Phạm Minh Khoa-18120418
Nhóm 14

Bài tập key-value

1. Cách thực thi file lua script
   - Tạo một file filename.lua
   - Viết script vào file. Ví dụ:

```
local ping = redis.call("ping")
return ping
```

   - Chạy file đó lên (dưới đây chạy trong terminal vs code):
     PS C:\Program Files\Redis> ./redis-cli --eval ./lua/test.lua 0
     - Trong đó ./redis-cli là đường dẫn gọi redis-cli nơi cài đăt redis.
     - --eval là lệnh thực thi script trong file test.lua
     - ./lua/test.lua đường dẫn tới script chứa mã thực thi
     - 0 là không có key truyền vào. Do lỗi gì đó không truyền được các ARGV chỉ truyền được KEYS. ./redis-cli --eval ./lua/test.lua key1 key2 thì trong file test.lua ta lấy key1, key2 thông qua KEYS[1], KEYS[2].
     - Cụ thể lệnh như này path/redis-cli –eval path/filename.lua numberkeys key1 keyn, argv1 argvn
   - Kết quả trả về là PONG

2. Phân tích
   - Thông tin lưu trữ bao gồm danh sách các sản phẩm được chọn, thông tin sản phẩm bao gồm tên, hình ảnh, giá , số lượng.
     ⇨ Thông tin sản phẩm gồm tập các giá trị. Áp dụng lữu trữ dưới dạng hash đối với thông tin sản phẩm.
     ⇨ Mỗi khách hàng sẽ có cookieid và có 1 giỏ hàng. Mỗi giỏ hàng của user có nhiều sản phẩm đã được chọn.
     ⇨ Lưu tất cả sản phẩm trong giỏ hàng của 1 user:
       - Key: users:{cookieid}:carts. Cookieid sẽ được thay giá trị tương ứng khi user truy cập.
       - Value: {productid}. Lưu dưới dạng list các productid, và chúng có thể trùng (những user có thể mua lại sản phẩm ở lần khác).
     ⇨ Lưu thông tin sản phẩm: (sử dụng hash)
       - Key: users:{cookieid}:carts:{productid}
       - Field-Value: name value1 imagesource value2 price value3 amount value4
   - Yêu cầu 2: Những lưu ý để chọn kiểu dữ liệu lưu trữ phù hợp
     - sorted sets và sets chỉ lưu giá trị không trùng lắp. List và hash lưu giá trị trùng lắp.
     - “Tìm những đơn hàng có số lượng sản phẩm lớn hơn 5” sẽ bị vướng trùng lắp giá trị khi lưu trữ cùng price và amount sản phẩm trong kiểu dữ liệu sorted set.

- Có thể sử dụng hai loại dấu nháy đơn và nháy kép để thể hiện chuỗi. Nếu ở ngoài là nháy đơn thì trong là tập hợp nháy kép và ngược lại.

3. Tạo cấu trúc lưu trữ dữ liệu lưu trữ và truy vấn

3.1.Yêu cầu 1

- Lưu tất cả sản phẩm trong giỏ hàng của 1 user (list):
  - Key: users:{cookieid}:carts. Cookieid sẽ được thay giá trị tương ứng khi user truy cập.
  - Value: {productid}. Lưu dưới dạng list các productid, và chúng có thể trùng (những user có thể mua lại sản phẩm ở lần khác).
- Lưu thông tin sản phẩm: (sử dụng hash)
  - Key: users:{cookieid}:carts:{productid}
  - Field-Value: name value1 imagesource value2 price value3 amount value4

3.2.Yêu cầu 2

- Ở đây đơn hàng khác giỏ hàng.
- Cấu trúc lưu trữ:
  - Các đơn hàng của một user (list):
    - Key: users:{cookieid}:orders
    - Value: {orderid}
  - Danh sách sản phẩm của 1 đơn hàng (hash):
    - Key: orders:{orderid}:products
    - Field-value: {productid:amount}
  - Lưu giá cho một sản phẩm trong 1 đơn hàng (hash):
    - Key: orders:{orderid}:products.price
    - Field-value: {productid:price}
  - Thêm thuộc tính chưa thanh toán (thêm tiền tố nopay trước orderid-một uuid) (hash):
    - Key: users:{cookieid}:orders.ispaid
    - Field-Value: {orderid-value}, value: 1-paid, 0-nopay

- Data mẫu

```
local hmset = "hmset"

--tao danh sach don hang có thuộc tính đã thanh toán chưa
--users: hung, huyen
local mykey="users:hung:orders.ispaid"
redis.call(hmset, mykey, "hung1", 1, "hung2", 1, "hung3", 0)
mykey="users:huyen:orders.ispaid"
redis.call(hmset, mykey, "huyen1", 0, "huyen2", 1)

--orderid
```

```
redis.call("rpush", "users:hung:orders", "hung1", "hung2", "hung3")
redis.call("rpush", "users:huyen:orders", "huyen1", "huyen2")

--danh sách sản phẩm của 1 đơn hàng
redis.call("hmset", "orders:hung1:products", "sach1", 2, "sach2", 1)
redis.call("hmset", "orders:hung2:products", "vo1", 2, "vo2", 1)
redis.call("hmset", "orders:hung3:products", "but1", 2, "but2", 1)

redis.call("hmset", "orders:huyen2:products", "giay1", 2, "giay2", 1)
redis.call("hmset", "orders:huyen1:products", "ao", 2, "quan", 1)

--luu giá
redis.call("hmset", "orders:hung1:products.price", "sach1", 20000, "sach2",
15000)
redis.call("hmset", "orders:hung2:products.price", "vo1", 14000, "vo2", 10000)
redis.call("hmset", "orders:hung3:products.price", "but1", 4000, "but2", 10000)

redis.call("hmset", "orders:huyen2:products.price", "giay1", 500000, "giay2",
1000000)
redis.call("hmset", "orders:huyen1:products.price", "ao", 200000, "quan", 300000)
```

3.2.1. Tìm những đơn hàng chưa được thanh toán, nếu chưa có thuộc tính phân biệt, hãy tạo thêm thuộc tính này.

```
--cookie
local users={"hung", "huyen"}
local orderids={}
--dang key:value lưu trữ của đơn hàng lưu vết đã thanh toán:
(users:{cookieid}:orders.ispaid):({orderid-value})

--lay các đơn hàng của các user
local key="users:"..users[1]..":orders.ispaid"
local orders
local temps={}
local lenOrders
local noPaidOrderId={} --danh sác các đơn hàng chưa thanh toán
--local ordersHung = redis.call('hgetall', key)
for i,user in ipairs(users)
do
    key="users:"..user..":orders.ispaid"
    orders = redis.call('hgetall', key)
    lenOrders = table.maxn(orders)
    for j=1,lenOrders,1
    do
        if(j%2 == 0 and orders[j] == "0")
        then
```

```
            table.insert(noPaidProductId, orders[j-1])
        end
    end
end
return noPaidProductId
```

3.2.2. Tìm những đơn hàng có số lượng sản phẩm lớn hơn 5.

- Chạy lệnh sau trên redis-cli để sửa amout của một vài sản phẩm lớn hơn 5
  hmset orders:hung3:products but1 6
  hmset orders:huyen1:products quan 7
- Script truy vấn

```
--cookie
local users={"hung", "huyen"}
local orderids={}
local orderid5={}
--dang key:value lưu trữ của đơn hàng lưu vết đã thanh toán:
(users:{cookieid}:orders.ispaid):({orderid-value})

--lay các đơn hàng của các user
local key="users:"..users[1]..":orders.ispaid"
local orders

--get all orderid
for i,user in ipairs(users)
do
    key="users:"..user..":orders"
    orders = redis.call('lrange', key, 0, -1)
    for j,id in pairs(orders)
    do
        table.insert(orderids, id)
    end
end

--find the orders which contain the product amounts > 5
for i,orderid in pairs(orderids)
do
    local mykey = "orders:"..orderid..":products"
    local products = redis.call("hgetall", mykey)
    local lenProducts = table.maxn(products)
    for j=1,lenProducts,1
    do
        if(j%2==0)
        then
            local amount=tonumber(products[j])
            if(amount>5)
```

```
            then
                table.insert(orderid5, orderid)
            end
        end
    end
end
return orderid5
```

3.2.3. Cho biết sản phẩm X xuất hiện trên bao nhiêu đơn hàng.

- Lệnh chạy trên cmd: ./redis-cli --eval ./lua/btvn#1/yeucau2-3.lua but1
    - X=but1

```
local X=KEYS[1]
local users={"hung", "huyen"}
local orderids={}
local orderid5={}

--lay các đơn hàng của các user
local key=""
local orders
local countOrders = 0

--get all orderid
for i,user in ipairs(users)
do
    key="users:"..user..":orders"
    orders = redis.call('lrange', key, 0, -1)
    for j,id in pairs(orders)
    do
        table.insert(orderids, id)
    end
end

--count the number of order that contain the X product id
for i,orderid in pairs(orderids)
do
    local mykey = "orders:"..orderid..":products"
    local products = redis.call("hkeys", mykey)
    for j,proID in pairs(products)
    do
        if(proID == X)
        then
            countOrders = countOrders + 1
            break
        end
```

```
    end
end
return countOrders
```

3.2.4. Tính tổng giá tiền cho một đơn hàng

- Lệnh: ./redis-cli --eval ./lua/btvn#1/yeucau2-4.lua huyen3
    - Với orderid=huyen3

```
local calMoneyOrder=0 --result
local orderid=KEYS[1] --get key from command
local key="" --key to query
local idProductAmount,idProductPrice,idProduct
local len=0

--get product id and amount
key="orders:"..orderid..":products"
idProductAmount=redis.call("hgetall", key)
--get length
len=table.maxn(idProductAmount)
--get product id and price
key="orders:"..orderid..":products.price"
idProductPrice=redis.call("hgetall", key)
--get all product id
idProduct=redis.call("hkeys", key)
--count the number of order that contain the X product id
for i,proId in pairs(idProduct)
do
    local amount,price=0,0
    --get amount
    for j=1,len,1
    do
        if(proId == idProductAmount[j])
        then
            amount=tonumber(idProductAmount[j+1])
            break
        end
    end
    --get price
    for j=1,len,1
    do
        if(proId == idProductPrice[j])
        then
            price=tonumber(idProductPrice[j+1])
            break
        end
    end
```

```
    calMoneyOrder = calMoneyOrder + price*amount
end
return calMoneyOrder
```

3.2.5. Liệt kê tất cả các đơn hàng của một người dùng

- Lệnh chạy trong redis-cli:
  eval 'local cookieid=ARGV[1] local key="users:"..cookieid..":orders" return redis.call("lrange",key,0,-1)' 0 huyen
- Với tham số truyền vào là cookie user: huyen

3.2.6. Xóa sản phẩm X trong đơn hàng Y

- Lệnh chạy trong redis-cli:
  eval 'local orderid,productid=ARGV[1],ARGV[2] local key="orders:"..orderid..":products" redis.call("hdel", key, productid)' 0 hung1 sach1
- Với hung1 là orderid, sach1 là productid

3.2.7. Tăng số lượng sản phẩm X trong đơn hàng Y lên 1 đơn vị

- Lệnh chạy trong redis-cli:

eval 'local orderid,productid=ARGV[1],ARGV[2] local key="orders:"..orderid..":products" redis.call("hincrby", key, productid,1)' 0 hung1 sach1

- Với hung1 là orderid, sach1 là productid

3.2.8. Xóa tất cả sản phẩm trong đơn hàng

- del keyname

3.2.9. Tìm đơn hàng có tổng giá trị lớn nhất

```
local function calMoneyOrder(orderid)
    local calMoneyOrder=0 --result
    local key="" --key to query
    local idProductAmount,idProductPrice,idProduct
    local len=0

    --get product id and amount
    key="orders:"..orderid..":products"
    idProductAmount=redis.call("hgetall", key)
    --get length
    len=table.maxn(idProductAmount)
    --get product id and price
    key="orders:"..orderid..":products.price"
    idProductPrice=redis.call("hgetall", key)
    --get all product id
    idProduct=redis.call("hkeys", key)
```

```
    --count the number of order that contain the X product id
    for i,proId in pairs(idProduct)
    do
        local amount,price=0,0
        --get amount
        for j=1,len,1
        do
            if(proId == idProductAmount[j])
            then
                amount=tonumber(idProductAmount[j+1])
                break
            end
        end
        --get price
        for j=1,len,1
        do
            if(proId == idProductPrice[j])
            then
                price=tonumber(idProductPrice[j+1])
                break
            end
        end
        calMoneyOrder = calMoneyOrder + price*amount
    end
    return calMoneyOrder
end

local listTotalMoneyOrder={}
local listOrderId={}
local key
local users={"hung", "huyen"}
local maxTotalMoneyOrderIds={}
local max=0
--dang key:value lưu trữ của đơn hàng lưu vết đã thanh toán:
(users:{cookieid}:orders.ispaid):({orderid-value})

--lay các đơn hàng của các user


--get all orderid
for i,user in ipairs(users)
do
    key="users:"..user..":orders"
    local orders = redis.call('lrange', key, 0, -1)
    for j,id in pairs(orders)
```

```
    do
        table.insert(listOrderId, id)
    end
end

--calculate
for i,id in pairs(listOrderId)
do
    listTotalMoneyOrder[id]=calMoneyOrder(id)
end

for id,total in pairs(listTotalMoneyOrder)
do
    if(total>=max)
    then
        max=total
    end
end

for id,total in pairs(listTotalMoneyOrder)
do
    if(total==max)
    then
        table.insert(listTotalMoneyOrder,id)
    end
end

return listTotalMoneyOrder
```

3.2.10. Tìm sản phẩm xuất hiện nhiều đơn hàng nhất.

- Có 2 hướng: tạo thêm key lưu trữ tất cả product id hoặc lọc ra các productid
- Cách làm tương tự những câu trên, sử dụng lua script