1.zAdd()

1.zAdd()

增加一个或多个元素，如果该元素已经存在，更新它的socre值虽然有序集合有序，但它也是集合，不能重复元素，添加重复元素只会更新原有元素的score值

[html] view plain copy

print?

```
1. $redis->zAdd('key', 1, 'val1');
2. $redis->zAdd('key', 0, 'val0');
3. $redis->zAdd('key', 5, 'val5');
4. $redis->zRange('key', 0, -1); // array(val0, val1, val5)
```

2.zRem()

从有序集合中删除指定的成员。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 0, 'val0');
2. $redis->zAdd('key', 2, 'val2');
3. $redis->zAdd('key', 10, 'val10');
4. $redis->zDelete('key', 'val2');
5. $redis->zRange('key', 0, -1); /* array('val0', 'val10') */
```

3.ZCARD()

返回存储在key对应的有序集合中的元素的个数。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 0, 'val0');
2. $redis->zAdd('key', 2, 'val2');
3. $redis->zAdd('key', 10, 'val10');
4. $redis->zSize('key'); /* 3 */
```

4.zCount()

返回key对应的有序集合中介于min和max间的元素的个数。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 0, 'val0');
2. $redis->zAdd('key', 2, 'val2');
3. $redis->zAdd('key', 10, 'val10');
4. $redis->zCount('key', 0, 3); /* 2, corresponding to array('val0', 'val2') */
```

5.zScore()

返回key对应的有序集合中member的score值。如果member在有序集合中不存在，那么将会返回nil。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 2.5, 'val2');
2. $redis->zScore('key', 'val2'); /* 2.5 */
```

6.ZINCRBY() score值相加

将key对应的有序集合中member元素的scroe加上increment。如果指定的member不存在，那么将会添加该元素，并且其score的初始值为increment。如果key不存在，那么将会创建一个新的有序列表，其中包含member这一唯一的元素。如果key对应的值不是有序列表，那么将会发生错误。指定的score的值应该是能够转换为数字值的字符串，并且接收双精度浮点数。同时，你也可用提供一个负值，这样将减少score的值。

[html] view plain copy

print?

```
1. $redis->delete('key');
2. $redis->zIncrBy('key', 2.5, 'member1');
3. /* key or member1 didn't exist, so member1's score is to 0 before the increment */
```

```
4. /* and now has the value 2.5  */
5. $redis->zIncrBy('key', 1, 'member1'); /* 3.5 */
```

7. zRange()

取得特定范围内的排序元素,0代表第一个元素,1代表第二个以此类推。-1代表最后一个,-2代表倒数第二个...

[html] view plain copy

print?

```
1. $redis->zAdd('key1', 0, 'val0');
2. $redis->zAdd('key1', 2, 'val2');
3. $redis->zAdd('key1', 10, 'val10');
4. $redis->zRange('key1', 0, -1); /* array('val0', 'val2', 'val10') */
5. // with scores
6. $redis->zRange('key1', 0, -1, true); /* array('val0' => 0, 'val2' => 2, 'val10' => 10) */
```

8. zRevRange()

返回key对应的有序集合中指定区间的所有元素。这些元素按照score从高到低的顺序进行排列。对于具有相同的score的元素而言，将会按照递减的字典顺序进行排列。该命令与ZRANGE类似，只是该命令中元素的排列顺序与前者不同。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 0, 'val0');
2. $redis->zAdd('key', 2, 'val2');
3. $redis->zAdd('key', 10, 'val10');
4. $redis->zRevRange('key', 0, -1); /* array('val10', 'val2', 'val0') */
5. // with scores
6. $redis->zRevRange('key', 0, -1, true); /* array('val10' => 10, 'val2' => 2, 'val0' => 0) */
```

9. zRangeByScore()

返回key对应的有序集合中score介于min和max之间的所有元素（包哈score等于min或者max的元素）。元素按照score从低到高的顺序排列。如果元素具有相同的score，那么会按照字典顺序排列。

可选的选项LIMIT可以用来获取一定范围内的匹配元素。如果偏移值较大，有序集合需要在获得将要返回的元素之前进行遍历，因此会增加O(N)的时间复杂度。可选的选项WITHSCORES可以使得在返回元素的同时返回元素的score，该选项自从Redis 2.0版本后可用。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 0, 'val0');
2. $redis->zAdd('key', 2, 'val2');
3. $redis->zAdd('key', 10, 'val10');
4. $redis->zRangeByScore('key', 0, 3); /* array('val0', 'val2') */
5. $redis->zRangeByScore('key', 0, 3, array('withscores' => TRUE); /* array('val0' => 0, 'val2' => 2) */
6. $redis->zRangeByScore('key', 0, 3, array('limit' => array(1, 1)); /* array('val2') */
7. $redis->zRangeByScore('key', 0, 3, array('withscores' => TRUE, 'limit' => array(1, 1)); /* array('val2' => 2) */
```

10. zRemRangeByScore()

移除key对应的有序集合中scroe位于min和max（包含端点）之间的所哟元素。从2.1.6版本后开始，区间端点min和max可以被排除在外，这和ZRANGEBYSCORE的语法一样。

[html] view plain copy

print?

```
1. $redis->zAdd('key', 0, 'val0');
2. $redis->zAdd('key', 2, 'val2');
3. $redis->zAdd('key', 10, 'val10');
4. $redis->zRemRangeByScore('key', 0, 3); /* 2 */
```