

BIG DATA #2

AGENDA

Dia 1 - Visão Geral e Hadoop

Dia 2 - Sparklyr, Parquet, MongoDB e outros

SPARK

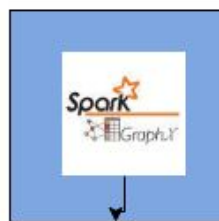
Desenvolvido em 2009 no AMPLab da Universidade de Berkeley e disponibilizado em código aberto pela fundação Apache em 2010

- Framework
- Processamento em memória
- Escrito em Scala



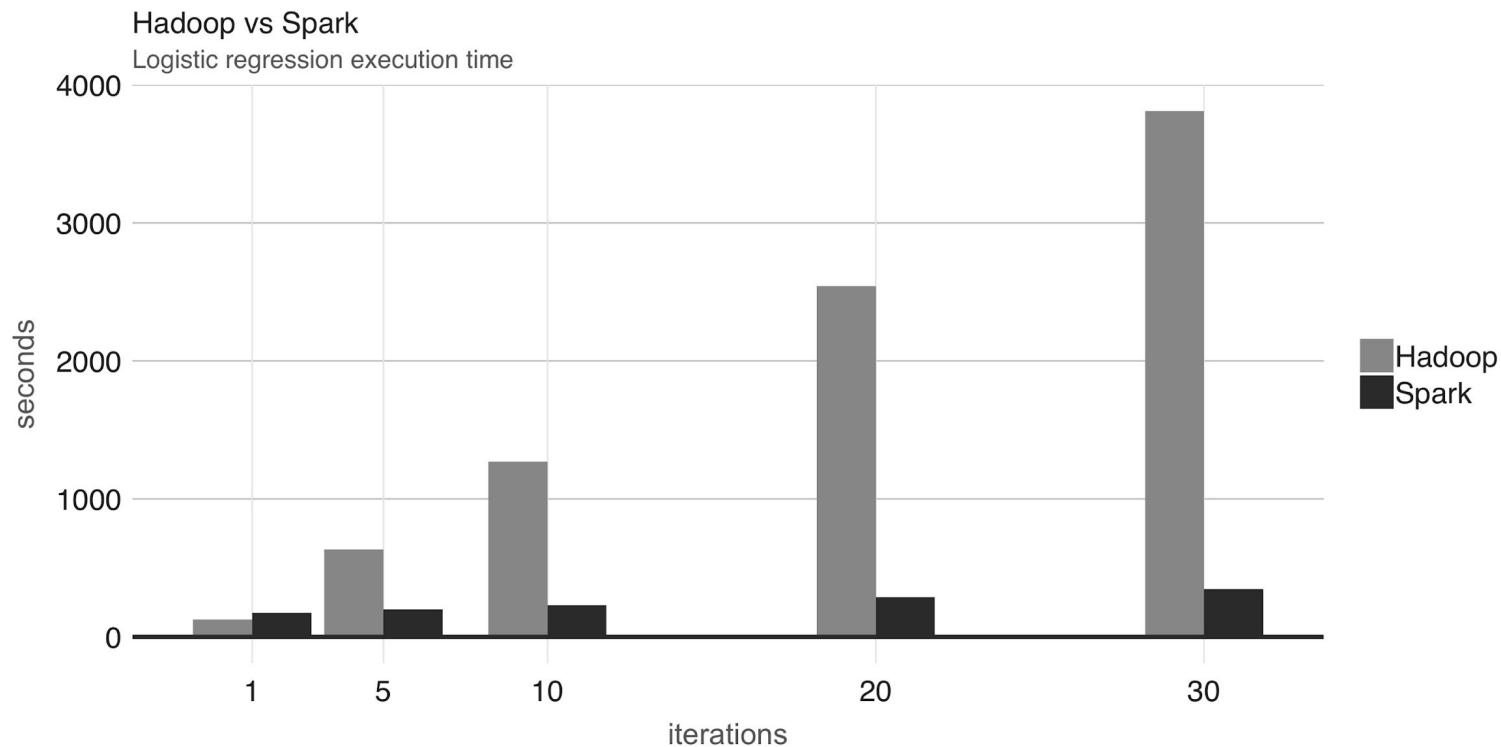


Environment



Source

SPARK



SPARK

	Hadoop Record	Spark Record
Data Size	102.5 TB	100 TB
Elapsed Time	72 mins	23 mins
Nodes	2100	206
Cores	50400	6592
Disk	3150 GB/s	618 GB/s
Network	10Gbps	10Gbps
Sort rate	1.42 TB/min	4.27 TB/min
Sort rate / node	0.67 GB/min	20.7 GB/min

SPARKLYR

<https://spark.rstudio.com/>

- Connect to Spark from R. The sparklyr package provides a complete dplyr backend.
- Filter and aggregate Spark datasets then bring them into R for analysis and visualization.
- Use Spark's distributed machine learning library from R.
- Create extensions that call the full Spark API and provide interfaces to Spark packages.



SPARKLYR (INSTALAÇÃO)

`system("java -version")` (Necessário ter Java 8 instalado)

`install.packages("sparklyr")`

`spark_install()`

`spark_installed_versions()`

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

base_sparklyr.R* script_sparklyr.R* Untitled1*

Source on Save Run Source

```
1 library(sparklyr)
2 library(dplyr)
3
4 #spark_install()
5 #spark_installed_versions()
6
7 sc <- spark_connect(master = "local")
8 |
```

8:1 (Top Level) R Script

Console Terminal Jobs

```
~/
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

  filter, lag

The following objects are masked from 'package:base':

  intersect, setdiff, setequal, union

> spark_installed_versions()
  spark hadoop dir
1 2.4.3 2.7 /home/tumenas/spark/spark-2.4.3-bin-hadoop2.7
> sc <- spark_connect(master = "local")
* Using Spark: 2.4.3
> |
```

Environment History Connections Tutorial

Spark Log SQL Help

local (No tables)

Files Plots Packages Help Viewer

New Folder Delete Rename More

Home

	Name	Size	Modified
<input type="checkbox"/>	.r		
<input type="checkbox"/>	.Rhistory	11.3 KB	Jan 27, 2021, 4:31 PM
<input type="checkbox"/>	Aulas		
<input type="checkbox"/>	derby.log	710 B	Jan 27, 2021, 11:19 PM
<input type="checkbox"/>	Desktop		
<input type="checkbox"/>	Documents		
<input type="checkbox"/>	Downloads		
<input type="checkbox"/>	exemplo_enem.csv	60.7 MB	Jan 24, 2021, 12:00 AM
<input type="checkbox"/>	logs		
<input type="checkbox"/>	mapper.py	555 B	Jan 22, 2021, 4:35 PM
<input type="checkbox"/>	Models		
<input type="checkbox"/>	Music		
<input type="checkbox"/>	nltk_data		
<input type="checkbox"/>	Papers		

Spark Jobs (?)

User: tumenas

Total Uptime: 12 s

Scheduling Mode: FIFO

Completed Jobs: 1

▶ Event Timeline

▼ Completed Jobs (1)

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	count at utils.scala:116 count at utils.scala:116	2021/01/27 23:19:50	0,7 s	2/2	<div>2/2</div>

spark_web(sc)

SPARKLYR

Inserir dados

```
cars <- copy_to(sc, mtcars)
```

```
cars
```

```
count(cars)
```

Gráfico com amostra

```
select(cars, hp, mpg) %>%
```

```
  sample_n(100) %>%
```

```
  collect() %>%
```

```
  plot()
```

Modelos

```
model <- ml_linear_regression(cars, mpg ~ hp)
```

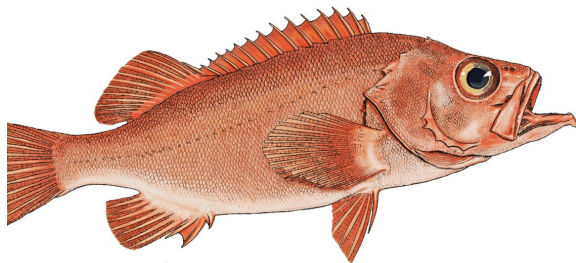
```
model
```

<https://therinspark.com/>

O'REILLY®

Mastering Spark with R

The Complete Guide to Large-Scale Analysis
and Modeling



Javier Luraschi,
Kevin Kuo & Edgar Ruiz
Foreword by Matei Zaharia

PARQUET

Lançado em 2013 (desenvolvido pelo Twitter + Cloudera).

Projetado como armazenamento colunar eficiente (em comparação a arquivos CSV ou TSV)



PARQUET

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	\$5.75
Data stored in Apache Parquet format*	130 GB	6.78 seconds	2.51 GB	\$0.01
Savings / Speedup	87% less with Parquet	34x faster	99% less data scanned	99.7% savings

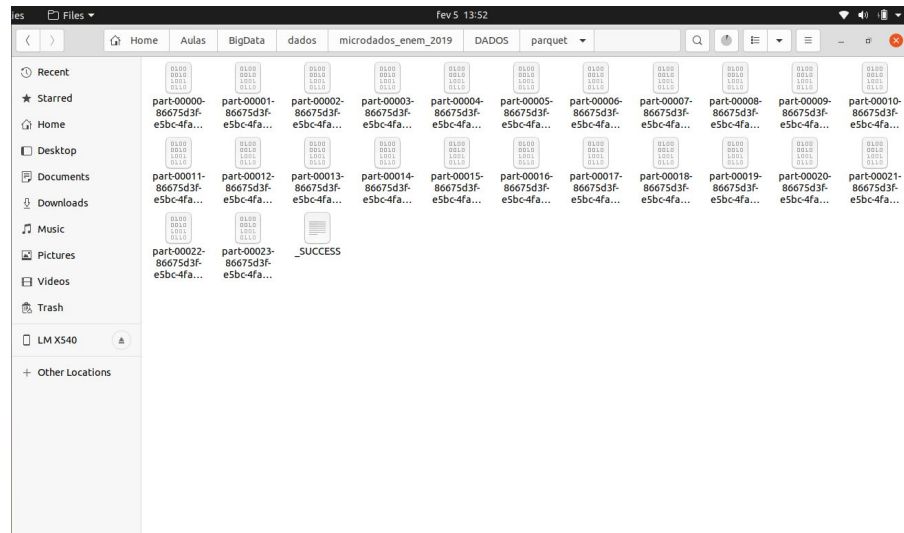
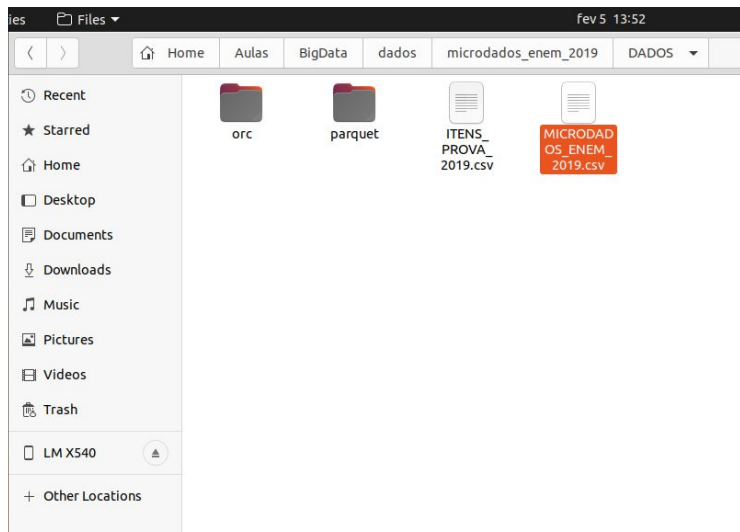
<https://dzone.com/articles/how-to-be-a-hero-with-powerful-parquet-google-and>

CONVERSÃO PARQUET (SPARKLYR)

```
# Leitura CSV Original
base_spark <- spark_read_csv(sc, name = nome_arquivo,
                             header = TRUE,
                             path= endereco,
                             delimiter = ";",memory=FALSE)
```

```
# Criar cópia em Parquet
spark_write_parquet(
  base_spark,
  endereco_parquet,
  mode = NULL,
  list(spark.sql.parquet.compression.codec="lzo"))
```

CONVERSÃO PARQUET (SPARKLYR)



3,2 Gb

638 Mb

ORC

- Lançado em 2013 (Hortonworks + Facebook)
- Optimized Row Columnar
- Transações ACID



CONVERSÃO ORC (SPARKLYR)

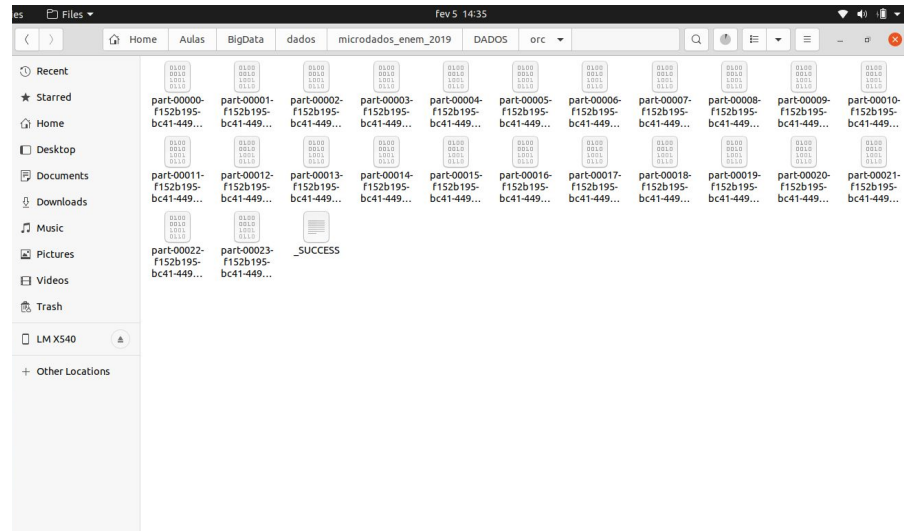
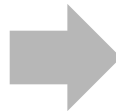
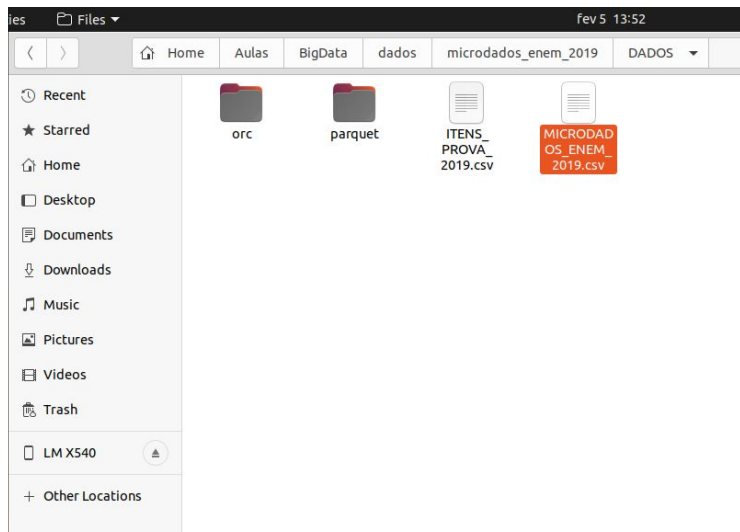
```
# Leitura CSV Original
```

```
base_spark <- spark_read_csv(sc, name = nome_arquivo,  
                             header = TRUE,  
                             path= endereco,  
                             delimiter = ";",memory=FALSE)
```

```
# Criar cópia em Parquet
```

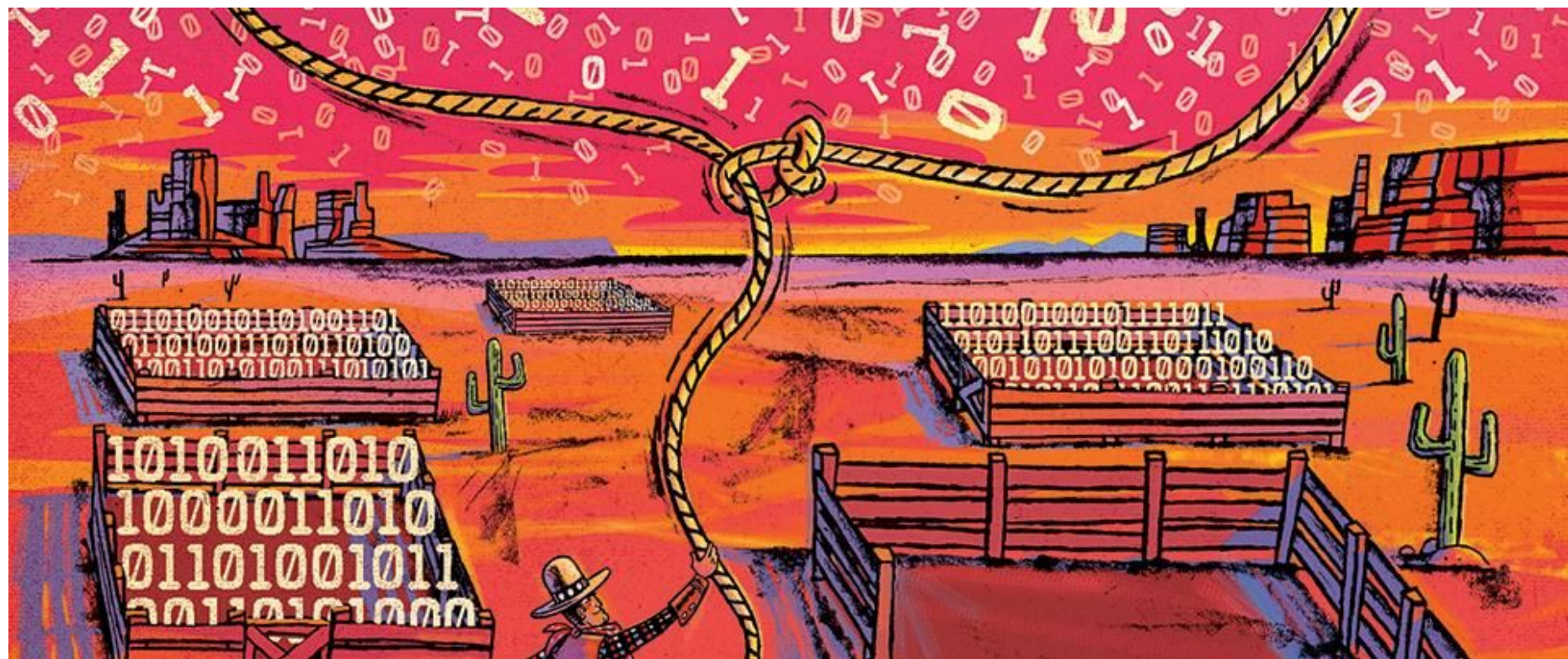
```
spark_write_orc(  
  base_spark,  
  endereco_orc,  
  mode = NULL)
```

CONVERSÃO ORC (SPARKLYR)



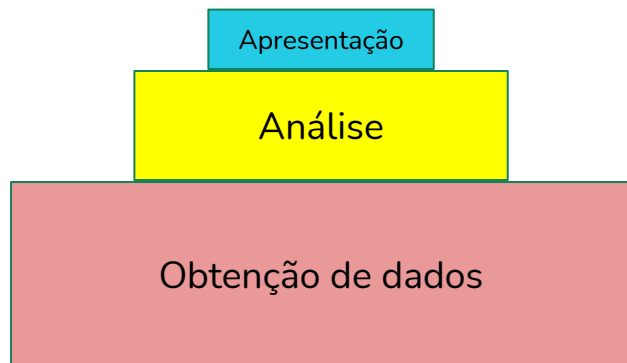
3,2 Gb

693 Mb

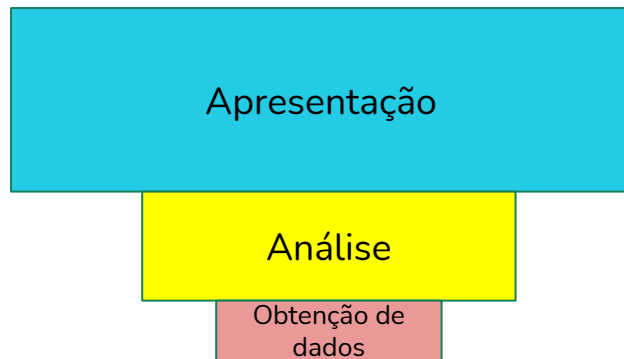


PROCESSO DE ANÁLISE DA DADOS

Esforço



Percepção de valor



REST

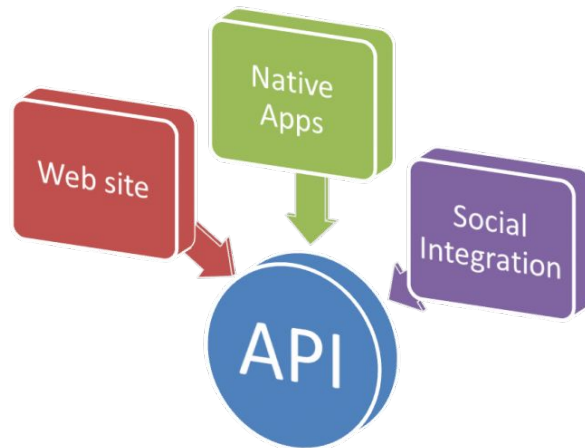
“**Representational State Transfer (REST)**, em português **Transferência de Estado Representacional**, é um estilo de arquitetura que define um conjunto de restrições e propriedades baseados em HTTP. Web Services que obedecem ao estilo arquitetural REST, ou web services **RESTful**, fornecem interoperabilidade entre sistemas de computadores na Internet. Os web services compatíveis com REST permitem que os sistemas solicitantes acessem e manipulem representações textuais de recursos da Web usando um conjunto uniforme e predefinido de operações sem estado. Outros tipos de web services, como web services SOAP, expõem seus próprios conjuntos arbitrários de operações.”

“Um conjunto de *operações bem definidas* que se aplicam a todos os *recursos* de informação: HTTP em si define um pequeno conjunto de operações, as mais importantes são **POST, GET**,...”

404

API

Application Programming Interface:
conjunto de padrões e rotinas estabelecidos
que permitem a **interoperabilidade entre**
aplicações.



TL;DR WEB ENTERTAINMENT

Iron Maiden makes millions of dollars by playing live for pirates (update)

122

By Rich McCormick | Dec 25, 2013, 9:59pm EST



SHARE



iron-maiden-flickr

MOST READ



Xiaomi's translucent Mi 8 Explorer Edition has an in-display fingerprint sensor and 3D face unlock



EXERCÍCIO API

QUAIS SÃO AS 10 MÚSICAS DO IRON MAIDEN MAIS TOCADAS NO BRASIL NO SPOTIFY?

SPOTIFY

```
install.packages('spotifyr')  
#library(devtools)  
#devtools::install_github('charlie86/spotifyr')
```

```
library(spotifyr)
```

```
Sys.setenv(SPOTIFY_CLIENT_ID = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx')  
Sys.setenv(SPOTIFY_CLIENT_SECRET = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx')
```

```
iron_maiden <- get_artist_audio_features('iron maiden')
```

```
iron_maiden_TopTracks <- get_artist_top_tracks('6mdiAmATAx73kdxrNrnIao', market = "BR",  
  authorization = get_spotify_access_token(),  
  include_meta_info = FALSE)
```



EXERCÍCIO API

```
library(httr)
```

```
astros = GET("http://api.open-notify.org/astros.json")
```

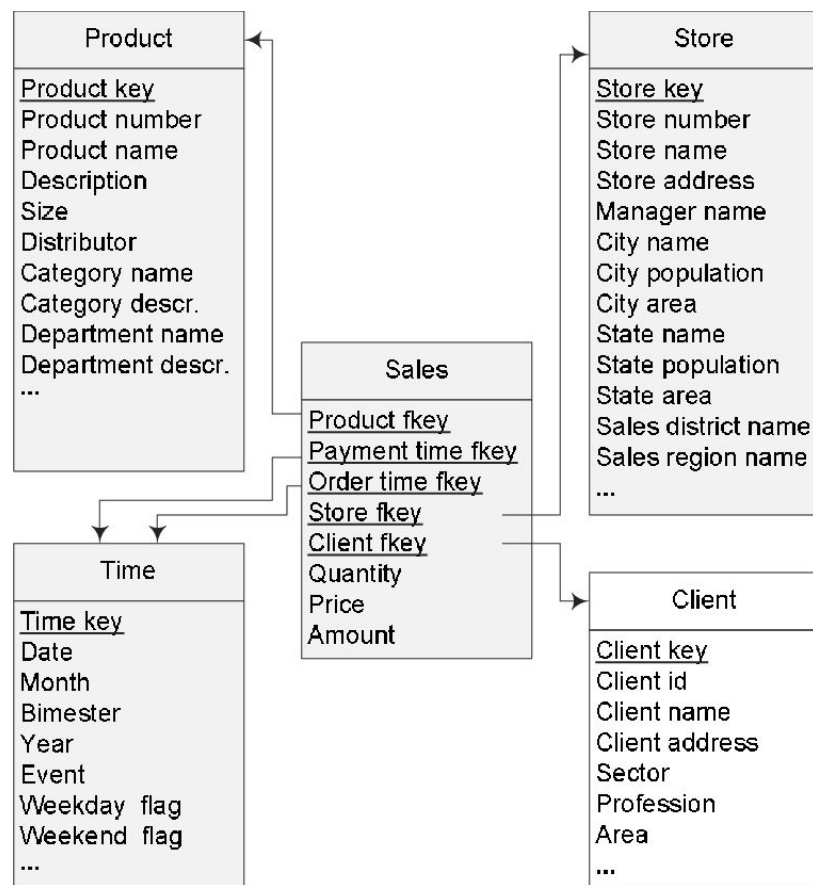
```
farol = GET("http://api.open-notify.org/iss-pass.json",
```

```
  query = list(lat = -13.009565000112607, lon = -38.53302776815651))
```

JSON

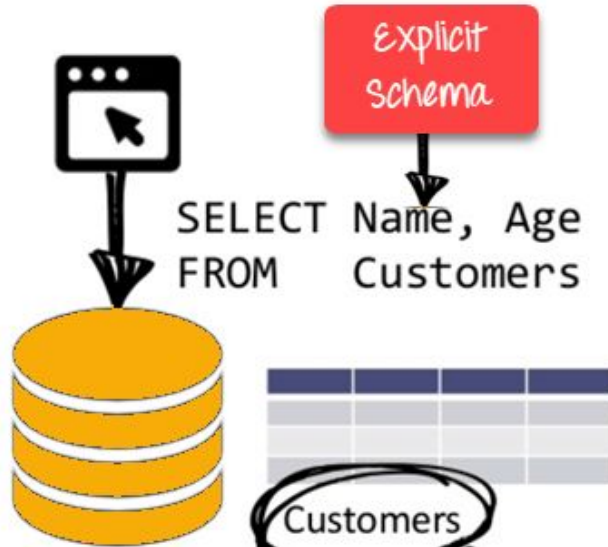
“JSON (JavaScript Object Notation) é um formato de representação e intercâmbio de dados complexos e heterogêneos “

```
{  
  "id": "00000234567894",  
  "name": "Jane Doe",  
  "birthday": "04/18/1978",  
  "gender": "female",  
  "type": "user",  
  "work": [{  
    "employer": {  
      "id": "106119876543210",  
      "name": "Doe Inc."  
    },  
    "start_date": "2007 - 08"  
  },  
  {  
    "start_date": "2004",  
    "end_date": "2007"  
  }  
}]  
}
```

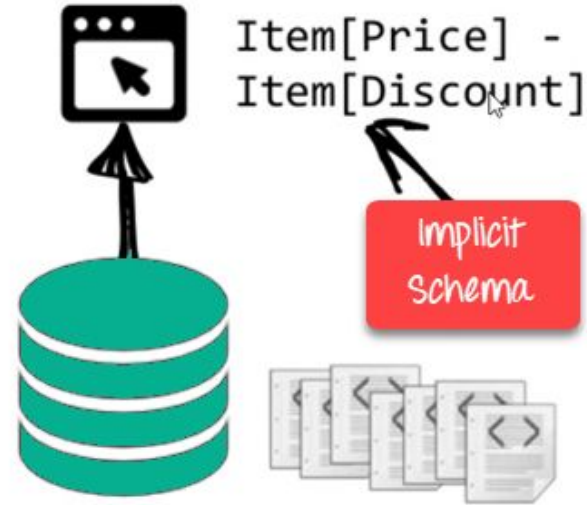


RDBMS x NoSQL

RDBMS:



NoSQL DB:



NoSQL (Not Only SQL)

APACHE
HBASE

 **Cassandra**



CouchDB
relax



riak



mongoDB

HYPERTABLE INC



Neo4j



redis



amazon
DynamoDB

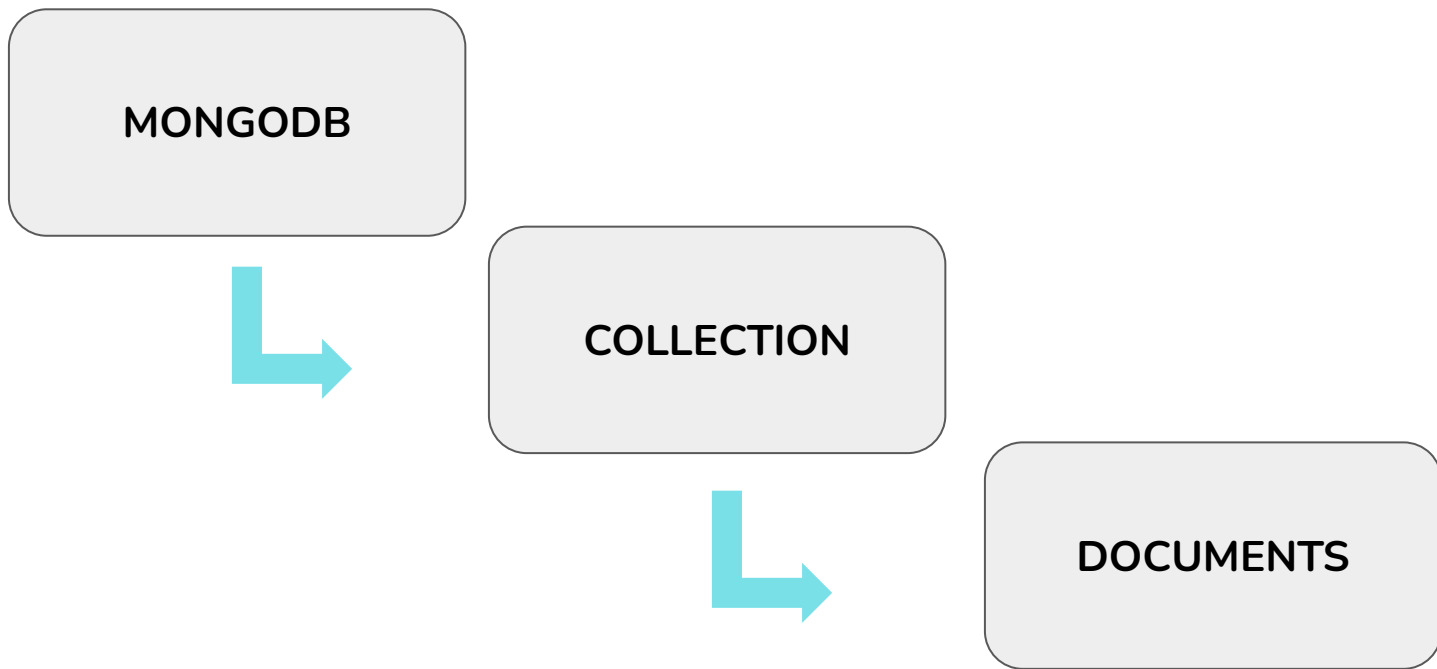
MONGODB

Lançado em 2009, é um banco de dados de código aberto, gratuito, de alta performance, sem schemas e orientado à documentos



<https://www.mongodb.com/>

MONGODB





[←](#) [→](#) [↺](#) [🏠](#) [🔒](#) <https://www.mongodb.com/try/download/community> [📄](#) [⋮](#) [🔖](#) [★](#) [↓](#) [📁](#) [📄](#) [👤](#) [B+](#) [☰](#)


[Share your story at MongoDB.live 2021](#) [Submit your talk](#)

[mongoDB](#) [Cloud](#) [Software](#) [Pricing](#) [Learn](#) [Solutions](#) [Docs](#) [🔍](#) [Contact](#) [Sign In](#) [Try Free](#)

Choose which type of deployment is best for you

**Cloud**
The easiest way to run MongoDB

**On-Premises**
Download on your own infrastructure

**Tools**
Do more with your data(base)

MongoDB Enterprise Server

MongoDB Community Server

MongoDB offers both an Enterprise and Community version of its powerful distributed document database. The community version offers the flexible document model along with ad hoc queries, indexing, and real time aggregation to provide powerful ways to access and analyze your data. As a distributed system you get high availability through built-in replication and failover along with horizontal scalability with native sharding.

Available Downloads

Version

4.4.3 (current)

Platform

Amazon Linux

Package

server

[🗨️](#)

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

MONGOLITE

```
install.packages("mongolite")  
devtools::install_github("jeroen/mongolite")
```

```
library(mongolite)  
m <- mongo(collection = "diamonds")
```

```
# Insert test data  
data(diamonds, package="ggplot2")  
m$insert(diamonds)
```

```
# Check records  
m$count()
```

```
# Perform a query and retrieve data  
out <- m$find('{"cut" : "Premium", "price" : { "$lt" : 1000 } }')
```

MONGOLITE TUTORIAL

<https://jeroen.github.io/mongolite/>