

# Treinamento MongoDB

Vicente Calfo

[vicentecalfo@gmail.com](mailto:vicentecalfo@gmail.com)

<https://www.linkedin.com/in/vicentecalfo/>

# Banco de Dados NoSQL

- NoSQL -> **N**ão **S**QL ou não relacional;
- Atualmente o termo NoSQL evoluiu para **Not Only SQL** ou não somente SQL;
- Banco de dados NoSQL tem sido usado para *Big Data* e *Data Science*;

SQL -> *Structured Query Language*

**Big Data** -> conjunto de dados maior e complexo (dados volumosos de diferentes bases de dados e processados com velocidade - 3Vs: Volume, Variedade e Velocidade)

**Data Science** -> A principal função da ciência de dados é transformar dados, estruturados ou não, em conhecimento para uma empresa ou projeto

# Banco de Dados Relacionais

- Bancos mais utilizados desde de 1970;
- Formato de Tabelas (linhas x colunas);
- Uso de chaves primarias (PK) e chaves estrangeiras (FK);
- Limitação de uso:
  - Escala (petabytes);
  - Perda de performance em relacionamento de grande conjunto de dados (*joins*);
  - Dificuldade na modelagem de esquemas complexos.

# Bancos NoSQL Linha do Tempo

- Termo surgido em 1998 por Carlo Strozzi para nomear uma estrutura de banco mais leve não exposta ao SQL e de código aberto;
- Em 2006 o Google publica um artigo: BigTable: A Distributed Storage System for Structred Data" - chamando a atenção para o tema NoSQL;
- Nos últimos anos os banco de dados NoSQL tem aumentado significativamente;

# Bancos NoSQL

- Não usam **SQL**;
- Formas de organização: **NÃO USAM TABELAS** - usam grafos, documentos e colunas visando melhor performance;
- Clusterização: Executar o banco de dados em várias máquinas ao mesmo tempo;
- Usam esquemas flexíveis (formatos diferentes para o mesmo tipo de registro);
- Escalabilidade Horizontal (amigável);

Escalabilidade vertical (scaling up) -> adicionar hardware mais poderosos (ex.: CPU, memórias, disco).



















Escalabilidade horizontal (scaling out) -> adicionar clones do mesmo servidor lado a lado e distribuir as solicitações.

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

# Ranking NoSQL

☐ include secondary database models

57 systems in ranking, March 2023

Rank			DBMS	Database Model	Score		
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022
1.	1.	1.	MongoDB 	Document, Multi-model 	458.78	+6.02	-26.88
2.	2.	2.	Amazon DynamoDB 	Multi-model 	80.77	+1.08	-1.03
3.	3.		Databricks	Multi-model 	60.86	+0.52	
4.	4.	↓ 3.	Microsoft Azure Cosmos DB 	Multi-model 	36.10	-0.40	-4.79
5.	5.	↓ 4.	Couchbase 	Document, Multi-model 	23.36	-1.50	-6.09
6.	6.	↓ 5.	Firebase Realtime Database	Document	18.78	+0.29	-0.80
7.	7.	↓ 6.	CouchDB	Document, Multi-model 	14.46	+0.01	-3.02
8.	8.	↑ 9.	Google Cloud Firestore	Document	11.36	-0.15	+2.21
9.	9.	↓ 7.	MarkLogic	Multi-model 	8.86	+0.02	-1.04
10.	10.	↓ 8.	Realm	Document	8.53	+0.28	-1.30
11.	11.	↑ 13.	Google Cloud Datastore	Document	6.62	-0.27	+1.14
12.	12.	↓ 10.	Aerospike 	Multi-model 	6.54	-0.02	+0.31
13.	13.	↓ 12.	Virtuoso 	Multi-model 	6.39	+0.29	+0.82
14.	14.	↓ 11.	ArangoDB 	Multi-model 	5.04	-0.26	-0.57
15.	15.	15.	OrientDB	Multi-model 	4.30	-0.24	-0.63

# Ranking Geral

410 systems in ranking, March 2023

Rank			DBMS	Database Model	Score		
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022
1.	1.	1.	Oracle +	Relational, Multi-model i	1261.29	+13.77	+9.97
2.	2.	2.	MySQL +	Relational, Multi-model i	1182.79	-12.66	-15.45
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model i	922.01	-7.08	-11.77
4.	4.	4.	PostgreSQL +	Relational, Multi-model i	613.83	-2.67	-3.10
5.	5.	5.	MongoDB +	Document, Multi-model i	458.78	+6.02	-26.88
6.	6.	6.	Redis +	Key-value, Multi-model i	172.45	-1.39	-4.31
7.	7.	7.	IBM Db2	Relational, Multi-model i	142.92	-0.04	-19.22
8.	8.	8.	Elasticsearch	Search engine, Multi-model i	139.07	+0.47	-20.88
9.	9.	↑ 10.	SQLite +	Relational	133.82	+1.15	+1.64
10.	10.	↓ 9.	Microsoft Access	Relational	132.06	+1.03	-3.37
11.	↑ 12.	↑ 14.	Snowflake +	Relational	114.40	-1.26	+28.17
12.	↓ 11.	↓ 11.	Cassandra +	Wide column	113.79	-2.43	-8.35
13.	13.	↓ 12.	MariaDB +	Relational, Multi-model i	96.84	+0.03	-11.47
14.	14.	↓ 13.	Splunk	Search engine	87.97	+0.89	-7.39
15.	15.	↑ 16.	Amazon DynamoDB +	Multi-model i	80.77	+1.08	-1.03

# MongoDB

- Nome derivado da palavra inglesa "Humongous" (imenso);
- Banco de dados NoSQL de código aberto;
- Gratuito;
- Alta performance;
- Sem esquemas;
- Orientado a documentos;
- Criados em 2009 para atender grandes volumes de dados.



# Ecossistema Mongo

- MongoDB Charts (visualização de dados através de dashboards);
- MongoDB Atlas (hospedagem de banco na nuvem);
- MongoDB DataLake (agregação);
- MongoDB Stitch (plataforma servless - autenticação e acesso ao DB);
- MongoDB Compass (ferramenta gráfica para conexão e manipulação de dados).

O **data lake** é um repositório centralizado projetado para armazenar, processar e proteger grandes quantidades de dados estruturados, semiestruturados e não estruturados. Ele pode armazenar dados em seu formato nativo e processar qualquer variedade desses dados, ignorando os limites de tamanho.

# Formato de documento - BSON

- Documentos em **BSON** (representação binária do padrão **JSON**);
- Tipos de dados padrão:
  - booleanos (true/false);
  - strings;
  - números;
  - null (nulo);
  - arrays (listas);
  - objetos/documentos (objects).

# JSON - Tipos

```
{
  "primeiroNome": "John",
  "segundoNome": "Smith",
  "idade": 32,
  "estrangeiro": false,
  "enderaco": {
    "rua": "Adalberto da Silveira",
    "numero": 25,
    "complemento": "apto 307",
    "cidade": "Rio de Janeiro",
    "estado": "RJ",
    "cep": null
  },
  "telefones": [{
    "tipo": "casa",
    "numero": "212 555-1234"
  },
  {
    "tipo": "celular",
    "numero": "646 555-4567"
  }
]
}
```

# MongoDB - Terminologia

Relação de nomes usados nos banco de dados relacionais e no MongoDB

Relacional DB	MongoDB
Banco de Dados	Banco de Dados
Tabela	Coleção
Linha (Registro)	Documento
Coluna	Campo

# MongoDB - Documento

- Cada documento sempre terá o campo: `_id`.

```
{
  "_id": ObjectId("536gfst785hfg896jgh90iju8gdtdf5"),
  "primeiroNome": "John",
  "segundoNome": "Smith",
  "idade": 32,
  "estrangeiro": false,
  "enderaco": {
    "rua": "Adalberto da Silveira",
    "numero": 25,
    "complemento": "apto 307",
    "cidade": "Rio de Janeiro",
    "estado": "RJ",
    "cep": null
  },
  "telefones": [{
    "tipo": "casa",
    "numero": "212 555-1234"
  },
  {
    "tipo": "celular",
    "numero": "646 555-4567"
  }
]
}
```

# MongoDB Atlas

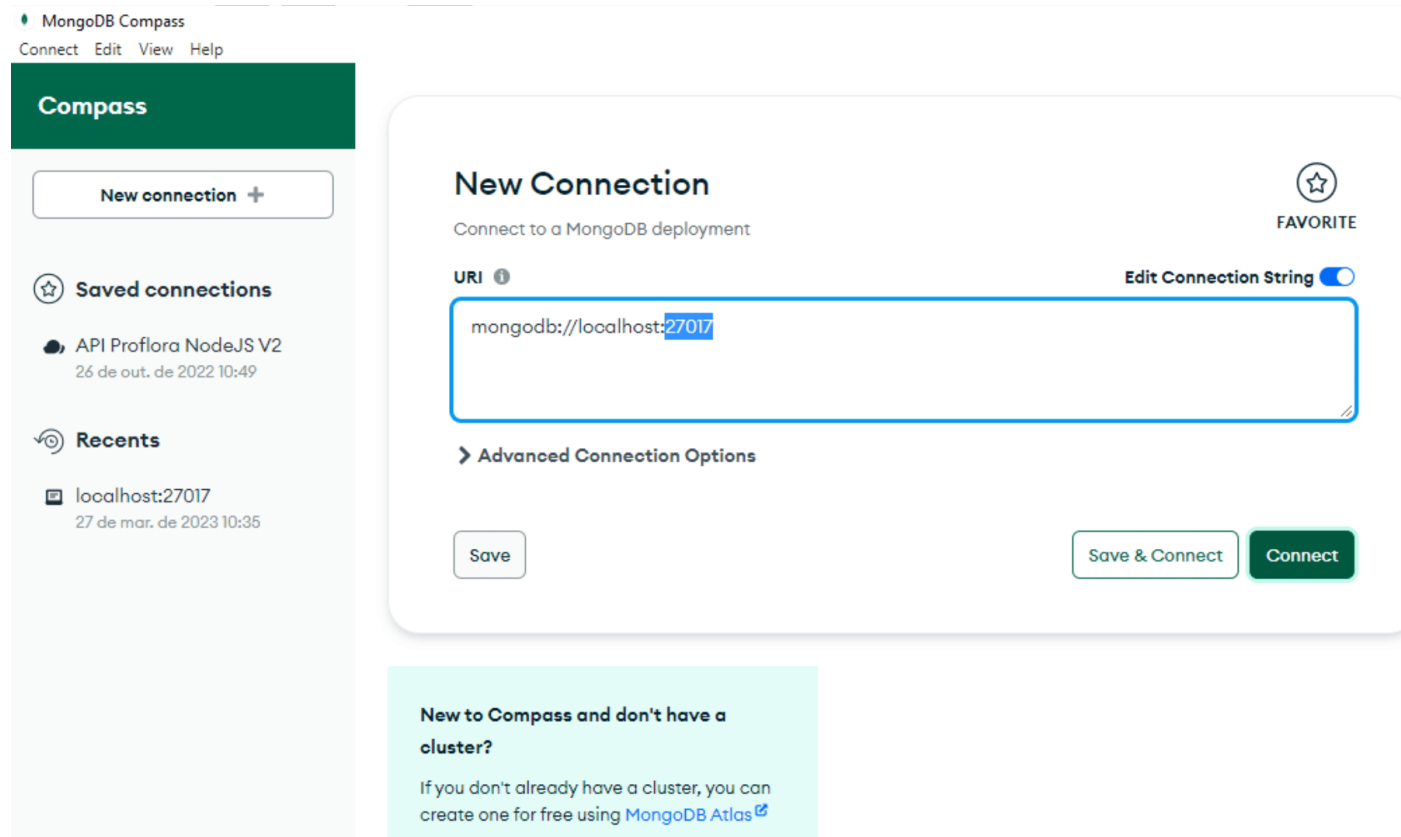
- Criar uma conta no MongoDB Atlas para descobrir o MongoDB
  - <https://www.mongodb.com/atlas/database>;
  - Usar sua conta do Google (para facilitar).

# Download MongoDB

- Link: <https://www.mongodb.com/try/download/community>

# MongoDB

- A porta padrão de instalação do MongoDB é a 27017;






# MongoDB Compass Terminal

MongoDB Compass - localhost:27017

Connect Edit View Help

 **localhost:27017** ...

My Queries

Databases

Performance

{ } My Queries

☰ Databases

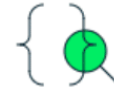


Search

▸ ☰ admin

▸ ☰ config

▸ ☰ local



**No saved queries yet.**

Start saving your aggregations and find queries, you'll see them here.

Not sure where to start? [Visit our Docs](#) →

>\_MONGOSH

> show databases

< admin 40.00 KiB

config 108.00 KiB

local 40.00 KiB

test>

# Comandos Básicos MongoDB

Para exhibir os banco de dados existentes

```
show databases
```

# Comandos Básicos MongoDB

Para escolher um banco de dados

```
use nome-do-database-desejado
```

# Comandos Básicos MongoDB

## Criar um banco de dados

Enquanto você não criar ao menos uma coleção o banco de dados não será criado.

**Obs.:** O mongo considera letras maiúsculas e minúsculas (camelcase) no momento de criação das coleções.

```
use nome-do-banco
```

```
db.createCollection('minha-colecao')
```

# Comandos Básicos MongoDB

Quando você apagar todas as coleções de um banco o banco será apagado também.

## Apagar uma coleção

```
db.nomedacollection.drop()
```

# Comandos Básicos MongoDB

## Apagar um Banco de Dados

```
db.dropDatabase()
```

# Comandos Básicos MongoDB

Listar todas as coleções de um banco de dados.

## Listar as coleções

```
show collections
```

# Comandos Básicos MongoDB

## Inserir Documento

```
db.colecao-desejada.insertOne({"name": "Vicente"})
```



# Comandos Básicos MongoDB

## Inserir Vários Documentos

```
db.colecao-desejada.insertMany([{"name":"Felipe"}, {"name":"André"}])
```

# Comandos Básicos MongoDB

## Buscar todos os Documentos

```
db.colecao-desejada.find()
```

# Comandos Básicos MongoDB

## Buscar um Documento em específico

```
db.colecao-desejada.find({"name":"André"})
```

# Comandos Básicos MongoDB

## Limitando a Busca de Documentos

```
db.colecao-desejada.find().limit(2)
```

# Comandos Básicos MongoDB

## Buscando um Documento por ID

```
db.colecao-desejada.find({_id:ObjectId("6421aa850ed20a9bf32cf818")})
```

# Comandos Básicos MongoDB

## Retornando apenas os campos desejados

O documento retornado só mostrará o campo "name".

O campo "\_id" sempre vai ser mostrado a não ser que explicitamente seja informado o contrário.

```
db.colecao-desejada.find({name:"Felipe"}, {"_id": 0, "name": 1})
```

# Comandos Básicos MongoDB

## Retornando a quantidade de documentos numa coleção

```
db.colecao-desejada.find().count()
```

# Comandos Básicos MongoDB

## Removendo um documento

```
db.colecao-desejada.deleteOne({_id:ObjectId("6421aa850ed20a9bf32cf818")})
```



# Comandos Básicos MongoDB

## Atualizando um documento

```
db.colecao-desejada.updateOne(  
  {_id:ObjectId("6421aa850ed20a9bf32cf818")}  
  {$set:{  
    'idade':48  
  }}  
)
```

# Operadores Atômicos

\$set

Ele é utilizado para especificar uma chave e atualizar a chave.  
Se a chave não exist ela é criada.

```
{ $set : { field : value } }
```

# Operadores Atômicos

\$unset

Para remover uma chave.

```
{ $unset : { field : 1} }
```

# Operadores Atômicos

\$inc

Para incrementar e decrementar valores

```
{ $inc : { field : 1 } }
```

```
{ $inc : { field : -1 } }
```

# Operadores Condicionais

\$gt

Operador de maior (>).

```
db.col.find({"idade" : {$gt : 30}})
```

# Operadores Condicionais

\$gte

Operador de maior igual ( $\geq$ ).

```
db.col.find({"idade" : {$gte : 40}})
```

# Operadores Condicionais

\$lt

Operador de menor (<).

```
db.col.find({"idade" : {$lt : 45}})
```

# Operadores Condicionais

\$lte

Operador de maior igual ( $\leq$ ).

```
db.col.find({"idade" : {$lte : 48}})
```