TRYBE
Modulo III – Back-end

Bloco 23 – NoSQL

1) Introdução NoSQL e MongoDB

Introdução NoSQL

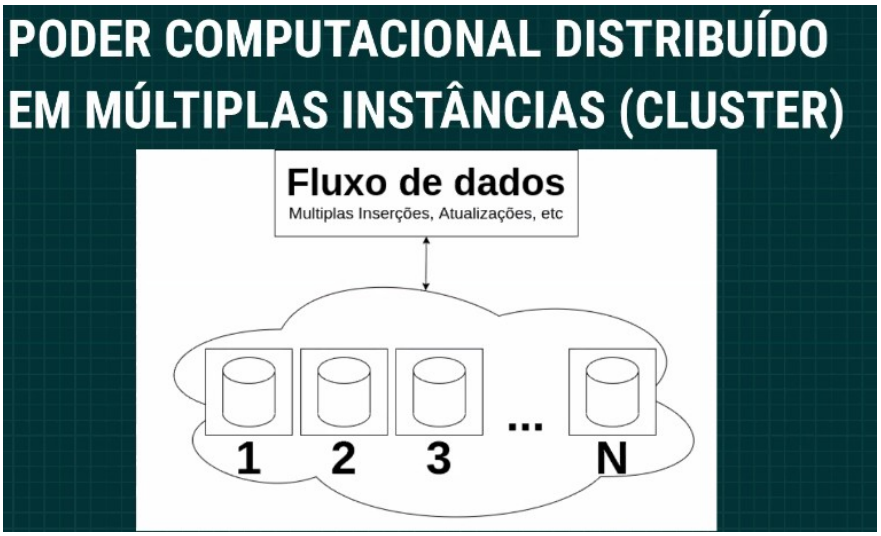
NoSQL = ***"Not Only SQL"***

Principios
Bancos relacionais: ACID (Atomicity, Consistency, Isolation, Durability)
Não relacionais: ***BASE (Base Availability, Soft State and Eventually Consistent)***

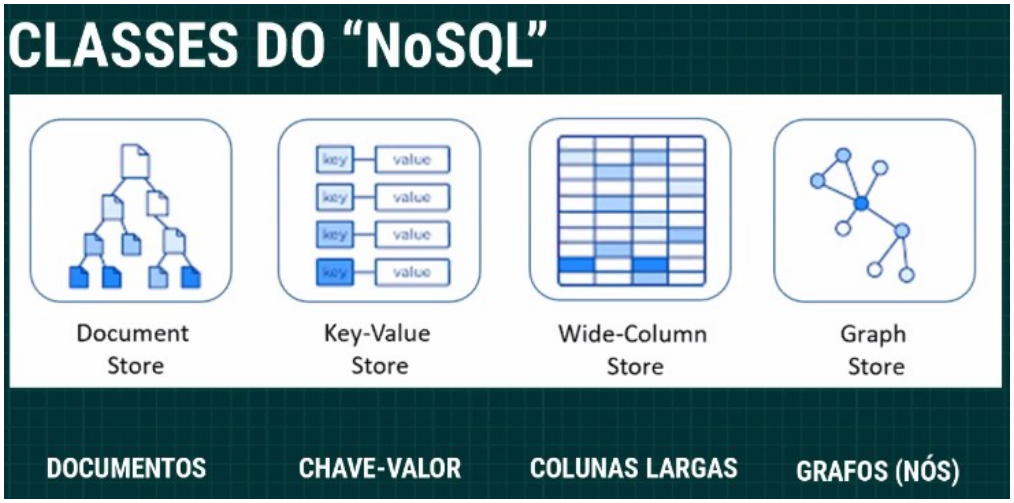
- PRINCÍPIO DA **COMPUTAÇÃO DISTRIBUÍDA;**
- **ESCALABILIDADE;**
- **PERFORMANCE;**
- **ESQUEMA FLEXÍVEL** (SCHEMA-LESS);
- **AGILIDADE**;
- FACILIDADE DE TRABALHAR COM **CLUSTERS**

// escalabilidade vertical e horizontal

Bancos de dados open source VS distribuídos.
Distribuídos operando em computação distribuída, aumentando escalabilidade e performance.
Cluster: capacidade de um conjunto de servidores ou instâncias se conectarem a um BD. Vantagens de Fault Tolerance e Load Balancing.
Instância: coleção de memória e processos que interagem com o BD, que é o conjunto de arquivos físicos que realmente armazenam os dados.



4 Classes (tipos de Bds) do NoSQL



1/ Chave / Valor - Key / Value
Dados mais simples armazenados num esquema de registros compostos por uma chave (identificador do registro) e um valor (todo o conteúdo pertencente àquela chave). In-memory.
Ex: Redis

2/ Família de Colunas - Column Family
Dados armazenados omo um conjunto de três "chaves": linha, coluna e timestamp.
Ex: Cassandra

3/ Documentos – Document
Dados de maior complexidade armazenados em estilo JSON, podendo ter vários níveis e subníveis.

4/ Grafos – Graph
Dados muito complexos compostos por nós (vértices do grafo), relacionamentos (arestas do grafo) e as propriedades ou atributos de ambos.
Ex: neo4j, GraphQL

Instalar MongoDB

Três tipos de instalação
- Standalone (ambientes de desenvolvimento)
- Replica Set (ambientes de produção
- Shard (modo para escalar a escrita de informações no banco)

Instalar (Standalone - MongoDB Community Edition – no Linux)

wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
(opcional) *sudo apt-get install gnupg*
echo "deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list

sudo apt-get update
sudo apt-get install -y mongodb-org

Iniciar MongoDB

sudo service mongod start
verificar com sudo *service mongod status*

```
juliette@juliette-HP-Laptop-15-dw0xxx:~$ sudo service mongod start
juliette@juliette-HP-Laptop-15-dw0xxx:~$ sudo service mongod status
● mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor prese>
     Active: active (running) since Wed 2020-10-21 16:19:04 -03; 19s ago
       Docs: https://docs.mongodb.org/manual
```

parar com *sudo service mongod stop*

saber versão instalada *mongod --version*

Configurar
Para iniciar junto ao sistema *sudo systemctl enable mongod.service* ou cancelar isso *sudo systemctl disable mongod.service*

Desinstalar
sudo service mongod stop
*sudo apt-get purge mongodb-org**
sudo apt-get autoremove
sudo apt-get autoclean
sudo rm -rf /var/log/mongodb
sudo rm -rf /var/lib/mongodb

Se conectar ao MongoDB Shell

Ou seja como usar o Mongo no CLI (Command Line Interface):

mongo

```
juliette@juliette-HP-Laptop-15-dw0xxx:~$ mongo
MongoDB shell version v4.2.10
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName
=mongodb
Implicit session: session { "id" : UUID("1380aed8-4fbc-409d-8651-c3551249e711")
}
MongoDB server version: 4.2.10
Welcome to the MongoDB shell.
```

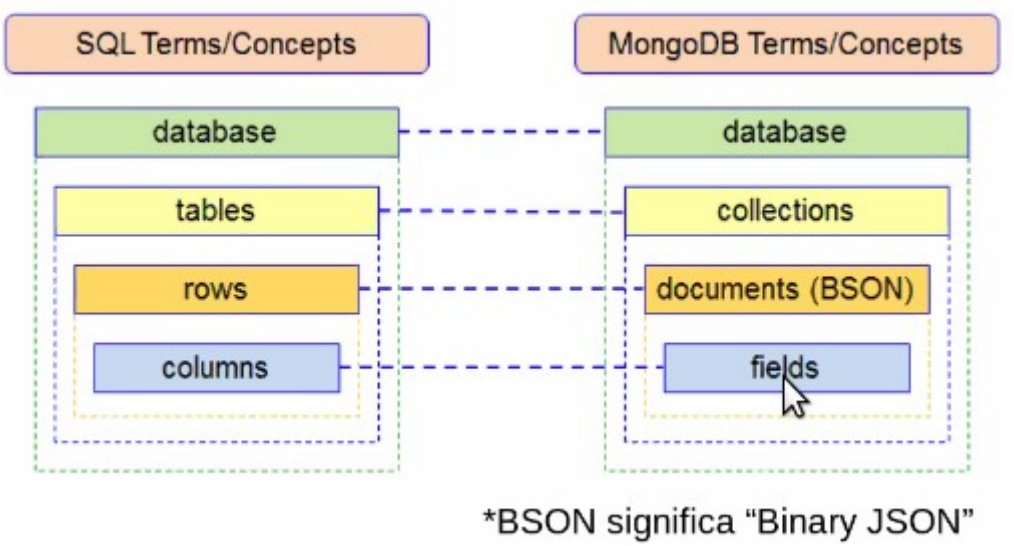
// pode digitar comandos dentro do mongo //

```
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
test    0.000GB
> use test;
switched to db test
> db.inventory.insertOne({ item: "journal", qty 25, category: { name: "movies" } });
2020-10-21T17:22:05.471-0300 E  QUERY    [js] uncaught exception: SyntaxError: missing : after property id :
@(shell):1:46
> db.inventory.insertOne({ item: "journal", qty: 25, category: { name: "movies" } });
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f909876548fd5e42fdd6de6")
}
> db.inventory.insertOne({ item: "journal", qty: 25, categorias: [{id: ObjectId("5f909876548fd5e42fdd6de6")},{ id
: ObjectId("5f909876548fd5e42fdd6de6")},{ id: ObjectId("5f909876548fd5e42fdd6de6")}]  });
```

exit

Configurar outro port do que o padrão 27017: *mongo --port 19000*

Databases, Coleções e Documentos



Normas

Documentos no Mongo
Nomeaçao no Mongo

Databases

use database-name para criar.
Nem precisa criar diretamente, Mongo cria jà ao primeiro insert.
show dbs para ver os BD que jà existem.

Collections

*** Criar collection** via insertOne ou createIndex:
db.minhaColecao.insertOne({ x: 1 })
db.minhaColecao2.createIndex({ y: 1 })
// cria tanto o bd quanto a collection
*** Criar collection com parâmetros** via createCollection:
db.createCollection("minhaColecao4", { collation: { locale: "fr" } });
// usar collMod para modificar params

Documentos

Mais rico e extenso do que linhas de tabelas no SQL.
insert recebe um objeto JSON como parâmetro.
Schema validation de documentos.
BSON = Binary JSON, um JSON com menos restrições de tipos de dados armazenados.

Insert

insertOne()
Faz o insert de um único documento por vez, aceita um objeto.
_id indice único criado automaticamente dentro da coleção.

```
> use sample
switched to db sample
> db
sample
> db.products.insertOne({productName: "Caixa",price: 20})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5e334555af94a4568305829c")
}
> db.products.insertOne({_id: 100,productName:"Caixa",price:20})
{ "acknowledged" : true, "insertedId" : 100 }
```

InsertMany()
Pode inserir milhares de documentos em uma única operação, aceita um array.
//erro se tiver _id duplicado – erro e aborta
//inserir de forma desordenada: erro documentado e continua a executar

```
],{ordered: false})
```

```
db.orders.insertMany([
    {
        "_id": 1,
        "customerId": 10,
        "customerName": "Leandro",
        "total": 20,
        "status":"aprovado"
    },
    {
        "_id": 2,
        "customerId": 33,
        "customerName": "Debora",
        "total": 200,
        "status":"aprovado"
    },
    {
        "_id": 1,
```

Find

Para depois de inserir dados, recuperar dados. Sintaxe *db.collection.find()* .

Parâmetros: query e projection

db.collection.find(query, projection)

Query:
Especifica os *filtros* da seleção usando os query operators, pode ser vazio ({}).

```
db.collection.find( { qty: { $gt: 4 } } )
```

Projection:
Especifica quais *campos* retornar, pode ser vazio para retornar todos, _id retornado por padrão.
{ "campo1": <valor>, "campo2": <valor> ... }
Onde valor pode ser 0 (false) ou 1 (true) ou projection operators.

```
db.voos.findOne(
  { "passageiros.pagos": { $gt: 7000 } },
  { vooId: 1, mes: 1, ano: 1, _id: 0 }
);
```

// retorna os campos de vooId, mês, ano, sem o _id que caso contrário é retornado por padrão

Usos do find
Para retornar todo - *db.collection.find()*
Para retornar o primeiro objeto e estudar assim a estrutura – *db.collection.findOne()*
Query por igualdade - *db.bios.find({ _id: 5 })*
Retornar somento campo - *db.bios.find({}, { name: 1 })*

Comandos úteis diversos

Gerenciamento do cursor
it para passar mais 20 documentos
db.collection.count() para countar os documentos

Legibilidade
.pretty();

```
> db.inventory.find({status: "D"}).pretty();
```

Pular documentos
.skip(número)

```
db.bios.find().limit(10).skip(5)
```

---

2) Filter operators

Operadores de Comparação

find(), count(), update() e distinct() aceitam operadores de comparação.

Operadores | Ordem de comparação para comparar objetos BSON, do menor ao maior

Name	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

1. MinKey (internal type)
2. Null
3. Numbers (ints, longs, doubles, decimals)
4. Symbol, String
5. Object
6. Array
7. BinData
8. ObjectId
9. Boolean
10. Date
11. Timestamp
12. Regular Expression
13. MaxKey (internal type)

Sintaxe

{ <campo>: { <operador>: <valor> } }
Prefixo *$*

```
db.collection.find( { qty: { $gt: 4 } } )
```

Operadores Lógicos

$not
{ campo: { $not: { <operador ou expressão> } } }
Seleciona os documentos que não correspondam ao < operador ou expressão >

$or
{ $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }
Seleciona os documentos que satisfaçam ao menos uma das expressões

$nor
{ $nor: [{ <expressão1> }, { <expressão2> }, ... { <expressãoN> }] }
Seleciona os documentos em que todas essas expressões falhem

$and
{ $and: [{ <expressão1> }, { <expressão2> } , ... , { <expressãoN> }] }
Seleciona os documentos que satisfaçam todas as expressões no array

Operador $exists

{ campo: { $exists: <boolean> } }
Quando true, retorna documentos que contêm o campo, quando false, que não.

```
db.inventory.find({ qty: { $exists: true, $nin: [ 5, 15 ] } })
```

// retorna documentos da coleção inventory onde campo qty existe e com valor diferente de 5 e 15

Método sort()

db.colecao.find().sort({ "campo": "1 ou -1"})
Para ordenar, 1 sendo ascendente e -1 descendente, usável apenas depois de resultado de busca.

```
db.example.find().sort({ "price": 1 }).pretty()
```

// ordena documentos da coleção example por price do menor ao maior

Remover documentos

db.colecao.deleteOne(query)
Para remover o *primeiro* campo que corresponda com a query.

db.colecao.deleteMany(query)
Para remover *todos* os campos que correspondam com a query.

Remover todos - *db.inventory.deleteMany({})*

Dicas diversas (exercicios – projeto dataflight)

“” para entrar no objeto: *db.superheroes.find({ "**aspects.height**": { $lt: 180 } }).pretty();*
Contar os documentos removidos: automatico com o comando delete.