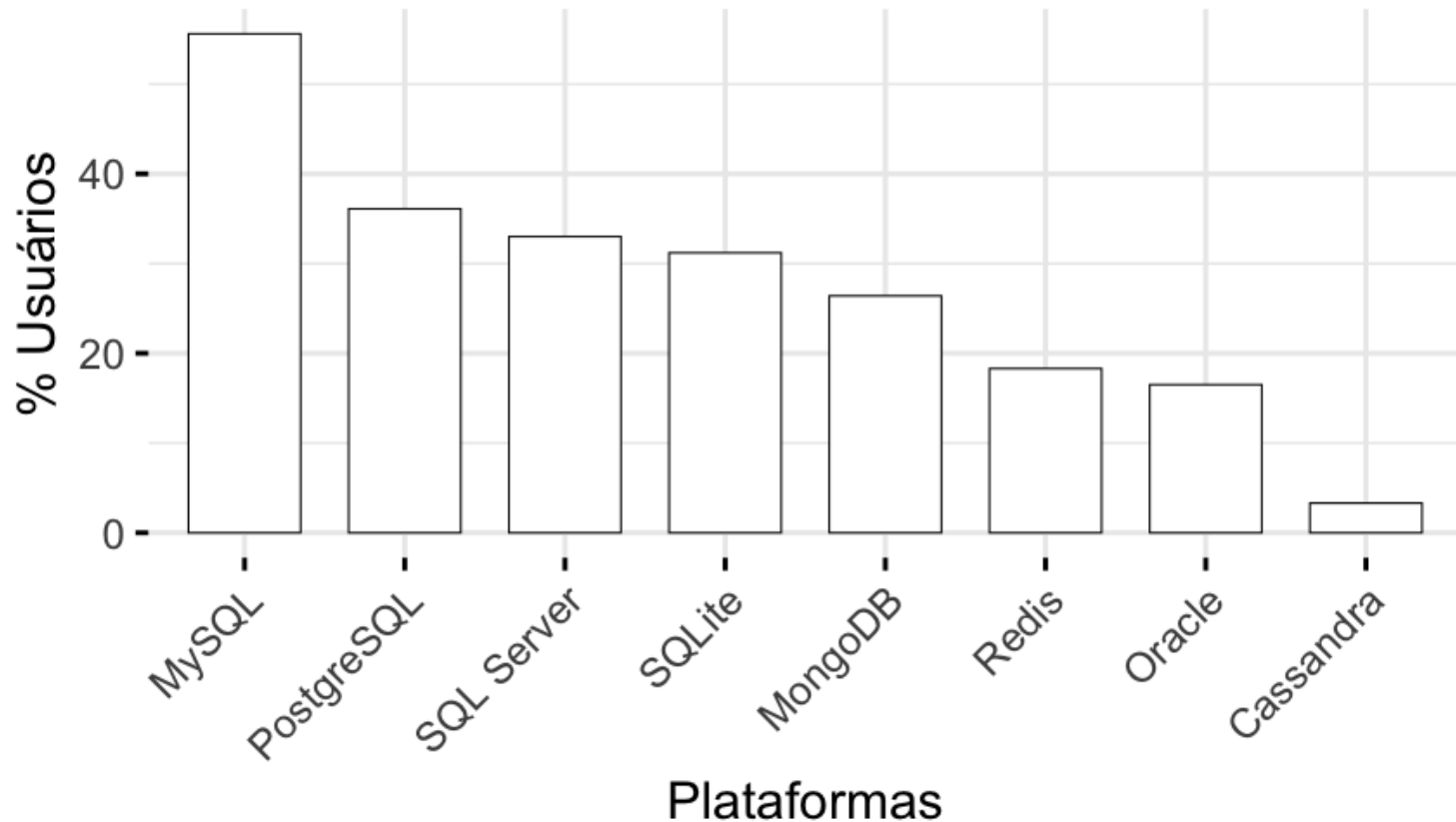# MongoDB

Uma Introdução

Popularidade no StackOverflow em 2020

Introdução ao MongoDB

- Código aberto;
- Gratuito;
- Alta performance;
- Sem esquemas;
- Orientado a documentos;
- Implementado em C++;

Orientado a Documentos

- Orientado a documentos JSON;
- Lembrete:
  - documentos JSON possuem estrutura hierárquica;
  - podem ser facilmente utilizados pelo R ou outras ferramentas para realização de analítica;
  - suportam hierarquias complexas e mantém índices;

O que é JSON

- JavaScript Object Notation
- Sintaxe para armazenamento e troca de dados
- É texto puro
- Amplamente utilizada em ambientes web servidor/cliente
- Regras da sintaxe:
    - Dados são definidos em pares do tipo chave/valor
    - Dados são separados por vírgulas
    - Chaves são utilizadas para armazenar objetos
    - Colchetes são utilizados para armazenar vetores

```
# dados em JSON
"primeiroNome":"João"

# objeto JSON
{"primeiroNome":"João", "ultimoNome":"Vieira"}

# vetor JSON
"alunos":[
    {"primeiroNome":"João", "ultimoNome":"Vieira"},
    {"primeiroNome":"Joana", "ultimoNome":"Silva"},
    {"primeiroNome":"Maria", "ultimoNome":"Gomes"}
]
```

Tipos de Dados em JSON

- *string*

```
{"nome":"Fernanda"}
```

- número

```
{"idade":27}
```

- objeto (JSON)

```
{
"funcionario": {"nome":"Fernanda",
                "idade":27,
                "cidade":"Campinas"}
}
```

Tipos de Dados em JSON

- *array*

```
{
"funcionarios": ["João",
                 "Fernanda",
                 "Maria"]
}
```

- *booleano*

```
{"ferias":true}
```

- nulo

```
{"nomeDoMeio":null}
```

Data Frames vs. JSON

```
## mtcars[1:3, 1:2]
```

	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4

```
library(jsonlite)
toJSON(mtcars[1:3, 1:2])
[{"mpg":21,   "cyl":6, "_row":"Mazda RX4"},
 {"mpg":21,   "cyl":6, "_row":"Mazda RX4 Wag"},
 {"mpg":22.8, "cyl":4, "_row":"Datsun 710"}]
```

Efeitos Práticos

- Cada documento é autossuficiente;
- Cada documento possui todas as informações de que possa precisar;
- Lembrete:
    - em SQL, evitam-se repetições e combinam-se tabelas via chaves;
- Evitam-se JOINs;
- Desenha-se a base de dados de forma que as *queries* busquem apenas uma chave e retornem todas as informações necessárias;
- Preço: espaço em disco;

Utilização de MongoDB

- Foco em *big data*;
- Escalonamento horizontal (*sharding*) - desempenho;
- Escalonamento vertical (*replica sets*) - multicore;
- Se os dados não possuem formato fixo, MongoDB é uma boa opção;
- (J/B)SON não possuem esquemas;
- Opção natural para sistemas web. Exemplo: Comércio eletrônico - detalhes de produtos;

Quando não utilizar MongoDB?

- Quando relacionamentos entre múltiplas entidades for essencial;
- Quando existirem múltiplas chaves externas e JOINs;
- Expectativas em MongoDB:
    - Documentos autossuficientes;
    - Mínimo de chaves;
    - etc;

Disponibilidade

- MongoDB Atlas - Database as a Service (AWS, GCP, Azure);
- Linux;
- MacOS;
- RHEL;
- Windows;

Uso do pacote `mongolite`

- Sempre monta-se uma conexão via `mongo()`;
- O arquivo pode ser remoto ou local;
- Contagem de registros via `con$count()`;
- Remoção de coleção via `con$drop()`;
- Inserção de coleção via `con$insert()`;

Enviando dados para o servidor MongoDB

```r
library(mongolite)
#url:"mongodb://usuario:senha@servidor:porta/base?retryWrites=false"
url="mongodb://readwrite:test@mongo.opencpu.org:43942/jeroen_test?ret
con <- mongo("mtcars", url = url)
if(con$count() > 0) con$drop()
con$insert(mtcars)
```

```
## List of 5
##  $ nInserted  : num 32
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

```r
stopifnot(con$count() == nrow(mtcars))
```

Uso do pacote `mongolite`

- Seleção de dados presentes no banco de dados fia `con$find()`;
- No pacote `mongolite`, remover o objeto de conexão, `con`, já desconecta a sua sessão do banco de dados;
- Mas também existe o método `disconnect()` para realizar a desconexão;

```
mydata <- con$find()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

```
stopifnot(all.equal(mydata, mtcars))
con$drop()
rm(con)
```

Coleções Maiores e Seleções mais Complexas

- Inserção de um conjunto de dados mais volumoso;

```r
library(nycflights13)
## subconjunto pq o servidor eh publico
flights = flights[sample(nrow(flights), 10000), ]
m <- mongo(collection = "nycflights", url=url)
m$drop()
m$insert(flights)
```

```
## List of 5
##  $ nInserted  : num 10000
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

Seleções mais Complexas

- find() é análogo ao SELECT * FROM tabela;
- É possível ordenar os dados já na seleção;
- As chamadas devem acontecer usando formato JSON;

```
m$count('{"month":1, "day":1}')
```

```
## [1] 19
```

```
jan1 <- m$find('{"month":1,"day":1}', sort='{"distance":-1}')
jan1[1:5, 1:7] %>% knitr::kable()
```

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time
2013	1	1	1720	1725	-5	2121
2013	1	1	1937	1905	32	2250
2013	1	1	628	630	-2	1016
2013	1	1	1059	1053	6	1342
2013	1	1	1730	1730	0	2126

Ordenação em Grandes Bases

- Bases volumosas exigem a existência de um índice para permitir a ordenação;
- O índice pode ser adicionado via `index()`;
- `find()` aceita o argumento `sort=`.

```r
# criacao de indice eh essencial para grandes volumes de dados
m$index(add = "distance")
```

```
##   v key._id key.distance       name                        ns
## 1 2       1           NA       _id_ jeroen_test.nycflights
## 2 2      NA            1 distance_1 jeroen_test.nycflights
```

```r
allflights <- m$find(sort='{"distance":-1}')
```

Seleção de Colunas Específicas

- Utiliza-se `find()`;
- Adiciona-se o argumento `fields=`, que recebe a lista (em JSON) das variáveis de interesse;
- Ao especificar colunas de interesse, o MongoDB retorna uma coluna adicional, `_id`, que corresponde a um identificador interno do banco de dados;

```
# Select columns
jan1 <- m$find('{"month":1,"day":1}',
               fields = '{"_id":0, "distance":1, "carrier":1}')
head(jan1, 5) %>% knitr::kable()
```

carrier	distance
EV	266
UA	2227
DL	1096
UA	997
US	214

Operadores em MongoDB

- Comparação:

  - `$eq`: equivalência
  - `$gt` (`$gte`): maior que (maior ou igual)
  - `$lt` (`$lte`): menor que (menor ou igual)
  - `$ne`: não-equivalentes

- Matemáticos

  - `$abs`: valor absoluto
  - `$ceil`: menor inteiro maior ou igual
  - `$floor`: maior inteiro menor ou igual
  - `$ln`: logaritmo natural
  - `$sqrt`: raiz quadrada

https://docs.mongodb.com/manual/reference/operator/aggregation/

Identificação de Ocorrências Únicas

- O método `distinct()` retorna o que são valores únicos de um certo campo;
- Ele pode receber condições para serem avaliadas durante a execução;

```
# List unique values
m$distinct("carrier")
```

```
##  [1] "WN" "AA" "US" "9E" "EV" "MQ" "DL" "UA" "B6" "VX" "FL" "AS" "HA" "YV"
## [16] "F9"
```

```
m$distinct("carrier", '{"distance":{"$gt":3000}}')
```

```
## [1] "UA" "HA"
```
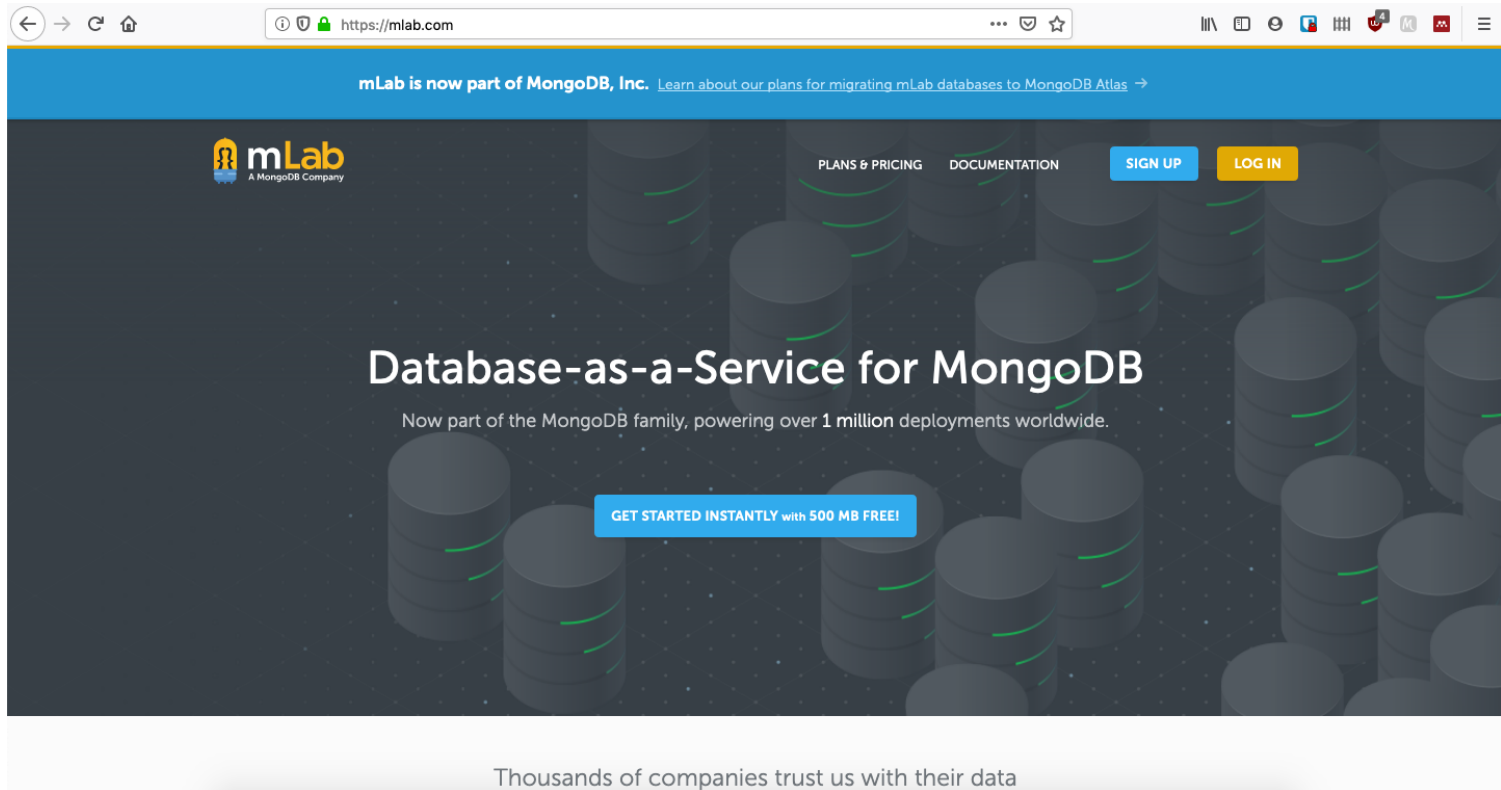
Tabulação de Dados em MongoDB

- O método `aggregate()` permite a tabulação de dados;
- Ele exige, em seu primeiro argumento, a apresentação de um *pipeline* para cálculos desejados;

```
out = m$aggregate('[{"$group": {"_id":"$carrier",
                    "count": {"$sum":1},
                    "average":{"$avg":"$distance"}}
          }]')
m$drop()
out[1:4,] %>% knitr::kable()
```

_id	count	average
F9	14	1620.0000
OO	2	453.5000
AS	10	2402.0000
9E	536	551.2071

Criando sua instância para testes: mongodb.com

Criando sua instância para testes: CREATE ACCOUNT

Criando sua instância para testes: FREE

Criando sua instância para testes: CREATE

Criando sua instância para testes: Espere até criação

Criando sua instância para testes: Cluster pronto

Criando sua instância para testes: Acesso

Criando sua instância para testes: Conexão

Criando sua instância para testes: Application

Criando sua instância para testes: Info para conexão

Criando sua instância para testes: ADD

Criando sua instância para testes: Banco e coleção

Criando sua instância para testes: Estrutura

Acesso via R

```r
library(mongolite)
url = paste0(
  "mongodb+srv://",
  "beniltonBD:senha123dificil@",
  "cluster0.agyxj.mongodb.net/",
  "?retryWrites=true&w=majority"
)
url
```

```
## [1] "mongodb+srv://beniltonBD:senha123dificil@cluster0.agyxj.mongodb.net/?
```

```r
myconn = mongo(collection="meuteste", db="testemongodb", url=url)
library(ggplot2)
myconn$insert(diamonds)
```

```
## List of 5
##  $ nInserted  : num 53940
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

Criando sua instância para testes: Dados

Criando sua instância para testes: Manipulação Extra

Criação do material

- Benilton Carvalho
- Guilherme Ludwig
- Tatiana Benaglia