

# Aula 06

---

- MongoDB;
- Coleção e Documentos;
- Modelagem MongoDB;
- Inclusão;
- Alteração, Exclusão e Seleção.

**MongoDB**

# O que é o MongoDB?

---

- É um banco de dados não relacional (NoSQL), e por isso não tem estrutura definida de tabelas, e nem muito menos tabelas;
- O MongoDB é orientado a documentos;
- No MongoDB os termos mudam;
- A organização dos dados também muda;
- Importante: O MongoDB é **case sensitive para nomes de bancos de dados**, ou seja "**livraria**" é um banco e "**Livraria**" é outro.

# Documentos e coleções

# Documentos

---

- É a unidade básica de dados no MongoDB;
- É basicamente uma estrutura de chave e valor (é análogo as colunas nos bancos de dados relacionais), semelhante ao JSON;
- Utiliza o BSON (Binary JSON), que é uma extensão do JSON, assim aumentando a quantidade de tipos de dados suportados:
  - O **JSON** normalmente tem tipo de dados como String, Number, Array, Object, null;
  - O **BSON** tem os mesmos do JSON porém tem outros a mais como ObjectId, Date, Timestamp, Regex;
- Fazendo um comparativo de alto nível os documentos estão no mesmo lugar das linhas das tabelas do modelo relacional;

# Documentos

---

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value  
← field: value  
← field: value  
← field: value

# Documentos

---

- Nos banco de dados relacionais temos as chaves primárias, que podemos definir da forma que acharmos melhor;
- Entretanto no MongoDB o campo análogo uma chave primária é o `_id`, que é sempre implementado, ou seja mesmo que você não indique explicitamente um valor, o MongoDB irá criar um automaticamente e anexar ao documento.

# Coleções

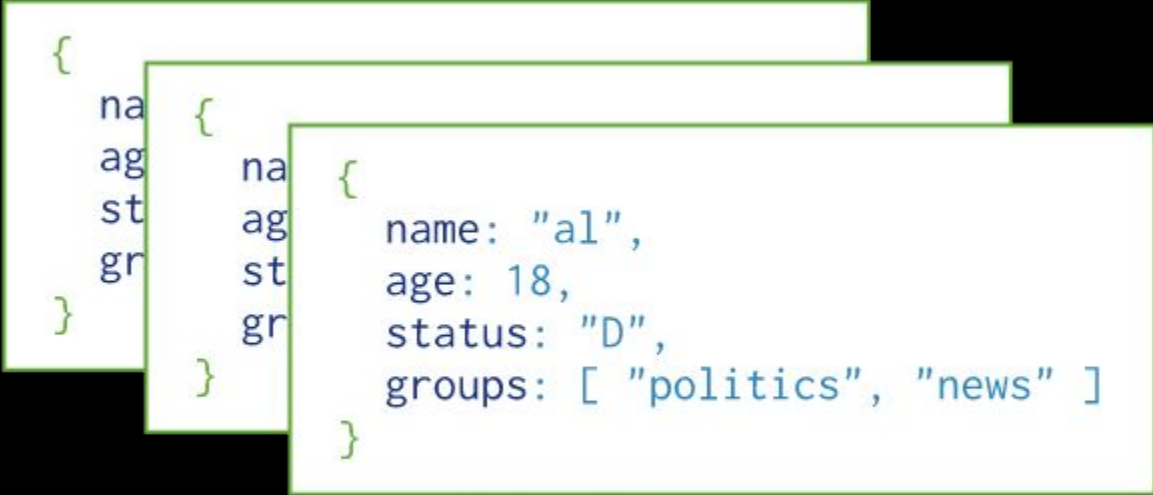
---

- Os documentos são armazenados em coleções;
- Ou seja, coleções são conjuntos de documentos;
- E as coleções podemos dizer é análogas as tabelas nos banco de dados relacionais;



# Coleções

---



```
{  
  na  
  ag  
  st  
  gr  
}  
  
{  
  na  
  ag  
  st  
  gr  
}  
  
{  
  name: "al",  
  age: 18,  
  status: "D",  
  groups: [ "politics", "news" ]  
}
```

Collection

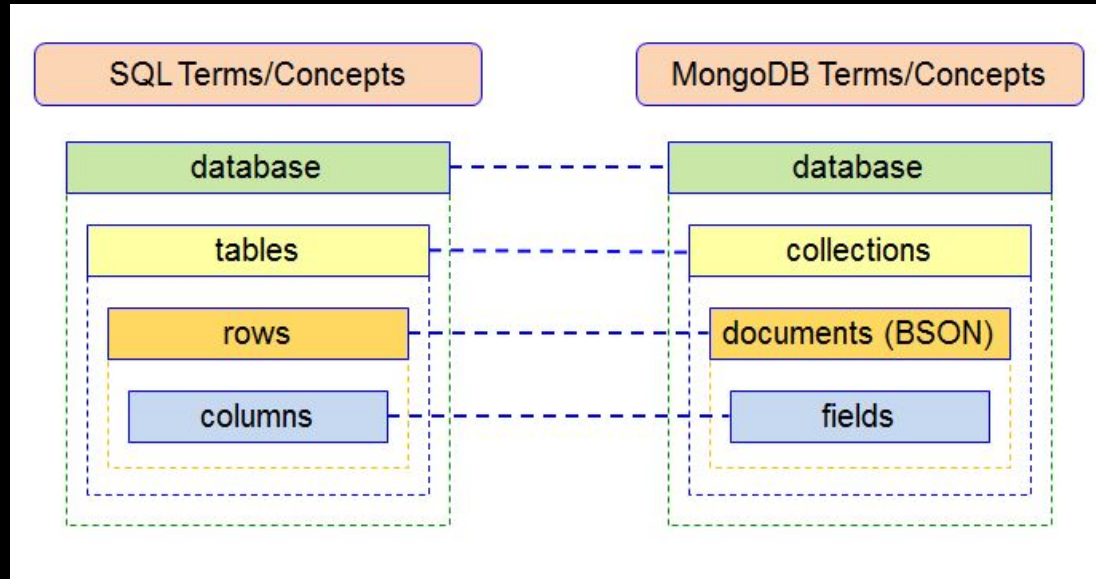
# Estrutura flexível

---

- Os **documentos** dentro de uma mesma **coleção** não precisam ter o mesmo conjunto de **campos**;
- O **tipo de dados** de um mesmo campo pode ser diferente dos demais **documentos** da mesma **coleção**;
- Com essa flexibilidade é mais fácil mapear os documentos para um objeto, mesmo com variação dos documentos;

# Analogia: SQL vs MongoDB

---



Fonte:

<https://studio3t.com/academy/topic/mongodb-vs-sql-concepts/>

# Modelagem MongoDB

# Dados integrados

---

- Os documentos possibilitam incorporar outros documentos em campo ou um array dentro do mesmo;
- Com isso os documentos incorporados ganham um relacionamento com os dados armazenados em uma única unidade de documento;
- Isso permite que seja feita apenas uma única operação para recuperar os dados relacionados;

# Dados integrados

---

```
{
  _id: <ObjectId>,
  username: "123xyz",
  contact: {
    phone: "123-456-7890",
    email: "xyz@example.com"
  },
  access: {
    level: 5,
    group: "dev"
  }
}
```

Embedded sub-document

Embedded sub-document

Fonte:

<https://www.mongodb.com/docs/manual/core/data-modeling-introduction/>

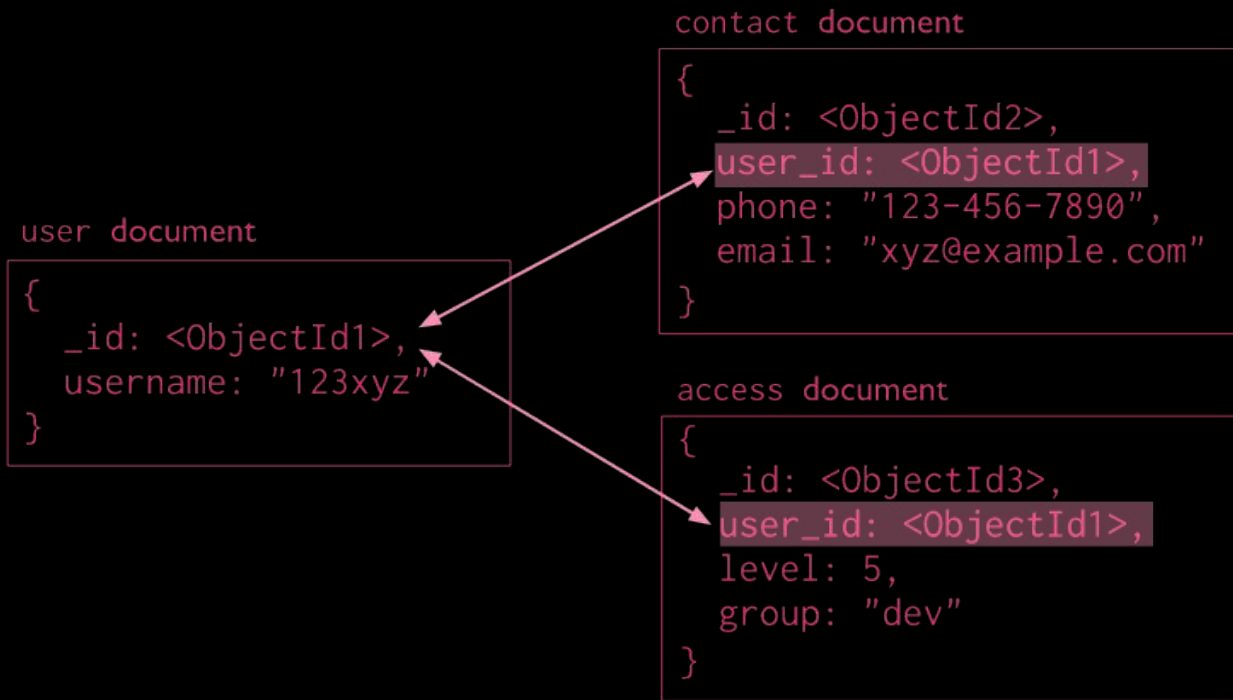
# Referências

---

- Também chamado de **links**, nesse padrão as referências armazenam as relações entre os dados;
- Assim, podemos incluir uma referência (link) de um documento para outro;
- Porém, quem estiver consultando os dados precisa resolver as referências para acessar os outros dados relacionados.
- É semelhante a uma estrutura **normalizada** dos dados, e se aproxima da forma utilizada pelos bancos de dados relacionais.

# Referências

---



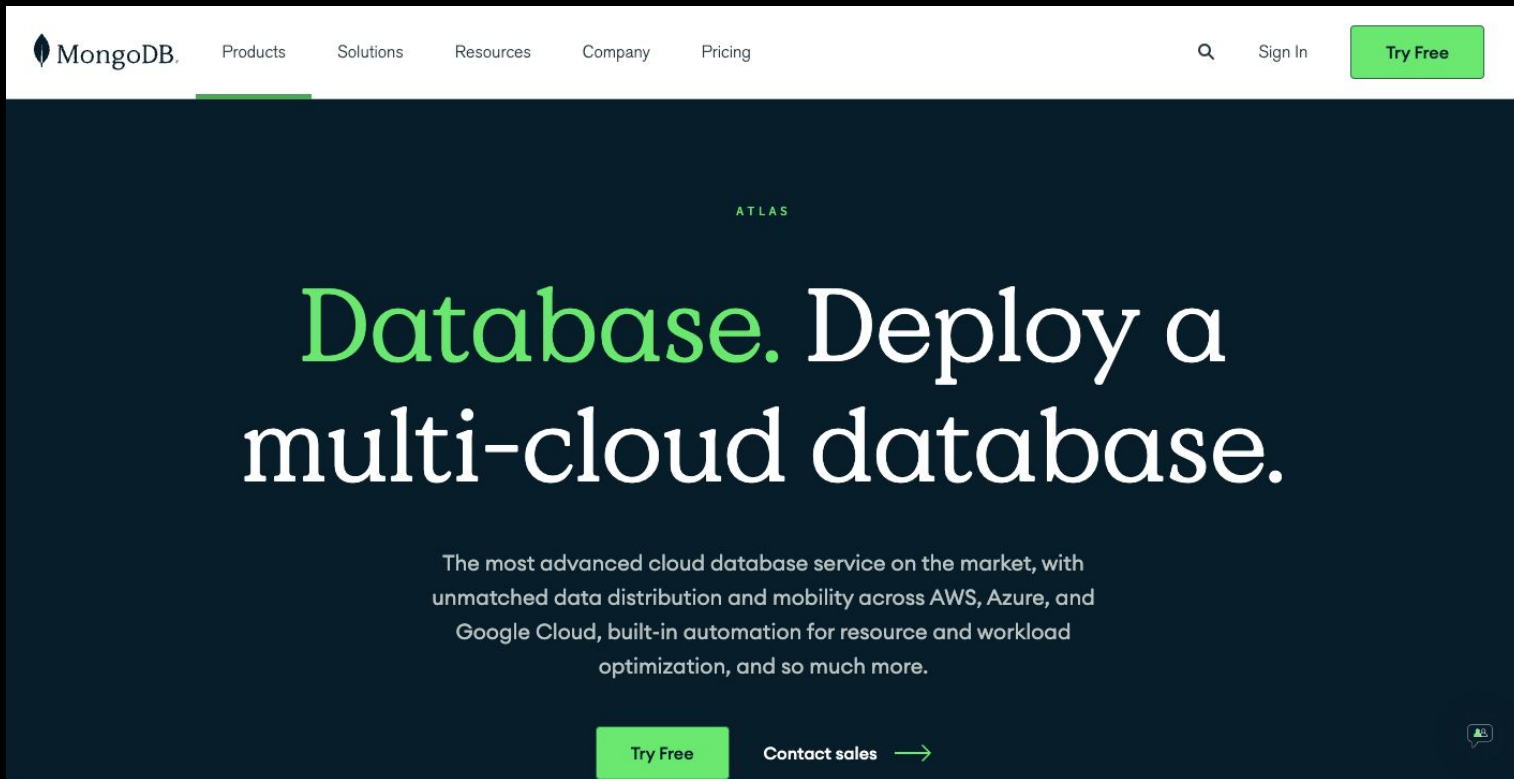
Fonte:

<https://www.mongodb.com/docs/manual/core/data-modeling-introduction/>



# Direto ao ponto com MongoDB Atlas

---



The screenshot shows the MongoDB Atlas website. The top navigation bar is white with the MongoDB logo on the left and links for Products, Solutions, Resources, Company, and Pricing in the center. On the right, there is a search icon, a 'Sign In' link, and a green 'Try Free' button. The main content area has a dark blue background. At the top center of this area is the word 'ATLAS' in small green capital letters. Below it is the main headline 'Database. Deploy a multi-cloud database.' where 'Database.' is in green and 'Deploy a multi-cloud database.' is in white. Under the headline is a paragraph of text: 'The most advanced cloud database service on the market, with unmatched data distribution and mobility across AWS, Azure, and Google Cloud, built-in automation for resource and workload optimization, and so much more.' At the bottom of the main content area, there is a green 'Try Free' button, a 'Contact sales' link followed by a green arrow, and a small chat icon in the bottom right corner.

MongoDB. Products Solutions Resources Company Pricing

Search Sign In Try Free

ATLAS

## Database. Deploy a multi-cloud database.

The most advanced cloud database service on the market, with unmatched data distribution and mobility across AWS, Azure, and Google Cloud, built-in automation for resource and workload optimization, and so much more.

Try Free Contact sales →

Chat icon

# **Gerenciando banco de dados**

# Como criar um banco de dados?

---

- Usando uma interface gráfica é bem simples e já deve ter um botão "Criar Banco de dados";
- Porém usando a CLI (command-line interface), o processo é diferente e tem uns detalhes importantes;
- Para listar os bancos de dados você usar o seguinte comando:

```
1 > show dbs
```

```
2
```

```
3 admin          0.000GB
```

```
4 local          3.104GB
```

# Como criar um banco de dados?

---

- Sempre teremos os bancos admin e local no mongo, já que fazem parte do MongoDB;
- **Não existe comando para criar banco de dados!**
- Para criar um novo banco de dados é usado o comando **use**:



```
1 > use novoBancoDeDados
```

# Como criar um banco de dados?

---

- Porém se executarmos novamente o comando `show dbs`:



```
1 > show dbs
2
3 admin          0.000GB
4 local          3.104GB
```

- Teremos o mesmo resultado, já que no mongo o banco de dados só é totalmente criado depois que você coloca alguma informação nele.
- E isso nos leva aos próximos passos: Como inserir? Alterar? Excluir? Buscar?

**Inclusão**

# Inclusão

---



Title

```
1 // Criando uma nova coleção
2 db.createCollection(nome, <opcional: Documento JSON com opções>)
3
4 // Exemplo
5 db.createCollection("pessoas")
6
7
8 // Importante: Aqui se aplica o mesmo da criação de banco de dados, se criar uma
   coleção mas não colocar documentos na mesma ela não é realmente criada.
```

# Inclusão

---




```
1 // Inserir um documento
2 db.nomeColecao.insertOne({ documento JSON })
3
4 // Inserir vários documentos de uma vez
5 db.NomeColeção.insertMany([ { documento_1 }, { documento_2 }, ... ])
6
7 // Exemplos
8 db.pessoas.insertOne({ "nome": "João da Silva" })
9 db.pessoas.insertMany([ { "nome": "Ana dos Santos" }, { "nome": "Maria Silva" } ])
```



# Voltando a criação do banco de dados

---

- Depois que adicionarmos coleções, podemos listar os banco de dados novamente, e veremos o nosso novo banco de dados na lista:



```
1 > show dbs
2
3 admin          0.000GB
4 local          3.104GB
5 novoBancoDeDados 0.000GB
```

# Como saber em qual banco de dados estou?

---

- Para saber qual banco de dados você está atualmente basta usar o comando **db**:



```
1 > db
```

```
2
```

```
3 novoBancoDeDados
```

**Alteração, exclusão e seleção**

# Alteração

---

```
1 // Alterando um único documentos
2 db.nomeDaColecao.updateOne(<filter>, <update>, <options>)
3
4 // Exemplo
5 db.categorias.updateOne(
6   { "_id": 1 },
7   {
8     $set: { "nome": "Ação" }
9   }
10 )
11
12 // Importante: O campo "_id" não pode ser alterado!
```

# Alteração

---

```
1 // Alterando vários documentos de uma vez
2 db.nomeDaColecao.updateMany(<filter>, <update>, <options>)
3
4 // Exemplo
5 db.livros.updateMany(
6   { "paginas": 400 },
7   {
8     $set: { "ano": 2021 }
9   }
10 )
```

# Alteração

---



```
1 // Excluindo um documento inteiro da coleção
2 db.nomeDaColecao.replaceOne(<filter>, <update>, <options>)
3
4 // Exemplo
5 db.categorias.replaceOne({ "_id": 1 }, { "nome": "Comédia" })
```

# Exclusão

---



```
1 // Excluindo um único documento
2 db.nomeDaColecao.deleteOne(<filter>)
3
4 // Exemplo
5 db.livros.deleteOne({ "_id": 3 })
6
7 // Sempre use algum campo único como o "_id" para uma exclusão precisa!
```

# Exclusão

---

```
1 // Excluindo todos os documentos
2 db.nomeDaColecao.deleteMany()
3
4 // Excluindo todos os documentos que correspondam a uma condição
5 db.livros.deleteMany({ "ano": 2021 })
6
7 // Excluindo todos os documentos de uma coleção, todos mesmo!
8 db.livros.deleteMany({})
9
10 // Cuidado!
```



# Seleção

---

```
1 // Selecionando todos os documentos de uma coleção
2 db.nomeDaColecao.find({})
3
4 // Exemplo
5 db.categorias.find({})
6
7 // Esse comando é análogo ao "SELECT * FROM categorias" do SQL
```

# Seleção

---

```
1 // Selecionando os documentos de uma coleção com filtro
2 db.nomeDaColecao.find({ <field>:<value> })
3
4 // Exemplo
5 db.livros.find({ "ano": 2019 })
6
7 // Esse comando é análogo ao
8 // "SELECT * FROM livros WHERE ano = 2019" do SQL
```

# Seleção

---



```
1 // Selecionando os documentos com operadores de consulta
2 db.nomeDaColecao.find({ <field1>: { <operator1>: <value1> }, ... })
3
4 // Exemplo
5 db.livros.find({ "ano": { $in: [ 2019, 2020 ] } })
6
7 // Esse comando é análogo ao
8 // "SELECT * FROM livros WHERE ano in (2019, 2020)" do SQL
```

# Seleção

---



```
1 // Selecionando os documentos com multiplos filtros com AND
2 db.nomeDaColecao.find({ <field_1>: <value_1> }, { <field_2>: <value_2> })
3
4 // Exemplo
5 db.livros.find({ "ano": 2019, "categoria_id": 1 })
6
7 // Esse comando é análogo ao
8 // "SELECT * FROM livros WHERE ano = 2019 AND categoria_id = 1" do SQL
```

# Seleção

---



```
1 // Selecionando os documentos usando o operador OR ($or)
2 db.nomeDaColecao.find({ $or: [ { <field_1>: <value_1> }, {<field_2>: <value_2> } ] } )
3
4 // Exemplo
5 db.livros.find({ $or: [ { "ano": 2019 }, { "categoria_id": 1 } ] } )
6
7 // Esse comando é análogo ao
8 // "SELECT * FROM livros WHERE ano = 2019 OR categoria_id = 1" do SQL
```

# Seleção de documentos incorporados

---



```
1 // Podemos usar o find para selecionar documentos incorporados
2 db.nomeDaColecao.find({ "nomeDocumentoIncorporado": { <field_1>: <value_1>, ... } })
3
4 // Importante: Todos os campos e valores do "nomeDocumentoIncorporado" precisam
   corresponder e ainda precisar ser colocados exatamente na mesma ordem que está salvo
```

# Exercícios de MongoDB