

# Motores Baseados em Hashing (Redis)

August 21, 2025

## 1 Conceito de Hashing

O hashing é uma técnica fundamental em ciência da computação para mapear dados de tamanho arbitrário (chaves) para valores de tamanho fixo (índices), usando uma função hash. Em bancos de dados e sistemas de armazenamento, isso permite localizar rapidamente onde um dado está armazenado.

extbfExemplo de função hash:

```
hash("usuario:1001") = 42
```

Assim, o valor referente à chave “usuario:1001” será armazenado na posição 42 da tabela hash.

extbfColisões: Quando duas chaves diferentes geram o mesmo índice, ocorre uma colisão. Motores baseados em hashing usam técnicas como encadeamento (listas ligadas) ou endereçamento aberto para resolver colisões.

## 2 Introdução

Motores baseados em hashing são sistemas de armazenamento de dados que utilizam funções hash para mapear chaves a posições na memória, permitindo acesso extremamente rápido e eficiente. O Redis é o principal exemplo moderno desse tipo de motor, sendo amplamente utilizado em aplicações que exigem alta performance, como cache, filas, contadores e armazenamento de sessões.

extbfExemplo prático de uso de hash:

```
// Inserção
SET usuario:1001 "Ana"
// Busca
GET usuario:1001
```

## 3 Como funciona um motor baseado em hashing?

1. O usuário envia uma chave (ex: “usuario:1001”).
2. O sistema aplica uma função hash à chave, obtendo um índice.
3. O valor é armazenado ou recuperado diretamente na posição correspondente da tabela hash.

extbfVantagem principal: O acesso é feito em tempo constante, independentemente do tamanho do banco de dados.

## 4 Vantagens e Desvantagens

extbfVantagens:

- **Acesso em tempo constante ( $O(1)$ ):** Operações de leitura e escrita são realizadas em tempo constante na maioria dos casos.
- **Eficiência para grandes volumes:** Ideal para aplicações que exigem alta performance e manipulação de grandes quantidades de dados.
- **Simplicidade:** Estrutura de dados simples, fácil de implementar e manter.

extbfDesvantagens:

- Não é ideal para buscas por intervalos ou ordenação de dados.
- Pode sofrer com colisões de hash, exigindo técnicas como encadeamento ou endereçamento aberto.

## 5 Exemplo de Colisão e Resolução

Suponha que as chaves “usuario:1001” e “produto:2002” resultem no mesmo índice após o hash. O Redis e outros motores usam técnicas como listas encadeadas para armazenar múltiplos valores no mesmo índice, garantindo que nenhum dado seja perdido.

extbfExemplo:

```
hash("usuario:1001") = 42
hash("produto:2002") = 42
// Ambos armazenados na posição 42, mas em listas separadas.
```

## 6 Redis como Motor Baseado em Hashing

O Redis utiliza tabelas hash para armazenar dados em memória, permitindo operações extremamente rápidas. Além disso, oferece estruturas de dados avançadas, como hashes, listas, conjuntos e sorted sets, mas o mecanismo de acesso principal é sempre baseado em hashing.

No Redis, cada chave é processada por uma função hash interna, que determina onde o valor será armazenado na memória. Isso garante que operações como SET, GET, HSET e HGET sejam realizadas de forma eficiente, mesmo com milhões de registros.

### Exemplo de uso de Hashes no Redis

Hashes no Redis permitem armazenar múltiplos campos e valores sob uma mesma chave, ideal para representar objetos:

```
HSET usuario:1001 nome "Ana" idade 25 cidade "Recife"
HGET usuario:1001 nome
HGETALL usuario:1001
```

extbfExemplo de uso em Python:

```
import redis
r = redis.Redis()
r.hset('usuario:1001', mapping={'nome': 'Ana', 'idade': 25, 'cidade': 'Recife'})
print(r.hgetall('usuario:1001'))
```

## Exemplo de uso como cache

```
// Armazenar resultado de consulta
SET resultado:consulta:123 "valor"
// Recuperar do cache
GET resultado:consulta:123
```

extbfAplicações típicas:

- Cache de páginas e resultados de consultas.
- Armazenamento de sessões de usuário.
- Contadores e rankings em tempo real.
- Fila de tarefas e sistemas de mensagens.

## Aplicações reais

- **Twitter:** Armazena timelines e contadores de seguidores.
- **GitHub:** Utiliza Redis para cache de páginas e sessões.
- **Stack Overflow:** Usa Redis para filas de tarefas e contadores de visualizações.

Assim, motores baseados em hashing como o Redis são fundamentais para aplicações modernas que exigem alta disponibilidade, baixa latência e escalabilidade.

## Imagens Ilustrativas e Explicações

extbfDescrição: Cada hash no Redis é um dicionário de campos e valores, ideal para armazenar objetos como perfis de usuário. As operações de leitura e escrita são extremamente rápidas (complexidade  $O(1)$ ).

extbfDescrição: O Redis é frequentemente utilizado como cache para acelerar o acesso a dados, reduzindo a carga sobre bancos de dados relacionais como MySQL.

extbfDescrição: O Redis suporta múltiplos tipos de dados, tornando-o versátil para diferentes aplicações, como contadores, filas, rankings e mais.

extbfDescrição: As tabelas acima trazem uma referência rápida dos principais comandos Redis para manipulação de strings, listas, conjuntos, hashes, scripts e operações de banco de dados.

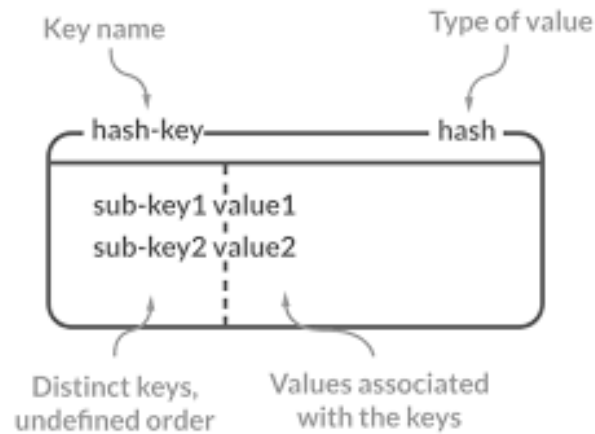


Figure 1: Exemplo visual de uma estrutura hash no Redis, mostrando chave, sub-chaves e valores.

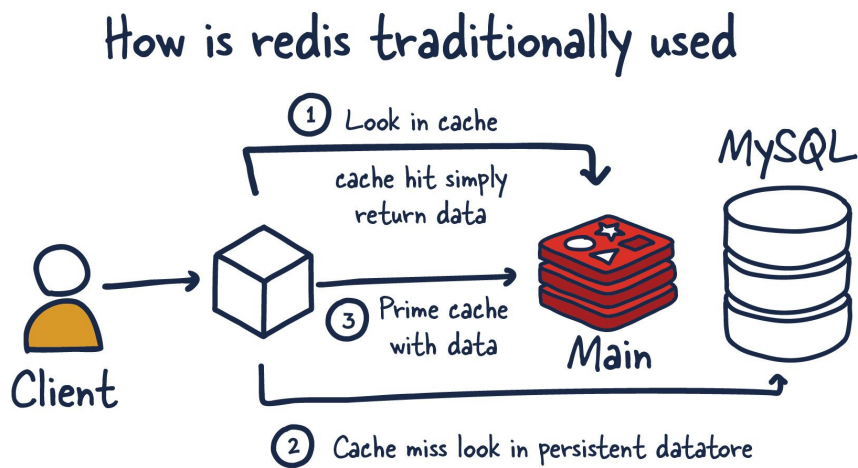


Figure 2: Fluxo tradicional de uso do Redis como cache entre cliente e banco de dados principal.

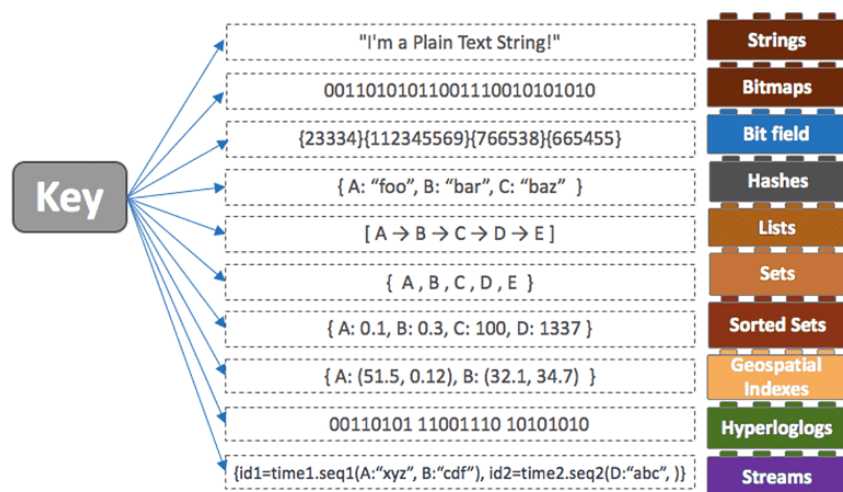



Figure 3: Diversos tipos de dados que podem ser armazenados no Redis, como strings, listas, hashes, sets, etc.

		<b>Redis Cheat Sheet</b> by James Hopkin (tasjaevan) via <a href="https://cheatography.com/18964/cs/2046/">cheatography.com/18964/cs/2046/</a>	
<b>Strings</b>		<b>Lists (cont)</b>	
APPEND	Append	LPUSH	Push onto start
BITCOUNT	Count set bits	LPUSHX	Push if list exists
BITOP	Bitwise operations	LRANGE	Access range
BITPOS	Find first set bit	LREM	Remove
DECR	Decrement integer	LSET	Set item by index
DECRBY	Subtract from integer	LTRIM	Remove start and/or end items
GET	Get by key	RPOP	Pop from end
GETBIT	Get bit by index	RPOPLPUSH	Rotate
GETRANGE	Get substring	RPUSH	Push onto end
GETSET	Set, returning old value	RPUSHX	Push onto end if list exists
INCR	Increment integer	<b>Client/Server</b>	
INCRBY	Add to integer	AUTH	Request authentication
INCRBYFLOAT	Add to float	ECHO	Return message
MGET	Get multiple	PING	Test connection
MSET	Set multiple	QUIT	Close connection
MSETNX	Set multiple if don't exist	SELECT	Set current database by index
PSETEX	Set with expiry (ms)	<b>Sets</b>	
SET	Set	SADD	Add item
SETBIT	Set bit by index	SCARD	Get size
SETEX	Set with expiry (seconds)	SDIFF	Get difference
SETNX	Set if doesn't exist	SDIFFSTORE	Store difference
SETRANGE	Set substring	SINTER	Intersection
STRLEN	Get length	SINTERSTORE	Store intersection
Strings can be used as numbers, arrays, bit sets and binary data		SISMEMBER	Check for item
<b>Lists</b>		SMEMBERS	Get all
BLPOP	Blocking left pop	SMOVE	Move item to another set
BRPOP	Blocking right pop	SPOP	Pop random item
BRPOPLPUSH	Blocking rotate	SRANDMEMBER	Get random item
LINDEX	Access by index	SREM	Remove matching
LINSERT	Insert next to	SSCAN	Iterate items
LLEN	Get length	SUNION	Union
LPOP	Pop from start	SUNIONSTORE	Store union
<b>Database</b>		<b>Database</b>	
DEL	Delete item	DEL	Delete item
DUMP	Serialise item	DUMP	Serialise item
EXISTS	Check for key	EXISTS	Check for key
EXPIRE	Set timeout on item	EXPIRE	Set timeout on item
EXPIREAT	Set timeout by timestamp	EXPIREAT	Set timeout by timestamp
KEYS	Get all keys matching pattern	KEYS	Get all keys matching pattern
MIGRATE	Transfer an item between Redis instances	MIGRATE	Transfer an item between Redis instances
MOVE	Transfer an item between databases	MOVE	Transfer an item between databases
OBJECT	Inspect item	OBJECT	Inspect item
PERSIST	Remove timeout	PERSIST	Remove timeout
PEXPIRE	Set timeout (ms)	PEXPIRE	Set timeout (ms)
PEXPIREAT	Set timeout (ms timestamp)	PEXPIREAT	Set timeout (ms timestamp)
PTTL	Get item time to live (ms)	PTTL	Get item time to live (ms)
RANDOMKEY	Get random key	RANDOMKEY	Get random key
RENAME	Change item's key	RENAME	Change item's key
RENAMENX	Change item's key if new key doesn't exist	RENAMENX	Change item's key if new key doesn't exist
RESTORE	Deserialize	RESTORE	Deserialize
SCAN	Iterate keys	SCAN	Iterate keys
SORT	Get or store sorted copy of list, set or sorted set	SORT	Get or store sorted copy of list, set or sorted set
TTL	Get item time to live	TTL	Get item time to live
TYPE	Get type of item	TYPE	Get type of item



By **James Hopkin** (tasjaevan)  
[cheatography.com/tasjaevan/](https://cheatography.com/tasjaevan/)

Published 8th May, 2014.  
 Last updated 12th May, 2016.  
 Page 1 of 2.

Sponsored by **ApolloPad.com**  
 Everyone has a novel in them. Finish Yours!  
<https://apollopadd.com>

Figure 4: Cheatsheet de comandos Redis (página 1).

## 7 O que é Redis?

Redis é um banco de dados em memória, open-source, que utiliza uma estrutura de dados baseada em hashing para armazenar e recuperar informações de forma extremamente rápida.

Scripts	
EVAL	Run
EVALSHA	Run cached
SCRIPT EXISTS	Check by hash
SCRIPT FLUSH	Clear cache
SCRIPT KILL	Kill running script
SCRIPT LOAD	Add to cache
Lua scripts access keys through the array KEYS and additional arguments through the array ARGV.	

Hashes	
HDEL	Delete item
HEXISTS	Check for item
HGET	Get item
HGETALL	Return all items
HINCRBY	Add to integer value
HINCRBYFLOAT	Add to float value
HKEYS	Return all keys
HLEN	Get number of items
HMGET	Get multiple items
HMSET	Set multiple items
HSCAN	Iterate items
HSET	Set item
HSETNX	Set item if doesn't exist
HVALS	Return all values

Sorted sets	
ZADD	Add item
ZCARD	Get number of items
ZCOUNT	Number of items within score range
ZINCRBY	Add to score
ZINTERSTORE	Store intersection
ZLEXCOUNT	Lexicographical range count
ZRANGE	Get items within rank range

Sorted sets (cont)	
ZLEXRANGE	Get items within lexicographical range
ZRANGEBYSCORE	Get items within score range
ZRANK	Get item rank
ZREM	Remove item(s)
ZREMRANGEBYLEX	Remove items within lexicographical range
ZREMRANGEBYRANK	Remove items within rank range
ZREMRANGEBYSCORE	Remove items within score range
ZREVRANGE	ZRANGE in reverse order
ZREVRANGEBYSCORE	ZRANGEBYSCORE in reverse order
ZREVRANK	ZRANK in reverse order
ZSCAN	Iterate items
ZSCORE	Get item score
ZUNIONSTORE	Store union
Lexicographical commands require all items to have the same score	

HyperLogLogs	
PFADD	Add items
PFCOUNT	Get approximate size
PFMERGE	Merge HyperLogLogs



By **James Hopkin** (tasjaevan)  
[cheatography.com/tasjaevan/](https://cheatography.com/tasjaevan/)

Published 8th May, 2014.  
 Last updated 12th May, 2016.  
 Page 2 of 2.

Sponsored by **ApolloPad.com**  
 Everyone has a novel in them. Finish Yours!  
<https://apollopad.com>

Figure 5: Cheatsheet de comandos Redis (página 2).

O Redis é amplamente utilizado em sistemas que exigem alta performance, como cache de páginas web, sistemas de filas, contadores de acessos, sessões de usuário e até mesmo como banco de dados principal para aplicações que priorizam velocidade.

Principais características:

- Armazenamento em memória (RAM), com persistência opcional em disco.

- Estruturas de dados avançadas: strings, hashes, listas, conjuntos, sorted sets, bitmaps, hyperloglogs, streams, entre outros.
- Suporte a operações atômicas e transações.
- Utilizado como cache, fila, broker de mensagens, ranking, contadores, etc.

Curiosidade: O nome “Redis” vem de “REmote DIctionary Server”.

extbfDica: O Redis pode ser configurado para persistir dados em disco, garantindo durabilidade mesmo após reinicializações.

## 8 Exemplo de Estrutura Hash

No Redis, um hash é ideal para representar objetos, como um perfil de usuário:

```
user:1001 => {
  nome: "Ana",
  idade: 25,
  cidade: "Recife"
}
```

## 9 Comandos Básicos Redis

Os comandos abaixo são essenciais para manipulação de dados no Redis:

- SET chave valor — Armazena um valor simples.
- GET chave — Recupera o valor de uma chave.
- HSET hash campo valor — Armazena um campo em um hash.
- HGET hash campo — Recupera o valor de um campo em um hash.

Exemplo:

```
SET contador 1
INCR contador
GET contador % 2
```

## 10 Exemplo prático de Hash

O comando HSET permite adicionar múltiplos campos a um hash de uma só vez. O HGETALL retorna todos os campos e valores associados à chave hash.

```
HSET usuario:1001 nome "Ana" idade 25 cidade "Recife"
HGETALL usuario:1001
% Retorno: nome => Ana, idade => 25, cidade => Recife
```

## 11 Vantagens do Redis

Além das vantagens listadas, o Redis possui suporte a replicação (master-slave), alta disponibilidade (Redis Sentinel) e particionamento de dados (Redis Cluster), facilitando a escalabilidade horizontal.

- Velocidade extrema: Todas as operações são feitas em memória RAM.
- Baixa latência: Ideal para aplicações que exigem resposta em milissegundos.
- Versatilidade: Suporta múltiplas estruturas de dados.
- Persistência opcional: Pode gravar snapshots (RDB) ou log de operações (AOF) para garantir durabilidade.
- Escalabilidade horizontal: Suporte a cluster e replicação.
- Comunidade ativa: Muitas integrações e ferramentas.

## 12 Comparativo

Banco	Tipo de Estrutura	Latência	Persistência	Uso comum
Redis	Hashing	Baixíssima	Opcional	Cache, filas
MySQL/InnoDB	B-Tree	Média	Sim	Relacional
DynamoDB	Hashing	Baixa	Sim	NoSQL distribuído

## 13 Exemplo de uso em fila

O Redis pode ser utilizado como uma fila de mensagens, permitindo a implementação de sistemas assíncronos e desacoplados.

```
LPUSH fila:emails "email1@example.com"
LPUSH fila:emails "email2@example.com"
RPOP fila:emails % Remove e retorna o último email inserido
```

## 14 Links úteis

O Redis é utilizado por empresas como Twitter, GitHub, Stack Overflow e Snapchat para sistemas de ranking, contadores de visualizações, sessões de usuário, filas de tarefas, pub/sub, entre outros.

- <https://redis.io/>
- <https://try.redis.io/>
- <https://redis.io/docs/data-types/strings/>
- <https://redislabs.com/redisinsight/>