链滴

redis 实现搜索热词统计

作者：JellyfishMIX
原文链接：https://ld246.com/article/1591166094977
来源网站：链滴

核心需求

一个项目中，遇到了搜索热词统计的需求，我使用了 Redis 的五大数据类型之一 Sorted Set 实现。前有两项数据需要统计：“当日搜索热词 top10”和“当周搜索热词 top10”。

关于这两项数据的统计方法，目前想到了两种实现方法：

1. 两个 Redis 的 Sorted Set 实现，一个 Sorted Set A 统计当天，0 点 top10 记录进 MySQL，Sort d Set 清零。一个 Sorted Set B 统计当周，每周日 top10 记录进 MySQL，Sorted Set B 清零。

2. 只使用用一个 Sorted Set 记录当天搜索热词，0 点 top10 记录进 MySQL，Sorted Set 清零。到日时，会有 7 * 10 行记录。把这 7 * 10 行遍历，每次便利都记录进 Sorted Set，全部遍历结束后，从 Sorted Set 中取出 top10 记录进 MySQL 的周热词统计表中。

Sorted Set 是 Redis 的数据结构，方法 1 会占用两份内存，一份当天的，一份当周的。方法 2 会提系统的复杂度，并且在统计周表时，可能会出现短时间内大量的计算（当然可以使用定时任务,把周表统计放到凌晨进行）。

最后选择了方案1，分开维护清晰明了。

至于内存占用问题，1MB = 1048576 字节，按两个字节存一个字算，理论上1MB 能存 1048576/2/8 = 65,536 个不重复的搜索关键词（当然使用Sorted Set肯定比纯字更多占用一些空间）。多投入一内存，能存下的数量还是很大的，通常可以撑到每周结束清理内存。一般的 CRUD 项目不用怎么考内存占用。

问题中涉及的相关知识

一个项目中，遇到了搜索热词统计的需求。我使用了 Redis 的五大数据类型之一 Sorted Set 实现。

Redis 有序集合(sorted set)

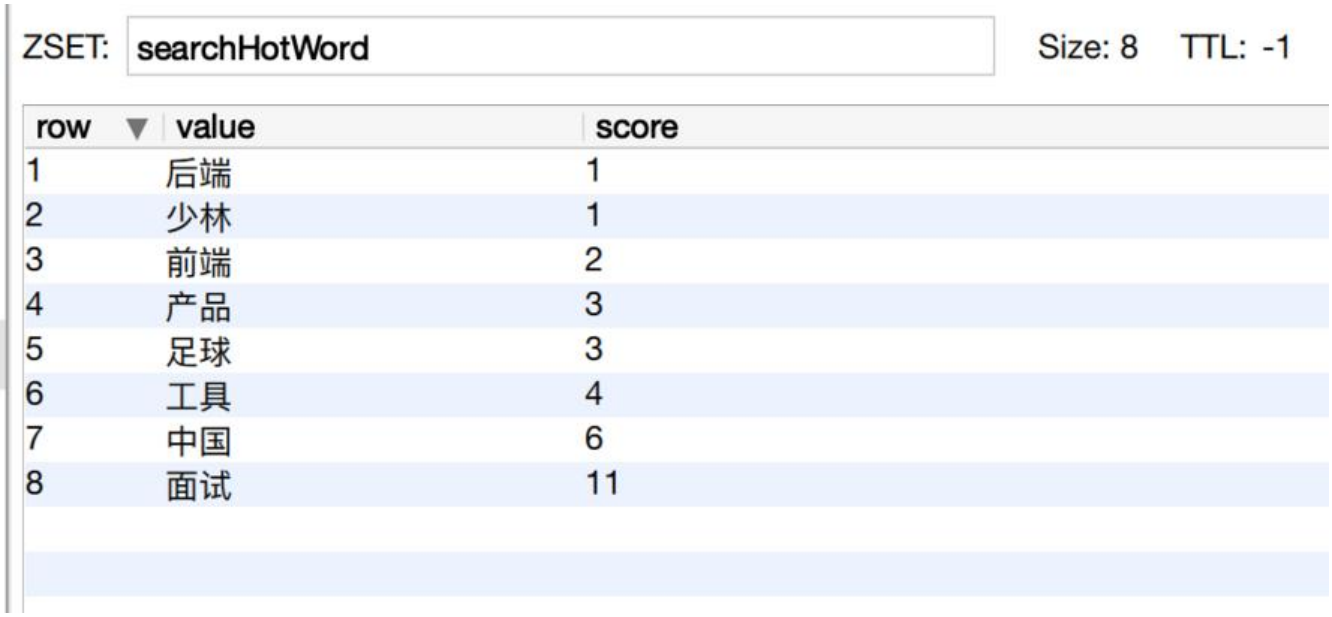
Redis 有序集合和集合一样也是 string 类型元素的集合，且不允许重复的成员。

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 - (4294967295, 每个集合可存储 40 多亿个成员)。

定义出自：菜鸟教程



ZSET: searchHotWord Size: 8 TTL: -1

row	value	score
1	后端	1
2	少林	1
3	前端	2
4	产品	3
5	足球	3
6	工具	4
7	中国	6
8	面试	11

如上图，Redis 的 Sorted Set 自带排序功能。

操作方法也比较简单，在本项目中，核心是两个方法：

zincrby，对于一个 Sorted Set，存在的就把分数加 x (x 可自行设定)，不存在就创建一个分数为 1 成员。

zrevrange，查询sorted set中指定范围的值。返回的有序集合中，score 大的在前面。此方法无需担用于指定范围的start和end出现越界报错问题。

在StringRedisTemplate中，Sorted Set被称为ZSet。更多redis (java客户端) 的Soeted Set使用方请见：Redis之ZSet数据结构使用姿势

代码

service示例代码

```
@Service("redisService")
public class RedisServiceImpl implements RedisService {
    @Autowired
    private StringRedisTemplate redisTemplate;

    /**
     * 使用Sorted Set记录keyword
     * zincrby命令，对于一个Sorted Set，存在的就把分数加x(x可自行设定)，不存在就创建一个分数1的成员
     *
     * @param keyword 搜索关键词
     */
    @Override
    public void searchZincrby(String keyword) {
        // 名为sortedSetName的Sorted Set不用预先创建，不存在会自动创建，存在则向里添加数据
        String sortedSetName = "searchHotWord";
        // x 的含义请见本方法的注释
        double x = 1.0;
```

```
        redisTemplate.opsForZSet().incrementScore(sortedSetName, keyword, x);
    }

    /**
     * zrevrange命令, 查询Sorted Set中指定范围的值
     * 返回的有序集合中，score大的在前面
     * zrevrange方法无需担心用于指定范围的start和end出现越界报错问题
     *
     * @param start 查询范围开始位置
     * @param end 查询范围结束位置
     * @return
     */
    @Override
    public Set<ZSetOperations.TypedTuple<String>> queryTopSearchHotWord(Integer start, I
teger end) {
    String sortedSetName = "searchHotWord";
        Set<ZSetOperations.TypedTuple<String>> resultSet =  redisTemplate.opsForZSet().rever
eRangeWithScores(sortedSetName, start, end);
        return resultSet;
    }

    /**
     * 删除指定的key
     *
     * @param keyName keyName
     */
    @Override
    public void deleteKey(String keyName) {
        redisTemplate.delete(keyName);
    }

}
```

controller示例代码

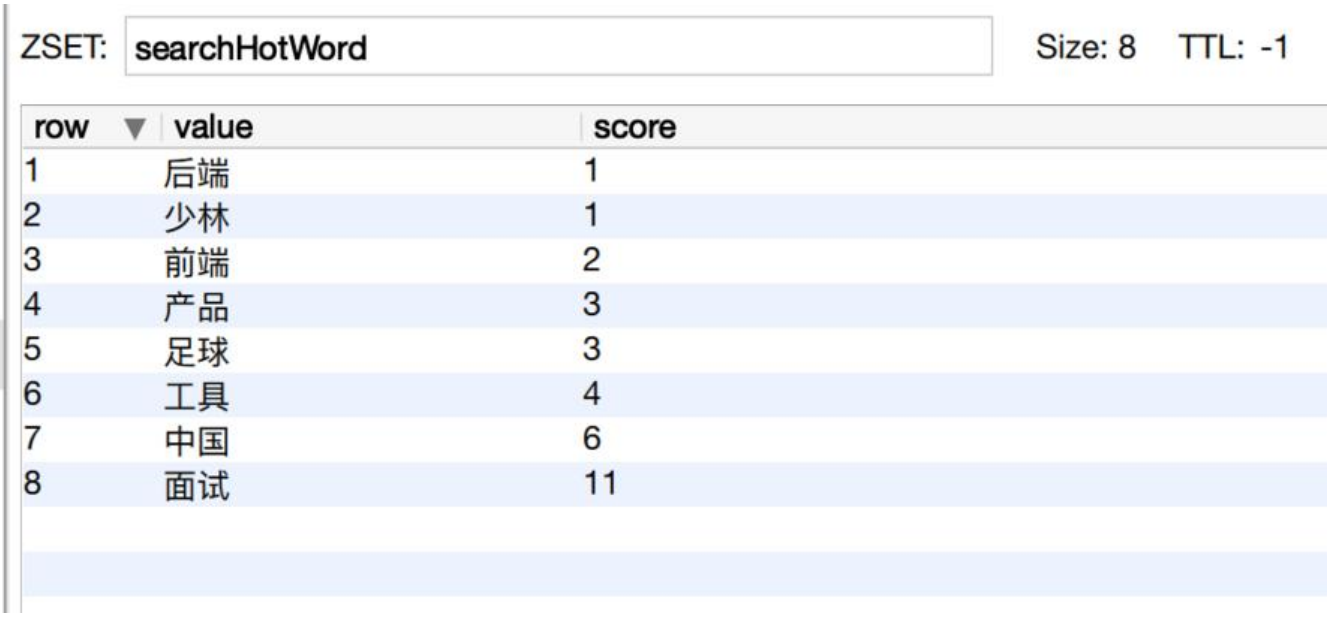
```
@RestController
@RequestMapping("/redis")
public class RedisController {
    @Autowired
    private RedisService redisService;

    /**
     * 测试redis记录keyword
     *
     * @param keyword 搜索关键词
     * @return
     */
    @GetMapping("/test_search")
    public ResultVO testSearch(@RequestParam("keyword") String keyword) {
        redisService.searchZincrby(keyword);
    // ResultVO和ResultVOUtil是自定义的class，为了方便展示结果，阅读时忽略即可
        return ResultVOUtil.success(1, "test-return");
    }
```

```
    /**
     * 测试redis查询指定范围的热词
     *
     * @param start 查询范围开始位置
     * @param end 查询范围结束位置
     * @return
     */
    @GetMapping("/test_query_top_search_hot_word")
    public ResultVO testQueryTopSearchHotWord(@RequestParam("start") Integer start,
                                              @RequestParam("end") Integer end) {
        Set<ZSetOperations.TypedTuple<String>> resultSet =  redisService.queryTopSearchHo
Word(start, end);
    // ResultVO和ResultVOUtil是自定义的class，为了方便展示结果，阅读时忽略即可
        return ResultVOUtil.success(1, "success", resultSet);
    }

    /**
     * 删除指定的key
     *
     * @param keyName keyName
     * @return
     */
    @PostMapping("/delete_key")
    public ResultVO deleteKey(@RequestParam("keyName") String keyName) {
        redisService.deleteKey(keyName);
    // ResultVO和ResultVOUtil是自定义的class，为了方便展示结果，阅读时忽略即可
        return ResultVOUtil.success(1, "test-return");
    }

}
```

测试代码的运行效果

模拟搜索一些keyword:

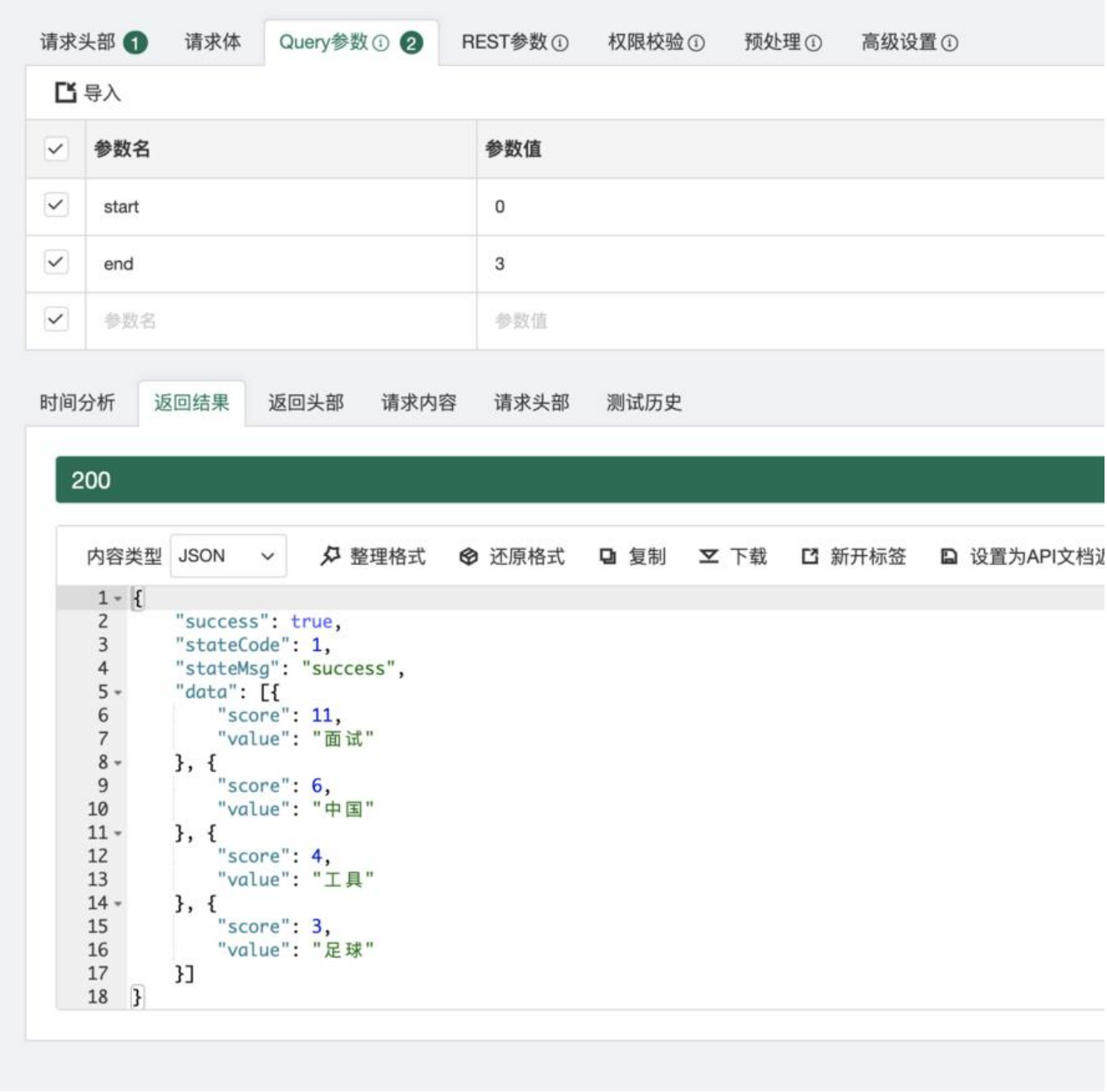
使用rdm查看reids的存储情况，搜索热词已经存在redis一个名为searchHotWord的Sorted Set中:



row	value	score
1	后端	1
2	少林	1
3	前端	2
4	产品	3
5	足球	3
6	工具	4
7	中国	6
8	面试	11

查询结果:

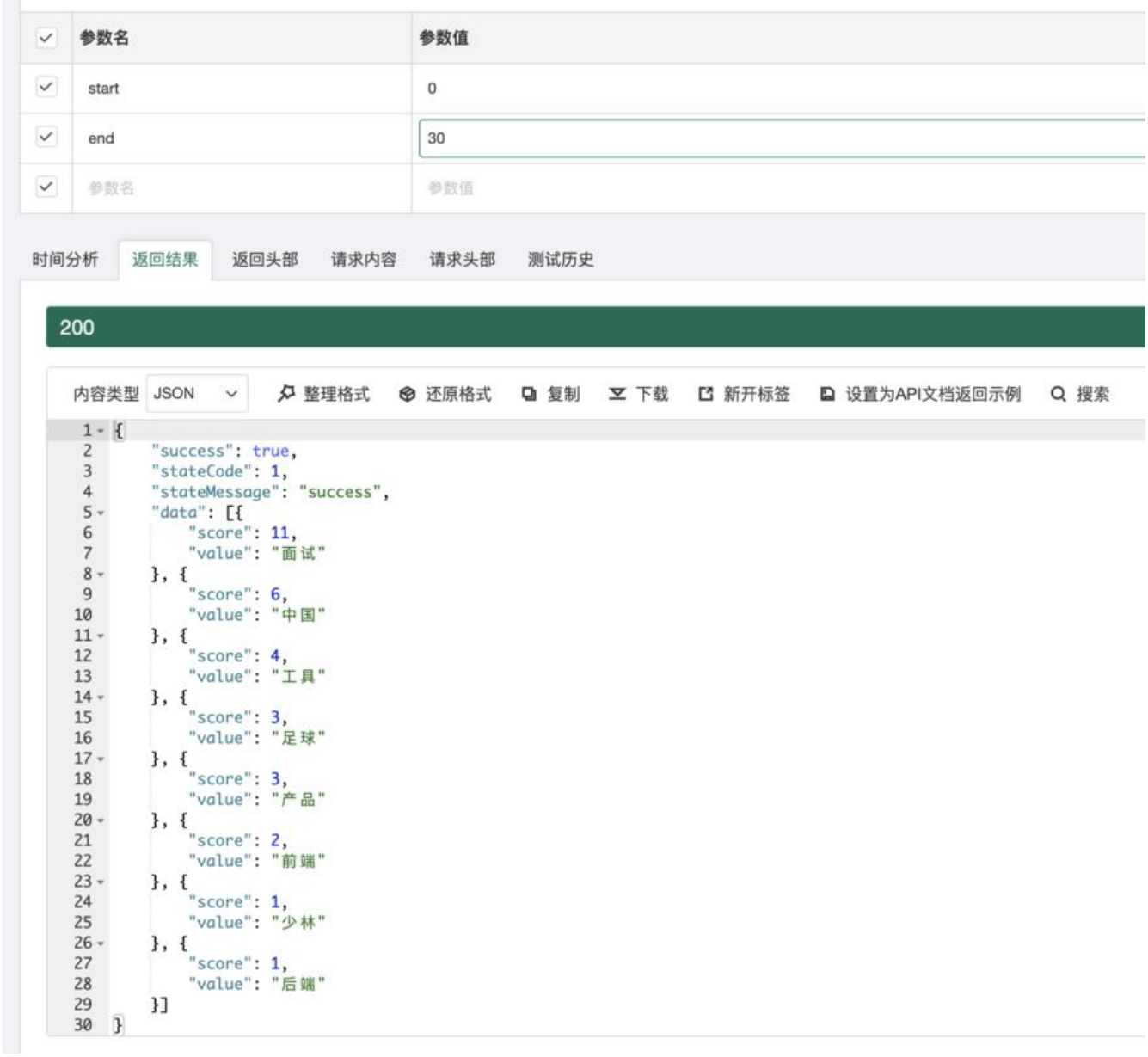
原文链接：redis 实现搜索热词统计



One more thing

zrevrange方法无需担心用于指定范围的 start 和 end 出现越界报错问题。

测试用的Sorted Set总共有8个数据，故意指定 start 和 end 在此长度范围之外：



原文链接：redis 实现搜索热词统计

经测试，在保证 start 和 end 均 >=0 的前提下，start 和 end 均无越界报错问题。