Redis入门记

by FangChannel

1.1 什么是Redis

远程字典服务器，**Re**mote **di**ctionary **s**erver
一个开源的基于内存的数据库，常用作键值存储、缓存和消息队列等
*什么是数据库见《一万小时计划11》
*按db-engines的排名，redis是连续几年最受欢迎的键值存储数据库

Redis通常将全部数据存储在内存中，也可以不时的将数据写入硬盘实现持久化，但仅用于重新启动后将数据加载回内存
*内存的速度比硬盘快一个数量级

1.1.1 发展简史

Salvatore Sanfilippo，意大利程序员
他最早使用传统数据库做了一个实时的 Web 日志分析器
因为对其性能不够满意，开发了 Redis
2020 年 6 月，Salvatore Sanfilippo 辞去了 Redis 维护者的职务

1.2 安装

1.2.1 Windows 10 16237+

开启WSL功能

*WSL，**W**indows **S**ubsystem for **L**inux，WSL 是一个用于在 Windows 上运行 Linux 可执行文件的兼容层；但也存在基于 Hyper-V 的 WSL 虚拟机

安装Ubuntu WSL版

*Ubuntu是基于Debian的Linux发行版，有机会详讲

https://www.microsoft.com/p/ubuntu/9nblggh4msv6

使用apt安装Redis

```
sudo apt update
sudo apt install redis-server
```

*apt是Debian的软件包管理器

启用Redis

```
sudo service redis-server start
```

1.2.2 老版本Windows

不推荐！微软官方提供的windows版redis，只有 Redis 3，且上次更新是2016

https://github.com/microsoftarchive/redis

1.2.3 MacOS

安装Homebrew

https://brew.sh

*Homebrew是一款开源的MacOS的软件包管理器

使用Homebrew安装Redis

```
brew install redis
```

启用Redis

```
brew services start redis
```

1.2.4 Linux

Ubuntu/Debian使用apt安装Redis

```
sudo apt update
```

sudo apt install redis-server

CentOS/Red Hat Enterprise Linux使用yum安装Redis
sudo yum update
*yum是也是一个软件包管理器
sudo yum install redis

启用Redis
sudo systemctl start redis

配置Redis自启动(可选)
sudo systemctl enable redis
喜欢动手的同学也可以在 /etc/redis 或 /etc 中找到 redis.conf
编辑 redis.conf，将 supervised 的值改为 systemd *默认为no

1.2.5 通用操作
前台启动redis服务器: redis-server
手动启动redis客户端: redis-cli

2.1 基本操作

官方文档：https://redis.io/documentation

2.1.1 数据库操作

*redis默认有16个数据库，编号为0~15，且默认访问0号数据库

select 数据库编号	选择指定数据库
dbsize	获取当前数据库键值对数量
flushdb	清空当前数据库
flushall	清空所有数据库

save	将数据保存至磁盘
bgsave	将数据异步保存至磁盘（后台：Background） *默认每两秒自动执行一次
lastsave	获取最后一次成功保存的unix秒

2.1.2 通用数据操作

*操作范围为当前数据库

keys 格式	查看符合指定格式的key，*为通配符
exists key1 [key2 ...]	查看是否存在一至多个指定的key
type key	按key查看value的数据类型
del key1 [key2 ...]	按key删除一至多个键值对

rename key1 key2	重命名key1，如果key2已经存在，其值会被覆盖
renamenx key1 key2	key2不存在时重命名key1

move key 数据库编号	按key将一个键值对移动到指定数据库
copy key1 key2	将key1的值拷贝给key2

2.2 字符串String

key-value，键-值

key1	value1
key2	value2
…	

2.2.1 基本操作

set key value [EX seconds|PX milliseconds|EXAT unix-time-seconds|PXAT unix-time-milliseconds|KEEPTTL] [NX|XX] [GET]	添加/修改一个键值对 EX(expire)：设置过期秒数 PX：设置过期毫秒数 EXAT：设置过期的unix秒 PXAT：设置过期的unix毫秒 NX：仅当成员不存在时添加成员 KEEPTTL：set时不重置ttl(见2.2.3) XX：仅当成员存在时修改成员 GET：修改一个键值对并返回原值，原值不存在则返回nil
get key	按key获取value
mset key1 value1 [key2 value2 …]	添加/修改一至多个键值对
mget key1 [key2 …]	按key获取一至多个value

append key value	在原有value后追加内容
strlen key	查看字符串长度
getrange key startindex endindex	获取范围时[startindex, endindex]的子串 *index从0开始，-n表示倒数第n个字符

*setnx key value	不再推荐，同set key value nx
*getset key value	不再推荐，同set key value get
msetnx key1 value1 [key2 value2 …]	批量版setnx

2.2.2 如果字符串的内容是数值

incr key	按key创建值为1的value，或使value增长(increase)1

incrby key 整数值	按key使value增长(increase)给定数值
incrbyfloat key 小数值	按key使value增长(increase)给定数值
decr key	按key使value减小(decrease)1
decrby key 数值	按key使value减小(decrease)给定数值

2.2.3 临时键值对

生存时间time to live，缩写为ttl，指键值对距离被删除的剩余秒数
*如果重新set，生存时间(time to live，ttl)将被重置

*以下操作支持各种数据类型

expire key 秒数	设定一个生存时间
ttl key	查看生存时间的剩余秒数 *key不存在则返回-2，永久键值对则返回-1
pexpire key 毫秒数	毫秒版expire
pttl key	毫秒版ttl
persist key	持久化(取消生存时间)

*setex key 秒数 value	不再推荐，同set key value ex 秒数
*psetex key 毫秒数 value	不再推荐，同set key value px 毫秒数

2.3 散列表Hash

key-field-value，键-字段-值

key1	field1	value1
	field2	value2
	…	
key2	field1	value1
	field2	value2
	…	
…		

2.3.1 基本操作

hset key field1 value1 [field2 value2 …] *老版本为hset为单个字段和值，多个为hmset	添加/修改一个键与一至多对字段和值
hget key field	按key和field获取一对value
hmget key field1 [field2 …]	按key和field获取一至多对value
hgetall key	按key获取field获取全部的field-value
hdel key field1 [field2 …]	删除一至多对field-value

hsetnx key field value	仅在field不存在时添加一对field-value

hkeys key	查看一个散列表中所有的field
hvals key	查看一个散列表中所有的value
hlen key	统计一个散列表中有多少对field-value
hexists key field	查看一个field是否存在

hstrlen key field	按key和field查看value的长度

2.3.2 如果value字符串的内容是数值

hincrby key field 整数值	按key和field使value增长(increase)给定数值
hincrbyfloat key field 小数值	按key和field使value增长(increase)给定数值

2.4 列表List

key-value0-value1-...，键-有序的值列队

key1	value0	value1	...
key2	value0	value1	...
...			

2.4.1 基本操作

rpush key value0 [value1 ...]	在列表右侧推入1至多个值
lpush key [... value1] value0	在列表左侧推入1至多个值
rpop key [数量]	从列表右侧弹出(指定数量的)值 *全部弹出后，key也会被删除
lpop key [数量]	从列表左侧弹出(指定数量的)值 *全部弹出后，key也会被删除

rpushx key value0 [value1 ...]	仅当列表存在时，在列表右侧推入1至多个值
lpushx key [... value1] value0	仅当列表存在时，在列表左侧推入1至多个值

lset key *index value	修改指定位置的值
linsert key before/after 定位value value	在定位value前/后插入一个值
lindex key *index	按索引查看值
lrange key *startindex *endindex	查看给定范围的值
llen key	查看队列长度
lrem key 数量 value	删除指定值 *数量为正代表从左侧开始删除，数量为负代表从右侧开始删除
ltrim key *startindex *endindex	将列表修剪到给定范围

*索引从0开始，-n表示倒数第n个

2.5 集合Set

key-stringX, stringY...，键-无序的不重复的成员

key1	stringX	stringY	...
key2	stringY	stringZ	...
...			

2.5.1 基本操作

sadd key stringX [stringY ...]	添加1至多个成员
srem key stringX [stringY ...]	删除1至多个成员
scard key	返回成员数量 *集合基数cardinality
sismember key string	查看是否存在指定成员
smismember key stringX [stringY ...]	批量查看是否存在指定成员
smembers key	查看集合中所有成员
srandmember key [数量]	随机查看指定数量的成员
spop key [数量]	随机取出指定数量的成员

smove key1 key2 string	将指定成员从集合1移至集合2
sinter key1 [key2 ...]	查看给定集合的交集
sinterstore newkey key1 [key2 ...]	存储给定集合的交集
sunion key1 [key2 ...]	查看给定集合的并集
sunionstore newkey key1 [key2 ...]	存储给定集合的并集
sdiff key1 [key2 ...]	查看给定集合的差集
sdiffstore newkey key1 [key2 ...]	存储给定集合的差集

2.6 有序集合ZSet

key-score1:stringX, score2:stringY…
键-按分数排序的不重复的成员

key1	score1	score2	…
	stringX	stringY	…
key2	score1	score2	…
	stringY	stringZ	…
…			

*分数为(范围为float64)的浮点数，且可以重复
*什么是float64/int53见《一万小时计划02》

2.6.1 基本操作

zadd key [nx/xx] [gt/lt] [ch] [incr] score1 stringX [score2 stringY…]	添加1至多个成员 NX：仅当成员不存在时添加成员 XX：仅当成员存在时修改成员 LT(less than)：仅当分数低于原有分数时才更新分数 GT(greater than)：仅当分数高于原有分数时才更新分数 *NX不可以与LT/GT同时使用 CH(changed)：返回变更的成员数量 *默认返回新增的成员数量 incr：累加分数 *选择incr时，只能操作一个分数–成员对
zrem key stringX [stringY …]	删除1至多个成员
zcount key minScore maxScore	返回指定分数区间内的成员数量 支持开区间：分数前加"(" 支持无穷大："–inf"负无穷大，"+inf"正无穷大
zscore key string	查看成员分数 *成员不存在时返回nil
zmscore key stringX [stringY …]	批量查看成员分数 *成员不存在时返回nil
zcard key	查看成员数量

zincrby key 数值 string	将指定成员的分数增加给定数值

2.6.2 按区间操作

zrange key start end [byscore/bylex] [rev] [limit 偏移量 查看数量] [withscores]	查看指定分数区间内的成员 BYSCORE：按分数生序排序 *BYSCORE支持开区间：分数前加"(" *BYSCORE支持无穷大："-inf"负无穷大，"+inf"正无穷大 BYLEX(lexicographical)：(分数相同时)按成员字符排序 *BYLEX指定的时字符串区间，且需要指定开闭："[string"表示闭区间，"(string"表示开区间 *BYLEX支持选取到开始或结束："-"表示开始，"+"表示结束 REV(reverse)：反转 *LIMIT用于指定查看范围，仅在开启BYSCORE/BYLEX时可用 WITHSCORES：带分数
*zrevrange key start end [withscores]	不再推荐，等效于zrange key start end rev [withscores]
zrangestore newkey key start end [byscore/bylex] [rev] [limit 偏移量 查看数量]	存储指定分数区间内的成员 参考zrange key start end [byscore/bylex] [rev] [limit 偏移量 查看数量]

*zrangebylex key startString endString [limit 偏移量 查看数量]	不再推荐，等效于zrange key startString endString bylex [limit 偏移量 查看数量]
*zrevrangebylex key startString endString [limit 偏移量 查看数量]	不再推荐，等效于zrange key startString endString bylex rev [limit 偏移量 查看数量]
zlexcount key startString endString	查看指定字符串区间内的成员数量 参考zrange key startString endString bylex
zremrangebylex key startString endString	删除指定字符串区间内的成员 参考zrange key startString endString bylex

*zrangebyscore key minScore maxScore [withscores] [limit 偏移量 查看数量]	不再推荐，等效于zrange key minScore maxScore bylex [limit 偏移量 查看数量] [withscores]
*zrevrangebyscore key minScore maxScore [withscores] [limit 偏移量 查看数量]	不再推荐，等效于zrange key minScore maxScore bylex rev [limit 偏移量 查看数量] [withscores]

zremrangebyscore key minScore maxScore	删除指定分数区间内的成员 参考zrange key minScore maxScore byscore

zrank key string	查看成员升序排名 *从第0名开始
zrevrank key string	查看成员降序排名 *从第0名开始
zremrangebyrank key startRank endRank	删除指定排名区间内的成员 *从第0名开始，-n表示倒数第n名

2.6.3 交集，并集，差集

zinter key的数量 key1 [key2 ...] [weights 权重1 [权重2 ...]] [aggregate sum|min|max] [withscores]	查看给定集合的交集 WEIGHTS：依次为每个集合的分数设置权重 *权重默认为1 AGGREGATE：新分数的计算方法 *默认为sum WITHSCORES：带分数 新分数=AGGREGATE(集合1中该成员的分数*权重1, 集合2中该成员的分数*权重2, ...)
zinterstore newkey key的数量 key1 [key2 ...] [weights 权重1 [权重2 ...]] [aggregate sum|min|max]	存储给定集合的交集
zunion key的数量 key1 [key2 ...] [weights 权重1 [权重2 ...]] [aggregate sum|min|max] [withscores]	查看给定集合的并集
zunionstore newkey key的数量 key1 [key2 ...] [weights 权重1 [权重2 ...]] [aggregate sum|min|max] [withscores]	存储给定集合的并集
zdiff key的数量 key1 [key2 ...] [withscores]	查看给定集合的差集
zdiffstore newkey key的数量 key1 [key2 ...]	存储给定集合的差集

2.7 遍历

2.7.1 遍历整个数据库

scan 游标 [match 格式] [count 游标] [type 指定类型]	遍历一定数量的key 游标：开始遍历的位置，从0开始， scan命令将返回遍历结束位置，如果返回0， 说明已经遍历到了最后一个 match：指定key的格式，*为通配符 count：指定遍历结束的游标，默认为10 type：指定遍历的类型

2.7.2 特定键值对的遍历

hscan key 游标 [match 格式] [count 游标]	遍历指定散列表中的字段与值
sscan key 游标 [match 格式] [count 游标]	遍历指定集合中的成员
zscan key 游标 [match 格式] [count 游标]	遍历指定有序集合中的成员与分数

完

预告：接下来将在《Golang学习记》中更新如何在Go语言中使用Redis

本文稿已上传至 https://github.com/fangdevelop/note