relacionamentos muitos-para-muitos
3. Quando uma parte é usada com frequência, mas a outra não é, e a memória é uma restrição

# Quando embarcar?

Ler tudo que eu preciso em tela e em uma única operação

1. Para integridade das operações de leitura
2. Para integridade de operações de escrita um-para-um e um-para-muitos
3. para dados que são excluídos ou arquivados juntos

# Padrões de modelagem para MongoDB

## Patterns

Approximation  
Attribute  
Bucket  
Computed  
Document Versioning  
Extended Reference  
Outlier  
Preallocated  
Polymorphic  
Schema Versioning  
Subset  
Tree and Graph

## Use Case Categories

	Catalog	Content Management	Internet of Things	Mobile	Personalization	Real-Time Analytics	Single View
Approximation	✓		✓	✓		✓	
Attribute	✓	✓					✓
Bucket			✓			✓	
Computed	✓		✓	✓	✓	✓	✓
Document Versioning	✓	✓			✓		✓
Extended Reference	✓			✓		✓	
Outlier			✓	✓	✓		
Preallocated			✓			✓	
Polymorphic	✓	✓		✓			✓
Schema Versioning	✓	✓	✓	✓	✓	✓	✓
Subset	✓	✓		✓	✓		
Tree and Graph	✓	✓					



# Padrões de modelagem para MongoDB

- ◇ **Catalog**: como catálogo de produtos e serviços;
- ◇ **Content Management**: como em ECMs (gerenciadores de conteúdo corporativo);
- ◇ **Internet of Things**: como em sistemas com sensores real-time, como na indústria 4.0;
- ◇ **Mobile**: como em aplicações móveis de baixa latência e alta escala;
- ◇ **Personalization**: como em sistemas com schema de dados personalizável para entregar conteúdo relevante aos usuários, muito usados no marketing digital;
- ◇ **Real-Time Analytics**: como em sistemas de estatísticas real-time;
- ◇ **Single View**: como em sistemas de dashboards, gerenciais e outros que agregam informações de diferentes fontes em uma base só;

<https://www.luiztools.com.br/post/padroes-para-modelagem-de-dados-documentos-em-mongodb/>

# Na aula passada...

- Criar o banco
- Criar coleções
- Mostrar coleções dentro do banco
- Adicionar documentos dentro das coleções
- Listar todos os documentos dentro de uma coleção

# Na aula passada

1. Como encontrar o banco de dados em que você está trabalhando no momento

db

# Na aula passada

2. Como listar todos os bancos de dados

```
show databases
```

## Na aula passada...

3. Ir para um banco de dados específico

```
use <nome_banco>
```

## Na aula passada...

4. Criar uma coleção

```
db.<nome_coleção>.insertOne(  
  
{  
  
  "nome": "Kelly",  
  
  "email": "kwiggers@utfpr.edu.br"  
  
}  
  
)
```

# Criar uma coleção sem limite de tamanho

```
db.createCollection("minhaColecao")
```

# Criar coleção com limite de tamanho

```
db.createCollection("minhaSegundaColecao",  
{  
  capped : true,  
  size : 2,  
  max : 2  
})
```

Aqui estamos criando uma coleção  
sem inserir dados  
Nesse exemplo, habilitamos os limites  
ao definir capped como true  
sendo:

- size:2 o limite de 2 megabytes
- max: 2 define o máximo de documentos, sendo 2



# Criar coleção com limite de tamanho

```
db.createCollection("empregado",
{
  capped: true,
  size: 2,
  max: 2
})
{ ok: 1 }
meubanco> show collections
empregado
pessoas
```

# Renomear uma coleção no banco

```
db.mycollection.renameCollection("mycollection1")
```

```
meubanco> db.empregado.renameCollection("empregados")  
{ ok: 1 }  
meubanco> show collections  
empregados  
pessoas
```

# Métodos de inserção

- insertOne()

```
db.myCollection.insertOne(  
  {  
    "name": "navindu",  
    "age": 22  
  }  
)
```

- insertMany()

```
db.myCollection.insertMany([  
  {  
    "name": "navindu",  
    "age": 22  
  },  
  {  
    "name": "kavindu",  
    "age": 20  
  },  
  
  {  
    "name": "john doe",  
    "age": 25,  
    "location": "colombo"  
  }  
)
```

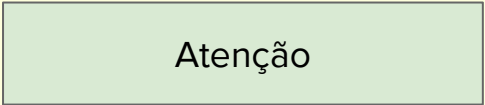
# Como atualizar documentos

```
db.COLLECTION_NAME.updateOne(  
  {SELECTION_CRITERIA},  
  {$set:{UPDATED_DATA}},
```

- Suponha que você queira atualizar o nome de uma pessoa

```
db.myCollection.updateOne({"nome":"Kelly"}, {$set: {"nome":"Lais"}})
```

```
meubanco> db.pessoas.updateOne({"nome": "Kelly"}, {$set: {"nome": "Lais"}})  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 0,  
  modifiedCount: 0,  
  upsertedCount: 0  
}
```

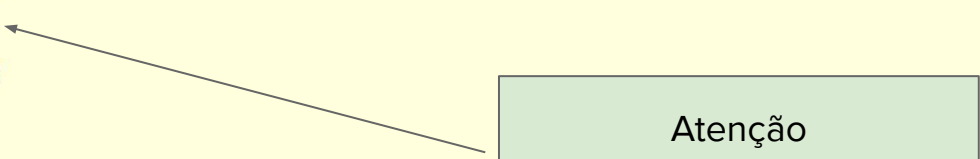


# Como atualizar documentos

```
db.COLLECTION_NAME.updateOne({SELECTION_CRITERIA}, {$set: {UPDATED_DATA}})
```

- Suponha que você queira atualizar o nome de uma pessoa

```
meubanco> db.pessoas.updateOne({"nome": "Kelly Lais"}, {$set: {"nome": "Kelly Lais Wiggers"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```



Atenção

# Como atualizar documentos

```
db.COLLECTION_NAME.updateOne(  
  {SELECTION_CRITERIA},  
  {$set:{UPDATED_DATA}},
```

- Suponha que você queira atualizar o nome de uma pessoa

```
db.myCollection.updateOne({"nome":"Kelly"}, {$set: {"nome":"Lais"}})
```

Aqui, o primeiro documento que tiver nome Kelly, irá atualizar para Lais.

Então como garantir que atualize um documento específico?

# Como atualizar documentos

- Suponha que você queira atualizar o nome de uma pessoa onde a tua query é a ID dela

```
meubanco> db.pessoas.updateOne({_id: ObjectId("6718d1373857bf08b2fe6912")}, {$set: {"nome": "Kelly"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

# Como atualizar documentos

- Suponha que você queira atualizar o nome de uma pessoa onde a tua query é a ID dela

```
meubanco> db.pessoas.find()
[
  {
    _id: ObjectId('6718d0ca3857bf08b2fe6911'),
    nome: 'Kelly Lais Wiggers',
    email: 'kwiggers@utfpr.edu.br'
  },
  {
    _id: ObjectId('6718d1373857bf08b2fe6912'),
    nome: 'Kelly',
    email: 'kwiggers@utfpr.edu.br'
  },
  {
    _id: ObjectId('6718d1843857bf08b2fe6913'),
    nome: 'Fulano de Tal',
    email: 'fulano@email.com'
  },
]
```



## E mais de um documento?

```
{
  _id: ObjectId('671a47b504895640f3fe6911'),
  nome: 'Bill Gates',
  email: 'bill@email.com',
  idade: '68'
},
{
  _id: ObjectId('672368a72de66b0a2cfe6911'),
  nome: 'Bill Gates',
  email: 'bill@email.com'
}
```

## E mais de um documento?

```
meubanco> db.pessoas.updateMany({"nome":"Bill Gates"}, {$set: {"nome": "Bill"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

## E mais de um documento?

```
{
  _id: ObjectId('671a47b504895640f3fe6911'),
  nome: 'Bill',
  email: 'bill@email.com',
  idade: '68'
},
{
  _id: ObjectId('672368a72de66b0a2cfe6911'),
  nome: 'Bill',
  email: 'bill@email.com'
}
```

# Como atualizar documentos

- E se tivéssemos um array?
- Por exemplo, um documento em pessoas, que possui nome, email e um array de notas

```
db.pessoas.insertOne({"nome": "Aluno", "email": "email@email.com", "notas": [44,78,38,80]})
```

- E agora preciso inserir mais uma nota no array:

```
db.pessoas.updateOne({}, {$push:{"notas": 94}})
```

quando deixamos vazio, pega o primeiro documento que aparece na lista

Veremos outras opções ao estudar consultas.

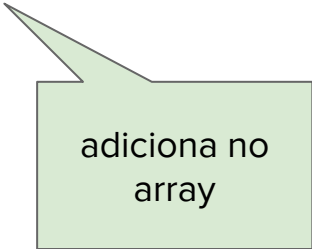
# Como atualizar documentos

- E se tivéssemos um array?
- Por exemplo, um documento em pessoas, que possui nome, email e um array de notas

```
db.pessoas.insertOne({"nome", "Aluno", "email": "email@email.com", "notas": [44,78,38,80]})
```

- E agora preciso inserir mais uma nota no array:

```
db.pessoas.updateOne({}, {$push:{"notas": 94}})
```



adiciona no  
array

Veja mais opções de push

<https://www.mongodb.com/pt-br/docs/manual/reference/operator/update/push/>

# Como atualizar documentos

```
meubanco> db.pessoas.find()  
[  
  {  
    _id: ObjectId('6718d0ca3857bf08b2fe6911'),  
    nome: 'Kelly Lais Wiggers',  
    email: 'kwiggers@utfpr.edu.br',  
    notas: [ 94 ]  
  },  
  ...  
]
```

adiciona no  
array

Veja mais opções de push  
<https://www.mongodb.com/pt-br/docs/manual/reference/operator/update/push/>

# Como atualizar documentos

```
meubanco> db.pessoas.updateOne({_id: ObjectId("672369eb2de66b0a2cfe6912")}, {$push:{"notas": 94}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

## Como atualizar documentos

```
{  
  _id: ObjectId('672369eb2de66b0a2cfe6912'),  
  nome: 'Aluno',  
  email: 'email@email.com',  
  notas: [ 44, 78, 38, 80, 94 ]  
}
```



# Para excluir uma coleção do banco

```
db.collection.drop()
```

O método `drop()` retorna `true` se a coleção foi apagada com sucesso, e `false` caso contrário.

Para excluir documentos individuais de uma coleção, pode-se utilizar os métodos **`deleteOne()`** ou **`deleteMany()`**

# Para excluir documentos de coleção do banco

- Para excluir todos os documentos de uma coleção, passe o filtro vazio do `deleteMany()`

```
db.myCollection.deleteMany({})
```

- Para excluir os documentos de uma coleção que possuem um nome específico:

```
db.myCollection.deleteMany({"nome": "Fulano"})
```

```
meubanco> db.pessoas.deleteMany({"nome": "Bill"})  
{ acknowledged: true, deletedCount: 2 }
```

# Para excluir apenas 1 documento

- Para excluir o primeiro documento da coleção que contém o nome “Ciclano”

```
db.myCollection.deleteOne({"nome": "Ciclano"})
```

O MongoDB preserva uma ordem de classificação natural para documentos. Essa ordenação é um recurso de implementação interna, e você não deve confiar em nenhuma estrutura específica dentro dela

# Exercício

1)

- Crie um banco de dados chamado "livraria".
- Dentro desse banco, crie três coleções: "livros", "autores" e "pedidos".

# Exercício

2)

- Insira os seguintes documentos na coleção "livros":

```
{  
  "titulo": "O Senhor dos Anéis",  
  "autor": "J.R.R. Tolkien",  
  "preco": 50,  
  "quantidade": 10  
}
```

```
{  
  "titulo": "Harry Potter",  
  "autor": "J.K. Rowling",  
  "preco": 40,  
  "quantidade": 15  
}
```

# Exercício

2)

- Insira os seguintes documentos na coleção "autores":

```
{  
  "nome": "J.R.R. Tolkien",  
  "dataNascimento": "1892-01-03"  
}
```

```
{  
  "nome": "J.K. Rowling",  
  "dataNascimento": "1965-07-31"  
}
```

# Exercício

2)

- Insira os seguintes documentos na coleção "pedidos":

```
{  
  "dataPedido": "2022-01-01",  
  "total": 100,  
  "livros": [  
    {"titulo": "O Senhor dos Anéis", "quantidade": 2},  
    {"titulo": "Harry Potter", "quantidade": 1}  
  ]  
}
```

```
}
```

# Exercício

3)

- Atualize o preço do livro "O Senhor dos Anéis" para 55.
- Adicione um novo autor à coleção "autores":

```
{  
  "nome": "George R.R. Martin",  
  "dataNascimento": "1948-09-20"  
}
```

- Atualize o pedido mais recente para adicionar um novo livro:

```
{  
  "titulo": "A Song of Ice and Fire",  
  "quantidade": 1  
}
```



# Exercício

- 4) - Exclua o livro "Harry Potter" da coleção "livros".
- Exclua um pedido da coleção pedidos

**Resolução será postada para aula remota.**