

# noSQL

Mestrado Integrado em Engenharia Informática

<https://hpeixoto.me/class/nosql>

Hugo Peixoto

[hpeixoto@di.uminho.pt](mailto:hpeixoto@di.uminho.pt)

2020/2021

# noSQL

PL05 - Introdução ao Modelo Documental

# Sumário

---



MongoDB - Arquitetura | Base de dados | Coleções | Modelo físico



MongoShell



MongoDB - Tutorial



FE04 - Ficha de exercícios 04





## Humongous DB

- Open-source
- Modelo Documental
- JSON alike
- Documentos embebidos



# MongoDB

## Principais Características:

- **Elevada Performance:** Indexação é possível em qualquer campo dos documentos;
- **Elevada Disponibilidade e Replicação:** O processo de replicação, de forma simplificada, é a capacidade de distribuir os dados por múltiplos servidores e manter tudo sincronizado. Existência de nó primário e nós secundários;
- **Ausência de Esquema:** Podem existir diferentes documentos dentro de uma base de dados, nem todos com a mesma estrutura;
- **Escalabilidade:** Possibilidade de adicionar vários nós para conter um enorme número de documentos.

# MongoDB - Arquitetura

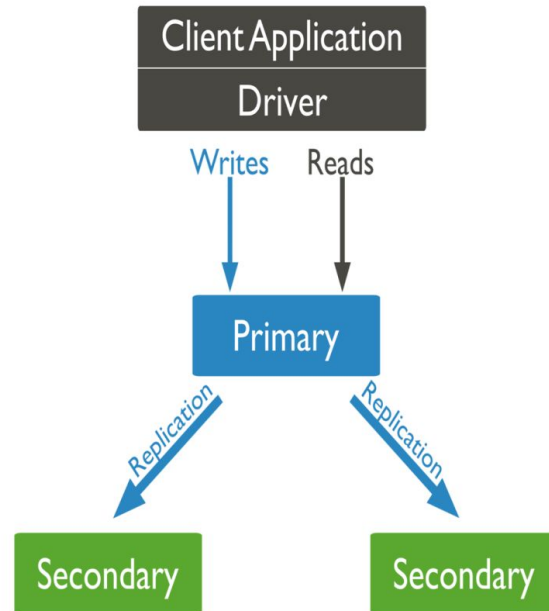
## Replicação

- Tolerância a falhas e redundância;
- Elevada disponibilidade;
- Invisível para a aplicação cliente;

Existência de 1 nó primário e 1..n nós secundários:

Nó primário: operações de leitura e escrita;

Nós secundários: replicação assíncrona;

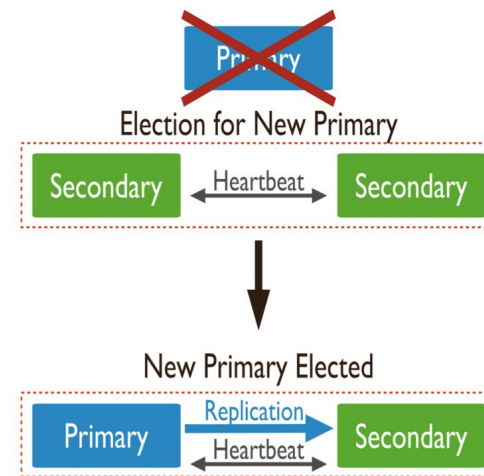




## Replicação

### Em caso de falha:

- Se o primário deixar de comunicar por um determinado período de tempo;
- Eleição do novo primário;
- Garantia de existência de heartbeat;
- Durante a eleição de um novo nó primário não podem ocorrer transações.







# MongoDB - Arquitetura

## Modelo:

- Cada documento tem no máximo 16mb;
- O formato dos documentos é BSON;
- Cada documento é armazenado numa coleção;
- Coleções:
  - Índices em comum;
  - São semelhantes a tabelas nas bases de dados relacionais;
  - Os documentos contidos numa coleção podem não ter estrutura uniforme.

# MongoDB - Arquitetura

## JSON - “JavaScript Object Notation”

Fácil para os humanos lerem e escreverem;

Fácil para as máquinas processarem e gerirem;

Estrutura:

- Conjunto nome/valor;
- Lista ordenada de valores (array);

## BSON - “Binary JSON”

Serialização binária de documentos JSON;

Principal objetivo é a eficiência;



# MongoDB - Arquitetura

## Exemplo:

```
{
  "_id" :      "37010"
  "city" :     "ADAMS",
  "pop" :      2660,
  "state" :    "TN",
  "councilman" : {
    name: "John Smith"
    address: "13 Scenic Way"
  }
}
```

# MongoDB - Arquitetura

## Campo `_id`:

Por defeito, todos os documentos contêm um campo `_id`:

- Serve como chave primária para o documento dentro da coleção;
- É único e não pode ser mudado ou transformado;
- O *data type* por defeito é ***ObjectId***:
  - Pequeno,
  - Único,
  - Fácil de gerar,
  - Fácil de ordenar (similar a ordenar por data de criação)



# MongoDB - Arquitetura

Tipo	Descrição
Double	used to store floating point values.
String	BSON strings are UTF-8. It is most commonly used datatype.
Object	used for embedded documents.
Array	used to store multiples of values and data types.
Binary data	used to store the binary data.
Undefined	used to store the undefined values.
Object id	used to store the document's id.
Boolean	used to store a boolean value.
Date	used to store the current date or time in UNIX time format.
Null	used to store a Null value.
Regular Expression	used to store regular expression.
JavaScript	used to store the javascript data without a scope.
Symbol	It is similar to the string data type, and not supported by a shell.
JavaScript (with scope)	It stores javascript data with a scope.
Timestamp	used to store a timestamp.
Min key	Min key compares the value of the lowest BSON element.
Max key	Max key compares the value against the highest BSON element.



# MongoDB - Arquitetura

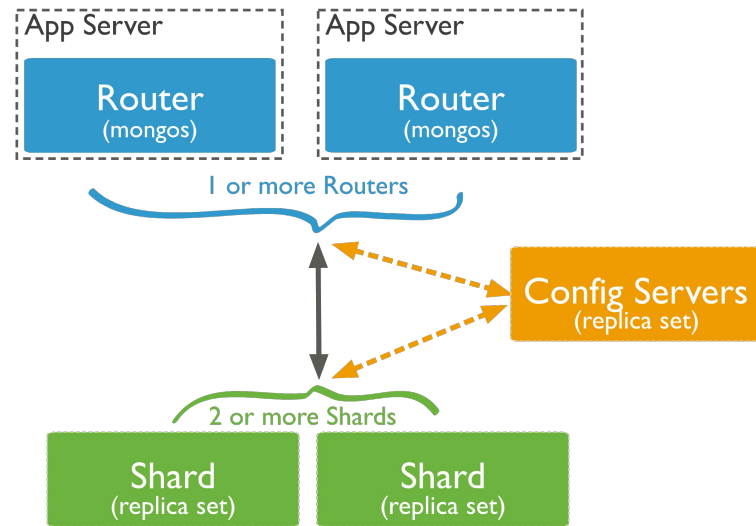
## MongoDB (Documental) Vs Bases de Dados Relacionais

RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Row	→	Document
Index	→	Index
Join	→	Embedded Document
Foreign Key	→	Reference

# MongoDB - Arquitetura

## Sharding

Método para distribuição dos dados por múltiplos servidores. MongoDB usa esta técnica para permitir o armazenamento de datasets enormes e garantir a execução de operações complexas sobre os mesmos.





# MongoDB - Arquitetura

## Sharding

**shard:** Cada shard contém um subset da totalidade dos dados. Podem ser definidos como replica sets.

**mongos:** Os mongos atuam como router para as queries a executar. Proporcionam uma interface entre as aplicações cliente e o cluster.

**config servers:** Cada servidor de configuração contém metadados e as configurações do servidor.





# Mongoshell

## Visualizar containers a correr:

```
# docker ps -a
```

## Executar shell do container mongodb:

```
# docker exec -it <container-id> bash
```

```
# mongo
```

```
root@ca364392ff28:/# mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c65a26eb-4b64-47da-9df1-b3c800098b36") }
MongoDB server version: 4.4.1
---
The server generated these startup warnings when booting:
  2020-11-15T21:22:04.482+00:00: Access control is not enabled for the database. Read and write access to data and configuration
  2020-11-15T21:22:04.483+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
  2020-11-15T21:22:04.483+00:00: /sys/kernel/mm/transparent_hugepage/defrag is 'always'. We suggest setting it to 'never'
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> █
```



# MongoDB - Tutorial

- Criar Base de dados:

```
> use nosql
```

Se o nome da base de dados usado já existir, será usada a base de dados existente!

- Verificar bd atual:

```
> db
```

- Listar todas as bases de dados:

```
> show dbs
```

A base de dados criada não aparece, pois temos de guardar o primeiro documento!

- Criar coleção e armazenar primeiro documento:

```
> db.pl.insert({  
    nome: "Hugo Peixoto",  
    age: "35"  
})
```

// Verificar a criação da BD:

```
> show dbs
```

```
> show dbs  
admin    0.000GB  
config   0.000GB  
local    0.000GB  
nosql    0.000GB
```

# MongoDB - Tutorial

## - Apagar base de dados criada:

```
// visualizar dbs existentes
```

```
> show dbs
```

```
// entrar na bd desejada
```

```
> use nosql
```

```
//apagar a Base de dados
```

```
> db.dropDatabase()
```

```
// Verificar que a db foi eliminada
```

```
> show dbs
```

## - Criar coleção:

### 1. Criar e armazenar primeiro documento:

```
// selecionar a base de dados
```

```
> use music
```

```
// inserir documento na coleção desejada (songs)
```

```
> db.songs.insert({  
    name: "Alive",  
    band: "Pearl Jam",  
    year: 1990  
})
```

Listar todos os documentos da coleção

```
> db.songs.find()
```



# MongoDB - Tutorial

- Criar coleção:

2. Criar sem gerar primeiro documento:

```
> use music
```

```
> db.createCollection("albums")
```

- Apagar Coleção:

```
// visualizar dbs existentes
```

```
> show dbs
```

```
// entrar na bd desejada
```

```
> use music
```

```
// visualizar as coleções existentes
```

```
> show collections
```

```
// apagar a coleção desejada
```

```
> db.albums.drop()
```

```
// visualizar as coleções existentes
```

```
> show collections
```



# MongoDB - Tutorial

## - Inserir documento:

Sintaxe:

```
> db.collection_name.insert()
```

```
// criar do documento
```

```
> db.songs.insert({  
  name: "Jeremy",  
  band: "Pearl Jam",  
  year: 1990,  
  crew: [{vocals: "Eddie Vedder", drums: "Dave Abbruzzese"}]  
})
```

```
// verificar a criação do documento
```

```
db.collection_name.find()
```



# MongoDB - Tutorial

- Inserir vários documentos:

```
var musicas= [{  
    "name" : "Riders on the Storm",  
    "band" : "The Doors",  
    "year": 1971  
},  
{  
    "name" : "Roadhouse Blues",  
    "band" : "The Doors",  
    "year": 1969  
}  
];
```

//inserir documentos

```
db.songs.insert(musicas);
```

// validar criação

```
db.songs.find()
```

// validar em json

```
db.songs.find().pretty()
```



# MongoDB - Tutorial

- Procurar documentos:

```
db.songs.find({name: "Alive"}).pretty()
```

- Maior que:

```
db.songs.find({year: {$gt:1980}}).pretty()
```

- Maior ou igual a:

```
db.songs.find({year: {$gte:1980}}).pretty()
```

- Menor que:

```
db.songs.find({year: {$lt:1980}}).pretty()
```

- Menor ou igual a:

```
db.songs.find({year: {$lte:1980}}).pretty()
```

- Não igual a:

```
db.songs.find({band: {$ne: "Pearl Jam"}}).pretty()
```



# MongoDB - Tutorial

- Atualizar documento:

```
db.songs.update({name: "Alive"}, {$set: {year: 1995}})
```

```
db.songs.update({band: "Pearl Jam"}, {$set: {band: "PJ"}})
```

- Atualizar múltiplos documentos:

```
db.songs.update({band: "Pearl Jam"}, {$set: {band: "PJ"}}, {multi:true})
```

- Criar no caso de não existir:

```
db.songs.update({band: "Nirvana"}, {$set: {foundation: 1990}},  
{upsert:true})
```

```
db.songs.update({name: "The man who sold the world"}, {$set: {band:  
"Nirvana", name: "The man who sold the world", year: 1991 }},  
{upsert:true})
```





# MongoDB - Tutorial

---

- Remover documento:

```
db.songs.remove({band: "Nirvana"})
```

Ao contrario do update, o remove apaga todas as entradas!!

- Apagar apenas 1 documento (primeiro que encontrar):

```
db.songs.remove({band: "Nirvana"}, 1)
```



# MongoDB - Tutorial

## Projections:

- Listar apenas o nome da banda, excluindo o \_id:

```
db.songs.find({}, {_id: 0, band: 1}).pretty();
```

- Listar o nome da banda (aparece o id):

```
db.songs.find({}, {band: 1, year: 1}).pretty();
```

## Limites:

- Listar apenas os primeiros 2 resultados:

```
db.songs.find({}).limit(2).pretty();
```

- Saltar os primeiros 2 resultados e listar os 2 seguintes:

```
db.songs.find({}).skip(2).limit(2).pretty();
```



# MongoDB - Tutorial

---

## Ordenar:

- Listar musicas pela data de lançamento descendente:

```
db.songs.find({}).sort({year: -1}).pretty();
```

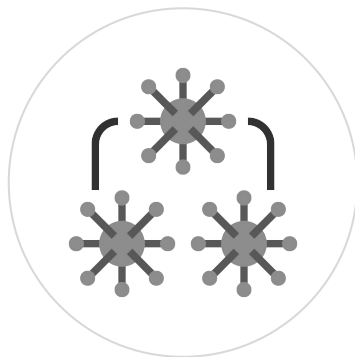
- Listar músicas pela data de lançamento ascendente:

```
db.songs.find({}).sort({year: 1}).pretty();
```

# FE04 - Introdução ao MongoDB

---

# FE04



# noSQL

Mestrado Integrado em Engenharia Informática

<https://hpeixoto.me/class/nosql>

Hugo Peixoto

[hpeixoto@di.uminho.pt](mailto:hpeixoto@di.uminho.pt)

2020/2021