活动排行榜系统设计

需求描述

你开发了一个游戏，日活跃用户在10万人以上。请设计一个活动排行榜系统，该排行榜需要实现以下功能或具有以下特点：

1. 在每月活动中，玩家得到的活动总分为 0 到 10000 之间的整数
2. 在每月活动结束之后，需要依据这一活动总分，从高到低为玩家建立排行榜

3. 如果多位玩家分数相同，则按得到指定分数顺序排序，先得到的玩家排在前面
4. 系统提供玩家名次查询接口，玩家能够查询自己名次前后10位玩家的分数和名次
5. 请使用 UML 图或线框图表达设计，关键算法可使用流程图或伪代码表达

需求分析

该系统需要实现的功能

1. 每月活动结束时，能够从高到低建立排名
2. 玩家可以获取自己的排名
3. 玩家可以获取自己名次前后的10名玩家

根据需求提取信息

1、在每月活动中，玩家得到的活动总分为 0 到 10000 之间的整数，日活人数10w+

从这里可以得到的信息是，玩家的数量可能远超积分的最大值，是否可以考虑将积分划分成10000个区间，然后将对应积分的用户，放入相应的区间。发现如果这样划分区间，可能会太多，所以可以考虑这样划分，0~1000积分的在一个区间内、1000~2000、2000~3000、.....、9000 ~ 10000，将用户按照积分划分开，类似于分治的思想

2、每月活动结束后，需要根据积分进行从高到低排名

对于排名，最容易想到的就是通过redis的zset来实现

3、如果多位玩家分数相同，则按得到指定分数顺序排序，先得到的玩家排在前面

分数相同的，谁先达到那个分数，谁就在前。那此时发现，如果用zset来实现排名，对于分数相同的情况，可能就满足不了谁先达到某个分数，谁就排名靠前了。既然是有先后，就可以考虑利用时间戳

4、系统提供玩家名次查询接口，玩家能够查询自己名次前后10位玩家的分数和名次

首先，根据玩家信息，获取它的名次，假设我们用redis的zset存（并且解决了分数相同的排名问题），知道用户名或ID（zset中的member去存用户名或ID），就可以通过zrevrank命令来获取到具体用户在zset中的排名。知道了该用户排名，就可以通过zrevrange key start end [withscores]，来获取指定排名范围的用户

整体设计与实现

考虑使用MySQL来实现

从【需求分析】部分可以知道，如果使用Redis的zset实现，不好解决相同积分情况的排名问题。如果使用MySQL的话，可以看看怎么实现上边的三个功能

表设计

可以创建一个**用户表**，用户基础信息+总积分；和一个**用户积分表**，考虑每个月单独使用一个用户积分表

```
//用户表
create table user_info (
  id bigint primary key auto_increment,
  user_ID varchar(20) unique key not null,
  user_name varchar(30) not null default 'xxx',
  user_total_score int unsigned default 0,
  create_time timestamp NOT NULL default CURRENT_TIMESTAMP comment '创建时间',
  update_time timestamp NOT NULL default CURRENT_TIMESTAMP comment '更新时间',
);

//用户积分表
```

```
create table user_score_2022_01 (
  id bigint primary key auto_increment,
  user_info_id bigint,
  user_ID varchar(20) unique key not null,
  user_name varchar(30) not null default 'xxx',//为了避免获取月排行榜时回表，这里加上用户名
  user_month_score int unsigned default 0,
  create_time timestamp NOT NULL default CURRENT_TIMESTAMP comment '创建时间',
  update_time timestamp NOT NULL default CURRENT_TIMESTAMP comment '更新时间',
);
```

可以每个月统计完之后，将月积分统计到用户表的总积分表中去

获取每月的排行榜

需要哪个月的排行榜，就去哪个月的对应积分表中去顺序输出

```
select user_ID, user_name from user_score_2022_01 order by score desc, update_time asc;
```

获取玩家排名

可以根据上边的SQL获取到排名的数据，然后对它进行遍历，然后不断的进行累加，直到找到要查询的用户id

```
//伪代码
func getUserRanking(userId string) int {
    userInfos := getList()//获取排名列表
    position:=0
    for ... {
        if userInfo[i].user_id == userId {
            return position+1
        }
        position++
    }
}
```

获取玩家前后10名玩家的信息

因为通过上边获取玩家排名，已经能找到具体某个玩家，而列表又是根据排名获取的，所以很容易可以将该玩家的前边和后边十个玩家输出

```
func getUserFrontAndBackTenUser(userId string) int {
    userInfos := getList()//获取排名列表
    position:=0
    for ... {
        if userInfo[i].user_id == userId {
            break
        }
        position++
    }

    front := position+1
    count_front, count_back := 10, 10
    for front >=0 && count_front > 0 {
        front--
        fmt.Println(userInfos[front])
        count_front++
    }
    ......
}
```

MySQL实现的缺点

1. 因为对于游戏积分，肯定会变更的十分频繁，MySQL并不适合频繁的更新和写操作
2. 对于获取月份排名，需要扫全表，效率非常低，如果数据量很大的话，就很很差
3. 获取用户排名接口执行时间一定会很长

考虑使用Redis来实现

需要解决的问题

1. 如果想实现快速的排名，很容易想到Redis的zset数据结构。但是首先需要解决**相同分数的排名问题**。题目中要求，先达到某个分数的用户，排名就靠前。这里有

时间先后，就可以考虑在用zset的score存积分的时候，加上时间戳，并且能够根据zset中的score，反推出积分是多少

2. 另外一个问题就是，**如果使用一个单独的zset，可能zset会很大**。可以考虑将用户按照积分区间进行划分，每个积分区间的用户，使用一个zset

解决相同分数排名问题

说明：相同积分排名的解决方案，我是在网上查到的，自己没有思考到好的解决方案。用自己的理解，整理如下（参考文章）

因为zset的score可以是相等的，如果score相等，它会根据member进行字典排序，所以需要解决分数相等时，排名能按照题目要求的来进行排名

很明显的是，如果这样去计算分数，达不到目的

```
score = 积分 + 时间戳
```

因为越先达到某个积分的用户，它的时间戳越小，所以积分+时间戳的方式实现不了

那如果想让时间戳小（先到达某个积分）的那个用户的score变大，我们可以用：**最大时间戳（MAX_TIMESTAMP）减去达到某个积分时的时间戳（TIMESTAMP）**。时间戳的毫秒精度是13位，score用长整形来存的话，是19位，那另外6位就可以存我们的用户积分，因为题目中说明了积分最大是10000，刚好是够的。所以，就可以得到下边这个计算score的公式

```
积分偏移 = math.Pow10(14)
score = MAX_TIMESTAMP - TIMESTAMP +  (积分*积分偏移)

这样的话，根据score，也能计算出积分

积分 = 分数 / 等级偏移（需要取整）

示例：
score = 9999999999999（13位） - 1649957191000 + （1000 * 10^14） = 100008350042808999
积分 = 100008350042808999 / math.Pow10(14) = 1000
```

这样存储score，就可以解决上边的问题

获取每月的排行榜

可以通过下边这种方式设置zset

```
积分 0 ~ 1000

key : user_score_2022_01_0_1000
score: 按照上边的公式计算积分
member: user_name

积分 1000 ~ 2000
key : user_score_2022_01_1000_2000
......

......

积分 9000 ~ 10000
key : user_score_2022_01_9000_10000
......
```

获取每月排行榜，可以从user_score_2022_01_9000_10000开始，将每个区间的有序集合，按照score从大到小的顺序输出

```
zrevrange user_score_2022_01_9000_10000 0 -1 withscores
......
zrevrange user_score_2022_01_0_1000 0 -1 withscores
```

获取玩家排名

1. 首先获取到用户的积分，这样能够知道它在哪个区间，找到对应的zset
2. 然后通过zrevrank命令，获取到某个用户在zset中的排名

```
zrevrank user_score_2022_01_2000_3000 cxs
```

3. 最后通过zcard，获取到user_score_2022_01_2000_3000后边几个积分区间的元素总个数（假设是X），在第二步已经获取到了用户在user_score_2022_01_2000_3000中的排名（假设是Y），最终用户cxs的排名就是X+Y

```
> zadd user_score_2022_01_2000_3000 2550 cxs
(integer) 1
> zrevrank user_score_2022_01_2000_3000 cxs
0
> zadd user_score_2022_01_2000_3000 2551 cxs_1
(integer) 1
> zadd user_score_2022_01_2000_3000 2552 cxs_2
(integer) 1
> zrevrank user_score_2022_01_2000_3000 cxs
2
```

获取玩家前后10名玩家的信息

在上边已经获取到了指定玩家的排名，获取该玩家前后玩家的信息，可以通过zrevrange来实现。它可以返回指定排名范围的成员

假设获取到了用户cxs的排名为m，那它前后十名玩家就可以按照下边的命令获取（如果m小于10，则取前10名的时候，start为-1即可，如果当前集合中当前玩家的前边不足10个玩家，去相邻的zset中取，下边代码中没有写这些情况的，实际编写过程需要考虑）

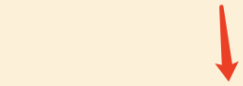
```
zrevrange user_score_2022_01_2000_3000 m-10 m withscores//前10名（去掉本身）
zrevrange user_score_2022_01_2000_3000 m m+10 withscores//后十名（去掉本身）
```

```
> zrevrange user_score_2022_01_2000_3000 7 11 withscores

1) "cxs_7"
2) 2557.0
3) "cxs_6"
4) 2556.0
5) "cxs_5"
6) 2555.0
7) "cxs_4"
8) 2554.0
9) "cxs_3"
10) 2552.0

> zrevrange user_score_2022_01_2000_3000 2 7 withscores

1) "cxs_12"
2) 2562.0
3) "cxs_11"
4) 2561.0
5) "cxs_10"
6) 2560.0
7) "cxs_9"
8) 2559.0
9) "cxs_8"
10) 2558.0
11) "cxs_7"
12) 2557.0
```



如何让我去设计底层实现我会怎么做

整体思路

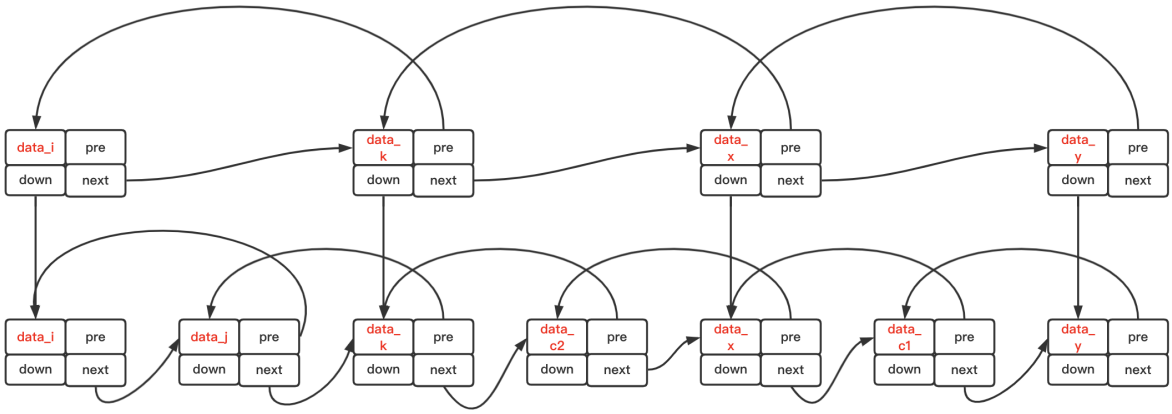
我最开始看这个题的时候，感觉自己想偏了，一直在想，我通过哪种基础的数据结构去实现这些排名相关的功能。而并不是从系统的层面去思考它的实现。按照基础数据结构，我是这样想的

1. 首先为了能够快速的根据用户名或ID查找到指定的用户，我会首先维护一个**hash表**，hash表的key，为用户名/ID经过hash算法之后得到的值，value为一个跳表的节点结构（下边代码中有跳表节点结构）
2. 这些hash表中的value，连接成一个有序的链表（根据积分），并且通过hash表的value连接成的链表，作为一个跳表结构的基础
3. 在基础链表上构建索引，主要是为了实现能够根据积分区间来获取到玩家的数据

在图上比较难话清楚完整的结构，线会比较杂，所以我这里将跳表和hash表的结构分开画了

跳表结构图

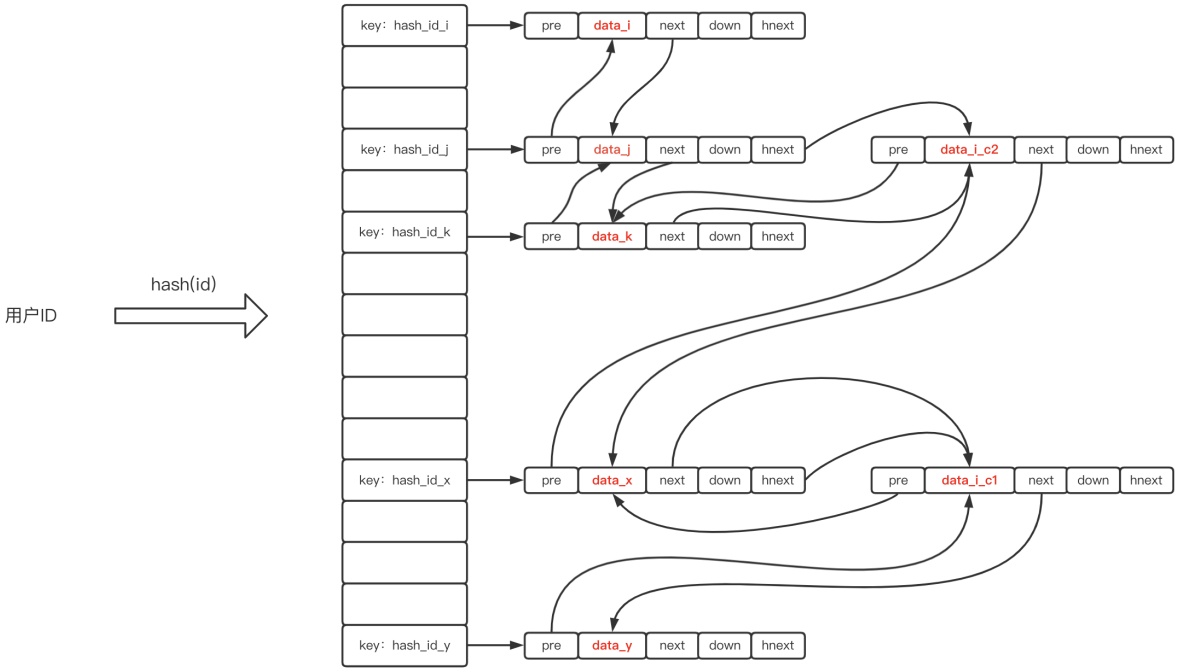
说明：节点的结构体设计在下边代码中



节点时以积分为顺序构建的（从大到小）

hash表结构图

说明：节点的结构体设计在下边代码中



hash表中vaulue相连的链表，就是跳表的最底层的基础链表

结构体设计

```
type skipListNode struct { //跳表节点设计
  uset_ID string
  user_name string
  score uint16
  pre *skipListNode //双向链表的前驱指针
  next *skipListNode //双向链表的后继指针
  hnext *skipListNode //处理hash冲突的拉链指针
  down *skipListNode //跳表的下沉指针
}

type skipList struct {//跳表
  level int
  headNodeArr []*skipListNode
}

type ranking struct {//hash结构
  hashMap map[string]*skipListNode
}
```

如何实现题目中的功能

关于相等的积分，谁先谁靠前的问题，可以在插入链表的时候进行判断，如果发现待插入节点的积分和它前边的这个节点的积分相等，就把新的节点插在后边即可

获取每月的排行榜

这个就可以直接遍历跳表最底层的双向链表

获取玩家排名

这种方式实现，目前没有想到更好的方法来获取玩家排名

获取玩家前后10名玩家的信息

根据玩家id，从hash表中找到对应玩家，从该玩家的位置往前和往后各遍历10个节点即可

说明

感觉这只是一个想法，实现起来应该是非常的困难，因为需要自己手动的实现一个跳表的相关操作