Redis命令速查表（cheatsheet）

本文档的内容覆盖 Redis 3.0 版本，文档释出日期为 2015 年 11 月 29 日。

字符串（string）

设置与获取

`SET key value [EX seconds] [PX milliseconds] [NX|XX]` —— 为字符串键设置值和过期时间（可选）

`GET key` —— 获取字符串键的值

`GETSET key new-value` —— 为字符串键设置新值，并返回键被设置之前的旧值

`SETNX key value` —— 仅在字符串键尚未有值的情况下，为它设置值

`SETEX key seconds value` —— 为字符串键设置值和秒级精度的过期时间

`PSETEX key milliseconds value` —— 为字符串键设置值和毫秒级精度的过期时间

批量设置与获取

`MSET key value [key value ...]` —— 一次为多个字符串键设置值

`MGET key [key ...]` —— 一次获取多个字符串键的值

`MSETNX key value [key value ...]` —— 仅在所有给定字符串键都尚未有值的情况下，为它们设置值

获取或修改内容

`STRLEN key` —— 获取字符串值的长度

`SETRANGE key offset value` —— 对字符串值在指定索引位置上的内容进行修改

`GETRANGE key start end` —— 获取字符串值在指定索引范围内的内容

`APPEND key value` —— 将指定的内容追加到字符串值的末尾

自增与自减

`INCR key` —— 为字符串键储存的整数值加上一

`DECR key` —— 为字符串键储存的整数值减去一

`INCRBY key increment` —— 为字符串键储存的整数值加上指定的整数增量

`DECRBY key decrement` —— 为字符串键储存的整数值减去指定的整数减量

`INCRBYFLOAT key increment` —— 为字符串键储存的数字值加上指定的浮点数增量

散列（hash）

设置与获取

`HSET hash key value` —— 为散列中的键设置值

`HSETNX hash key value` —— 仅在散列中的给定键尚未有值的情况下，为该键设置值

`HGET hash key` —— 返回散列中与给定键相关联的值

`HMSET hash key value [key value]` —— 一次为散列中的多个键设置值

`HMGET hash key [key ...]` —— 一次获取散列中多个键的值

自增与自减

`HINCRBY hash key increment` —— 为散列中给定键储存的整数值加上指定的整数增量

`HINCRBYFLOAT hash key increment` —— 为散列中给定键储存的数字值加上指定的浮点数增量

检测与管理

`HEXISTS hash key` —— 检查给定键在散列中是否存在

`HLEN hash` —— 返回散列包含的键值对数量

`HDEL hash key [key ...]` —— 删除散列中的一个或多个键，以及这些键的值

批量获取散列键值

`Has hash` —— 返回散列包含的所有键

`HVALS hash` —— 返回散列包含的所有键的值

`HGETALL hash` —— 返回散列包含的所有键值对

`HSCAN hash cursor [MATCH pattern] [COUNT count]` —— 以渐进的方式返回散列包含的键值对

列表（list）

推入元素

`LPUSH list item [item ...]` —— 将一个或多个元素推入到列表的左端

`LPUSHX list item` —— 仅在列表已经存在的情况下，将一个元素推入到列表的左端

`RPUSH list item [item ...]` —— 将一个或多个元素推入到列表的右端

`RPUSHX list item` —— 仅在列表已经存在的情况下，将一个元素推入到列表的右端

弹出元素

`LPOP list` —— 移除并返回列表左端第一个元素

`RPOP list` —— 移除并返回列表右端第一个元素

`BLPOP list [list ...] timeout` —— 在指定的时限内，弹出首个非空列表的最左端元素

`BRPOP list [list ...] timeout` —— 在指定的时限内，弹出首个非空列表的最右端元素

弹出元素然后推入元素

`RPOPLPUSH source_list target_list` —— 弹出源列表的最右端元素，并将该元素推入到目标列表的左端

`BRPOPLPUSH source_list target_list timeout` —— 在指定的时限内，尝试弹出源列表的最右端元素，并将该元素推入到目标列表的左端

元素的获取与管理

`LINDEX list index` —— 获取列表在给定索引上的元素

`LLEN list` —— 返回列表包含的元素数量

`LRANGE list start end` —— 返回列表在指定索引范围内的所有元素

`LINSERT list BEFORE|AFTER target item` —— 将给定的元素插入到目标元素的前面或者后面

`LREM list count item` —— 从列表中移除给定的元素

`LSET list index item` —— 把列表在指定索引上的值修改为给定的元素

`LTRIM list start end` —— 对列表进行修剪，只保留指定索引范围内的元素

集合（set）

元素的添加与移除

`SADD set element [element ...]` —— 将一个或多个元素添加到集合当中

`SPOP set` —— 随机地移除并返回集合中的某个元素

`SMOVE source_set target_set element` —— 将指定的元素从源集合移动到目标集合

`SREM set element [element ...]` —— 移除集合中的一个或多个元素

元素的获取与检测

`SCARD set` —— 返回集合包含的元素数量

`SISMEMBER set element` —— 检查集合是否包含了给定的元素

`SRANDMEMBER set [count]` —— 随机地返回集合包含的元素

`SMEMBERS set` —— 返回集合包含的所有元素

`SSCAN set cursor [MATCH pattern] [COUNT count]` —— 以渐进的方式返回集合包含

的元素

集合运算

`SDIFF set [set ...]` —— 计算并返回多个集合的差集计算结果

`SDIFFSTORE target_set set [set ...]` —— 对多个集合执行差集计算，并将结果储存到目标集合当中

`SINTER set [set ...]` —— 计算并返回多个集合的交集计算结果

`SINTERSTORE target_set set [set ...]` —— 对多个集合执行交集计算，并将结果储存到目标集合当中

`SUNION set [set ...]` —— 计算并返回多个集合的并集计算结果

`SUNIONSTORE target_set set [set ...]` —— 对多个集合执行并集计算，并将结果储存到目标集合当中

有序集合（sorted set）

成员的检测与管理

`ZADD sorted_set score member [[score member] [score member] ...]` —— 将给定的成员及其分值添加到有序集合

`ZINCRBY sorted_set increment member` —— 为成员的分值加上指定的整数增量

`ZSCORE sorted_set member` —— 返回给定成员的分值

`ZCARD sorted_set` —— 返回有序集合包含的成员数量

`ZRANK sorted_set member` —— 返回有序集合成员在按照分值从小到大进行排列时，给定的成员在有序集合中所处的排名

`ZREVRANK sorted_set member` —— 返回有序集合成员在按照分值从大到小进行排列时，给定的成员在有序集合中所处的排名

批量处理多个成员

`ZCOUNT sorted_set min max` —— 返回有序集合中，分值介于指定区间之内的成员数量

`ZRANGE sorted_set start end [WITHSCORES]` —— 按照分值从小到大的顺序，返回指定索引范围之内的成员及其分值（可选）

`ZREVRANGE sorted_set start end [WITHSCORES]` —— 按照分值从大到小的顺序，返回指定索引范围之内的成员及其分值（可选）

`ZRANGEBYSCORE sorted_set min max [WITHSCORES] [LIMIT offset count]` —— 按照分值从小到大的顺序，返回指定分值范围之内的成员

`ZREVRANGEBYSCORE sorted_set max min [WITHSCORES] [LIMIT offset count]` —— 按照分值从大到小的顺序，返回指定分值范围之内的成员

`ZSCAN sorted_set cursor [MATCH pattern] [COUNT count]` —— 以渐进的方式，返回有序集合包含的成员及其分值

`ZREM sorted_set member [member ...]` —— 从有序集合中移除指定的一个或多个成员

`ZREMRANGEBYRANK sorted_set start end` —— 移除有序集合中，位于指定排名范围内的成员，其中成员按照分值从小到大进行排列

`ZREMRANGEBYSCORE sorted_set min max` —— 移除有序集合中，分值位于指定范围内的成员

集合运算

`ZINTERSTORE target number [sorted_set ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]` —— 对给定数量的有序集合执行交集计算，并将计算的结果储存到目标有序集合里面

`ZUNIONSTORE target number [sorted_set ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]` —— 对给定数量的有序集合执行并集计算，并将计算的结果储存到目标有序集合里面

根据成员的大小对其进行处理

`ZLEXCOUNT sorted_set min max` —— 统计有序集合里面，位于指定大小范围内的成员的数量

`ZRANGEBYLEX sorted_set min max [LIMIT offset count]` —— 按照从小到大的顺序，返回有序集合里面位于指定大小范围之内的成员

`ZREMRANGEBYLEX sorted_set min max` —— 从有序集合里面，移除位于指定大小范围之内的成员

位图（bitmap）

设置或获取单个位

`SETBIT bitmap index value` —— 为位图在指定索引上的二进制位设置值

`GETBIT bitmap index` —— 获取位图在给定索引上的二进制位的值

对多个位进行计算或操作

`BITCOUNT bitmap [start] [end]` —— 统计位图中值为 1 的二进制位的数量

`BITOP AND destination bitmap [bitmap ...]` —— 对任意多个位图执行逻辑并计算，并将结果储存到指定的位图里面

`BITOP OR destination bitmap [bitmap ...]` —— 对任意多个位图执行逻辑或计算，并将结果储存到指定的位图里面

`BITOP XOR destination bitmap [bitmap ...]` —— 对任意多个位图执行逻辑异或计算，并将结果储存到指定的位图里面

`BITOP NOT destination bitmap` —— 对给定的位图执行逻辑非计算，并将结果储存到指定的位图里面

HyperLogLog

添加元素

`PFADD hll element [element ...]` —— 将一个或多个元素添加到 HyperLogLog 里面

统计元素数量

`PFCOUNT hll` —— 统计 HyperLogLog 已包含的唯一元素数量

执行合并操作

`PFMERGE destination hll [hll ...]` —— 将多个 HpyerLogLog 合并为一个 HyperLogLog ，并将其储存到指定的键里面

地理位置（GEO）

添加或获取地理位置

`GEOADD geoset longitude latitude location [longitude latitude location ...]` —— 将给定的一个或多个地理位置添加到地理位置集合里面

`GEOPOS geoset location [location ...]` —— 从地理位置集合里面取出一个或多个地理位置的经纬度

计算范围或距离

`GEODIST geoset location1 location2 [unit]` —— 计算两个给定位置之间的距离

`GEORADIUS geoset longitude latitude radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [ASC|DESC] [COUNT count]` —— 以给定的经纬度为圆心，返回地理位置集合里面，所有位于圆心指定半径范围之内的地理位置及其经纬度

`GEORADIUSBYMEMBER geoset location radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [ASC|DESC] [COUNT count]` —— 以给定的地理位置为圆心，返回地理位置集合里面，所有位于圆心指定半径范围之内的其他地理位置及其经纬度

计算地理位置的Geohash值

`GEOHASH geoset location [location ...]` —— 为给定的一个或多个地理位置计算Geohash值

数据库（database）

获取

`KEYS pattern` —— 从数据库里面获取所有符合给定模式的键

`SCAN cursor [MATCH pattern] [COUNT count]` —— 以渐进的方式获取数据库中的键

`RANDOMKEY` —— 从数据库里面随机地返回一个键

`SORT key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern ...]] [ASC | DESC] [ALPHA] [STORE destination]` —— 对给定的键进行排序

检测

`EXISTS key` —— 检查给定的键是否存在于数据库

`DBSIZE` —— 返回当前正在使用的数据库包含的键值对数量

`TYPE key` —— 返回给定键储存的值的类型

管理

`RENAME key new-key` —— 为给定键设置一个新名字

`RENAMENX key new-key` —— 仅在新名字尚未被使用的情况下，为给定键设置一个新名字

`MOVE key db` —— 将当前数据库中的给定键移动到指定的数据库

`DEL key [key ...]` —— 从数据库中删除给定的一个或多个键

`FLUSHDB` —— 删除当前数据库中的所有键

`FLUSHALL` —— 删除服务器中，所有数据库的所有键

过期时间（expiration）

设置过期时间

`EXPIRE key seconds` —— 为键设置秒级精度的过期时间

`PEXPIRE key milliseconds` —— 为键设置毫秒级精度的过期时间

`EXPIREAT key timestamp-in-seconds` —— 为键设置秒级精度的过期 UNIX 时间戳

`PEXPIREAT key timestamp-in-milliseconds` —— 为键设置毫秒级精度的过期 UNIX 时间戳

查看剩余生存时间

`TTL key` —— 以秒级精度返回给定键的剩余生存时间

`PTTL key` —— 以毫秒级精度返回给定键的剩余生存时间

移除过期时间

`PERSIST key` —— 移除键的过期时间

事务（transaction）

基本事务操作

`MULTI` —— 开始一次事务

`EXEC` —— 执行事务

`DISCARD` —— 取消事务

乐观锁事务操作

`WATCH key [key ...]` —— 监视给定的键，看它们在事务执行之前是否已被修改

`UNWATCH` —— 取消对所有键的监视

脚本（script）

执行脚本

`EVAL script number_of_keys key [key ...] arg [arg ...]` —— 执行给定的 Lua 脚本

`EVALSHA sha1 number_of_keys key [key ...] arg [arg ...]` —— 执行与给定 SHA1 校验和相对应的已载入 Lua 脚本

脚本管理

`SCRIPT LOAD script` —— 载入给定的 Lua 脚本

`SCRIPT EXISTS script [script ...]` —— 检查给定的 Lua 脚本是否已被载入

`SCRIPT KILL` —— 杀死当前正在执行的 Lua 脚本

`SCRIPT FLUSH` —— 移除所有已载入脚本

发布与订阅（pub/sub）

发布消息

`PUBLISH channel message` —— 向指定频道发布一条消息

订阅消息

`SUBSCRIBE channel [channel ...]` —— 订阅给定的一个或多个频道

`PSUBSCRIBE pattern [pattern ...]` —— 订阅给定的一个或多个模式

退订消息

`UNSUBSCRIBE [channel [channel ...]]` —— 退订给定的一个或多个频道，如果没有给定频道则退订全部频道

`PUNSUBSCRIBE [pattern [pattern ...]]` —— 退订给定的一个或多个模式，如果没有给定模式则退订全部模式

查看订阅信息

`PUBSUB CHANNELS [pattern]` —— 列出当前被订阅的频道

`PUBSUB NUMSUB [channel channel ...]` —— 返回给定频道的订阅者数量

`PUBSUB NUMPAT` —— 返回当前被订阅模式的数量

客户端与服务器（client&server）

连接管理

`AUTH password` —— 使用给定的密码连接服务器

`ECHO message` —— 让服务器打印指定的消息，用于测试连接

`PING` —— 向服务器发送一条 PING 消息，用于测试连接或者测量延迟值

`QUIT` —— 请求服务器关闭与当前客户端的连接

`SELECT number` —— 切换至指定的数据库

客户端管理

`CLIENT SETNAME name` —— 为当前客户端设置名字

`CLIENT GETNAME` —— 返回当前客户端的名字

`CLIENT LIST` —— 返回正在连接服务器的所有客户端的相关信息

`CLIENT KILL ip:port` —— 关闭指定的客户端

数据持久化

`SAVE` —— 创建一个 RDB 快照文件，这个命令在执行期间将阻塞所有客户端

`BGSAVE` —— 在后台异步地创建一个 RDB 快照文件，这个命令不会阻塞客户端

`BGREWRITEAOF` —— 对 AOF 文件进行重写，减少它的体积

`LASTSAVE` —— 返回服务器最后一次成功执行持久化操作的 UNIX 时间戳

配置选项管理

`CONFIG SET option value` —— 为给定的配置选项设置值

`CONFIG GET option` —— 返回给定配置选项的值

`CONFIG REWRITE` —— 对服务器的配置选项文件进行重写，并将改写后的文件储存在硬盘里面

`CONFIG RESETSTAT` —— 重置服务器的某些统计数据

服务器管理

`INFO [section]` —— 返回与服务器相关的统计信息

`TIME` —— 返回服务器当前的 UNIX 时间戳

`SHUTDOWN [SAVE|NOSAVE]` —— 关闭服务器

调试

`SLOWLOG GET [number]` —— 查看 slow log

`SLOWLOG LEN` —— 返回目前记录的 slow log 数量

`SLOWLOG RESET` —— 清空所有 slow log

`MONITOR` —— 实时打印出服务器执行的命令

`DEBUG SEGFAULT` —— 让 Redis 执行一个不合法的内存访问，导致它崩溃

`DEBUG OBJECT key` —— 查看与给定键有关的调试信息

`OBJECT REFCOUNT key` —— 查看键的值被引用的次数

`OBJECT ENCODING key` —— 查看键的值所使用的内部表示（底层实现）

`OBJECT IDLETIME key` —— 查看给定键的空闲时间

关于（about）

更新记录（change log）

2015 年 11 月 29 日，释出覆盖 Redis 3.0 版本的文档。