

## Introdução

NoSQL



## NoSQL

- Not Only SQL
  - Banco de dados não relacionais
- Características
  - Alto desempenho
  - Arquitetura distribuída
  - Schema não é rígido
- Tipos de banco de dados
  - SQL - Relacional
  - NoSQL



Exibir todos



## NoSQL Armazenamento

### ○ SQL - Relacional

- Orientado a Linha



### ○ NoSQL

- Orientado a Colunas



- Chave-Valor



- Orientado a Grafos



- Orientado a Documentos

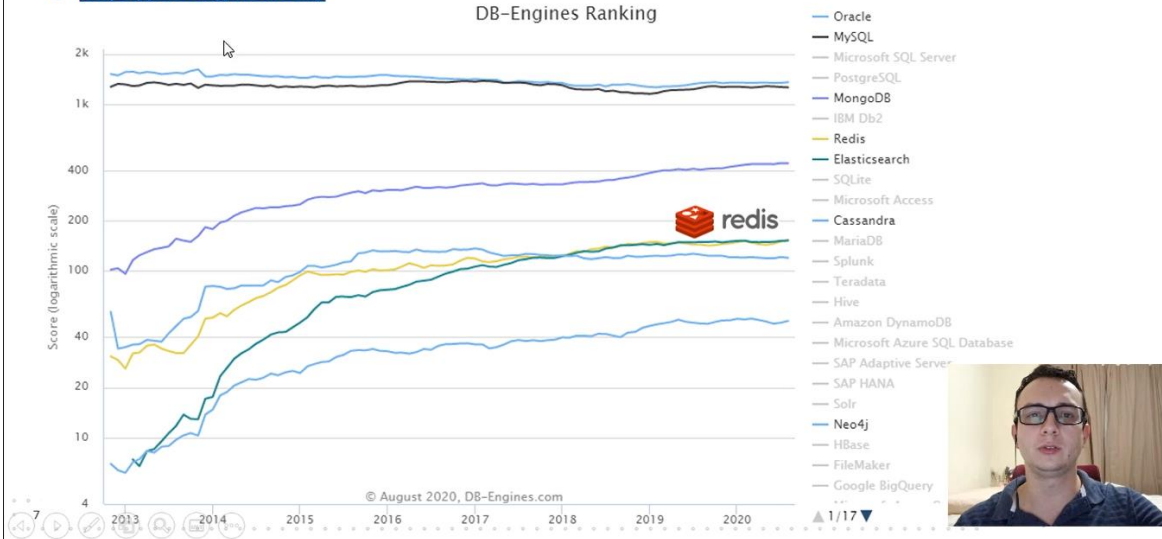


01:48



## Ranking Banco de dados

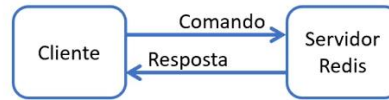
- <https://db-engines.com/>



## Redis

- Banco de dados
  - NoSQL
  - Chave-Valor mais popular do mundo
    - Valor -> diversas estrutura de dados
  - Armazenamento em memória
    - Acesso rápido aos dados
      - Escrita
      - Leitura
    - Possível persistir os dados fisicamente
      - Não é um BD para armazenar todos os dados

- Servidor TCP
  - Modelo cliente-servidor



- Comandos são atômicos
  - Aplicação single-threaded
  - Um comando por vez
- Comunicação
  - Aplicação
  - Redis CLI



9

## Redis Cliente

- Clientes disponíveis para as linguagens de programação:

ActionScript	ActiveX/COM+	Bash	Boomi	C	C#
C++	Clojure	Common Lisp	Crystal	D	Dart
Delphi	Elixir	emacs lisp	Erlang	Fancy	gawk
GNU Prolog	Go	Haskell	Haxe	Io	Java
Julia	Lasso	Lua	Matlab	mruby	Nim
Node.js	Objective-C	OCaml	Pascal	Perl	PHP
PL/SQL	Pure Data	Python	R	Racket	Rebol
Ruby	Rust	Scala	Scheme	Smalltalk	Swift
Tcl	VB	VCL	Xojo	Zig	



10

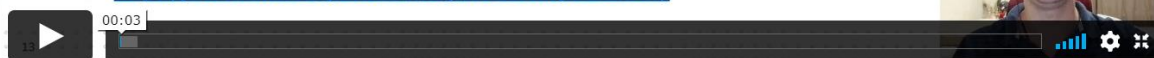
## Redis História

- Criado por Salvatore Sanfilippo
  - Primeira versão liberada em 2009
- Patrocinado pela Pivotal Software e pela VMware
  - A partir de 2015 mudou para a Redis Labs
- Site Oficial
  - <https://redis.io/>



## Instalação Redis

- Site Oficial
  - <https://redis.io/download>
- Localmente no Linux
  - `wget http://download.redis.io/releases/redis-6.0.6.tar.gz`
  - `$ tar xzf redis-6.0.6.tar.gz`
  - `$ cd redis-6.0.6`
  - `$ make`
- Docker
- Cloud - Serviço do Redis Labs
  - Desenvolvimento
  - Produção
  - <https://redislabs.com/redis-enterprise-cloud/overview/>
- Versão Redis
  - `redis-server --version`  
Redis server v=6.0.6



## Preparação Ambiente

### ○ Instalação

- Docker: <https://docs.docker.com/get-docker/>
- Docker-compose: <https://docs.docker.com/compose/install/>
- SO
  - Windows
    - Docker Desktop (Hyper-V ou Hyper-V com WSL2)
    - Docker Toolbox (VirtualBox)
  - Linux - Seguir o passo a passo (PassosInstalacaoDockerLinux.txt)
  - Mac - Docker Desktop

14



## Preparação Ambiente

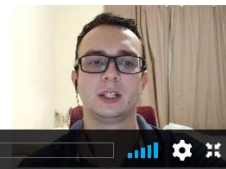
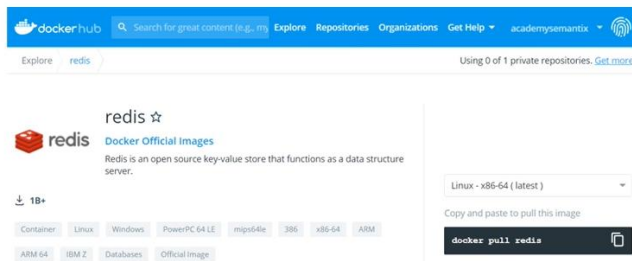
### ○ Download da imagem: [https://hub.docker.com/\\_/redis](https://hub.docker.com/_/redis)

- `docker pull redis`

### ○ Criar a seguinte estrutura de pasta:

redis

docker-compose.yml



## Opções Docker Compose

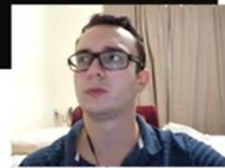
- Iniciar todos os serviços
  - \$ docker-compose up -d
- Parar os serviços
  - \$ docker-compose stop
- Iniciar os serviços
  - \$ docker-compose start
- Término do treinamento
  - Matar os serviços
    - \$ docker-compose down
  - Apagar todos os volumes sem uso
    - \$ docker volume prune

- cat docker-compose.yml

```
version: '3.1'

services:
  redis:
    container_name: redis
    image: redis
    ports:
      - 6379:6379
    volumes:
      - data:/data
    entrypoint: redis-server --appendonly yes
    restart: always

volumes:
  data:
```



16

## Acessos Ambiente docker

- Visualizar os container
  - Ativos
    - \$ docker ps
  - Todos
    - \$ docker ps -a
- Executar comandos no container
  - \$ docker exec -it <container> <comando>
- Visualizar os logs
  - \$ docker logs <container>
- Enviar arquivos
  - \$ docker cp <diretório> <container>:/<diretório>

- Acesso Redis Cli e Server
  - \$ docker exec -it redis bash
    - # redis-cli
    - # redis-server
  - Instalação local
    - src/redis-server
    - src/redis-cli



## Exercícios Instalação

1. Instalação do docker e docker-compose
2. Baixar a imagem do redis
3. Iniciar o Redis através do docker-compose
4. Listar as imagens em execução
5. Verificar a versão do Redis
6. Acessar o Redis CLI



Pressione **Esc** para sair do modo tela cheia



# Semantix

## Redis

Aula 2



## Redis Tipos de dados

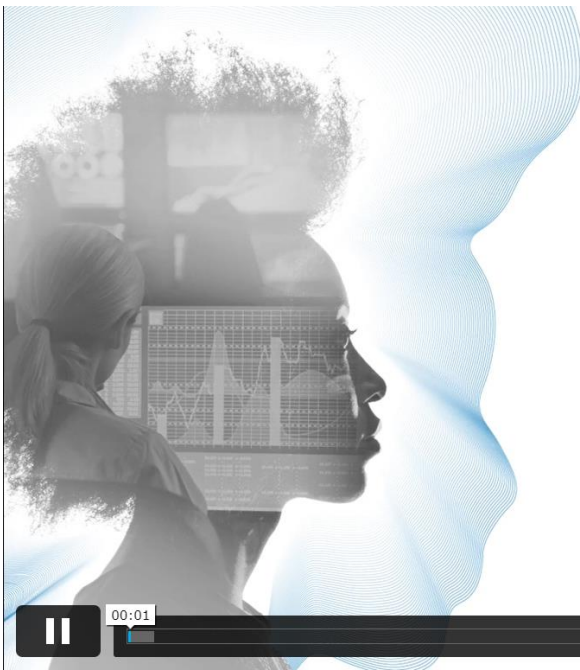
- Servidor de estruturas de dados que oferece suporte a diferentes tipos de valores
  - Não é um armazenamento de chave-valor simples
    - String-string
  - Aceita estruturas de dados mais complexas
    - Strings
    - Listas
    - Sets
    - Sets Ordenados
    - Hash
    - HyperLogLogs
    - Streams



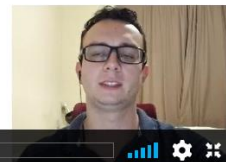


## Redis Chave

- As chaves são *binary safe*
  - Sequência binária
  - Regras
    - Tamanho máximo 512 MB
      - Não indicado chaves > 1 MB
      - Gastar memória e largura de banda
    - Dicas
      - Instanciar “object-type:id”
        - “user:1000” ao invés de “u1000”



## Strings



## Redis String

- Único tipo de dados *Memcached*
  - Cache para páginas Web
- Chaves também são string
  - Mapear string para outra string
- Sintaxe
  - Definir um valor de string: SET <chave> <valor>
  - Recuperar um valor de string: GET <chave>

```
root@2f3da0f8aeed:/data# redis-cli
127.0.0.1:6379> set minhaChave valorChave
OK
127.0.0.1:6379> get minhaChave
"valorChave"
```



## Redis String

- Opções para a chave String
  - NX – Falhar se a chave existir
  - XX (Default) – Substituir o valor da chave

```
127.0.0.1:6379> set minhaChave novoValor nx
(nil)
127.0.0.1:6379> set minhaChave novoValor xx
OK
```

- Verificar o tamanho do valor
  - Sintaxe: strlen <chave>

```
127.0.0.1:6379> get minhaChave
"novoValor"
127.0.0.1:6379> strlen minhaChave
(integer) 9
```



## Redis String

### String como um inteiro

- Sintaxe para incrementos e decrementos do valor
  - incr <chave>
  - decr <chave>
  - incrby <chave> <incremento>
  - decrby <chave> <decremento>

### Incr atômico

- Ex.
  - Ao mesmo tempo 2 clientes leem a chave 10 e ambos incrementem para 11
  - O valor final é 12
    - Não são executados os comandos ao mesmo tempo

```
127.0.0.1:6379> set contador 10
OK
127.0.0.1:6379> incr contador
(integer) 11
127.0.0.1:6379> incrby contador 5
(integer) 16
127.0.0.1:6379> decr contador
(integer) 15
127.0.0.1:6379> decrby contador 5
(integer) 10
127.0.0.1:6379> get contador
"10"
```



## Redis String

### Definir e recuperar várias chaves em um comando

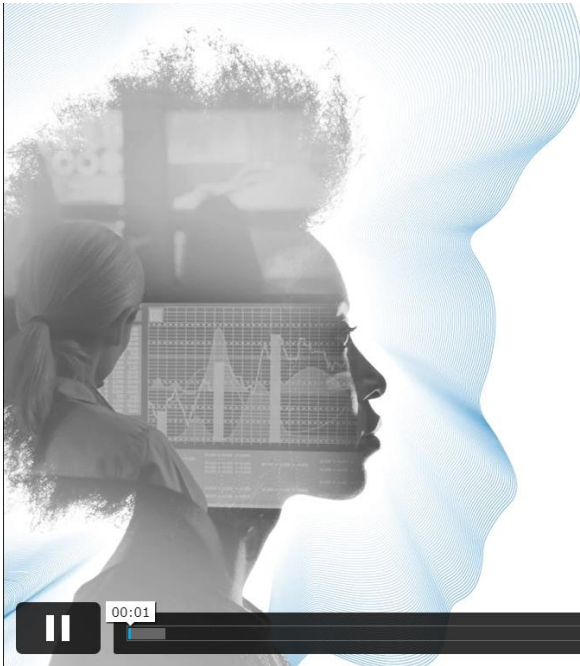
- Reduzir latência

### Sintaxe

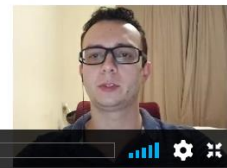
- MSET <chave 1> <valor> <chave 2> <valor> ... <chave N> <valor>
- MGET <chave 1> <chave 2>... <chave N>

```
127.0.0.1:6379>
127.0.0.1:6379> mset chave1 10 chave2 20 chave3 30
OK
127.0.0.1:6379> mget chave1 chave2 chave3
1) "10"
2) "20"
3) "30"
```





## Opções Básicas com Chaves



00:01



### Redis Opções com chaves

- Verificar se a chave existe
  - Sintaxe
    - exists <chave>
- Deletar chave
  - Sintaxe
    - del <chave>
- Tipo da chave
  - Sintaxe
    - type <chave>

```
127.0.0.1:6379> set minhaChave teste
OK
127.0.0.1:6379> type minhaChave
string
127.0.0.1:6379> exists minhaChave
(integer) 1
127.0.0.1:6379> del minhaChave
(integer) 1
127.0.0.1:6379> exists minhaChave
(integer) 0
127.0.0.1:6379> type minhaChave
none
127.0.0.1:6379> get minhaChave
(nil)
```



## Redis Persistência de Chaves



Remover tempo para a chave expirar

- Sintaxe
  - persist <chave>

```
127.0.0.1:6379> set minhaChave teste2 ex 50
OK
127.0.0.1:6379> ttl minhaChave
(integer) 46
127.0.0.1:6379> ttl minhaChave
(integer) 41
127.0.0.1:6379> persist minhaChave
(integer) 1
127.0.0.1:6379> ttl minhaChave
(integer) -1
127.0.0.1:6379> get minhaChave
"teste2"
```



# Semantix

## Redis

Aula 4



00:01



## Mensagens Pub/Sub

Pressione **Esc** para sair do modo tela cheia

- Implementar o paradigma de mensagens **Publish/Subscribe** (Publicar/Assinar)
  - Mensagens dos remetente (editor) não são enviadas diretamente para um destinatário (assinante)
  - Mensagens são publicadas através de um canal
    - Sem o editor saber quem são os assinantes
- Sintaxe
  - Publicar mensagem
    - `publish <canal> <mensagem>`
  - Assinar um ou mais canais
    - `subscribe <canal1> ... <canalN>`



## Mensagens Pub/Sub

- Ex.
  - Publicar/Assinar

```
127.0.0.1:6379> subscribe canal1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "canal1"
3) (integer) 1
```

```
127.0.0.1:6379> subscribe canal1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "canal1"
3) (integer) 1
1) "message"
2) "canal1"
3) "msg de teste"
```

```
127.0.0.1:6379> publish canal1 'msg de teste'
(integer) 1
127.0.0.1:6379>
```





## Mensagens Pub/Sub pattern

Pressione **Esc** para sair do modo tela cheia

- Assinar canais através de um padrão
  - Sintaxe
    - `psubscribe <padrão1> ... <padrãoN>`

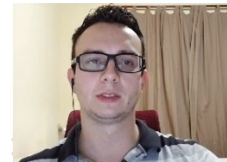
```
127.0.0.1:6379> psubscribe canal*
Reading messages... (press Ctrl-C to quit)
1) "psubscribe"
2) "canal*"
3) (integer) 1
```



## Mensagens Cancelamento

- Cancelar a assinatura dos canais
  - Sintaxe
    - Canais específicos
      - `unsubscribe <canal1> ... <canalN>`
    - Todos os canais
      - `unsubscribe`
- Cancelar a assinatura dos canais através de um padrão
  - Sintaxe
    - `punsubscribe <padrao1> ... <padraoN>`

```
127.0.0.1:6379> unsubscribe canal1
1) "unsubscribe"
2) "canal1"
3) (integer) 0
```





## Redis Revisão

- Tutorial Redis online

- <http://try.redis.io>



**\* TRY REDIS \***

Welcome to **Try Redis**, a demonstration of the **Redis** database!

Please type **TUTORIAL** to begin a brief tutorial, **HELP** to see a list of supported commands, or any valid Redis command to play with the database.

> **TUTORIAL**

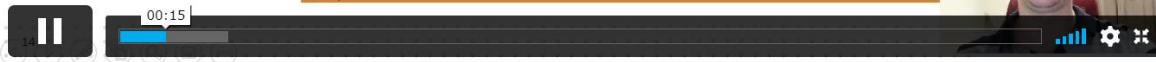
Redis is in the family of databases called key-value stores.

The essence of a key-value store is the ability to store some data, called a value, inside a key. This data can later be retrieved only if we know the exact key used to store it.

Often Redis it is called a data structure server because it has outer key-value shell, but each value can contain a complex data structure, such as a string, a list, a hashes, or ordered data structures called sorted sets as well as probabilistic data structures like hyperloglog.

As a first example, we can use the command **SET** to store the value "fido" at key "server.name":

```
SET server:name "fido"
```

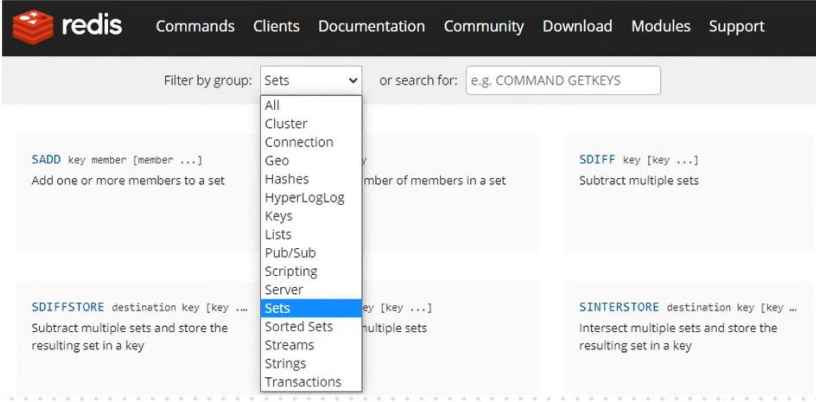




## Redis Revisão

### ○ Lista completa de comandos do Redis

- <https://redis.io/commands>

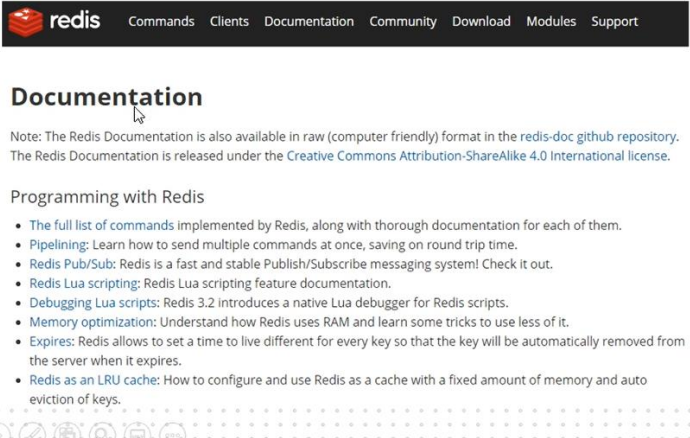


The screenshot shows the Redis website's 'Commands' page. A dropdown menu is open under the 'Filter by group' section, listing various Redis command categories: All, Cluster, Connection, Geo, Hashes, HyperLogLog, Keys, Lists, Pub/Sub, Scripting, Server, Sets (highlighted), Sorted Sets, Streams, Strings, and Transactions. The main content area displays several command cards, including SADD, SDIFF, SDIFFSTORE, and SINTERSTORE. A video player at the bottom shows a man speaking.

## Redis Revisão

### ○ Documentação oficial do Redis

- <https://redis.io/documentation>



The screenshot shows the Redis website's 'Documentation' page. The 'Documentation' header is highlighted. Below it, a note states: 'Note: The Redis Documentation is also available in raw (computer friendly) format in the redis-doc github repository. The Redis Documentation is released under the Creative Commons Attribution-ShareAlike 4.0 International license.' The 'Programming with Redis' section lists several topics: The full list of commands implemented by Redis, Pipelining, Redis Pub/Sub, Redis Lua scripting, Debugging Lua scripts, Memory optimization, Expires, and Redis as an LRU cache. A video player at the bottom shows a man speaking.