**五.sorted\_set**

**1.需求**

新的存储需求：数据排序有利于数据的有效展示，需要提供一种可以根据自身特征进行排序的方式。

需要的存储结构：新的存储模型，可以保存可排序的数据

sorted\_set类型：在set的存储结构基础上添加可排序字段

**2.sorted\_set类型数据的基本操作**

**添加数据：**

zadd key score1 member1 score2 member2 …

**获取全部数据：**

zrange key start stop [withscores] --升序排列

zrevrange key start stop [withscores] --降序排列

**删除数据：**

zrem key member [member …]

**按条件获取数据：**

zrangebyscore key min max [withscores] [limit] --输出在**[min, max)**范围之间的数据，limit begin count begin -- 排序后的索引 count--3个

zrevrangebyscore key max min [withscores]

**条件删除：**

zremrangebyrank key start stop -- 按照排序后的索引进行范围删除

zremrangebyscore key min max

**获取集合数据总量：**

zcard key

zcount key min max --范围内计数

**集合交、并操作：**

zinterstore destination numkeys key [key…]

zunionstore destination numkeys key [key…]

**注意：**

**min与max用于限定搜索查询的条件**

**start与stop用于限定查询范围，作用于索引，表示开始和结束索引**

**offset于count用于限定查询范围，作用于查询结果，表示开始位置和数据总量**

**3.sorted\_set的扩展操作**

**场景1）**

聊天室活跃度统计；票选十大杰出青年；

业务分析：

为所有参与排名的资源建立排序依据

**解决方案：**

1. 获取数据对应的索引（排名）

zrank key member --查对应值排序后的索引

zrevrank key member

1. score值获取与修改

zscore key member --查对应值的score

zincrby key increment member --对score进行自增操作

**ps13：**

**redis应用于计数器组合排序功能对应的排名**

**4.sorted\_set操作的注意事项**

1）score保存的数据存储空间是64位，如果是整数范围是

2）score保存的数据也可以是一个双精度double值，基于双精度浮点数的特征，可能会丢失精度，使用时候要慎重

3）sorted\_set底层存储还是基于set结构的，因此数据不能重复，如果重复添加相同的数据，score值将会被反复覆盖，保留最后一次修改的结果

**5.应用场景**

**场景1）**

基础服务 + 增值服务 比如：腾讯视频 -- 普通用户和vip用户

解决方案：

1. 对于基于时间限定的任务处理，将处理时间记录为score值，利用排序功能区分处理的先后顺序

2. 记录下一个要处理的时间，当到期后处理对应任务，移除redis中的记录，并记录下一个要处理的时间

3. 当新任务加入时，判定并更新当前下一个要处理的任务时间

4. 当提升sorted\_set的性能，通常将任务根据特征存储成若干个sorted\_set。如：1小时内，1天内等，操作时逐级提升，将即将操作的若干任务纳入到1小时处理的队列中

**获取当前系统时间：**

time

**ps14：**

**redis应用于定时任务执行顺序管理或任务过期管理**

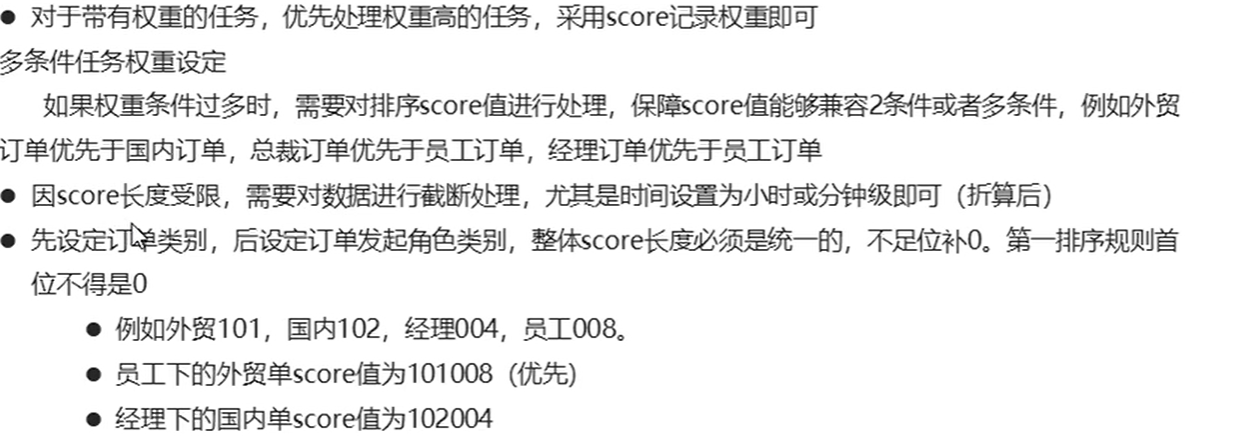
**场景2）**

任务/消息权重设定应用

当任务或者消息待处理，形成了任务/消息队列时，对于高优先级的任务要保障对其优先处理，如何实现任务权重管理？

**解决方案：**

1. 对于带有权重的任务，优先处理权重高的任务，采用score记录权重。

权重由多个字段组成时，需要将各个字段拼接起来

**注意：但是注意每个字段的字符数目应该是相同的，不够补0，否则比较的时候各个字段没有对齐，比较就失效**

**ps15：**

**redis应用于即时任务/消息队列执行管理**