
FIAP
GRADUAÇÃO

FIAP

Arquiteturas Disruptivas e *Big Data*

Redis

Milton Goya

Versão 01 – fevereiro de 24

FIAP

NoSQL
BD Chave:Valor
redis

3

FIAP

NoSQL – Chave / Valor - redis

Principais Bancos Chave-Valor

Rank Mar 2023	Rank Feb 2023	Rank Mar 2022	DBMS	Database Model	Score Mar 2023	Score Feb 2023	Score Mar 2022
1.	1.	1.	Redis	Key-value, Multi-model	172.45	-1.39	-4.31
2.	2.	2.	Amazon DynamoDB	Multi-model	80.77	+1.08	-1.03
3.	3.	3.	Microsoft Azure Cosmos DB	Multi-model	36.10	-0.40	-4.79
4.	4.	4.	Memcached	Key-value	22.61	-0.56	-3.06
5.	5.	↑ 6.	Hazelcast	Key-value, Multi-model	8.63	-0.48	-1.43
6.	6.	↓ 5.	etcd	Key-value	8.60	+0.09	-3.23
7.	7.	↑ 10.	Aerospike	Multi-model	6.54	-0.02	+0.31
8.	8.	↑ 9.	Ehcache	Key-value	6.03	-0.01	-0.52
9.	9.	↑ 14.	Google Cloud Bigtable	Multi-model	5.44	-0.47	+1.10
10.	10.	↓ 8.	Ignite	Multi-model	5.29	-0.21	-1.58
11.	↑ 12.	↓ 7.	Riak KV	Key-value	5.28	+0.14	-1.75
12.	↓ 11.	↓ 11.	ArangoDB	Multi-model	5.04	-0.26	-0.57
13.	↑ 15.	↑ 15.	RocksDB	Key-value	4.57	+0.11	+0.51
14.	↓ 13.	↓ 13.	OrientDB	Multi-model	4.30	-0.24	-0.63
15.	↓ 14.		GemFire	Key-value, Multi-model	4.26	-0.26	

- https://db-engines.com/en/ranking/key-value+store

FIAP

O que é o redis?

- O nome Redis é um acrônimo para **Re**mote **Di**ctionary **S**erver
- Criado pelo programador italiano Salvatore Sanfillip, contratado pela empresa VMware
- Armazena dados no formato valor-chave em memória
- Oferece tempos de resposta menores que milissegundos, permitindo milhões de solicitações por segundo para aplicativos em tempo real em Jogos, Ad-Tech, Serviços Financeiros, Assistência Médica e IoT.
- O Redis é uma opção popular para armazenamento em cache, gerenciamento de sessões, jogos, tabelas de classificação, análises em tempo real, geoespaciais, veiculação de bate-papo, bate-papo/mensagens, streaming de mídia e aplicativos de pub/sub.
- Possui mecanismo de persistência de dados em disco;

FIAP

Características

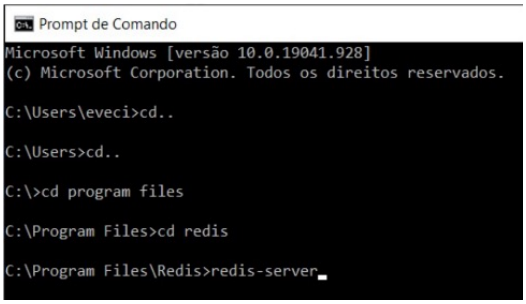
- **Velocidade**:
  - O Redis armazena todo o conjunto de dados na memória principal.
  - Carrega até 110.000 SETs/segundo e 81.000 GETs/segundo
- **Persistência**:
  - Todos os dados residem na memória
  - As alterações são salvas de forma assíncrona no disco
- **Estruturas de dados**:
  - strings, hashes, conjuntos, listas, conjuntos classificados com consultas de intervalo, bitmaps, hiperloglogs e índices geoespaciais.
- **Operações atômicas**:
  - Operações Redis trabalhando nos diferentes Tipos de Dados são atômicas
- **Idiomas suportados**:
  - ActionScript, C, C++, C #, Clojure, Common Lisp, D, Dart, Erlang, Go, Haskell, Iax, Java, JavaScript (Node.js), Julia, Lua, Objective-C, Perl, PHP, Dados Puros, Python, R, Raquete, Ruby, Rust, Scala, Smalltalk e Tcl.

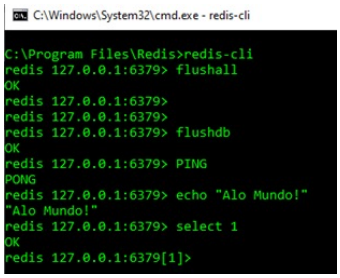
Características

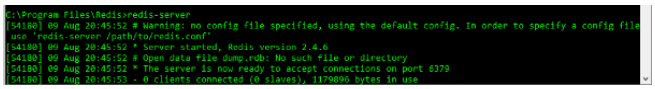
- **Replicação master/slave**: Redis segue uma replicação Master/Slave muito simples e rápida. São necessários apenas uma linha no arquivo de configuração para configurá-la e 21 segundos para que um escravo conclua a sincronização inicial do conjunto de chaves de 10 MM em uma instância do Amazon EC2.
- **Sharding**: Redis suporta sharding. É muito fácil distribuir o conjunto de dados em várias instâncias do Redis, como outro armazenamento de valor-chave.
- **Portátil**: o Redis é escrito em ANSI C e funciona na maioria dos sistemas POSIX, como Linux, BSD, Mac OS X, Solaris etc. Redis é relatado para compilar e trabalhar sob o WIN32 se compilado com o Cygwin, mas não há suporte oficial para o Windows atualmente.

Versão para Windows

- https://github.com/MSOpenTech/redis/releases/download/win-3.2.100/Redis-x64-3.2.100.msi







FIAP

O que é o redis?

- É um Banco de Dados NoSQL do tipo Chave / Valor em memória;
- Possui mecanismo de persistência de dados em disco;
- Possui tipos de dados interessantes:
  - **Binary-Safe Strings**: todas as chaves no Redis são binary-safe strings, ou seja, você pode usar qualquer sequência binária como uma chave, de uma string como "oi" ao conteúdo de um arquivo JPEG.
  - **LISTs**: coleções de elementos de string ordenadas de acordo com a ordem de inserção. Eles são basicamente listas vinculadas.
  - **SETs**: coleções de elementos únicos e não ordenados de strings.

FIAP

O que é o redis?

- Possui tipos de dados interessantes:
  - **Sorted SETs**: semelhantes a conjuntos, mas cada elemento de sequência está associado a um valor numérico flutuante, chamado pontuação. Os elementos são sempre ordenados por sua pontuação, portanto, ao contrário de Sets, é possível recuperar um intervalo de elementos (por exemplo, você pode perguntar: me dê o top 10 ou o 10 inferior).
  - **HASHs**: que são mapas compostos de campos associados a valores. O campo e o valor são strings. Isso é muito semelhante aos hashes Ruby ou Python.
  - **BITMAPs**: é possível, usando comandos especiais, manipular valores de String como uma matriz de bits: você pode definir e limpar bits individuais, contar todos os bits definidos como 1, localizar o primeiro conjunto ou indefinido, e assim por diante.
  - **HYPERLOGLOGs**: esta é uma estrutura de dados probabilística que é usada para estimar a cardinalidade de um conjunto.

FIAP

O que é o redis?

- Totalmente *open source*, contando com uma comunidade ativa e amigável;
- Desenvolvimento constante e com respostas a novos requisitos de maneira rápida;
- Patrocinado pela VMWare;
- Alguns casos no mundo real: github, craigslist;

FIAP

redis – Algumas Características

- Dados em memória (quase tudo) ;
- Todos os dados são eventualmente persistidos (podendo inclusive ser configurado para persistir imediatamente);
- Manipula grandes cargas de dados de maneira fácil;
- Ideal para grandes cargas de escrita;
- Suporta operações atômicas;
- Oferece suporte a transações;
- Possui várias bibliotecas para suportar diversas linguagens;

FIAP

redis – Escalabilidade

- Suporta replicação do tipo *Master-Slave*;
- Nós *slave* podem ser transformados em nó *master* em tempo de execução;
- Atualmente não suporta *clusterização* "real";

FIAP

redis – Persistência

- Todos os dados são sincronizados com o disco (eventualmente ou imediatamente);
- Configura-se o intervalo de persistência conforme sua necessidade de risco X desempenho;
- Os dados são escritos todos de uma vez através de um processo ou escritos sob demanda em um *change log* (AOF – *Append-Only File*);
- O modo *append-only* suporta escritas em disco por transação, sendo assim não há perda de dados, no entanto o custo é de 99% de perda de desempenho; ☺
- O AOF cresce bastante, no entanto o redis consegue diminuí-lo em tempo de execução;

redis – Persistência

- É possível “salvar” o estado atual explicitamente, em segundo plano ou em bloco;
- Configuração Padrão:
  - Estado salvo após 900s (15min) se, ao menos, 1 chave foi alterada;
  - Estado salvo após 300s (5min) se, ao menos, 10 chaves foram alteradas;
  - Estado salvo após 60s (1min) se, ao menos, 10000 chaves foram alteradas;

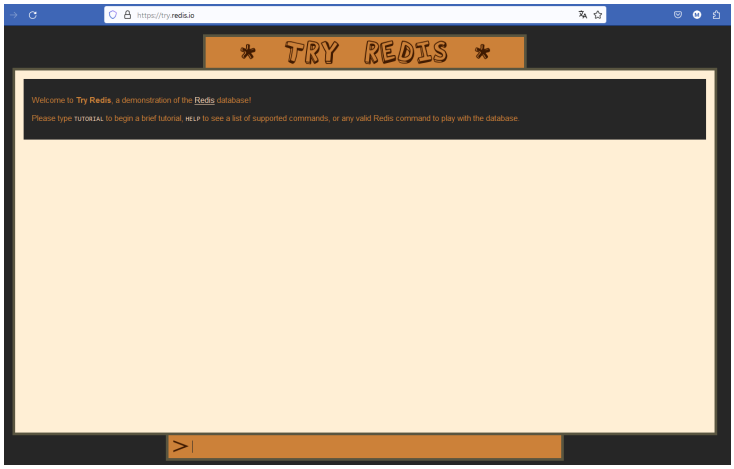
redis – Memória Virtual

- Caso o banco de dados seja muito grande, o redis pode utilizar *swap* por conta própria;
- As chaves são mantidas em memória, porém as menos usadas são enviadas para disco;
- I/O em *swap* acontece em processos separados;
- Para este tipo de situação o melhor é utilizar outra tecnologia ou escalar verticalmente o servidor; ☹

FIAP

Terminal Online Redis

- https://try.redis.io/



FIAP

redis – Operações Básicas com *strings*

- GET: Retorna o valor de uma chave. Caso a chave não exista será retornado `nil`.

```
redis> GET cliente_01
(nil)
redis> GET msg_01
"Hello"
```

- SET: Define um valor para uma chave. Caso a chave já possua um valor, o mesmo será sobrescrito.

```
redis> SET msg_01 "Hello"
OK
redis> GET msg_01
"Hello"
```

FIAP

redis – Operações Básicas com *strings*

- MGET: Retorna o valor de todas as chaves especificadas. Caso não exista, retorna `nil`.

```
redis> SET chave1 "Hello"
OK
redis> SET chave2 "World"
OK
redis> MGET chave1 chave2 chave3
1) "Hello"
2) "World"
3) (nil)
```

- SETNX: Define um valor para uma chave somente se a mesma não existir. Caso exista retorna erro.

```
redis> SET chave1 "Hello"
*OK*
redis> SETNX chave1 "World"
*FAILS*
```


redis – Operações Básicas com *strings*

- INCR: Incrementa em 1 o valor de uma chave. Caso a chave não exista, define como 0 antes de executar a operação.

```
redis> SET chave1 "10"
OK
redis> INCR chave1
(integer) 11
redis> GET chave1
"11"
```

- ECHO: Retorna uma mensagem definida.

```
redis> ECHO "Hello World!"
"Hello World!"
```

FIAP

redis – Operações Básicas com *Lists*

- LPUSH: Insere todos os valores definidos no topo de uma lista *(List)*.

```
redis> LPUSH chave1 "world"
(integer) 1
redis> LPUSH chave1 "hello"
(integer) 2
redis> LRANGE chave1 0 -1
1) "hello"
2) "world"
```


FIAP

redis – Operações Básicas com *Lists*

- LRANGE: Retorna os valores especificados por índice em uma lista (*List*).

```
redis> RPUSH chave1 "um"
(integer) 1
redis> RPUSH chave1 "dois"
(integer) 2
redis> RPUSH chave1 "tres"
(integer) 3
redis> LRANGE chave1 0 0
1) "um"
redis> LRANGE chave1 -3 2
1) "um"
2) "dois"
3) "tres"
redis> LRANGE chave1 -100 100
1) "um"
2) "dois"
3) "tres"
redis> LRANGE chave1 5 10
(empty list or set)
```

FIAP

redis – Operações Básicas com *Sets*

- SADD: Adiciona um valor específico a uma chave do tipo *Set*.

```
redis> SADD chave_01 "Hello"
(integer) 1
redis> SADD chave_01 "World"
(integer) 1
redis> SADD chave_01 "World"
(integer) 0
redis> SMEMBERS chave_01
1) "World"
2) "Hello"
```

FIAP

redis – Operações Básicas com *Sets*

- SINTER: Retorna os valores resultantes da intersecção de duas chaves do tipo *Set*.

```
redis> SADD chave1 "a"
(integer) 1
redis> SADD chave1 "b"
(integer) 1
redis> SADD chave1 "c"
(integer) 1
redis> SADD chave2 "c"
(integer) 1
redis> SADD chave2 "d"
(integer) 1
redis> SADD chave2 "e"
(integer) 1
redis> SINTER chave1 chave2
1) "c"
```


redis – Operações Básicas com *Sorted Sets*

- ZADD: Adiciona todos os valores definidos em uma ordem especificada a uma chave do tipo *Sorted Set.*

```
redis> ZADD produto 1 "laranja"
(integer) 1
redis> ZADD produto 1 "banana"
(integer) 1
redis> ZADD produto 3 "abacate" 2 "goiaba"
(integer) 2
redis> ZRANGE produto 0 -1 WITHSCORES
1) "laranja"
2) "1"
3) "banana"
4) "1"
5) "goiaba"
6) "2"
7) "abacate"
8) "3"
```

FIAP

redis – Operações Básicas com *Sorted Sets*

- ZSCORE: Retorna a ordem de um valor em uma chave do tipo *Sorted Set.*

```
redis> ZADD pets 1 "gato"
(integer) 1
redis> ZSCORE pets "gato"
"1"
```

- ZRANK: Retorna a classificação de um valor em uma chave do tipo *Sorted Set*, ordenado do menor para o maior.

```
redis> ZADD pets 1 "gato"
(integer) 1
redis> ZADD pets 2 "papagaio"
(integer) 1
redis> ZADD pets 3 "tubarão"
(integer) 1
redis> ZRANK pets "tubarão"
(integer) 3
redis> ZRANK pets "dinossauro"
(nil)
```

FIAP

redis – Operações Básicas com *Hashes*

- HSET: Define o valor para um campo em uma chave do tipo *Hash.*

```
redis> HSET usuario1 msg1 "Hello"
(integer) 1
redis> HGET usuario1 msg1
"Hello"
```

- HGETALL: Retorna todos os campos e valores de uma chave do tipo *Hash.*

```
redis> HSET usuario1 msg1 "Hello"
(integer) 1
redis> HSET usuario1 msg2 "World"
(integer) 1
redis> HGETALL usuario1
1) "msg1"
2) "Hello"
3) "msg2"
4) "World"
```

FIAP

redis – *Publish / Subscribe*

- É possível inscrever-se em canais para receber notificações sempre que alguma mensagem chegar no canal. Ou também publicar no canal.

```
redis> subscribe feed:joe feed:moe feed:boe
// Agora é só esperar

redis> publish feed:joe "all your base are belong to me"
(integer) 1 //received by 1

1. "message"
2. "feed:joe"
3. "all your base are belong to me"
```

FIAP

redis – Exemplo Prático de Aplicação

```
# Adicionando alguns seguidores
>>> client.rpush('user:1:followers', 2)
>>> numFollowers = client.rpush('user:1:followers', 3)
>>> msgId = client.incr('messages:id') #ATOMIC OPERATION
# Adicionando Mensagem
>>>  client.hmset('messages:%s'  %  msgId,  {'text':  'hello  world',
   'user': 1})
# Distribuindo para os seguidores
>>> followers = client.lrange('user:1:followers', 0, numFollowers)
>>> pipe = client.pipeline()
>>> for f in followers:
pipe.rpush('user:%s:feed' % f, msgId)
>>> pipe.execute()
>>> msgId = client.incr('messages:id') # Incrementar id
# Repetir várias vezes
# Agora capturar as mensagens do usu[ario 2]
>>> client.sort(name = 'user:2:feed', get='messages:*->text')
['hello world', 'foo bar']
```

FIAP

Referências Bibliográficas

- VOLK, Dvir. redis “Little Server of Awesome”. 2011. Disponível em: http://pt.slideshare.net/dvirsky/introduction-to-redis. Acesso em: 01 agosto 2018.
- tutorialspoint. Redis Tutorial. Disponível em: http://www.tutorialspoint.com/redis/index.htm. Acesso em: 01 agosto 2018.
- redis. Disponível em: redis.io. Acesso em: 01 agosto 2017.