Universidade de Aveiro, DETI

Complementos de Bases de Dados

Guião das aulas práticas

LEI – Licenciatura em Engenharia Informática

Ano: 2021/2022

Lab 1 Bases de Dados Chave-Valor

Objetivos

Os objetivos deste trabalho são:

- Compreender os fundamentos das bases de dados de chave-valor.
- Instalar e utilizar uma solução de código aberto.
- Desenvolver soluções para diversos casos de uso

Nota prévia

Este módulo deverá ser preferencialmente desenvolvido em Linux. Caso pretenda usar Windows verifique as notas sobre compatibilidade do software que irá usar.
Submeta o código/resultados/relatórios no elearning. Utilize uma pasta (1, 2, ..) para cada exercício, compactadas num único ficheiro.
Bom trabalho!

1.1 REDIS

Redis (*REmote DIctionary Service*) foi desenvolvido em 2009 e tem sofrido atualizações sucessivas, sendo um dos mais populares repositórios do tipo chave-valor em memória. Pode ser utilizado como base de dados, como *cache* de dados, ou como sistema de mensagens (*message broker*).

a) Instale o Redis no seu computador pessoal (https://redis.io/). Execute o servidor:

```
$ redis-server
```

b) Estude o funcionamento do sistema testando os comandos mais usados, através de linha de comandos.

```
$ redis-cli
```

c) Consulte os slides disponibilizados para a disciplina (*CBS_05_KeyValue*) e sítios web com documentação sobre o Redis. Alguns exemplos:

- *Redis web site,* https://redis.io
- *Introduction to Redis*, http://intro2libsys.info/introduction-to-redis/
- *Redis Java*, https://www.tutorialspoint.com/redis/redis_java.htm
- *Redis Tutorial - w3resource*, http://www.w3resource.com/redis/index.php

Alguns conceitos para os quais deve ter particular atenção:

- Escrita e leitura, persistência, TTL, tipos
- Estruturas (Hash, List, Set, Sorted Set, Streams)
- Operações sobre estruturas (ranges, unions, intersections, subtractions)

- ACLs, Transações e Namespace

No final deste exercício, no seu diretório raiz encontrará um ficheiro com o nome "`.rediscli_history`". Copie-o para outro ficheiro com o nome "`redis_11_<NMEC>.txt`" (onde `<NMEC>` deve ser substituído pelo seu nº mecanográfico) e submeta este ficheiro.

1.2 Redis – Inserção massiva

O Redis permite inserir dados a partir de um ficheiro, através de linha de comandos. Use o ficheiro "`names.txt`" como e crie chaves com o seguinte formato:

```
SET A <total de nomes que começa pela letra 'A'/'a'>
SET B <total de nomes que começa pela letra 'B'/'b'>
...
```

Deve entregar os ficheiros:

`names_counting.txt` – contagens a partir do ficheiro original no formato de upload
`README.txt` – comando(s) usados para carregar o ficheiro no Redis

1.3 Redis – Acesso programático

a) Instale um driver de Redis para Java (e.g. Jedis, ou outro disponível em https://redis.io/clients) e crie um pequeno programa para ligação ao servidor Redis, repetindo algumas das operações anteriores. Use como base o exemplo seguinte (*baseado em Jedis*):

```
import redis.clients.jedis.Jedis;

public class Forum {

   private Jedis jedis;

   public Forum(){
      this.jedis = new Jedis("localhost");
      System.out.println(jedis.info());
   }

   public static void main(String[] args) {
      new Forum();
   }
}
```

b) Crie um programa que escreva e leia usando i) uma lista e ii) um hashmap. Tome como base o exemplo seguinte que escreve e lê sobre um Set.
(Nota: quando mudar entre tipos, garanta que usa nomes diferentes para as para evitar colisões entre tipos).

```
package redis;

import java.util.Set;
import redis.clients.jedis.Jedis;

public class SimplePost {

   private Jedis jedis;
```

```
    public static String USERS = "users"; // Key set for users' name

    public SimplePost() {
       this.jedis = new Jedis("localhost");
    }

    public void saveUser(String username) {
       jedis.sadd(USERS, username);
    }
    public Set<String> getUser() {
       return jedis.smembers(USERS);
    }

    public Set<String> getAllKeys() {
       return jedis.keys("*");
    }

    public static void main(String[] args) {
       SimplePost board = new SimplePost();
       // set some users
       String[] users = { "Ana", "Pedro", "Maria", "Luis" };
       for (String user: users)
          board.saveUser(user);
       board.getAllKeys().stream().forEach(System.out::println);
       board.getUser().stream().forEach(System.out::println);

    }
}
```

1.4 Autocomplete

a) Usando o servidor Redis e o driver cliente (Jedis) crie um programa para fornecer uma lista de termos, ordenados por ordem alfabética, para uma função de *autocomplete*. Use o ficheiro "`names.txt`" como base para os termos a procurar. Note que apenas irá utilizar os nomes como chave (sem valor associado). A iteração final com o utilizador pode ser a seguinte:

```
Search for ('Enter' for quit): susann
susann
susanna
susannah
susanne

Search for ('Enter' for quit): zora
zora
zorah
zorana
```

b) O ficheiro "`nomes-pt-2021.csv`" contém uma lista dos nomes pessoais e o total de registos que foram registados em 2021, Desenvolva uma variante da alínea anterior onde o resultado da pesquisa para *autocomplete* é ordenado por popularidade decrescente do nome.

1.5 Sistema de mensagens

a) Construa um sistema de mensagens usando Redis e o driver cliente (Jedis). Deve prever as seguintes funcionalidades:

- Adição de utilizadores (identificados pelo nome).

- Associação entre utilizadores (e.g. se *userA* segue *userB*, então *userA* deve ter informação sobre todas as mensagens enviadas por *userB* para o sistema).
- Envio de mensagens, por exemplo:
  ```
  storeMsg(userA, "Isto vai ser fácil!").
  ```
- Leitura de mensagens (por utilizador, etc.).

Elabore e desenvolva algumas funcionalidades adicionais.
Descreva as estruturas que criou/usou (e.g. num ficheiro *readme.txt*).