# Redis

**1、常用数据结构**

string 最常见key-value结构。

常用命令：

SET key value

GET key

MSET key1 value1 key2 value2... 一次存储多个key-value

MGET key1 key2... 一次获取多个key-value

hash 哈希表，会不有重复的field。

常用命令：

HGET key field

HSET key field value

HGETALL key

HDEL key field

HKEYS key

list 列表，并非传统先进先出的列表，是一个前后都可以插入数据的双端列表。类似于数据结构中的双向链表。

常用命令：

LPUSH key value 列表首部加入元素

RPUSH key value 列表尾部加入元素

LLEN key列表长度

LPOP key 弹出列表首部元素

RPOP key 弹出列表尾部元素

LRANGE key start end 输出列表中下标从start到end的元素，下标从0开始。

set 集合，成员是唯一的，集合本身无序。

常用命令：

SADD key value1 value2... 向集合中加入一个或者多个成员

SREM key value1 value2... 移除集合中的一个或者多个成员

SMEMBERS key 返回集合中的所有成员

SISMEMBER key value 判断value是不是集合成员

sorted set 有序集合，通过score字段进行排序。

常用命令：

ZADD key score1 value1 score2 value2... 向有序集合中加入一个或多个成员，或者更新成员分数

ZCOUNT key min max 获取有序集合中指定分数区间的成员个数

ZRANK key value 返回有序集合中指定成员的索引，索引从0开始

ZSCORE key value 返回有序集合中成员的分数值

ZREM key value1 value2... 移除有序集合中一个或者多个成员

1. **事务**

事务是一次性执行多个命令。事务具有原子性，即在事务执行完成之前不会执行其他的命令。

常用命令：

MULTI 开始事务

SET xxx xxx

GET xxx xxx

...（输入事务需要执行的命令）

EXEC 执行事务

事务执行出错的处理

EXEC之前出错：

MULTI

SET abc

GET abc

EXEC

这里SET命令存在语法错误，所以最后EXEC是不会执行之前任何命令的。

EXEC执行中出错

MULTI

SADD member 1 2 3 4

RPUSH user 5 6

SADD password 111 222

EXEC

这里RPUSH错误对user使用了列表命令。EXEC执行这条命令会出错，但不影响其他命令的执行。

WATCH命令监控了某个key，如果这个key在EXEC命令之前被修改，则整个事务会被取消，EXEC返回nil-reply，不论事务中包含多少个命令。

事务中的命令要求操作的对象必须在相同的slot（redis分区中的槽）之中，否则会按照上面第一种出错情况处理。

1. **管道pipeline**

pipeline的作用在于将多个命令一起发送到redis执行，节省RTT时间。

由于redis本身是单进程单线程，对于客户端发来的命令是顺序执行的。而客户端发送redis命令是阻塞式的，每发送一个命令要等这个命令执行结果返回之后才会发送下一个命令。

这种方式中间会消耗大量的RTT时间，而pipeline就是将多个命令合并成一个请求发送给redis，redis执行完成之后响应客户端。

pipeline和事务的区别在于，pipeline不保证原子性，redis在执行pipeline发送的多条命令过程中可能执行其他客户端发来的命令。

hiredis的pipeline实现是redisAppendCommand，这个函数调用的时候实际上是缓存了要发送给redis的命令。函数执行完成命令并没有发送到redis上，而是在用户调用redisGetReply的时候才将命令发送给redis并等待应答。

1. **发布订阅**
2. **数据持久化**

两种方式：snapshotting和append-only file

snapshotting（快照）：设置n秒内超过m个key被修改就自动做快照。

append-only file（AOF）：redis每收到一条写命令都将修改的内容写入到文件中。

可以指定强制写入硬盘的时机：

appendfsync alway，收到写命令立即写入磁盘

appendfsync everysec，每秒写磁盘一次

appendfsync no，不强制，依赖操作系统策略

1. **数据过期机制**

定期删除+惰性删除

定期删除：redis默认100ms随机抽取一些key进行检查，发现过期就删除。

惰性删除：在get操作的时候判断key是否过期，如过期就删除。

1. **内存淘汰机制**

6种内存淘汰机制

no-envication 内存不足，写入操作报错

allkeys-lru 内存不足，在key空间内移除使用最少的key

allrandom-lru 随机删除几个key

voliate-lru 在设置过期时间的key空间中，删除使用最少的key

voliate-random 在设置过期时间的key空间中，随机删除key

vliate-ttl 在设置过期时间的key空间中， 删除更早过期时间的key

# Redis-Sentinel