Redis Sscan 命令 Redis Zadd 命令

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 $2^{32} - 1$ (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。

Redis 有序集合命令

下表列出了 redis 有序集合的基本命令:

序号	命令及描述
1	ZADD key score1 member1 [score2 member2] 向有序集合添加一个或多个成员，或者更新已存在成员的分数
2	ZCARD key 获取有序集合的成员数
3	ZCOUNT key min max

	计算在有序集合中指定区间分数的成员数
4	ZINCRBY key increment member 有序集合中对指定成员的分数加上增量 increment
5	ZINTERSTORE destination numkeys key [key ...] 计算给定的一个或多个有序集的交集并将结果集存储在新的有序集合 key 中
6	ZLEXCOUNT key min max 在有序集合中计算指定字典区间内成员数量
7	ZRANGE key start stop [WITHSCORES] 通过索引区间返回有序集合成指定区间内的成员
8	ZRANGEBYLEX key min max [LIMIT offset count] 通过字典区间返回有序集合的成员
9	ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT] 通过分数返回有序集合指定区间内的成员
10	ZRANK key member 返回有序集合中指定成员的索引
11	ZREM key member [member ...] 移除有序集合中的一个或多个成员
12	ZREMRANGEBYLEX key min max 移除有序集合中给定的字典区间的所有成员
13	ZREMRANGEBYRANK key start stop 移除有序集合中给定的排名区间的所有成员
14	ZREMRANGEBYSCORE key min max 移除有序集合中给定的分数区间的所有成员
15	ZREVRANGE key start stop [WITHSCORES] 返回有序集中指定区间内的成员，通过索引，分数从高到底
16	ZREVRANGEBYSCORE key max min [WITHSCORES] 返回有序集中指定分数区间内的成员，分数从高到低排序
17	ZREVRANK key member 返回有序集合中指定成员的排名，有序集成员按分数值递减(从大到小)排序

18	ZSCORE key member 返回有序集中，成员的分数值
19	ZUNIONSTORE destination numkeys key [key ...] 计算给定的一个或多个有序集的并集，并存储在新的 key 中
20	ZSCAN key cursor [MATCH pattern] [COUNT count] 迭代有序集合中的元素（包括元素成员和元素分值）

← Redis Sscan 命令 Redis Zadd 命令 →

2 篇笔记

写笔记

原文中说，**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)**其实不太准确。
其实在redis sorted sets里面当items内容大于64的时候同时使用了hash和skiplist两种设计实现。这也会为了排序和查找性能做的优化。所以如上可知：
添加和删除都需要修改skiplist，所以复杂度为O(log(n))。
但是如果仅仅是查找元素的话可以直接使用hash，其复杂度为O(1)
其他的range操作复杂度一般为O(log(n))
当然如果是小于64的时候，因为是采用了ziplist的设计，其时间复杂度为O(n)

麻酱 1年前 (2017-12-03)

补充测试结果:

```
127.0.0.1:6379> ZADD SDATA 1 S1
(integer) 1
127.0.0.1:6379> ZADD SDATA 2 S2
(integer) 1
127.0.0.1:6379> ZADD SDATA 3 S3
(integer) 1
127.0.0.1:6379> ZADD SDATA 4 A1
(integer) 1
127.0.0.1:6379> ZADD SDATA 4 A2
(integer) 1
127.0.0.1:6379> ZADD SDATA 4 A3
(integer) 1
127.0.0.1:6379> ZADD SDATA 4 A4
(integer) 1
127.0.0.1:6379> ZRANGE SDATA 0 10 WITHSCORES
 1) "S1"
 2) "1"
 3) "S2"
```

```
 4) "2"
 5) "S3"
 6) "3"
 7) "A1"
 8) "4"
 9) "A2"
10) "4"
11) "A3"
12) "4"
13) "A4"
14) "4"
```

张小三 3个月前 (12-05)