# Liunx下安装Redis

# redis介绍

Redis是当前比较热门的NOSQL系统之一,它是一个KEY-VALUE存储系统。和Memcache类似，但是很大程度补偿了Mamcahe的不足,它支持存储的value类型对更多,包括string、list、set、zset和hash、这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作。在此基础上，redis支持不同方式的排序。

和Memcache一样，redis数据都是缓存在计算机内存中，不同的是，Memcache只能将数据缓存到内存中，无法自动定期写入磁盘，这就表示，一断电或重启，内存清空，数据丢失。所以Memcache的应用场景适用于缓存无需持久化的数据。而redis不同的是它会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，实现数据的持久化。

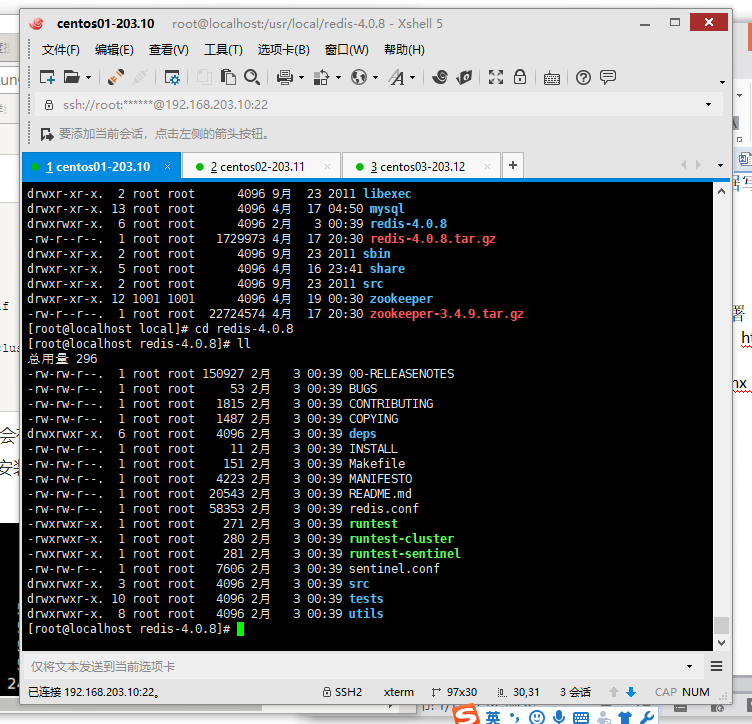
## Redis单体应用

下面介绍Liunx环境下，redis的安装与部署

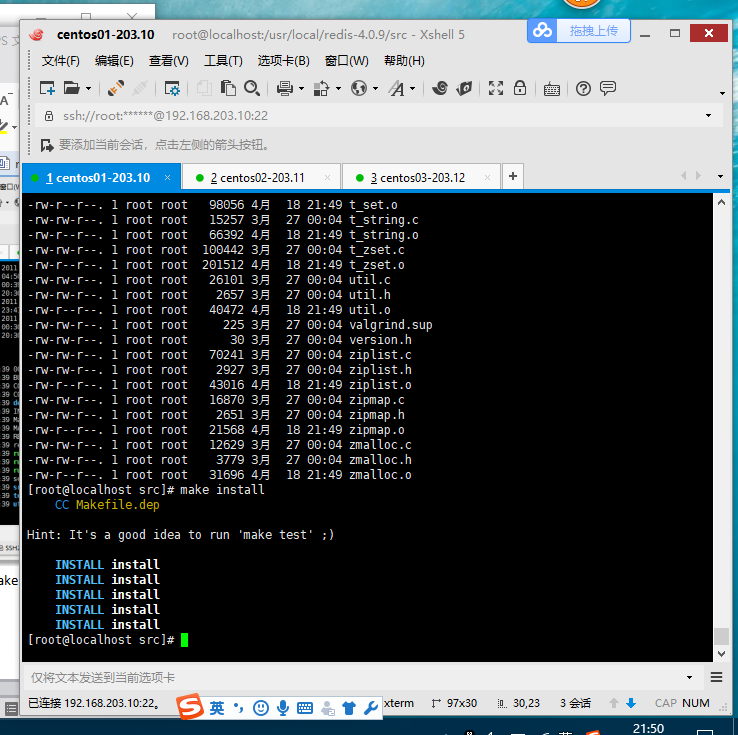
1. 首先上官网下载redis压缩包，地址：http：//redis.io/download下载文档版本4.0.8或.9即可
2. 通过远程管理工具，将压缩包看到liunx服务器中，执行压缩操作

tar -zxvf 对应的压缩包

1. 执行make对压缩后的文件进行编译



1. 进入src目录下执行make install



安装完成后，界面如上，如果出现安装错误则查看最后一页中yum问题修复方案

1. reid的部署

安装完成后，下面对redis进行部署

首先为了放屁管理，将redis文件中的conf配置文件和常用命令移动到同一文件中

创建bin和etc文件

代码如下

|  |
| --- |
| mkdir -p /usr/localredis/bin  mkdir -p /usr/local/redis/ect |

执行liunx文件移动命令:

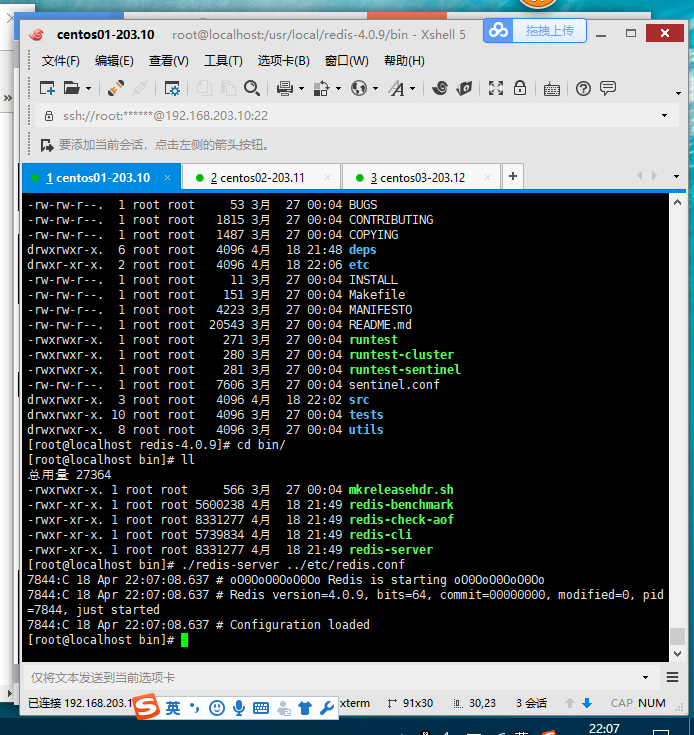
|  |
| --- |
| mv redis.conf etc/  cd src/  mv mkreleasehdr.sh redis-benchmark redis-check-aof redis-cli redis-server /user/local/redis/bin |

后台启动redis服务

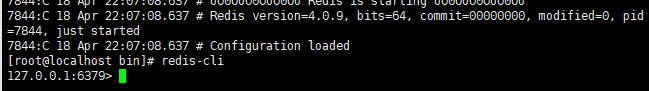
切换到/usr/local/redis/etc目录，编辑redis.conf文件，将daemonize属性改为yes（表明需要在后台运行）

|  |
| --- |
| vim redis.conf |

切换到/usr/local/redis/bin目录下执行redis-server命令，使用/usr/local/redis/etc/redis.conf配置文件来启动redis服务



服务器启动成功后，执行redis-cli启动redis客户端，查看端口号，默认6379



后续：https://www.cnblogs.com/KunGe-13/p/8340309.html

## Redis集群应用

|  |
| --- |
| centos-yum update问题-103  在centOs上安装软件时。提示报错信息：   Could not retrieve mirrorlisthttp://mirrorlist.centos.org/?release=6&arch=x86\_64&repo=os error was  14: PYCURL ERROR 6 - "Couldn't resolvehost 'mirrorlist.centos.org'"  Could not retrieve mirrorlisthttp://mirrorlist.centos.org/?release=6&arch=x86\_64&repo=extras errorwas  14: PYCURL ERROR 6 - "Couldn't resolvehost 'mirrorlist.centos.org'"  Could not retrieve mirrorlisthttp://mirrorlist.centos.org/?release=6&arch=x86\_64&repo=updates errorwas  14: PYCURL ERROR 6 - "Couldn't resolvehost 'mirrorlist.centos.org'"  http://mirrors.163.com/centos/6.2/os/x86\_64/repodata/repomd.xml:[Errno 14] PYCURL ERROR 6 - "Couldn't resolve host 'mirrors.163.com'"  Trying other mirror.  Error: Cannot retrieve repository metadata(repomd.xml) for repository: base. Please verify its path and try again  [root@hbase01-222 ~]# service iptables stop  [root@hbase01-222 ~]# nslookup  mirrorlist.centos.org  ;; connection timed out; no servers couldbe reached  到这里发现问题了，我的域名解析不了，yum的配置文件里指向的域名就识别不了了。重新设置下dns指向。  [root@hbase01-222 ~]# vim /etc/resolv.conf  # Generated by NetworkManager    nameserver 8.8.8.8  [root@hbase01-222 ~]# yum install gcc-c++ |

## Redis操作命令

Key操作

注:Redis是一个key-value的键值对的内存数据库，最基本的一些操作就是对这些key的操作

DEL

删除key, del key1 key2

EXISTS

检查key是否存在，exists key

EXPIRE

设置或更新到期时间，到期后自动清除，单位秒设置为 -1表示永不过期，EXPIER key

PERSIST

移除过期时间，key永久保存，其实就是过期时间设置为 -1，永久不过期

PTTL

以毫秒为单位返回key剩余的过期时间。

TTL

以秒为单位，返回给定key的摄于生存时间。

EXPIREAT

设置key的过期时间，不过设置的是时间戳。

KEYS pattern

查找匹配给定模式pattern的所有key

KEYS \* 匹配数据库中所有的key

KEYS h?llo 匹配hello ，hallo 和 hxllo等

KEYS h\*llo 匹配hllo和heeeello等

KEYS h[as]llo 匹配hello和hallo,但不匹配hillo。

特殊符号用\隔开

MOVE KEY db

把制定的key指定到数据库db中(默认的redis有16个数据库)

SECLETE index

切换数据库（redis默认16个数据库 编号从0开始）

RANDOMKEY

随机获取一个key

RENAME KEY newkey

修改key名字，如果newkey已经存在，则删除newkey。

RENAMENX KEY newkey

仅当newkey不存在时，将key改名为newkey。

TYPE key

返回key的类型

String操作

Get

获取

MEGET

获取多个

SET

设置（新增 修改）

MSET

设置一个

GETRANGE

获取范围段

比如value是 qwertyuiop

正序 倒序两种情况

Getrange key 0-4 （qwert）

Getrange key -1 -5 （yuiop）

Getrange key 0 -1 （qwertyuiop）

P w e r t y u i o p

0 1 2 3 4 5 6 7 8 9

-10 -9 -8 -7 -6 -5 -4 -3 -2 -1

GETSET

设置一个值，然后在返回原来老值。

SETEX KEY secondes value

设置一个key，并定制他的过期时间，单位秒

Psetex key milliseconds value

设置一个key，并制定他的过期时间，单位毫秒

SETNX

只有字key不存在是设置value，就是新增一个（包含更新）

MSETNX

同时设置一个或多个key-value对，当且仅当所有给定key都不存在

APPEND key value

追加到末尾。

STRLEN key

长度

INCR key

把key中的数字值添加1

INCRBY key increment

将key所存储的值增加一定数量

INCRBYFLOAY key increment

增加一点的浮点数

DECR key

将key存储数字键1

DEVRBY key decrement

减去给你的减量值

Hash操作

Hash是我们最常用的，因为他可以存储对象，字典等。hash里面是一个string类型的key value映射集合，每一个都叫做域。

HMSET key field value[field value....]

同事将多个field-value（域-值）对设置到哈希表key中

HSET key field value

将哈希表key中的域field的设置为value

如果key不存在，一个新的哈希表被创建并进行HSET操作

如果域field已经存在于哈希表中，旧值将被覆盖

HSETNX key field value

将哈希表key中的域field的设置为value，当且仅当域field不存在，若域field已经存在，改操作无效。

如果key不存在，一个新哈希表被创建并执行HSETNX命令，就是不能更新操作。

HMGET key filed[field ...]

返回哈希表key中，一个或多个给定域的值。

如果给定的域不存在于哈希表，那么返回一个nil值的表。

HGET key field

返回哈希表key中给定域field值

HGETALL key

返回哈希表key中，所有的域和值

在返回值里，紧跟每个域名（field name）之后是域的值（value），所有返回值得长度是哈希表大小的两倍

HEXISTS key field

查看哈希表key中，给定域field是否存在。

如果哈希表含有给定域，返回1

如果哈希表不含有给定域，或key不存在返回0

HDEL key field[field ...]

删除域

HKEYS key

返回哈希表中key中的所有域

HVALS key

返回哈希表key中域的数量

HINCRBY key field increment

为哈希表key中的域field的值增量increment。

增量也可以为负数，相当于对给定域进行减法操作

如果key不存在，一个新的哈希表被创建并执行HINCREBYmingl。

如果域field存在那么执行命令前，域的值被初始化为0.

HINCRBYFLOAY key field increment

为哈希表key中域field加上浮点数量increement

如果哈希表中没有域field那么HINCREBYFLOAT会将域field的设置为0，然后在执行加法操作。

如果键key不存在，那么HINCRBYFLOAY会创建一个哈希表，在创建域field，最后在执行加法操作。

List操作

列表的特点就是可以头部（左边）或尾部（右边）插入，入栈push，出栈pop操作

LPUSH key value[value ..]

将一个或多个值value插入到列表key的表头

如果key不存在，一个空列表会被创建并执行LPUSH操作

当key存在但不是类别类型时，返回一个错误。

LSET key index value

通过索引设置值

LPUSHX key value

将value插入到列表key的表头，当且仅当key存在并且是一个列表

RPUSH key value[value ..]

将一个或多个值value插入到列表key的表尾（最右边）

如果key不存在，一个空列表会被创建 并执行RPUSH操作。

ROUSHX key value

将值value插入到列表key 的表尾，当且仅当key存在并且是一个列表。

Lpop key

移除并放回类别key的尾元素

BLPOP key[key ...] timeout

移出并获取列表的第一个元素，如果列表没有元素会阻塞列表直到等待超时或发现可单出元素位置。

BRPOP key[key ...] timeout

移出并获取列表的最后一个元素，如果列表没有元素会堵塞列表知道等待超时或发现可弹出元素为止。

RPOPLPUSH source destination

移出列表的最后一个元，并将改元素添加到另一个列表并返回

BRPOPLLPUSH source destination timeout

从列表中弹出一个元素插入到另外一个列表中并返回它，如果类别没有元素会阻塞列表直到等待超时或发现可弹出元素为止。

LINDEX key index

返回列表key中，下标为index的元素

LINSERT key BEFORE|AFTER pivot value

将值value插入到列表key当中，位于值prvot之前或之后。

当pivot不存在于列表key是，不执行任何操作，多个pivot时值操作第一个。

当key不存在时，key被视为空类别，不执行任何操作。

如果key不是类别类型，然后一个错误。

LLEN key

里边数量

LREM key count value

移出count个value值

LRANGE key start stop

查看范围段的value（可以用来分页）

Lrange list 0 5 （第一个到第六个）

LTRIM key start stop

对一个类别进行修剪（trim），就是说让列表直保留指定区区间元素，不在指定区间之内的元素将被删除。

Set操作

Redis的set是string类型的有序集合，集合成员是唯一的，这就意味着集合不能出现重复的数据。

集合除了增删改操作外，主要的是一些集合的运算操作。

SADD key memder[memder ...]

将一个或多个member元素计入到集合key当中，已经存在于集合的member元素被忽略。

假如key不存在，则创建一个值包含member元素成员的集合。

SCARD key

返回集合key的基数（集合元素的数量）

SISMEBER key member

判断member元素是否集合key的成员，返回1表示存在0表示不存在。

SMEMBERS key

返回集合key中的所有成员。

不存在的key被视为空集合。

SMOVE sourve destination member

将member元素冲source集合移动到destination集合

SMOVE是原子性操作。

如果 source 集合不存在或不包含指定的 member 元素，则 SMOVE 命令不执行任何操作，仅返回 0 。否则， member 元素从 source 集合中被移除，并添加到 destination 集合中去。  
当 destination 集合已经包含 member 元素时， SMOVE 命令只是简单地将 source 集合中的 member 元素删除。  
当 source 或 destination 不是集合类型时，返回一个错误

**SRANDMEMBER key [count]**

如果命令执行时，只提供了 key 参数，那么返回集合中的一个随机元素。

从 Redis 2.6 版本开始， SRANDMEMBER 命令接受可选的 count 参数：  
如果 count 为正数，且小于集合基数，那么命令返回一个包含 count 个元素的数组，数组中的元素各不相同。如果 count 大于等于集合基数，那么返回整个集合。  
如果 count 为负数，那么命令返回一个数组，数组中的元素可能会重复出现多次，而数组的长度为 count 的绝对值。  
该操作和 SPOP 相似，但 SPOP 将随机元素从集合中移除并返回，而 SRANDMEMBER 则仅仅返回随机元素，而不对集合进行任何改动。

**SREM key member [member ...]**

移除集合 key 中的一个或多个 member 元素，不存在的 member 元素会被忽略。  
当 key 不是集合类型，返回一个错误。

集合运算

**SDIFF key [key ...]**

返回一个集合的全部成员，该集合是所有给定集合之间的差集。

**SDIFFSTORE destination key [key ...]**

这个命令的作用和 SDIFF 类似，但它将结果保存到 destination 集合，而不是简单地返回结果集。  
如果 destination 集合已经存在，则将其覆盖。  
destination 可以是 key 本身。(如果destination不存在  则新建一个)

**SINTER key [key ...]**

返回一个集合的全部成员，该集合是所有给定集合的交集。  
不存在的 key 被视为空集。  
当给定集合当中有一个空集时，结果也为空集(根据集合运算定律)。

**SDIFFSTORE destination key [key ...]**

这个命令的作用和 SDIFF 类似，但它将结果保存到 destination 集合，而不是简单地返回结果集。  
如果 destination 集合已经存在，则将其覆盖。  
destination 可以是 key 本身。

**SUNION key [key ...]**

返回一个集合的全部成员，该集合是所有给定集合的并集。  
不存在的 key 被视为空集。

**SUNIONSTORE destination key [key ...]**

这个命令类似于 SUNION 命令，但它将结果保存到 destination 集合，而不是简单地返回结果集。  
如果 destination 已经存在，则将其覆盖。  
destination 可以是 key 本身。

ZSET操作

Redis有序集合和集合一样也是string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正式通过分数来为集合中的成员进行从小到大的排序。

Zset和set操作基本一样，比set多了一点。

**ZADD key score member [[score member] [score member] ...]**

将一个或多个 member 元素及其 score 值加入到有序集 key 当中。  
如果某个 member 已经是有序集的成员，那么更新这个 member 的 score 值，并通过重新插入这个 member 元素，来保证该 member 在正确的位置上。  
score 值可以是整数值或双精度浮点数。  
如果 key 不存在，则创建一个空的有序集并执行 ZADD 操作。  
当 key 存在但不是有序集类型时，返回一个错误。

**ZCARD key**

返回有序集 key 的个数。

**ZSCORE key member**

返回有序集 key 中，成员 member 的 score 值。  
如果 member 元素不是有序集 key 的成员，或 key 不存在，返回 nil 。

**ZCOUNT key min max**

返回有序集 key 中， score 值在 min 和 max 之间(默认包括 score 值等于 min 或 max )的成员的数量。  
关于参数 min 和 max 的详细使用方法，请参考 ZRANGEBYSCORE 命令。

**ZINCRBY key increment member**

为有序集 key 的成员 member 的 score 值加上增量 increment 。  
可以通过传递一个负数值 increment ，让 score 减去相应的值，比如 ZINCRBY key -5 member ，就是让 member 的 score 值减去 5 。  
当 key 不存在，或 member 不是 key 的成员时， ZINCRBY key increment member 等同于 ZADD key increment member 。  
当 key 不是有序集类型时，返回一个错误。  
score 值可以是整数值或双精度浮点数。

**ZRANGE key start stop [WITHSCORES]**

这个可以用来做分页。

通过索引区间返回有序集合成指定区间内的成员。

返回有序集 key 中，指定区间内的成员。  
其中成员的位置按 score 值递增(从小到大)来排序。  
具有相同 score 值的成员按字典序(lexicographical order )来排列。  
如果你需要成员按 score 值递减(从大到小)来排列，请使用 ZREVRANGE 命令。  
下标参数 start 和 stop 都以 0 为底，也就是说，以 0 表示有序集第一个成员，以 1 表示有序集第二个成员，以此类推。  
你也可以使用负数下标，以 -1 表示最后一个成员， -2 表示倒数第二个成员，以此类推。  
超出范围的下标并不会引起错误。  
比如说，当 start 的值比有序集的最大下标还要大，或是 start > stop 时， ZRANGE 命令只是简单地返回一个空列表。  
另一方面，假如 stop 参数的值比有序集的最大下标还要大，那么 Redis 将 stop 当作最大下标来处理。  
可以通过使用 WITHSCORES 选项，来让成员和它的 score 值一并返回，返回列表以 value1,score1, ..., valueN,scoreN 的格式表示。  
客户端库可能会返回一些更复杂的数据类型，比如数组、元组等。

**ZREVRANGE key start stop [WITHSCORES]**

可以做分页

返回有序集 key 中，指定区间内的成员。  
其中成员的位置按 score 值递减(从大到小)来排列。  
具有相同 score 值的成员按字典序的逆序(reverse lexicographical order)排列。  
除了成员按 score 值递减的次序排列这一点外， ZREVRANGE 命令的其他方面和 ZRANGE 命令一样。

**ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]**

返回有序集 key 中，所有 score 值介于 min 和 max 之间(包括等于 min 或 max )的成员。有序集成员按 score 值递增(从小到大)次序排列。  
具有相同 score 值的成员按字典序(lexicographical order)来排列(该属性是有序集提供的，不需要额外的计算)。  
可选的 LIMIT 参数指定返回结果的数量及区间(就像SQL中的 SELECT LIMIT offset, count )，注意当 offset 很大时，定位 offset 的操作可能需要遍历整个有序集，此过程最坏复杂度为 O(N) 时间。  
可选的 WITHSCORES 参数决定结果集是单单返回有序集的成员，还是将有序集成员及其 score 值一起返回。  
该选项自 Redis 2.0 版本起可用。

**ZREVRANGEBYSCORE key max min [WITHSCORES] [LIMIT offset count]**

返回有序集 key 中， score 值介于 max 和 min 之间(默认包括等于 max 或 min )的所有的成员。有序集成员按 score 值递减(从大到小)的次序排列。  
具有相同 score 值的成员按字典序的逆序(reverse lexicographical order )排列。  
除了成员按 score 值递减的次序排列这一点外， ZREVRANGEBYSCORE 命令的其他方面和 ZRANGEBYSCORE 命令一样。

**ZRANK key member**

返回有序集 key 中成员 member 的排名。其中有序集成员按 score 值递增(从小到大)顺序排列。  
排名以 0 为底，也就是说， score 值最小的成员排名为 0 。  
使用 ZREVRANK 命令可以获得成员按 score 值递减(从大到小)排列的排名。

**ZREVRANK key member**

返回有序集 key 中成员 member 的排名。其中有序集成员按 score 值递减(从大到小)排序。  
排名以 0 为底，也就是说， score 值最大的成员排名为 0 。  
使用 ZRANK 命令可以获得成员按 score 值递增(从小到大)排列的排名。

**ZLEXCOUNT KEY MIN MAX**

计算有序集合中指定字典区间内成员数量。

**ZREM key member [member ...]**

移除有序集 key 中的一个或多个成员，不存在的成员将被忽略。  
当 key 存在但不是有序集类型时，返回一个错误。

**ZREMRANGEBYRANK key start stop**

移除有序集 key 中，指定排名(rank)区间内的所有成员。  
区间分别以下标参数 start 和 stop 指出，包含 start 和 stop 在内。  
下标参数 start 和 stop 都以 0 为底，也就是说，以 0 表示有序集第一个成员，以 1 表示有序集第二个成员，以此类推。  
你也可以使用负数下标，以 -1 表示最后一个成员， -2 表示倒数第二个成员，以此类推。

**ZREMRANGEBYSCORE key min max**

移除有序集 key 中，所有 score 值介于 min 和 max 之间(包括等于 min 或 max )的成员。  
自版本2.1.6开始， score 值等于 min 或 max 的成员也可以不包括在内，详情请参见 ZRANGEBYSCORE 命令。

下面 集合运算

**ZINTERSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]**

计算给定的一个或多个有序集的**交集**，**其中给定 key 的数量必须以 numkeys 参数指定**，并将该交集(结果集)储存到 destination 。  
默认情况下，结果集中某个成员的 score 值是所有给定集下该成员 score 值之和.  
关于 WEIGHTS 和 AGGREGATE 选项的描述，参见 ZUNIONSTORE 命令。

**ZUNIONSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]**

计算给定的一个或多个有序集的**并集**，其中给定 key 的数量必须以 numkeys 参数指定，并将该并集(结果集)储存到 destination 。  
默认情况下，结果集中某个成员的 score 值是所有给定集下该成员 score 值之 和 。  
WEIGHTS  
使用 WEIGHTS 选项，你可以为 每个 给定有序集 分别 指定一个乘法因子(multiplication factor)，每个给定有序集的所有成员的 score 值在传递给聚合函数(aggregation function)之前都要先乘以该有序集的因子  
如果没有指定 WEIGHTS 选项，乘法因子默认设置为 1 。  
AGGREGATE  
使用 AGGREGATE 选项，你可以指定并集的结果集的聚合方式。  
默认使用的参数 SUM ，可以将所有集合中某个成员的 score 值之 和 作为结果集中该成员的 score 值；使用参数 MIN ，可以将所有集合中某个成员的 最小 score 值作为结果集中该成员的 score 值；而参数 MAX 则是将所有集合中某个成员的 最大 score 值作为结果集中该成员的 score 值。