1. redis的API
2. redis介绍

redis是一种基于键值对（key-value）数据库，其中value可以为string、hash、list、set、zset等多种数据结构，可以满足很多应用场景。还提供了键过期，发布订阅，事务，流水线，等附加功能,

流水线: Redis 的流水线功能允许客户端一次将多个命令请求发送给服务器， 并将被执行的多个命令请求的结果在一个命令回复中全部返回给客户端， 使用这个功能可以有效地减少客户端在执行多个命令时需要与服务器进行通信的次数。

1. 特性：

1〉速度快，数据放在内存中，官方给出的读写性能10万/S，与机器性能也有关

a,数据放内存中是速度快的主要原因

b,C语言实现，与操作系统距离近

c,使用了单线程架构，预防多线程可能产生的竞争问题

2〉键值对的数据结构服务器

3〉丰富的功能：见上功能

4〉简单稳定：单线程

5〉持久化：发生断电或机器故障，数据可能会丢失，持久化到硬盘

6〉主从复制：实现多个相同数据的redis副本

8〉高可用和分布式：哨兵机制实现高可用，保证redis节点故障发现和自动转移

9〉客户端语言多：java php python c c++ nodejs等

1. 使用场景：
2. 缓存：合理使用缓存加快数据访问速度，降低后端数据源压力
3. 排行榜：按照热度排名，按照发布时间排行，主要用到列表和有序集合
4. 计数器应用：视频网站播放数，网站浏览数，使用redis计数
5. 社交网络：赞、踩、粉丝、下拉刷新
6. 消息队列：发布和订阅
7. 正确安装与启动
8. linux上安装，windows也能装，但我们以linux环境为主

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

1. 配置、启动、操作、关闭

|  |  |
| --- | --- |
| 可执行文件 | 作用 |
| redis-server | 启动redis |
| redis-cli | redis命令行客户端 |
| redis-benchmark | 基准测试工具 |
| redis-check-aof | AOF持久化文件检测和修复工具 |
| redis-check-dump | RDB持久化文件检测和修复工具 |
| redis-sentinel | 启动哨兵 |

1. redis-server启动：
2. 默认配置：redis-server, 日志输出版本信息，端口6379
3. 运行启动：redis-server --port 6380 不建议
4. 配置文件启动： redis-server /opt/redis/redis.conf，灵活，生产环境使用这种

4,redis-cli 启动

1. >交互式：redis-cli -h {host} -p {prot}连接到redis服务，没有h默认连127.0

redis-cli -h 127.0.0.1 -p 6379 //没有p 默认连6379

1. >命令式：redis-cli -h 127.0.0.1 -p 6379 get hello //取key=hello的value
2. >停止redis服务： redis-cli shutdown

注意: a，关闭时：断开连接，持久化文件生成，相对安全

b，还可以用kill关闭，此方式不会做持久化，还会造成缓冲区非法关闭，可能会造成AOF和丢失数据

c，关闭前生成持久化文件：

使用redis-cli -a 123456 登录进去，再shutdown nosave|save

4，>重大版本：

1，版本号第二位为奇数，为非稳定版本（2.7、2.9、3.1）

2，第二为偶数，为稳定版本（2.6、2.8、3.0）

3，当前奇数版本是下一个稳定版本的开发版本，如2.9是3.0的开发版本

1. 重要的指令使用:

1>全局命令

1，查看所有键：keys \* set school dongnao set hello world

2，键总数 dbsize //2个键，如果存在大量键，线上禁止使用此指令

3，检查键是否存在：exists key //存在返回1，不存在返回0

4，删除键：del key //del hello school, 返回删除键个数，删除不存在键返回0

5，键过期：expire key seconds //set name test expire name 10 //10秒过期

ttl 查看剩余的过期时间

6，键的数据结构类型：type key //type hello //返回string,键不存在返回none

1. 单线程架构

列举例子：三个客户端同时执行命令

客户端1：set name test

客户端2：incr num

客户端3：incr num

执行过程：发送指令－〉执行命令－〉返回结果

执行命令：单线程执行，所有命令进入队列，按顺序执行，使用I/O多路复用解决I/O问题，后面有介绍(通过select/poll/epoll/kqueue这些I/O多路复用函数库，我们解决了一个线程处理多个连接的问题)

单线程快原因：纯内存访问， 非阻塞I/O（使用多路复用），单线程避免线程切换和竞争产生资源消耗

问题：如果某个命令执行，会造成其它命令的阻塞

1. **字符串<String>**

3.1，字符串类型：实际上可以是字符串（包括XML JSON），还有数字（整形 浮点数），二进制（图片 音频 视频），最大不能超过512MB

3.2，设值命令：set age 23 ex 10 //10秒后过期 px 10000 毫秒过期

setnx name test //不存在键name时，返回1设置成功；存在的话失败0

set age 25 xx //存在键age时，返回1成功

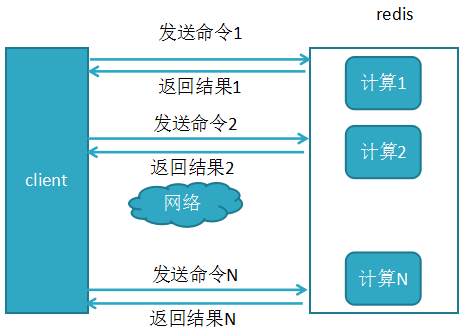
场景：如果有多客户同时执行setnx,只有一个能设置成功，可做分布式锁

获值命令：get age //存在则返回value, 不存在返回nil

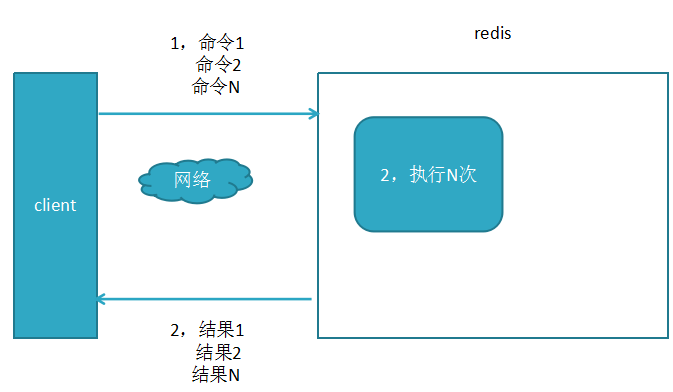
批量设值：mset country china city beijing

批量获取：mget country city address //返回china beigjin, address为nil

若没有mget命令，则要执行n次get命令



使用mget＝1次网络请求+redis内部n次查询



3.3，计数：incr age //必须为整数自加1，非整数返回错误，无age键从0自增返回1

decr age //整数age减1

incrby age 2 //整数age+2

decrby age 2//整数age -2

incrbyfloat score 1.1 //浮点型score+1.1

3.4，append追加指令：set name hello; append name world //追加后成helloworld

3.5，字符串长度：set hello “世界”；strlen hello//结果6，每个中文占3个字节

3.6，截取字符串：set name helloworld ; getrange name 2 4//返回 llo

3.7，内部编码：int:8字节长整理//set age 100; object encoding age //返回int

embstr:小于等于39字节串set name bejin; object encodeing name//embstr

raw:大于39字节的字符串set a fsdfwerwerweffffffffffdfs//返回raw

3.8，应用场景：

1. 键值设计：业务名:对象名:id:[属性]

数据库为order, 用户表user，对应的键可为 order:user:1 或order:user:1:name

注意：redis目前处于受保护模式，不允许非本地客户端链接，可以通过给redis设置密码，然后客户端链接的时候，写上密码就可以了

127.0.0.1:6379> config set requirepass 123456 临时生效

或者修改redis.conf requirepass 123456,启动时./redis-server redis.conf指定conf

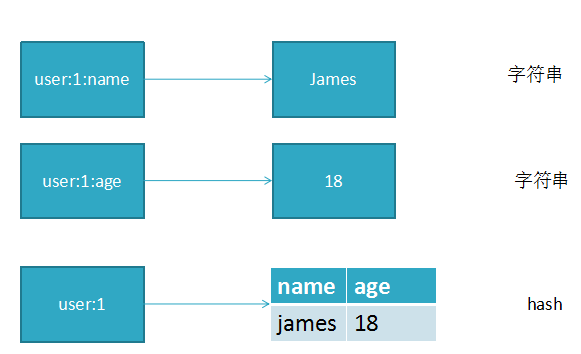
./redis-cli -p 6379 -a 12345678 //需要加入密码才能访问

切换数据库:select 2

**场景实践cache-demo**

JedisAllCommandTest OrderListHashTest SiteVisitNumTest

1. **哈希hash：是一个string类型的field和value的映射表，hash特适合用于存储对象。**



4.1命令 hset key field value

设值：hset user:1 name james //成功返回1，失败返回0

取值：hget user:1 name //返回james

删值：hdel user:1 age //返回删除的个数

计算个数：hset user:1 name james; hset user:1 age 23;

hlen user:1 //返回2，user:1有两个属性值

批量设值：hmset user:2 name james age 23 sex boy //返回OK

批量取值：hmget user:2 name age sex //返回三行：james 23 boy

判断field是否存在：hexists user:2 name //若存在返回1，不存在返回0

获取所有field: hkeys user:2 // 返回name age sex三个field

获取user:2所有value：hvals user:2 // 返回james 23 boy

获取user:2所有field与value：hgetall user:2 //name age sex james 23 boy值

增加1：hincrby user:2 age 1 //age+1

hincrbyfloat user:2 age 2 //浮点型加2

4.2内部编码：ziplist<压缩列表>和hashtable<哈希表>

当field个数少且没有大的value时，内部编码为ziplist

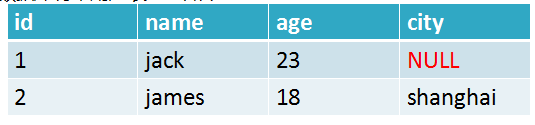
如：hmset user:3 name james age 24; object encoding user:3 //返回ziplist

当value大于64字节，内部编码由ziplist变成hashtable

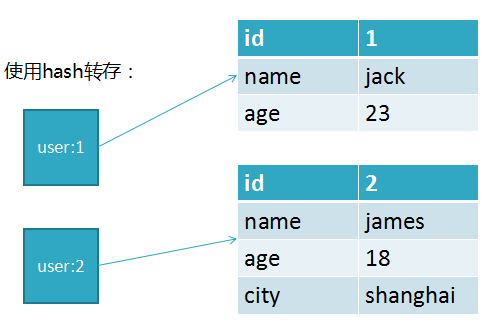
如：hset user:4 address “fsgst64字节”; object encoding user:3 //返回hashtable

4.3应用场景：

比如将关系型数据表转成redis存储：



使用hash后的存储方式为：



如果有值为NULL，那么如下：

HASH类型是稀疏，每个键可以有不同的filed, 若用redis模拟做关系复杂查询开发因难，维护成本高

4.4三种方案实现用户信息存储优缺点：

1，原生：set user:1:name james;

set user:1:age 23;

set user:1:sex boy;

优点：简单直观，每个键对应一个值

缺点：键数过多，占用内存多，用户信息过于分散，不用于生产环境

1. 将对象序列化存入redis

set user:1 serialize(userInfo);

优点：编程简单，若使用序列化合理内存使用率高

缺点：序列化与反序列化有一定开销，更新属性时需要把userInfo全取出来进行反序列化，更新后再序列化到redis

1. 使用hash类型：

hmset user:1 name james age 23 sex boy

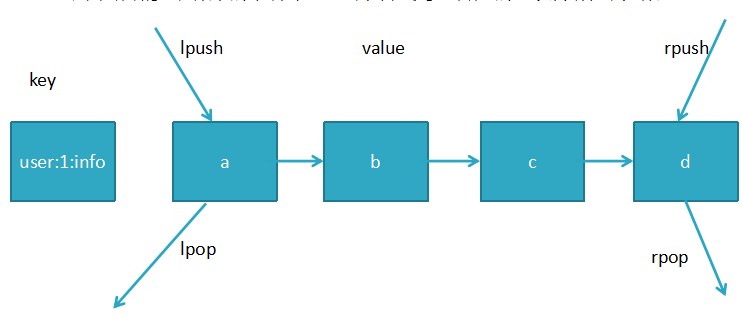
优点：简单直观，使用合理可减少内存空间消耗

缺点：要控制ziplist与hashtable两种编码转换，且hashtable会消耗更多内存

**总结：**对于更新不多的情况下，可以使用序列化，对于VALUE值不大于64字节可以使用hash类型

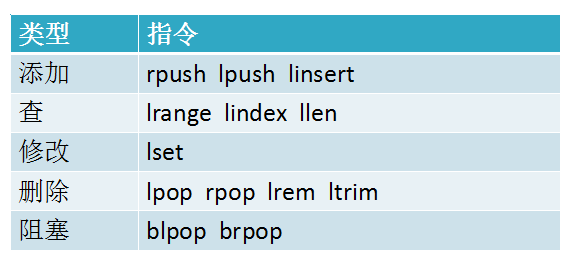
1. **列表<list>**

5.1用来存储多个有序的字符串，一个列表最多可存2的32次方减1个元素



因为有序，可以通过索引下标获取元素或某个范围内元素列表，列表元素可以重复

5.2列表命令：



添加命令：rpush james c b a //从右向左插入cba, 返回值3

lrange james 0 -1 //从左到右获取列表所有元素 返回 c b a

lpush key c b a //从左向右插入cba

linsert james before b teacher //在b之前插入teacher, after为之后，使 用lrange james 0 -1 查看：c teacher b a

查找命令： lrange key start end //索引下标特点：从左到右为0到N-1

lindex james -1 //返回最右末尾a，-2返回b

llen james //返回当前列表长度

lpop james //把最左边的第一个元素c删除

rpop james //把最右边的元素a删除

lrem key count value//删除指定元素

如：lpush test b b b b b j x z //键test放入z x j b b b b b

lrange test 0 -1 //查询结果为 z x j b b b b b

lrem test 4 b //从左右开始删除b的元素,删除4个，

若lrem test 8 b, 删除8个b, 但只有5个全部删除

lrange test 0 -1 //删除后的结果为 b j x z

lrem test 0 b //检索所有b全部删除 j x z

lpush user b b b b b j x z //键user从左到右放入 z x j b b b b b

ltrim user 1 3 //只保留从第2到第4的元素，其它全删

lrange user 0 -1 //查询结果为 x j b, 其它已全被删掉

lpush user01 z y x //键user01从左到右放入x y z

lset user01 2 java // 把第3个元素z替换成java

lrange user01 0 -1 //查询结果为 x y java

**应用场景设计： cacheListHashApplicationTest用例**

每个用户有多个订单key为 order:1 order:2 order:3, 结合hmset

1, hmset order:1 orderId 1 money 36.6 time 2018-01-01

hmset order:2 orderId 2 money 38.6 time 2018-01-01

hmset order:3 orderId 3 money 39.6 time 2018-01-01

把订单信息的key放到队列

lpush user:1:order order:1 order:2 order:3

每新产生一个订单，

hmset order:4 orderId 4 money 40.6 time 2018-01-01

追加一个order:4放入队列第一个位置

lpush user:1:order order:4

当需要查询用户订单记录时：

List orderKeys = lrange user:1 0 -1 //查询user:1 的所有订单key值

for(Order order: orderKeys){

hmget order:1

}

5.3列表内部编码：

1，当元素个数少且没大元素，编码为ziplist,减少内存的使用

rpush list a b c

object encoding list //返回ziplist

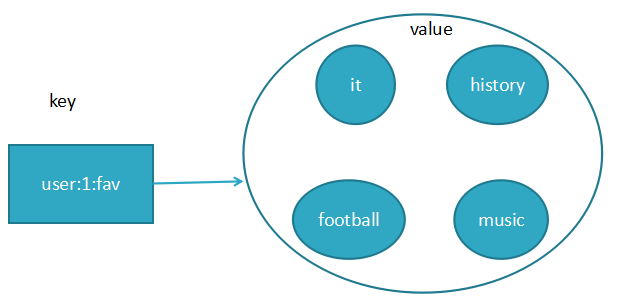
2，当元素超过512个，或元素超过64字节，内部编码变成linkedlist链表;

rpush list a1 a2 ....a513 或rpush list xxxxxxxxxxxxxx

object encoding list //linkedlist

1. **集合<SET> 用户标签，社交，查询有共同兴趣爱好的人,智能推荐**

保存多元素，与列表不一样的是不允许有重复元素，且集合是无序，一个集合最多可存2的32次方减1个元素，除了支持增删改查，还支持集合交集、并集、差集；



6.1命令：

exists user //检查user键值是否存在

sadd user a b c//向user插入3个元素，返回3

sadd user a b //若再加入相同的元素，则重复无效，返回0

smember user //获取user的所有元素,返回结果无序

srem user a //返回1，删除a元素

scard user //返回2，计算元素个数

sismember user a //判断元素是否在集合存在，存在返回1，不存在0

srandmember user 2 //随机返回2个元素，2为元素个数

spop user 2 //随机返回2个元素a b,并将a b从集合中删除

smember user //此时已没有a b, 只有c

集合的交集：

sadd user:1 zhangsan 24 girl

sadd user:2 james 24 boy//初始化两个集合

sinter user:1 user:2 //求两集合交集， 此时返回24

sadd user:3 wang 24 girl //新增第三个元素

sinter user:1 user:2 user:3 //求三个集合的交集，此时返回24

集合的并集（集合合并去重）：

sunion user:1 user:2 user:3 //三集合合并(并集)，去重24

sdiff user:1 user:2//1和2差集,(zhangsan 24 girl)-(james 24 boy)=zhangsan girl

将交集(jj)、并集(bj)、差集(cj)的结果保存：

sinterstore user\_jj user:1 user:2 //将user:1 user:2的交集保存到user\_jj

sunionstore user\_bj user:1 user:2 //将user:1 user:2的(并)合集保存user\_bj

sdiffstore user\_cj user:1 user:2 //将user:1-user:2的差集保存user\_cj

smemebers user\_cj // 返回zhangsan girl

6.1内部编码：

sadd user 1 2 3 4 //当元素个数少(小于512个)且都为整数，redis使用intset减少内存的使用

sadd user 1 2...513 //当超过512个或不为整数（比如a b）时，编码为hashtable

object encoding user //hashtables