Redis的应用与数据设计

我的redis数据设计小试

List

用户列表：user\_唯一识别码（user\_code）

{

user\_id

user\_name

user\_nick

is\_logIn

age

...

}

订单列表：order\_唯一订单流水号（SerialOrderNum）

{

order\_id

order\_beloger

...

}

Hash

订单-用户HashSet：哈希名为md5加密的sql语句(select \* from data\_order)

{

键为md5加密的sql语句（select \* from data\_order where order\_id=...） 值为订单号

}

订单-总金额HashSet：哈希名为order\_amount

{

键为订单号 值为该订单总金额

}

ZSet

店内的单个商品成交量ZSet：可以实现热门商品的排序Hot\_Ware

{

ware\_amount1

ware\_amount2

...

}

Set

某项活动的参加人员，人员是唯一的。比如：拼多多的拼单活动。

{

user\_code1

user\_code2

...

}

用redis创建订单号

利用redis的单线程特性创建分布式唯一id

/\*\*

\* 流水订单号生成器

\*/  
public class SerialOrderNumGenerator {  
private static long nodeValueForIncr=0;  
private static final String nodeKeyForIncr="nodeOrderForIncr";  
 /\*构造函数\*/  
public SerialOrderNumGenerator(){}  
 /\*订单号32 bit组成

省份编码3;地区编码3;商品类型编码3;商品编码4;时间戳序列号13;

从redis中获取序列号6;\*/  
public static String createSerialOrderNum(String sqlLang,

String provinceCode,String areaCode,String wareTypeCode,

String wareCode){

String serialOrderNum=getProvinceCode(provinceCode)+

getAreaCode(areaCode)+getWareTypeCode(wareTypeCode)+

getWareCode(wareCode)+getCurrentMilliTime()+

createRedisSerialOrderNum();

return serialOrderNum;  
 }  
 /\*省份编码对应序列号3位\*/  
private static String getProvinceCode(String provinceCode){

int codeNumber=999;

if (provinceCode.equals("fujian")) {

codeNumber=26;

}else if (provinceCode.equals("hubei")) {

codeNumber=29;

}

return String.format("%03d",codeNumber);

}  
 /\*地区编码对应序列号3位\*/  
private static String getAreaCode(String areaCode){

int codeNumber=999;

if (areaCode.equals("fuzhou")) {

codeNumber=26;

}else if (areaCode.equals("xiamen")) {

codeNumber=29;

}

return String.format("%03d",codeNumber);

}  
 /\*商品类型编码对应序列号3位\*/  
private static String getWareTypeCode(String wareTypeCode){

int codeNumber=9999;

if (wareTypeCode.equals("digit")) {

codeNumber=26;

}else if (wareTypeCode.equals("fruit")) {

codeNumber=29;

}

return String.format("%03d",codeNumber);

}  
 /\*商品编码对应序列号4位\*/  
private static String getWareCode(String wareCode){

int codeNumber=9999;

if (wareCode.equals("manggo")) {

codeNumber=26;

}else if (wareCode.equals("lemon")) {

codeNumber=29;

}

return String.format("%04d",codeNumber);

}  
 /\*\* 时间戳序列13位\*/  
private static long getCurrentMilliTime(){

return System.currentTimeMillis();

}  
 /\*\* 从redis生成序列号6位\*/  
private static String createRedisSerialOrderNum(){

int codeNumber=999999;

Jedis jedis= RedisPoolUtil.getJedis();

jedis.incr(nodeKeyForIncr);

codeNumber=Integer.valueOf(jedis.get(nodeKeyForIncr));

RedisPoolUtil.returnResource(jedis);

return String.format("%06d",codeNumber);

}

而需要注意的是nodeValueForIncr是需要每天定时更新重置的。

/\*\*

\*保存订单信息到redis数据库

\*/

public class SaveOrderToRedis {  
private final Logger log=LoggerFactory.getLogger(SaveOrderToRedis.class);

/\*构造函数\*/  
public SaveOrderToRedis(OrderInfo orderInfo,String sqlLang,

String serialOrderNum){

if(saveSerialOrderNum(sqlLang,serialOrderNum)){

Jedis jedis=RedisPoolUtil.getJedis();

String orderKey="";

try {

MessageDigest md5= MessageDigest.getInstance("MD5");

BASE64Encoder base64Encoder=new BASE64Encoder();

orderKey=base64Encoder.encode(md5.digest(sqlLang.getBytes()));

} catch (NoSuchAlgorithmException e) {

log.error("获取MD5加密摘要失败:"+e.getMessage());

}

//保存订单信息到redis列表中;订单号加密后作为List表名

//组装列表:归属人、总金额、

jedis.lpush(orderKey);

jedis.lset(orderKey,1,orderInfo.getOrderBelonger());

jedis.lset(orderKey,2,String.valueOf(orderInfo.getOrderAmount()));

//关闭redis连接

RedisPoolUtil.returnResource(jedis);

}

}

/\*将订单号作为值,存入hashSet,键为md5加密的sql语句\*/  
private boolean saveSerialOrderNum(String sqlLang,String serialOrderNum){

Jedis jedis=RedisPoolUtil.getJedis();

//对sql语句进行加密,作为hashSet键key

try {

MessageDigest md5= MessageDigest.getInstance("MD5");

BASE64Encoder base64Encoder=new BASE64Encoder();

String sqlCodeAsKey=base64Encoder.encode(md5.digest(sqlLang.getBytes()));

//存入名为serialOrderNumHashKey的hashSet;作为订单号存储的HashSet

jedis.hset("serialOrderNumHashKey",sqlCodeAsKey,serialOrderNum);

} catch (NoSuchAlgorithmException e) {

log.error("获取MD5加密摘要失败:"+e.getMessage());

}

RedisPoolUtil.returnResource(jedis);

return true;

}  
}

由于redis不支持数据包含空格，所以保存sql语句要进行加密并且结果唯一，MD5的唯一性具有很强的保障。

## 参考：

在该博文中可获得redis其他数据结构和命令详解；并且其他博文也有较丰富的讲解

[**Redis数据结构解析之（二）**](http://www.cnblogs.com/knowledgesea/p/4972771.html) <https://www.cnblogs.com/knowledgesea/p/4972771.html>

Redis 的 5 个常见应用场景 <http://blog.jobbole.com/88383>

[**Redis使用场景**](http://blog.csdn.net/hguisu/article/details/8836819) <http://blog.csdn.net/hguisu/article/details/8836819>

[Redis各个数据类型对应的应用场景](http://www.cnblogs.com/yy1234/p/7809551.html) <https://www.cnblogs.com/yy1234/p/7809551.html>

[使用redis缓存加索引处理数据库百万级并发](http://www.cnblogs.com/fanwencong/p/5782860.html) <https://www.cnblogs.com/fanwencong/p/5782860.html>

[**深入浅出Redis-哨兵集群**](http://www.cnblogs.com/jaycekon/p/6237562.html) <https://www.cnblogs.com/jaycekon/p/6237562.html>

[Redis Sentinel(哨兵)：集群解决方案](http://p.primeton.com/articles/559e431d608f8f5438000059)2年前 <http://p.primeton.com/articles/559e431d608f8f5438000059>

Redis伪集群的搭建 <http://blog.csdn.net/nayouzenmyang/article/details/54232957>

Redis的客户端Jedis操作<https://www.cnblogs.com/liuling/p/2014-4-19-04.html>

<https://www.jianshu.com/p/7913f9984765>

<http://www.runoob.com/redis/redis-java.html>

Jedis连接池配置 <http://blog.csdn.net/zhoubinbin111111/article/details/52370706>

[高并发下怎么生成唯一订单号](http://www.cnblogs.com/jianchileiliang/p/6796590.html) <https://www.cnblogs.com/jianchileiliang/p/6796590.html>

利用redis创建订单号 <http://blog.csdn.net/qq_39291929/article/details/78827066>

Redis在集群环境中生成唯一ID <http://blog.csdn.net/csujiangyu/article/details/52348003>

基于redis的分布式ID生成器 <http://blog.csdn.net/hengyunabc/article/details/44244951>