



Fatec Votorantim
Faculdade de Tecnologia

CPS

SÃO PAULO

Desenvolvimento de Software Multiplataforma

Banco de Dados Não Relacional

Fatec Votorantim – Prof. Ms. Ricardo Leme

Olá!

**Eu sou o Prof.
Ricardo Leme**



ricardo.leme@fatec.sp.gov.br



<https://github.com/ricardoleme>




<https://www.linkedin.com/in/ricardo-leme/>

Algo em comum...

Eu e o jovem ao lado
temos algo em
comum...

Agnosia Facial 😐



- 
- ▶ Experiência de mais de 30 anos com desenvolvimento
 - ▶ Professor Universitário desde 2003
 - ▶ Mestre em Ciência da Computação pela UFSCar
 - ▶ Entusiasta de Tecnologias Open Source



Firebase



mongoDB



“
A razão pela qual os bancos de dados NoSQL são chamados assim é porque a próxima pergunta, depois de explicar *'não SQL'*, é quase sempre, *'Então, o que você está usando'*?

- Dwight Merriman
(co-fundador da MongoDB Inc.)



Plano de Ensino

Competências Profissionais desenvolvidas

- ▶ Desenvolver projetos de Banco de Dados utilizando diferentes abordagens de modelagem e implementação a fim de garantir a qualidade dos dados.
- ▶ Utilizar adequadamente as técnicas de armazenamento e tratamento de dados não-estruturados, visando qualidade, manutenção e segurança;
- ▶ Utilizar adequadamente os princípios de armazenamento e tratamento de dados a fim de suportar a recuperação de dados utilizados em aplicações.



Plano de Ensino

Objetivos de Aprendizagem

- ▶ Caracterizar Banco de Dados Relacional e Não Relacional, de acordo com a especificação do projeto.
- ▶ Utilizar Banco de Dados Não Relacional.
- ▶ Utilizar Sistemas de Banco de Dados paralelos e distribuídos.
- ~~▶ Compreender os conceitos de Data Warehouse e Mineração de Dados.~~
- ▶ Identificar métodos seguros para gerenciamento do Banco de Dados.



Informações Importantes

Sobre a condução da disciplina



Chamada

- ▶ Será realizada em dois momentos (**antes do início** das duas primeiras aulas **e no fim** das duas últimas)
- ▶ As regras de **reprovação por falta** (mínimo de 75% de presença continuam as mesmas!)
- ▶ Faremos a **parada no intervalo**,



Avaliações e Trabalhos

- ❖ A avaliação será composta de dois trabalhos, duas avaliações e do PI:
 - **07/04** - Entrega de um projeto em MongoDB e NodeJS implementando um backend Restful.
 - **14/04** - Primeira Avaliação teórica (múltipla escolha)
 - **16/06** - Segunda Avaliação teórica (múltipla escolha)
 - **23/06** - Implementação de autenticação JWT, documentação da API implementada e integração com frontend.
 - Projeto Interdisciplinar

Média Final =

$(\text{Prova 1} + \text{Trabalho 1} + \text{Prova 2} + \text{Trabalho 2} + \text{PI}) / 5$



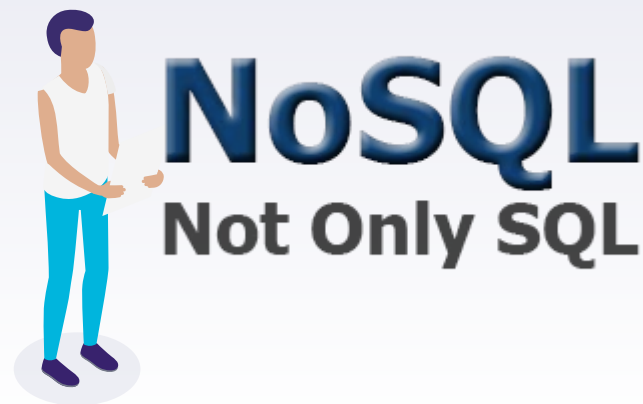
As aulas exigirão o uso do computador!

- ❖ Para o desenvolvimento do projeto, utilizaremos GIT, Github, Node e npm.
- ❖ Utilizaremos o MongoDB **local*** e o VS Code com a extensão abaixo:



The screenshot shows the extension page for 'MongoDB for VS Code' on the Visual Studio Marketplace. On the left is a green leaf icon. The title 'MongoDB for VS Code' is in white, with an orange 'Preview' badge to its right. Below the title, it says 'MongoDB | 95,890 installs | 4.5 stars (15) | Free'. A description in white text reads: 'Connect to MongoDB and Atlas directly from your VS Code environment, navigate your databases and collections, inspect your schema and use playgrounds to prototype queries and aggregations.' At the bottom, there is a green 'Install' button and a link 'Trouble Installing?' with an external link icon.

<https://marketplace.visualstudio.com/items?itemName=mongodb.mongodb-vscode>



O que é?

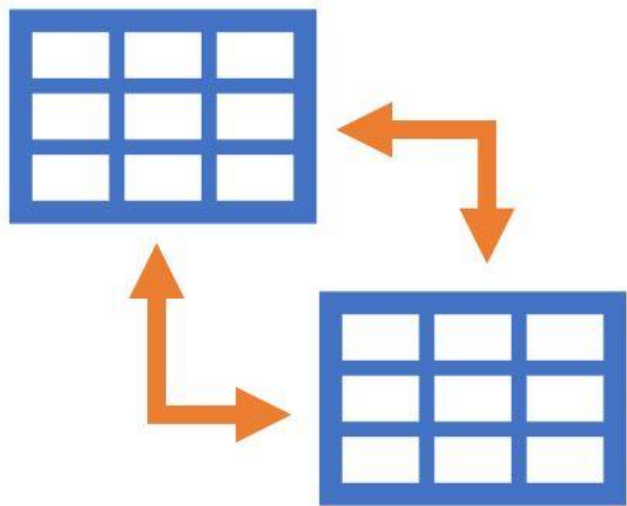
O NoSQL é um tipo diferente de sistema de armazenamento de dados em computadores. Enquanto os bancos de dados tradicionais têm **regras rígidas** sobre como organizar e acessar dados, o NoSQL **é mais flexível**. Ele permite armazenar e acessar informações de maneira mais livre e rápida, o que é útil para lidar com grandes volumes de dados e em situações onde a estrutura dos dados pode mudar com frequência.

NoSQL

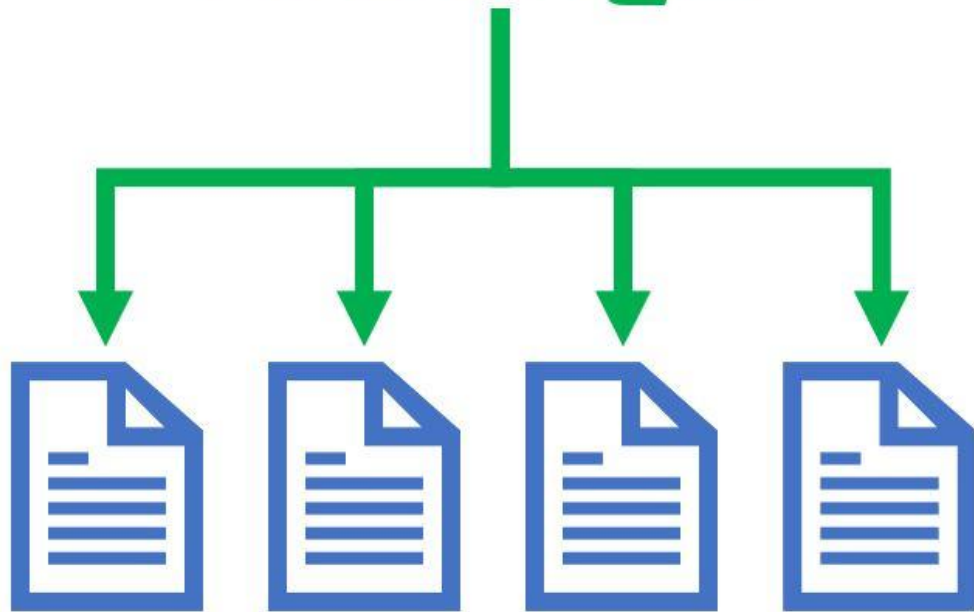


Muito do que vamos ver na utilização de bancos de dados NoSQL não é sobre resolver problemas que são impossíveis de serem resolvidos com um banco relacional, mas sim sobre como podemos ter soluções mais elegantes e mais práticas, além de muitas vezes também mais performáticas e escaláveis com o NoSQL.

SQL VS *NoSQL*



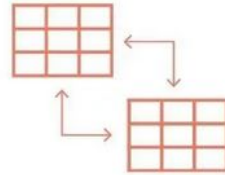
(Structured Data)



(Un-Structured Data)

SQL

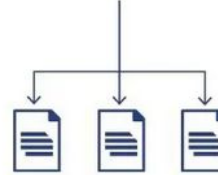
Structured



VS

NoSQL

Unstructured



Fixed rows & columns - Static schema

Dynamic schema

Structured data

Unstructured data

Suited for complex transactions

Suited for big data analytics

Best for e-commerce platforms

Best for social media platforms

MySQL, PostgreSQL

MongoDB, Cassandra

Principais Bancos NoSQL



Azure Cosmos DB



Apache Cassandra



DynamoDB



Couchbase



Firebase



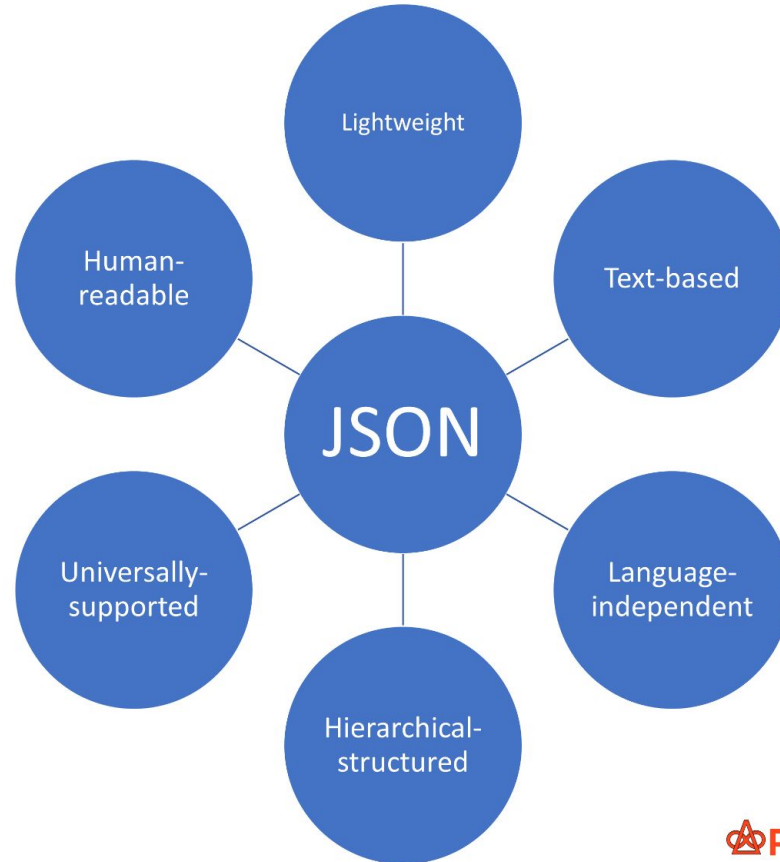
redis

Fonte: <https://db-engines.com/en/ranking/document+store>

Documento no formato JSON

```
{
  "id": 55,
  "País": "Brasil",
  "Região": "América do Sul",
  "População": 201032714,
  "PrincipaisCidades": [
    {
      "NomeCidade": "São Paulo",
      "População": 1182876,
    },
    {
      "NomeCidade": "Rio de Janeiro",
      "População": 6323037,
    }
  ]
}
```

JSON - JavaScript Object Notation



MongoDB

O MongoDB é um SGBD NOSQL *open-source* e orientado a documentos.

Alguns de seus diferenciais são:

- ▶ Alto desempenho: documentos embutidos e índices atuando sobre eles;
- ▶ Rica linguagem de consulta: permite operações CRUD, agregações de dados, busca por texto e consultas geoespaciais;
- ▶ Alta disponibilidade: *replica set*; (escalonamento)
- ▶ Escalabilidade horizontal: sharding. (particionamento)












Como está o uso do MongoDB?

- ▶ Com a popularidade e a consolidação da linguagem SQL no mercado, este tipo de questionamento é comum.
- ▶ **DB-ENGINES RANKING**: ranking de popularidade dos SGBD mais utilizados, atualizado mensalmente.
- ▶ Pode ser acessado em: <https://db-engines.com/en/ranking>;
- ▶ Considera uma série de critérios para obter uma pontuação capaz de classificar os SGBD.

Db Engines Ranking (General)

Rank			DBMS	Database Model	Score
Feb 2025	Jan 2025	Feb 2024			
1.	1.	1.	Oracle	Relational, Multi-model ⓘ	1254.82
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	999.99
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model ⓘ	786.87
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	659.62
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	396.63

Db Engines Ranking (Document Store)

Rank			DBMS	Database Model	Score		
Feb 2025	Jan 2025	Feb 2024			Feb 2025	Jan 2025	Feb 2024
1.	1.	1.	MongoDB 	Document, Multi-model 	396.63	-5.87	-23.73
2.	2.	 3.	Databricks 	Multi-model 	90.03	+2.19	+13.13
3.	3.	 2.	Amazon DynamoDB 	Multi-model 	75.58	+2.58	-7.31
4.	4.	4.	Microsoft Azure Cosmos DB 	Multi-model 	22.13	-0.83	-9.86
5.	5.	5.	Couchbase 	Multi-model 	15.63	-0.38	-4.85
6.	6.	6.	Firebase Realtime Database	Document	13.32	+0.21	-3.02
7.	7.	7.	CouchDB	Document, Multi-model 	7.81	+0.03	-4.86
8.	 9.	8.	Google Cloud Firestore	Document	6.92	+0.19	-3.95
9.	 8.	9.	Realm	Document	6.74	-0.25	-1.22
10.	10.	 11.	Aerospike 	Multi-model 	5.09	+0.05	-2.00

Critérios do Db Ranking



- ▶ Menções do SGBD em mecanismos de busca;
- ▶ Interesse geral no SGBD (Google Trends);
- ▶ Frequência de discussões técnicas sobre o SGBD (StackOverflow e DBAStackExchange);
- ▶ Número de ofertas de emprego relacionadas ao SGBD;
- ▶ Número de perfis em redes profissionais onde o SGBD é mencionado (LinkedIn e Upwork);
- ▶ Relevância em redes sociais (Twitter/X).

Veja como estão as vagas no mercado de trabalho:

<https://www.infojobs.com.br/empregos.aspx?Palabra=mongodb>

O que é o conceito do documento?




Não Relacional



X

Relacional

O que é o conceito do documento?

NOTA FISCAL ELETRÔNICA SÉRIE 1 Nº 000.096.31 DATA DE EMISSÃO 23/03/2016 DATA DE RECEBIMENTO		RECIBO DO DESTINATÁRIO OS PRODUTOS RECEBIDOS CONFORME A NOTA FISCAL INDICADO ABAIXO IDENTIFICAÇÃO E ASSINATURA DO RECEBEDOR		NOTA FISCAL ELETRÔNICA SÉRIE 1 Nº 000.096.31 DATA DE EMISSÃO 23/03/2016 DATA DE RECEBIMENTO		RECIBO DO TRANSPORTADOR OS PRODUTOS RECEBIDOS CONFORME A NOTA FISCAL INDICADO ABAIXO ASSINATURA	
LOGOMARCA DA EMPRESA EMISSORA DA NF-e				DANFE DOCUMENTO AUXILIAR DA NOTA FISCAL ELETRÔNICA 0 - ENTRADA 1 - SAÍDA 1 Nº 000.096.31 SÉRIE 1 FL 1 / 1		 CHAVE DE ACESSO 4316 0304 1234 7100 0148 5500 1000 096312170554 8158 Consulta de autenticidade no portal nacional da NF-e http://www.nfe.fazenda.gov.br/portal ou no site da Sefaz Autorizadora	
NATUREZA DA OPERAÇÃO VENDA MERCADO REGIME DE FISCALIZAÇÃO 0000000000				INSCRIÇÃO ESTADUAL DO SUBSTITUTO TRIBUTÁRIO 00.000.000/0000-00		PROTOCOLO DE AUTORIZAÇÃO DE USO 1431600401619888-2303/2016134148 CNPJ 00.000.000/0000-00	
DESTINATÁRIO / REMETENTE NOME / RAZÃO SOCIAL DATAM EX TECNOLOGIA DA INFORMACAO LTDA ENDEREÇO Avenida PRESIDENTE VARGAS, 1 MUNICÍPIO RIO GRANDE				INSCRIÇÃO ESTADUAL 00.000.000/0000-00		CNPJ / CPF 00.000.000/0000-00 DATA DA EMISSÃO 23/03/2016	
BAIRRO / DISTRITO VILA JUNAÇÃO UF RS				CEP 96.202-188 REGISTRO ESTADUAL 13.41.40		DATA DA ENTRADA / SAÍDA 23/03/2016	
FATURA / DUPLICATA NÚMERO 96312-A VENCIMENTO 20/04/2016 VALOR 198,48				NÚMERO 96312-A VENCIMENTO 20/04/2016 VALOR 198,48		NÚMERO 96312-A VENCIMENTO 20/04/2016 VALOR 198,48	
CÁLCULO DO IMPOSTO BASE DE CÁLCULO DO ICMS 0,00 VALOR DO ICMS 0,00 VALOR DO ICMS SUBSTITUIÇÃO 0,00 VALOR TOTAL DOS PRODUTOS 198,48				VALOR DO ICMS 0,00 VALOR DO ICMS SUBSTITUIÇÃO 0,00 VALOR TOTAL DOS PRODUTOS 198,48		VALOR DO ICMS 0,00 VALOR DO ICMS SUBSTITUIÇÃO 0,00 VALOR TOTAL DOS PRODUTOS 198,48	
TRANSPORTADOR / VOLUME TRANSPORTADOS NOME / RAZÃO SOCIAL E.V. BARROS ENDEREÇO RUA...				FICHA DE CONTAGEM 1 DESP/REM 1		CÓDIGO ANTT 1 PLACA DO VEÍCULO 1 UF RS	
QUANTIDADE 1 ESPÉCIE 1 MARCA 1 NÚMERO 1 PESO BRUTO 1 PESO LÍQUIDO 1				QUANTIDADE 1 ESPÉCIE 1 MARCA 1 NÚMERO 1 PESO BRUTO 1 PESO LÍQUIDO 1		QUANTIDADE 1 ESPÉCIE 1 MARCA 1 NÚMERO 1 PESO BRUTO 1 PESO LÍQUIDO 1	
DADOS DOS PRODUTOS / SERVIÇOS CÓDIGO PRODUTO 3667 DESCRIÇÃO DOS PRODUTOS / SERVIÇOS 07 PROTETOR DE REDE 55 16A/5W MAG 3001 1 (cm) NBR Valor aprox. Tributos: 15,43				CÓDIGO PRODUTO 3667 DESCRIÇÃO DOS PRODUTOS / SERVIÇOS 07 PROTETOR DE REDE 55 16A/5W MAG 3001 1 (cm) NBR Valor aprox. Tributos: 15,43		CÓDIGO PRODUTO 3667 DESCRIÇÃO DOS PRODUTOS / SERVIÇOS 07 PROTETOR DE REDE 55 16A/5W MAG 3001 1 (cm) NBR Valor aprox. Tributos: 15,43	
CÓDIGO PRODUTO 3667 DESCRIÇÃO DOS PRODUTOS / SERVIÇOS 07 PROTETOR DE REDE 55 16A/5W MAG 3001 1 (cm) NBR Valor aprox. Tributos: 15,43				CÓDIGO PRODUTO 3667 DESCRIÇÃO DOS PRODUTOS / SERVIÇOS 07 PROTETOR DE REDE 55 16A/5W MAG 3001 1 (cm) NBR Valor aprox. Tributos: 15,43		CÓDIGO PRODUTO 3667 DESCRIÇÃO DOS PRODUTOS / SERVIÇOS 07 PROTETOR DE REDE 55 16A/5W MAG 3001 1 (cm) NBR Valor aprox. Tributos: 15,43	
DADOS ADICIONAIS INFORMAÇÕES COMPLEMENTARES Valor aprox. Total Tributos: 18,35				INFORMAÇÕES COMPLEMENTARES Valor aprox. Total Tributos: 18,35		INFORMAÇÕES COMPLEMENTARES Valor aprox. Total Tributos: 18,35	

Obtendo o MongoDB

O MongoDB pode ser obtido de duas maneiras diferentes:

- ▶ Download On-Premises - <https://www.mongodb.com/try/download/community>
- ▶ Cloud via MongoDB Atlas - <https://www.mongodb.com/try> (free até 512 Mb)



Instale a
extensão
oficial no
VsCode



MongoDB for VS Code `mongodb.mongodb-vscode` Preview

MongoDB | 30.292 | ★★★★★ | Repository | License | v0.1.1

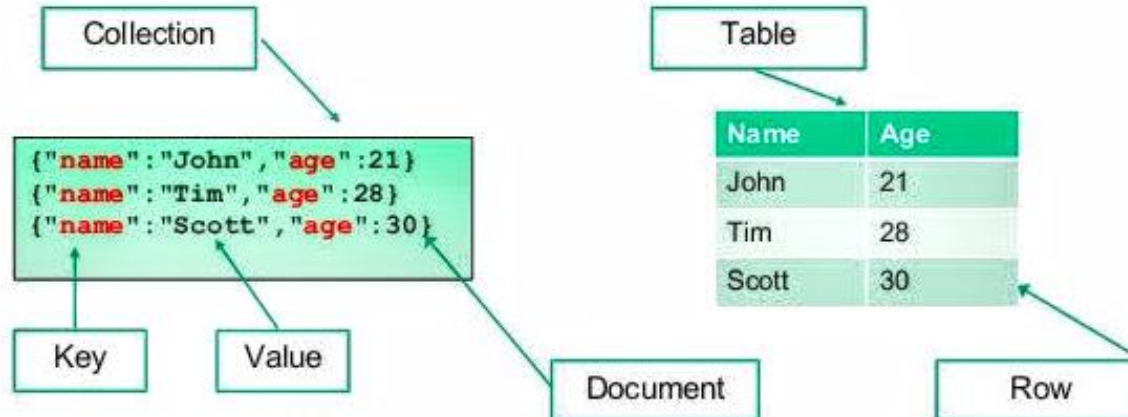
Connect to MongoDB and Atlas directly from your VS Code environment, navigate your databases and collections, inspect your schema and use playgrounds to prototype queries and aggregations.

Uninstall

Principais termos e conceitos

Basic Translation Terms/Concepts

Mongo/NoSQL Terms	Traditional SQL Terms
Database	Database
Collection	Table
Document	Row
Field	Column



Criando um banco de dados

O MongoDB abstrai diversos comandos DDL (Data Definition Language) do SQL

- ▶ Estruturas são criadas conforme estas são necessárias
- ▶ Exemplo: Para criar um banco de dados, basta você utilizar o comando **para acessar um banco de dados que ainda não existe.**



```
CREATE  
DATABASE  
DB_AULA;
```



```
USE  
DB_AULA
```

Coleções e documentos

- ▶ Como em outros modelos orientado a documentos, o MongoDB organiza os dados em **coleções de documentos**.
- ▶ Cada documento possui um atributo identificador (`_id`) e uma quantidade qualquer de outros atributos.
- ▶ Não é necessário (Mas é possível) especificar o ID dos documentos.
- ▶ Não é **obrigatório** especificar o tipo dos atributos.
- ▶ Documentos diferentes que fazem parte de uma mesma coleção podem ter atributos diferentes.

Estruturas do ObjectId

Um ObjectId é um tipo BSON binário de 12 bytes representados em 24 caracteres hexadecimais:

```
{  
  "_id": ObjectId("54759eb3c090d83494e2d804")  
}
```

Esses bytes são gerados automaticamente e separados em grupos com funcionalidades específicas:

Tamanho	Descrição
4 bytes	4 bytes que representa os segundos desde a época do Unix
3 bytes	3-byte com identificador de máquina
2 bytes	2-byte com o identificador único do processo
3 bytes	3-byte contador que começa com um número aleatório por coleção

Tipos de Dados (BSON Types)

- ▶ **String:** Este é o tipo de dados mais comumente usado para armazenar dados. String no MongoDB deve ser um UTF-8 válido.
- ▶ **Integer:** Este tipo é usado para armazenar um valor numérico. Integer pode ser 32 bits ou 64 bits dependendo do seu servidor.
- ▶ **Boolean:** Este tipo é usado para armazenar um valor booleano (verdadeiro/falso).
- ▶ **Double:** Este tipo é usado para armazenar valores de ponto flutuante.
- ▶ **Arrays:** Este tipo é usado para armazenar arrays, listas ou múltiplos valores dentro de uma chave(key).

Data Type	Description
String	Unicode character string.
Integer	32-bit integer number.
Long	64-bit integer number.
Double	64-bit decimal number.
Boolean	Boolean value (true or false).
Date	Date and time. Stored as a number of milliseconds since January 1, 1970.
Object	Nested JSON document.
Array	List of values, can include values of different types.
Binary	Binary data, such as images or files.
ObjectId	Unique identifier for a BSON document.
Decimal128	128-bit high-precision decimal number.
Regular Expression	Regular expression used for pattern matching in strings.
Null	Null value.
Undefined	Undefined value.

Tipos de Dados (BSON Types)

- ▶ **Timestamp:** Isto pode ser útil para a gravação de quando um documento foi modificado ou acrescentado.
- ▶ **Object:** Este tipo de dado é usado para incorporar documentos.
- ▶ **Date:** Este tipo de dados é utilizado para armazenar a data ou a hora atual no formato de UNIX. Você pode especificar o seu próprio date_time através da criação do objeto Date e passando o dia, mês e ano para ele.
- ▶ **Object ID:** Este tipo de dados é usado para armazenar os identificadores (_id) dos documentos.
- ▶ **Binary data:** Este tipo de dados é usado para armazenar um dado binário.
- ▶ **Regular expression:** Este tipo de dados é usado para armazenar expressões regulares.

“insert” no MongoDB

- ▶ Para criar uma coleção, basta inserir um documento nela. (esqueça o *CREATE TABLE*)
- ▶ Existem duas operações de inserção no MongoDB:
 - ▶ Inserção de um único documento: **insertOne**. (Recebe como parâmetro um único {} documento.)
 - ▶ Inserção de múltiplos documentos de uma só vez: **insertMany**. (Recebe como parâmetro um *array* [] de documentos.)

```
1 // Inserindo dados via MongoDB Playground
2 // Selecionando o database a ser utilizado
3 use('projetos');
4
5 // Exemplo com insertOne
6 db.filmes.insertOne({"titulo":"Guerra nas Estrelas", "ano":1978, "diretor":"George Lucas"})
7
8 // Exemplo com insertMany
9 db.filmes.insertMany([{"titulo":"O Exorcista", "ano":1973, "diretor":"William Friedkin"},
10 {"titulo":"007 - Sem Tempo Para Morrer", "ano":2020, "diretor":"Cary Joji Fukunaga"}])
```

“select” no MongoDB

▶ O MongoDB possui dois métodos principais para retornar informações de documentos.

- ▶ O método **find()** retorna um ponteiro para todos os documentos que atendem aos critérios especificados.
- ▶ O método **findOne()** retorna um único documento que atende aos critérios especificados. (Caso exista mais de um documento atendendo aos critérios, o método `findOne()` retorna apenas o primeiro.)

```
Currently connected to Prof. Ricardo Leme. Click here to change connection.  
1 // Listando dados via MongoDB Playground  
2 // Selecionando o database a ser utilizado  
3 use('projetos');  
4  
5 // Equivalente ao comando sql:  
6 // select * from filmes;  
7 db.filmes.find()  
8  
9 // select titulo, diretor from filmes where ano = 1978  
10 db.filmes.find({"ano": 1978}, {"_id": 0, "titulo": 1, "diretor": 1})  
11  
12 // select * from filmes where titulo like '%estrela%';  
13 db.getCollection("filmes").find({"titulo": /estrela/})  
14 db.filmes.find({"titulo": /estrela/i}) //i = insensitive case
```

Operadores de Comparação

O MongoDB possui os seguintes operadores de comparação:

Nome	Descrição	Equivalente no SQL
\$eq	Exibe valores que são iguais ao valor especificado	=
\$gt	Exibe valores que são maiores ao especificado	>
\$gte	Exibe valores que são maiores ou iguais ao especificado	>=
\$in	Exibe valores iguais aos especificados no array	in
\$lt	Exibe valores que são menores ao especificado	<
\$lte	Exibe valores que são menores ou iguais ao especificado	<=
\$ne	Exibe valores diferentes ao especificado	<>
\$nin	Exibe valores diferentes aos especificados no array	not in

Operadores Lógicos

O MongoDB possui os seguintes operadores lógicos:

Nome	Descrição	Equivalente no SQL
\$and	Retorna todos os documentos que são verdadeiros em ambas as cláusulas	and
\$or	Retorna todos os documentos que são verdadeiros em pelo menos uma das cláusulas	or
\$not	Retorna todos os documentos diferentes da cláusula solicitada.	not
\$nor	Retorna todos os documentos que são falsos em ambas as cláusulas	nor

Operadores de Elemento

▶ O MongoDB possui os seguintes operadores de elemento

Nome	Descrição	Equivalente no SQL
\$exists	Retorna todos os documentos com o campo especificado	Não há
\$type	Retorna todos os documentos cujo campo é do tipo especificado	Não há

Gerando dados aleatórios para testes

- ▶ <https://next.json-generator.com>

JSON GENERATOR

[Update](#)[Fork](#)[Share](#)[Help](#)

```
1  [
2  {
3    'repeat(50)': {
4      _id: '{{objectId()}}',
5      ativo: '{{bool()}}',
6      totalVendas: '{{floating(452, 99000, 2, "0.00')}}',
7      foto: 'http://placeholder.it/32x32',
8      numeroCalçado: '{{integer(34, 46)}}',
9      sexo: '{{random("masculino", "feminino", "não informado')}}',
10     nome: {
11       primeiro: '{{firstName()}}',
12       sobrenome: '{{surname()}}'
13     },
14     empresa: '{{company().toUpperCase()}}',
15     email(tags) {
16       return
17     },
18     telefone: '+55 {{phone()}}',
19     endereco: '{{street()}}', {{integer(10, 999)}} - {{city()}}', {{state()}}', {{integer(100, 10000)}}',
20     sobre: '{{lorem(1, "paragraphs')}}',
21     inclusao: '{{date(new Date(2019, 1, 1), new Date().getTime())}}',
22     latitude: '{{floating(40, 50)}}',
23     longitude: '{{floating(-75, -45)}}',
24     tags: [
25       {
26         'repeat(2)': '{{lorem(1, "words')}}'
27       }
28     ],
29     dependentes: [
30       {
31         'repeat(1,3)': {
32           id: '{{index()}}',
33           nome: '{{firstName()}} {{surname()}}'
34         }
35       }
36     ],
37     timeFutebol(tags) {
38       const times = ['Vasco', 'Ceará', 'Atlético Paranaense', 'Ituano'];
39       return times[tags.integer(0, times.length - 1)];
40     }
41   }
42 ]
43 }
```

Exemplo do documento gerado

```
{
  "_id": "5f32941ca65e26a861edecdd"
  ativo: true
  totalVendas: "81916.98"
  foto: "http://placeholder.it/32x32"
  numeroCalcado: 42
  sexo: "feminino"
  nome: Object
    primeiro: "Sybil"
    sobrenome: "Bentley"
  fumante: "Sim"
  empresa: "CYTREK"
  email: "sybil.bentley@cytrek.biz"
  telefone: "+55 (944) 474-3487"
  endereco: "Lynch Street, 615 - Kula, North Carolina, 5938"
  sobre: "Irure minim culpa tempor ad veniam sint ad mollit pariatur ad laborum ..."
  inclusao: 1596897192998
  latitude: "45.4764"
  longitude: "-73.5708"
  tags: Array
    0: "laboris"
    1: "adipisicing"
  dependentes: Array
    0: Object
      id: 0
      nome: "Melba Solis"
    1: Object
      id: 1
      nome: "Butler Kirk"
    2: Object
      id: 2
      nome: "Byrd Carey"
  timeFutebol: "Ceará"
```

Exemplos de consultas

```
//Quais clientes calçam 42?
db.clientes.find( {numeroCalcado:42} )

//Quais clientes que tem o primeiro nome Harrell?
db.clientes.find( {"nome.primeiro": /harrel/i} )

//Quais clientes calçam entre 36 e 40?
db.clientes.find( {numeroCalcado: {
  $gte: 36,
  $lte: 40
}})

//Quais clientes são do sexo Feminino e não torcem para o Ituano?
db.clientes.find( { $and: [
  { sexo: { $eq: 'feminino' } },
  { timeFutebol: { $ne: 'Ituano' } }
] } )

//Quais clientes tiveram um total de vendas maior que 85.000 ou são da empresa Tsunamia?
db.clientes.find( { $or: [
  { totalVendas: { $gt: "85000" } },
  { empresa: /tsunamia/i }
] } )
```

Exemplos de consultas

```
//Quais clientes tem o nome ou sobrenome Vega e moram em uma latitude maior que 48?
db.clientes.find( {
  $and : [
    {
      $or : [
        {"nome.primeiro" : "Vega"},
        {"nome.sobrenome" : "Vega"}
      ]
    },
    {
      latitude: {$gte : "48"}
    }
  ]
} )

//Quais documentos possuem o atributo fumante?
db.clientes.find( { fumante: { $exists: true } } )

//Quais documentos o atributo totalVendas é do tipo double?
db.clientes.find( { "totalVendas" : { $type : "double" } } );
```

Ordenação

- ▶ O MongoDB possui o método **sort()** para alterar a ordenação dos documentos.
- ▶ Para trazer ordenado em ordem descendente (Z→A) utilize -1
- ▶ Para trazer ordenado em ordem ascendente (A→Z) utilize 1

```
//Nome, Empresa e email dos clientes que calçam 42 ordenados pelo nome de forma ascendente
db.clientes.find({
  numeroCalcado:42
},
{ _id:0, nome:1, empresa:1, email:1} )
.sort({nome:1})
```

```
//Nome, Empresa e sexo dos clientes que são masculino ou feminino ordenados pela empresa de forma descendente
db.clientes.find({
  sexo:{
    $in: ["masculino","feminino"]
  }
},
{ _id:0, nome:1, empresa:1, sexo:1} )
.sort({empresa:-1})
```

Array

- ▶ O MongoDB permite localizar documentos com uma determinada quantidade de elementos dentro do array.

Exemplo: Localizando apenas os clientes que possuem 3 dependentes

```
db.clientes.find( { "dependentes": { $size: 3 } },  
{nome:1, dependentes:1} )
```

“*update*” no MongoDB

- ▶ O MongoDB possui três métodos para atualização de dados em um documento.
- ▶ Os métodos **updateOne()** e **updateMany()** localizam o documento segundo os critérios especificados e fazem as alterações descritas.
- ▶ Diferença: quantidade de documentos afetada. Enquanto o **updateOne()** afeta somente um documento que atenda aos critérios, o **updateMany()** afeta todos.
- ▶ O método **replaceOne()** localiza um único documento que atenda aos critérios especificados e o substitui por um novo documento. (O atributo `_id` do documento permanece o mesmo)

"update" na prática

```
1 // Update no MongoDB
2 use('projetos');
3
4 //updateOne
5 db.filmes.updateOne(
6   {titulo: "Guerra nas Estrelas"},
7   {
8     $set: {ano: 1977, genero: "Ficção"}
9   }
10 )
11
12 //updateMany
13 db.filmes.updateMany(
14   {diretor: "George Lucas"},
15   {
16     $set: {blockbuster: true}
17   }
18 )
19
20 //replace
21 db.filmes.replaceOne(
22   {ano: 1973},
23   [{genero: "Terror", titulo: "The Exorcist"}]
24
25 )
```


“*delete*” no MongoDB

- ▶ O MongoDB possui dois métodos para a remoção de documentos.
- ▶ Os métodos **deleteOne()** e **deleteMany()** localizam o documento segundo os critérios especificados e o removem da base de dados.
- ▶ Diferença: quantidade de documentos afetada.
- ▶ Enquanto o **deleteOne()** afeta somente um documento que atenda os critérios, o **deleteMany()** afeta todos.

“delete” na prática

```
1 // Delete no MongoDB
2 use('projetos');
3
4 //DeleteOne
5 db.filmes.deleteOne(
6   {titulo: "Guerra nas Estrelas"}
7 )
8
9 //DeleteMany
10 db.filmes.deleteMany(
11   {diretor: "George Lucas"}
12 )
13
14 //Se não for passado nada na seleção, tudo é excluído!
15 db.filmes.deleteMany({})
```

That's all folks!
Let's code.

