Redis数据库

1.Redis数据库简介

Redis 是完全开源的，遵守 BSD 协议，是一个高性能的 key-value 数据库。

Redis 与其他 key - value 缓存产品有以下三个特点：

- Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。
- Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
- Redis支持数据的备份，即master-slave模式的数据备份。

Redis 优势

- 性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。
- 丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。
- 原子 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。
- 丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

2.Redis数据类型

Redis支持五种数据类型：string(字符串), hash(哈希), list(列表), set(集合)以及zset(sorted set：有序集合)。

类型	简介	特性	场景
String(字符串)	二进制安全	可以包含任何数据,比如jpg图片或者序列化的对象,一个键最大能存储512M	---
Hash(字典)	键值对集合,即编程语言中的Map类型	适合存储对象,并且可以像数据库中update一个属性一样只修改某一项属性值(Memcached中需要取出整个字符串反序列化成对象修改完再序列化存回去)	存储、读取、修改用户属性
List(列表)	链表(双向链表)	增删快,提供了操作某一段元素的API	1,最新消息排行等功能(比如朋友圈的时间线) 2,消息队列
Set(集合)	哈希表实现,元素不重复	1、添加、删除,查找的复杂度都是O(1) 2、为集合提供了求交集、并集、差集等操作	1、共同好友 2、利用唯一性,统计访问网站的所有独立ip 3、好友推荐时,根据tag求交集,大于某个阈值就可以推荐
Sorted Set(有序集合)	将Set中的元素增加一个权重参数score,元素按score有序排列	数据插入集合时,已经进行天然排序	1、排行榜 2、带权重的消息队列

3.Redis命令

- 启动redis客户端

```
$ redis-cli
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> PING

PONG
```

4.Redis 键(Key)

语法

Redis 键命令的基本语法如下:

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

实例

```
redis 127.0.0.1:6379> SET runoobkey redis
OK
redis 127.0.0.1:6379> DEL runoobkey
(integer) 1
```

在以上实例中 **DEL** 是一个命令， **runoobkey** 是一个键。 如果键被删除成功，命令执行后输出 **(integer) 1**，否则将输出 **(integer) 0**

5.Redis 字符串(String)

Redis 字符串数据类型的相关命令用于管理 redis 字符串值，基本语法如下：

语法

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

实例

```
redis 127.0.0.1:6379> SET runoobkey redis
OK
redis 127.0.0.1:6379> GET runoobkey
"redis"
```

在以上实例中我们使用了 **SET** 和 **GET** 命令，键为 **runoobkey**。

6.Redis 哈希(Hash)

Redis hash 是一个 string 类型的 field（字段） 和 value（值） 的映射表，hash 特别适合用于存储对象。

Redis 中每个 hash 可以存储 232 - 1 键值对（40多亿）。

实例

```
127.0.0.1:6379>  HSET people.zhangsan id 1
(integer) 1
# HGETALL key 获取在哈希表中指定 key 的所有字段和值
127.0.0.1:6379>  HGETALL people.zhangsan
1) "id"
2) "1"
3) "sex"
4) "male"
# HGET key field 获取存储在哈希表中指定字段的值
127.0.0.1:6379>  HGET people.zhangsan id
"1"
# HSET key field value 将哈希表 key 中的字段 field 的值设为 value
127.0.0.1:6379>  HSET people.zhangsan id 2
(integer) 0
```

7.Redix 列表(List)

Redis列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）

一个列表最多可以包含 232 - 1 个元素 (4294967295, 每个列表超过40亿个元素)。

实例

```
redis 127.0.0.1:6379> LPUSH runoobkey redis
(integer) 1
redis 127.0.0.1:6379> LPUSH runoobkey mongodb
(integer) 2
redis 127.0.0.1:6379> LPUSH runoobkey mysql
(integer) 3
redis 127.0.0.1:6379> LRANGE runoobkey 0 10

1) "mysql"
2) "mongodb"
3) "redis"
```

在以上实例中我们使用了 **LPUSH** 将三个值插入了名为 **runoobkey** 的列表当中。

8.Redis 集合(Set)

Redis 的 Set 是 String 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。

Redis 中集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)。

集合中最大的成员数为 232 - 1 (4294967295, 每个集合可存储40多亿个成员)。

实例

```
redis 127.0.0.1:6379> SADD runoobkey redis
(integer) 1
redis 127.0.0.1:6379> SADD runoobkey mongodb
(integer) 1
redis 127.0.0.1:6379> SADD runoobkey mysql
(integer) 1
redis 127.0.0.1:6379> SADD runoobkey mysql
(integer) 0
redis 127.0.0.1:6379> SMEMBERS runoobkey

1) "mysql"
2) "mongodb"
3) "redis"
```

在以上实例中我们通过 **SADD** 命令向名为 **runoobkey** 的集合插入的三个元素。

9.Redis 有序集合(sorted set)

Redis 有序集合和集合一样也是 string 类型元素的集合,且不允许重复的成员。

不同的是每个元素都会关联一个 double 类型的分数。redis 正是通过分数来为集合中的成员进行从小到大的排序。

有序集合的成员是唯一的,但分数(score)却可以重复。

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)。 集合中最大的成员数为 232 - 1 (4294967295, 每个集合可存储40多亿个成员)。

实例

```
redis 127.0.0.1:6379> ZADD runoobkey 1 redis
(integer) 1
```

```
redis 127.0.0.1:6379> ZADD runoobkey 2 mongodb
(integer) 1
redis 127.0.0.1:6379> ZADD runoobkey 3 mysql
(integer) 1
redis 127.0.0.1:6379> ZADD runoobkey 3 mysql
(integer) 0
redis 127.0.0.1:6379> ZADD runoobkey 4 mysql
(integer) 0
redis 127.0.0.1:6379> ZRANGE runoobkey 0 10 WITHSCORES

1) "redis"
2) "1"
3) "mongodb"
4) "2"
5) "mysql"
6) "4"
```

在以上实例中我们通过命令 **ZADD** 向 redis 的有序集合中添加了三个值并关联上分数。

10.JAVA API连接Redis设置

1. 修改redis.conf

   ```
   vim /etc/redis/redis.conf
   ```

   注释掉bind 127.0.0.1
2. 关闭防火墙

   ```
   ufw disable
   ```

3. 重启redis

   ```
   service redis-server restart
   ```

10.JAVA API连接

- 添加maven依赖

  ```
  <dependency>
      <groupId>redis.clients</groupId>
      <artifactId>jedis</artifactId>
      <version>3.3.0</version>
      <type>jar</type>
      <scope>compile</scope>
  </dependency>
  ```

- 建立连接

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

public class RedisStringJava {
    public static void main(String[] args) {
        //连接本地的 Redis 服务
        Jedis jedis = new Jedis("localhost");
        System.out.println("连接成功");
        //设置 redis 字符串数据
        jedis.set("runoobkey", "www.runoob.com");
        // 获取存储的数据并输出
        System.out.println("redis 存储的字符串为: "+ jedis.get("runoobkey"));
    }
}
```

- 添加数据

```
import java.util.Map;
import redis.clients.jedis.Jedis;
public class jedis_test {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Jedis jedis = new Jedis("localhost");
        jedis.hset("people.zhangsan", "id","1");
        jedis.hset("people.zhangsan", "sex","male");
        Map<String,String>  value = jedis.hgetAll("people.zhangsan");
        for(Map.Entry<String, String> entry:value.entrySet())
        {
            System.out.println(entry.getKey()+":"+entry.getValue());
        }
    }
}
```

- 查找数据

```
import java.util.Map;
import redis.clients.jedis.Jedis;

public class jedis_query {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Jedis jedis = new Jedis("192.168.6.130");
        String value=jedis.hget("people.zhangsan", "id");
        System.out.println("zhangsan's id is:"+value);
    }
}
```