Análise e Organização de Informações com Banco de Dados Redis

Otávio A. S. Silva[1], **Rafael N. Freitas**[1], **Marcos P. Andrade**[1]

[1]Instituto Federal do Sul de Minas Gerais - Campus Passos
Rua da Penha, 394 - Penha II, Passos-MG, 37903-358

{otavio.silva,rafael.nfreitas,marcos.andrade}@alunos.ifsuldeminas.edu.br

Abstract. *The main characteristic of NoSQL databases is that they have great performance, being very useful when dealing with a large volume of data. In this context, Redis is a key-value based NoSQL bank that efficiently performs reading and writing operations. This system is divided into two parts of storage, which are in memory and on disk, and its executable client can be installed on both Linux and Windows. In addition, Redis can be used in different applications due to the ease of integrating it with different programming languages. From this, this work describes the main features of this bank, and exposes an example of a real scenario with the PHP language.*

Resumo. *A principal caracteristica dos bancos de dados NoSQL é o fato de possuirem grande desempenho, sendo bastante úteis quando se trata de um grande volume de dados. Nesse contexto, o Redis é um banco NoSQL baseado em chave-valor que realiza de forma eficiente as operações de leitura e escrita. Esse sistema é dividido em duas partes de armazenamento, que são em memória e em disco, e o seu cliente executável pode ser instalado tanto no Linux quanto no Windows. Além disso, o Redis pode ser utilizado em diferentes aplicações devido a facilidade de integrá-lo com diversas linguagens de programação. A partir disso, este trabalho descreve as principais funcionalidade deste banco, e expõe um exemplo de cenário real com a lingaugem PHP.*

1. Introdução

O Redis (Remote Dictionary Service) é um banco de dados NoSQL (Não somente SQL) criado por Salvatore Sanfilippo, sendo lançado no ano de 2009. De acordo com [P. J. SAMPAIO 2015], Salvatore trabalhava com um sistema de análise de dados em tempo real e com isso era necessário realizar operações rápidas de leitura e escrita. O grande problema era que nenhum banco de dados mantinha um desempenho satisfatório devido a grande quantidade de dados que ele utilizava. Dessa forma, coube à ele desenvolver um banco que atendesse suas necessidades da forma mais rápida possível.

Conforme [WALKER-MORGAN 2010], o Redis pode ser descrito como um sistema de armazenamento em cache baseado em chave-valor e de código aberto. Normalmente, esse tipo de sistema pode ser dividido em dois grupos de armazenamento, que são em memória e em disco. Nesse sentido, o armazenamento em memória é o que garante o alto desempenho, enquanto o em disco garante que o dado não se perca em uma eventual queda de servidor, já que a memória é volátil.

Ademais, de acordo com a empresa [AMAZON 2020], o alto desempenho na manipulação de dados faz com que o banco em questão seja utilizado em diversas

aplicações de tempo real. Entre essas, destacam-se armazenamento em cache, chats e sistemas de mensagens, placares de jogos, armazenamento de sessões, dados espaciais etc.

Vale salientar ainda, que grandes empresas fazem uso deste sistema para alguma dessas aplicações. Nesse contexto, segundo o site oficial do [REDISLABS 2020], predominam-se algumas do ramo de TI (Tecnologia da Informação) como Twitter, GitHub, Snapchat, StackOverflow, Instagram etc.

Com relação a disponibilidade, o Redis Labs é distribuidor oficial e o suporte existe somente para o sistema operacional Linux. Apesar disso, segundo [BEMFICA 2018], o grupo de desenvolvedores Microsoft Open Tech mantém uma versão compatível com o Windows 64-bits.

1.1. Objetivos

Relatar as principais caracteristicas e funcionalidade do Redis, bem como demonstrar como é feita a instalação do sistema. Em seguida, apresentar um cenário real sobre o COVID-19 em Minas Gerais, em que os dados são armazenados no banco com a linguagem PHP, e posteriormente exibidos em uma página WEB.

2. Fundamentação Teórica

Como foi dito, o Redis faz uso de armazenamento baseado em estruturas chave-valor. Assim, [P. J. SAMPAIO 2015] pontuam que o fato da maior parte do seu processamento ocorrer na memória, facilita o acesso a essas estruturas, garantindo alta velocidade na manipulação dos dados. Nesse viés, [CARLSON 2013] relata que essa estrutura de dados resolve grande parte dos problemas de armazenamento sem a necessidade de mais implantações, como ocorre em outros bancos de dados.

Entre os tipos de dados que o banco suporta, de acordo com a própria documentação, destacam-se caracteres, hashes, listas, listas de vetores, geolocalização, transações, mapas, publicação e subscrição. A partir dessas estruturas é possível realizar operações como por exemplo inserção, remoção, consulta etc.

É importante pontuar também, que o Redis pode ser codificado em diversas linguagens de programação. Dessa forma, é possível fazer a integração do banco com linguagens bastante conhecidas como C, C++, Java, Lua, PHP, Python, etc.

3. Detalhes da Instalação

Como o Windows já estava disponível no momento, optou-se por utilizar e instalar a versão disponibilizada pelo grupo Microsoft Open Tech no GitHub (Figure 1). Assim, com excessão das políticas de acesso que foram feitas em um provedor de hospedagem online, todos os testes e operações aqui exemplificadas se deram por meio de um cliente no Windows.

Após a instalação, executou-se o servidor do Redis para que o cliente (Figure 2) estivesse pronto para uso. Desse modo, poderiam ser feitos testes com as principais operações de manipulação deste banco de dados, que são 'set', 'get', 'exists', 'del', 'rpush' e 'hmset'.

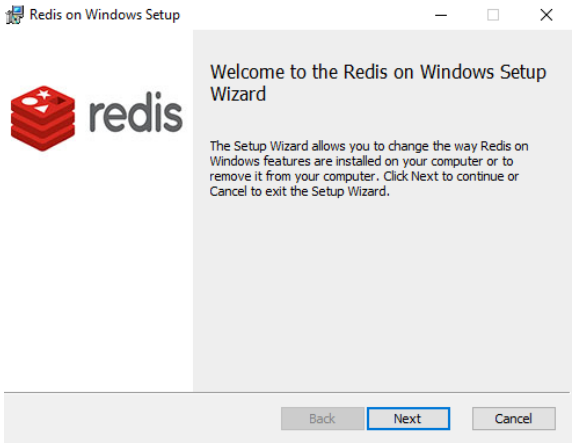

Figura 1. Instalador

O ’set’ tem a função de inserir uma chave-valor, enquanto o ’get’ faz uma espécie de consulta ao valor de uma determinada chave. Por outro lado, a função ’exists’ verifica se determinada chave está contida no banco e a função ’del’ deleta uma chave, sendo que ambas retornam 1 para verdadeiro e 0 para falso.

Ademais, existem as estruturas conhecidas como lista e hash. O comando rpush tem a função de criar uma lista em que uma chave possui vários valores do mesmo tipo. Já o comando ’hmset’ consiste em criar uma hash que armazene valores diferentes que se relacionam à uma mesma chave. No cliente, para exibir os dados de uma lista e de uma hash é necessário executar os comandos ’lrange’ e ’hget’, respectivamente.

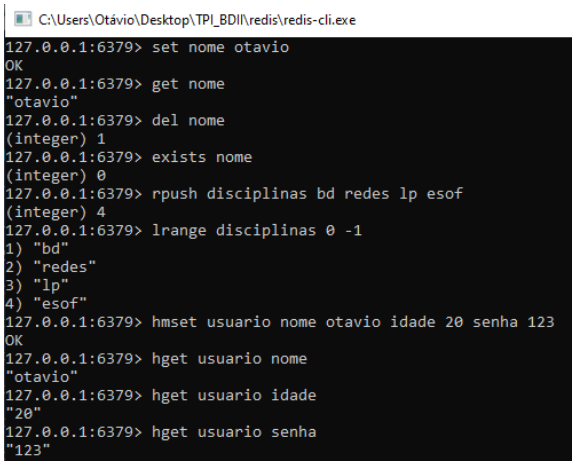

Figura 2. Cliente Redis

Outrossim, é possível implementar políticas de acesso de usuários e controles de transações no Redis. Essas políticas se dão por meio de comandos ACL (Access Control Lists), que são iniciados juntamente com servidor do sistema.

Entre esses comandos, o ’setuser’ é responsável por criar um usuario com sua respectiva senha, definir quais chaves podem ser acessadas por este e quais operações podem ser feitas com essas respectivas chaves. Ressalta-se, que é necessário antes das operações, autenticar o usuário por meio do comando ’AUTH’.

Destaca-se também, que o uso de ACL está disponível apenas a partir da versão 6.0 do Redis, e por isso não é compatível com o Windows, já que a versão para este sistema é a 3.2. Apesar disso, foi possível instalar o Redis na distribuição Kali Linux disponibilizada pelo [ONWORKS 2020], que é um provedor de hospedagem online para testes de distribuições Linux, e assim implementar as políticas de acesso. Todos os comandos de instalação (Figure 3) foram executados no terminal do sistema, de acordo com a documentação do [REDISLABS 2020].

```
$ wget http://download.redis.io/releases/redis-6.0.3.tar.gz
$ tar xzf redis-6.0.3.tar.gz
$ cd redis-6.0.3
$ make
```

Figura 3. Comandos Linux

No exemplo a seguir (Figure 4) feito no Kali Linux do OnWorks, o usuário 'otavio' possui permissões para realizar operações de 'get' e 'set' em todas as chaves do banco, enquanto o usuário 'rafael' possui permissão apenas para a operação 'get'. No contexto em questão, foi feito uma operação em que o usuário 'rafael' não possuia permissão, e com isso emitiu-se uma mensagem de erro.

```
127.0.0.1:6379> acl setuser otavio on >otavio123 nocommands +get +set allke
ys
OK
127.0.0.1:6379> acl setuser rafael on >rafael123 nocommands +get allkeys
OK
127.0.0.1:6379> auth otavio otavio123
OK
127.0.0.1:6379> set idade 20
OK
127.0.0.1:6379> auth rafael rafael123
OK
127.0.0.1:6379> set idade 25
(error) NOPERM this user has no permissions to run the 'set' command or its
 subcommand
127.0.0.1:6379> get idade
"20"
127.0.0.1:6379>
```

Figura 4. Controle de Usuários

Para finalizar o processo de instalação, integrou-se o Redis com a linguagem de programação PHP (Figure 5). Para testar o funcionamento, como o sistema operacional em execução era o Windows, se fez necessário ativar a extensão php-redis junto ao arquivo php.ini e adicionar o arquivo php-redis.dll junto a pasta principal do Php no WAMP.

```php
<?php

  $host = 'localhost';
  $port = 6379;
    /*abre conexao com redis*/
  $redis = new Redis();
  if ($redis->connect($host, $port) == false) {
    die($redis->getLastError());
  }

?>
```

Figura 5. Conexão Redis com PHP

4. Implementação do Cenário Proposto

O grande sucesso do Redis é sem dúvida o fato de apresentar alta velocidade e desempenho na manipulação dos dados. Por essa razão, este banco é muito utilizado em aplicações

de tempo real que possuem uma grande quantidade de dados. Pensando nisso, optou-se por implementar um cenário em que o Redis armazenasse dados do Covid-19 de todas as cidades de Minas Gerais, para que posteriomente fossem disponibilizado por meio de uma página PHP.

Inicialmente, realizou-se a conexão do Redis em PHP levando em conta que a extensão php-redis já estava instalada e configurada. Após isso, requisitou-se uma API que retorna um JSON que continha todos os dados necessários (Figure 6). Essa API é disponibilizada de forma gratuita pelo site brasil.io, que oferece diferentes tipos de dados públicos para serem exibidos conforme o desenvolvedor desejar, desde que a plataforma seja citada na aplicação.

```php
//capturando dados json
$filename = "https://brasil.io/api/dataset/covid19/caso
/data/?format=json&is_last=True&state=MG";
$content = file_get_contents($filename);
$jsonObj = json_decode($content);
$results = $jsonObj->results;
```

Figura 6. Coleta de dados com JSON

Em seguida, operou-se o comando 'hmset' para inserir os dados de todas as cidades capturada com JSON (Figure 7). Nesse âmbito, a variável 'count' é responsável por controlar o laço de repetição de acordo com a quantidade cidades.

Pontua-se ainda, que nem todos os dados do JSON foram aproveitados. Nesse sentido, inseriu-se a população geral, casos confirmados, mortes, taxa de mortalidade e data da última atualização de cada boletim municipal.

```php
//setando no banco redis
for ($i=0;$i<$jsonObj->count;$i++){
    if ($results[$i]->city!=NULL){
        $redis->set($results[$i]->city, $i); //utilizado na pesquisa
        $redis->hmset($i,
        [
            'city' => $results[$i]->city,
            'population' => $results[$i]->estimated_population_2019,
            'confirmed' => $results[$i]->confirmed,
            'deaths' => $results[$i]->deaths,
            'death_rate' => $results[$i]->death_rate,
            'date' => $results[$i]->date,
        ]);
    }else{
        $redis->set('pgeral', $results[$i]->estimated_population_2019);
        $redis->set('casos', $results[$i]->confirmed);
        $redis->set('mortes', $results[$i]->deaths);
    }
}
```

Figura 7. Inserção de dados no Redis

Para exibição do valor das chaves, bastou consultá-los individualmente por meio do comando 'hget' e inseri-los em uma tabela html (Figure 8). Neste caso, como se tratava de vários valores, se fez necessário um laço de repetição considerando a quantidade total de cidades.

Ressalta-se por fim, que a página de apresentação pode ser estilizada (Figure 9) com o objetivo de melhorar a exibição dos dados. O código fonte da página exemplo pode ser visualizado através do link no GitHub.

```
while ($i<$redis->get('count')){
    echo '<tr class="conteudo"><td>'.$redis->hget($i, 'city').'</td>'; //nome cidade
    echo '<td><font color="green">'.$redis->hget($i, 'population').'</font></td>';
    echo '<td><font color="orange">'.$redis->hget($i, 'confirmed').'</font></td>';
    echo '<td><font color="red">'.$redis->hget($i, 'deaths').'</font></td>'; //morte
    echo '<td><font color="red">'.$redis->hget($i, 'death_rate').'</font></td>'; //t
    echo '<td><font color="yellow">'.$redis->hget($i, 'date').'</font></td>'; //data
    $i++;
}
```

Figura 8. Exibição do valor das chaves

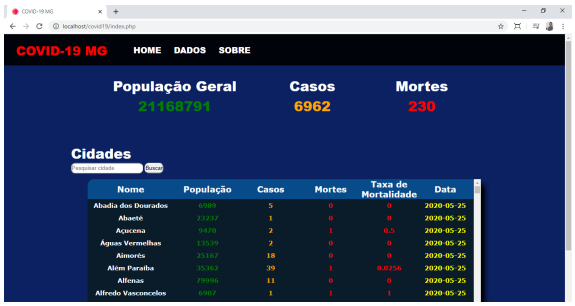

Figura 9. Exemplo de Página Estilizada

5. Conclusão

Em suma, o Redis provou ser eficiente quando operou testes de leitura e escrita. Além do mais, pôde ser exemplificado na prática o quão fácil é integrá-lo com uma linguagem de programação e implementá-lo em casos reais.

Por desfecho, destaca-se que houveram várias dificuldades durante a elaboração do trabalho, uma vez que não se tinha conhecimento algum sobre o sistema em questão. Além disso, a bibliografia no geral é limitada e apresenta difícil compreensão, especialmente quando se trata da documentação do banco. Todavia, acredita-se que o desenvolvimento como um todo foi satisfatório, principalmente com relação a elaboração do cenário real, já que se trata de um tema atual.

6. Referências

Referências

AMAZON (2020). Redis.

BEMFICA, D. (2018). Começando com o redis.

CARLSON, J. (2013). Redis in action.

ONWORKS (2020). Kali linux.

P. J. SAMPAIO, I. O. K. (2015). Desempenho de aplicações web: Um estudo comparativo utilizando o software redis.

REDISLABS (2020). Documentation.

WALKER-MORGAN, D. (2010). Nosql im uberblick.