

Banco de Dados NoSQL

(NS0202)

Prof. Giovane Barcelos
giovane_barcelos@uniritter.edu.br

NoSQL - Redis

Conteúdo Programático

- 1. Introdução ao Redis**
- 2. Primeiros Passos**
- 3. Tipos de Dados**
- 4. Persistência de Dados**
- 5. Aplicação prática com Node.js**

2. Primeiros Passos

Vídeo para Instalação do Redis no Windows

http://bit.do/YouGO_InstalacaoRedisWindows

2. Primeiros Passos

Vídeo para Instalação do Redis no Linux

http://bit.do/YouGO_InstalacaoRedisLinux

2. Primeiros Passos

Vamos treinar com alguns comandos no *redis-cli*

ping
echo "ola"
SET foo 100
GET foo
INCR foo
DECR foo
SET bar 200
EXISTS bar
DEL bar
EXISTS bar

redis-cli ECHO Oi
redis-cli GET foo
redis-cli INCR foo > commands.txt

monitor

FLUSHALL
GET foo

SET server:name myserver
GET server:name
SET server:port 6379

SET resource:foo ola
EXPIRE resource:foo 120
TTL resource:foo
SET resource:foo ola
TTL resource:foo

2. Primeiros Passos

Quais são os comandos básicos?

- **SET** – Define uma variável String
- **GET** – Obtém o valor de uma variável
- **DEL** – Exclui uma variável
- **EXISTS** – Verifica se existe uma variável
- **INCR** – Aumenta em 1
- **INCRBY** – Incrementa por quantidade definida
- **DECR** – Diminui em 1
- **DECRBY** – Diminuição por quantidade especificada

2. Primeiros Passos

MSET

- **Define várias chaves para seus respectivos valores**
- **Substitui valores existentes por novos**

MSET key1 “**val1**” key2 “**val2**”

2. Primeiros Passos

MSETNX

- Define várias chaves para os respectivos valores desde que não exista nenhuma das chaves
- NÃO substituirá os valores existentes
- NÃO executará se existir uma única chave cadastrada

MSETNX key1 “**val1**” key2 “**val2**”

2. Primeiros Passos

MGET

- **Retorna valores de todas as chaves especificadas**
- **Null é retornado se a chave não tiver um valor**

MGET key1 key2

2. Primeiros Passos

APPEND

- Se a chave já existe e é uma *string* o valor será anexado no final da *string*
- Se a chave NÃO existir. Funciona como SET

APPEND mykey “acrescentarstring”

2. Primeiros Passos

GETRANGE

- Retorna a *substring* de um valor de *string* determinado por início e fim
- Os deslocamentos negativos podem ser usados para começar a partir do final da *string*

GETRANGE mykey 0 -1

2. Primeiros Passos

RENAME

- **Renomeia uma chave**
- **Retorna erro se a chave não existir**
- **Se existir substitui a chave**

RENAME mykey myrenamedkey

2. Primeiros Passos

RENAMENX

- Renomeia a chave para *novachave* se a *novachave* não existir
- Retorna um erro se a chave não existir

RENAMENX mykey myrenamedkey

2. Primeiros Passos

GETSET

- Define automaticamente uma chave para um valor e retorna o valor antigo
- Pode ser usado com o *INCR* para contar com reinicialização automática

GETSET mykey “myval”

2. Primeiros Passos

SETEX

- Define para uma chave manter um valor de *string* *num* tempo limite após uma determinada quantidade de segundos

SETEX mykey 10 “oi”

equivalente a ...

SET mykey “oi”

EXPIRE mykey 10

2. Primeiros Passos

PSETEX

- O mesmo que o SETEX, exceto que ele usa milissegundos em vez de segundos

PSETEX mykey 6000 “oi”

PTTL é usado para obter o tempo restante em milissegundos

2. Primeiros Passos

PERSIST

- **Remove o tempo limite disponível em uma chave**

PERSIST mykey

2. Primeiros Passos

SETNX

- **Funciona como SET se a chave NÃO existir**
- **Se a chave já existe, não mudará o valor**

SETNX mykey “oi”

2. Primeiros Passos

Vamos praticar um pouco mais ...

MSET key1 "Olá" key2 "Mundo"

MGET key1 key2

MSETNX key3 "oi"

MSETNX key3 "olá" key4 "aqui"

SET greeting "Olá"

GET greeting

APPEND greeting " Mundo"

GET greeting

APPEND foo "bar"

GET foo

RENAME greeting greet

GET greet

RENAMENX greet greeting

RENAMENX key1 greting

SET mystring "Minha string"

GETRANGE mystr 0 -1

GETRANGE mystr 0 5

GETRANGE mystr 3 8

GETSET foo

GET foo

SET mykey "olá"

EXPIRE mykey 10

TTL mykey

SETEX mykey 10 "olá"

PSETEX mykey 1000 "olá"

PTTL mykey

SETEX mykey 120 "olá"

PERSIST mykey

SETNX newkey "foobar"

GET foobar

SETNX newkey "barfoo"

2. Primeiros Passos

Questões

➤ **QuestionarioRedis02**

2. Primeiros Passos

Como funciona a pesquisa *SCAN* no Redis?

- **Itera sobre um conjunto de chaves no banco de dados**
- **Retorna apenas uma pequena parcela de registros por chamada**
- **Pode ser usado em produção**
- **Toma o cursor / posição como um parâmetro**



2. Primeiros Passos

Iterator baseado em cursor com **SCAN**

- O servidor retorna um **cursor** atualizado com **cada chamada** do comando
- Isso pode ser usado no argumento da próxima chamada
- A iteração é iniciada quando o cursor está configurado para 0 (zero)
- Termina quando o cursor retorna do servidor 0 (zero)
- O primeiro "1)" valor retornado no exemplo ("17") é o próximo cursor e o segundo grupo é um *array* com as chaves

```
redis 127.0.0.1:6379> scan 0
```

```
1) "17"  
2) 1) "key:12"  
   2) "key:8"  
   3) "key:4"  
   4) "key:14"  
   5) "key:16"  
   6) "key:17"  
   7) "key:15"  
   8) "key:10"  
   9) "key:3"  
  10) "key:7"  
  11) "key:1"
```

```
redis 127.0.0.1:6379> scan 17
```

```
1) "0"  
2) 1) "key:5"  
   2) "key:18"  
   3) "key:0"  
   4) "key:2"  
   5) "key:19"  
   6) "key:13"  
   7) "key:6"  
   8) "key:9"  
   9) "key:11"
```

2. Primeiros Passos

Garantias do SCAN

- **As iterações completas irão recuperar todos os elementos que estão presentes na coleção do início até o fim**
- **Nunca retorna nenhum elemento que NÃO esteja presente na coleção do início ao fim**

2. Primeiros Passos

Opção *COUNT*

- ***COUNT* pode ser definido em um comando *SCAN* para substituir o padrão retornado por iteração**
- **O usuário pode especificar a quantidade de trabalho realizado em cada chamada**
- **O *COUNT* padrão é 10**
- ***COUNT* pode ser alterado de uma iteração para a próxima**

***SCAN* 0 *COUNT* 20**

2. Primeiros Passos

Opção *MATCH*

- **Itera elementos que correspondem a um padrão especificado**

SCAN 0 MATCH alguma coisa
SCAN 0 MATCH k*

2. Primeiros Passos

SCAN com outros tipos de dados

- **SSCAN**: usado com conjuntos. Retorna uma lista com os membros do *set* (Ex: `sscan myset 0 match f*`)
- **HSCAN**: usado com *hashes*. Retorna uma matriz de elementos com campo e valor (Ex: `hscan hash 0`)
- **ZSCAN**: Usado com conjuntos ordenados. Retorna uma matriz de elementos com pontuação associada (Ex: `zscan mytest 0`)

2. Primeiros Passos

Padrões para KEYS

- **Retorna todas as chaves que correspondem a um padrão específico**
- **Deve ser evitado em ambientes de produção**
- **Suporta padrões com expressões regulares**

KEYS *

2. Primeiros Passos

RANDOMKEY

- **Retorna uma chave aleatória do banco de dados**

RANDOMKEY

2. Primeiros Passos

Vamos praticar um pouco de pesquisa

MSET key1 "1" key2 "2" key3 "3" key4 "4"
key5 "5" key6 "6" key7 "7" key8 "8"
key9 "9" key10 "10" key11 "11"
key12 "12" key13 "13"

SCAN 0

SCAN 13

SCAN 0 **COUNT** 5

SCAN **COUNT** 100

SCAN 0 **MATCH** key1*

SCAN 0 **MATCH** key1* **COUNT** 2

KEYS *

KEYS key1*

RANDOMKEY

2. Primeiros Passos

CONFIG GET

- Usado para ler os parâmetros de configuração de um servidor *Redis* em execução
- Desde o Redis 2.6 todos os parâmetros de configuração são suportados
- Apresenta tudo em um único argumento

CONFIG GET port : obtém a configuração da porta

CONFIG GET * : lista todos os parâmetros

CONFIG GET *max-*-entries*

2. Primeiros Passos

CONFIG SET

- **Usado para reconfigurar o servidor em tempo de execução sem ter que fazer um reinício**

CONFIG SET configoption "novovalor"

2. Primeiros Passos

INFO

- **Retorna informações e estatísticas sobre um servidor**
- **Parâmetro opcional para selecionar a seção específica de informação**
- **server | clients | memory | persistence | stats | replication | cpu | commandstats | cluster | keyspace | all | default**

INFO

INFO server

2. Primeiros Passos

CONFIG RESETSTAT

- **Reseta as estatísticas reportadas usando o comando INFO**
 - ✓ **Número de comandos processados**
 - ✓ **Número de conexões recebidas**
 - ✓ **Número de chaves expiradas**
 - ✓ **Número de conexões rejeitadas**

CONFIG RESETSTAT

2. Primeiros Passos

COMMAND

- Retorna detalhes sobre todos os comandos do Redis
- Cada resultado de nível superior contém 6 sub resultados:
 1. Nome do comando
 2. Especificação *arity*: número de parâmetros
 3. Sinalizadores / assinatura do comando em um *array*
 4. Posição da primeira chave na lista de argumentos
 5. Posição da última chave na lista de argumentos
 6. Contagem de etapas para localizar ou repetir chaves

COMMAND

2. Primeiros Passos

COMMAND INFO

- **Retorna detalhes para um comando específico**
- **O mesmo que *COMMAND*, mas com foco em um comando específico**

COMMAND INFO GET

2. Primeiros Passos

COMMAND COUNT

- **Retorna o número de comandos disponíveis no servidor**

COMMAND COUNT

2. Primeiros Passos

CLIENT LIST

- **Retorna informações e estatísticas dos clientes conectados a um servidor**
 - ✓ **Identificação (*Id*)**
 - ✓ **Nome (*Name*)**
 - ✓ **Sinalizadores (*Flags*)**
 - ✓ **Tempo de vida (*Age*)**
 - ✓ **Endereço / Porta (*Address/Port*)**
 - ✓ **Último comando**
 - ✓ **Tempo ocioso (*Idle*)**

CLIENT LIST

2. Primeiros Passos

CLIENT SETNAME

- **Atribui um nome a conexão do cliente atual**
- **Exibido na saída de CLIENT LIST**

CLIENT SETNAME nomeDoCliente

2. Primeiros Passos

CLIENT GETNAME

- **Retorna o nome da conexão atual do cliente**
- **Retorna nulo se nenhum nome estiver configurado**

CLIENT GETNAME

2. Primeiros Passos

CLIENT KILL

- Fecha uma conexão específica
- Pode usar o endereço / porta ou ID

CLIENT KILL 127.0.0.1:numPorta

CLIENT KILL id <numeroid>

2. Primeiros Passos

Vamos praticar um pouco (*Client e Config*)

CONFIG GET port

CONFIG GET *

CONFIG GET *max-*-entries*

CONFIG SET lua-time-limit 6000

INFO

INFO cpu

COMMAND

COMMAND INFO GET

COMMAND COUNT

CLIENT LIST

CLIENT SETNAME foo

CLIENT SET NAME bar

CLIENT GETNAME

CLIENT KILL 127.0.0.1:xxxxxx

Lembre-se

“Que a força esteja contigo.”

Star Wars

