**DNA元基索引ETL中文脚本编译机**

罗瑶光

[313699483@qq.com](mailto:313699483@qq.com)

浏阳德塔软件开发有限公司

**Proposal 0.0.1**

**1 介绍**

DNA元基索引ETL中文脚本编译机 前身是《Deta Socket流 PLSQL 数据库》的 Query指令集编译机。在养疗经的内存计算中，作者开始逐步的将编译机的命令中文化和 参与ETL， TCP，内存 和 中药表格筛选 搜索计算，于是开始命令扩充和整体逻辑优化。将shell命令进行 元基分类标识 和 索引管理，于是这个项目发展起来。

项目时间 2021年9月22日~ 2021年10月16日

**2 动机**

2.1 作者思考命令行编程 （Programmable Language SQL）PLSQL 进行数据库操作，同理可以进行其他类的数据操作如内存数据。需要进行论证。

2.2 作者的父亲经常对作者说 养疗经的操作选项组件太多， 界面繁琐会让人眼花。作者思考需要一种便携统一的方式来简化使用逻辑。

2.3 作者使用ETL Unicron 总是不经意的思考每个节点就要设计一个节点界面，消耗大量前端人力和时间， 作者思考需要一种便携统一的方式来简化使用逻辑。

2.4 作者在WCC 2021 长沙开幕式 听了周向宇先生一堂课 说 古人的古书语文作品中蕴含数学逻辑， 如愚公移山的故事蕴藏 数列的极限和迭代逻辑等 F（n）= f（n + 1）。。。 作者思考把 《Deta Socket流 PLSQL 数据库》的PLSQL指令翻译成 中文试试， 以后能使用命令的就不再只有程序员的群体了 比如父亲。

2.5 作者在《DNA元基催化与肽计算 第三修订版》的元基卷积ETL 两章有描述元基矩阵记忆流 和 元基DNN计算流 两种节点模式，于是思考如何开始论证 思考 计算流和记忆流 的表达模式。

2.6 作者的 DNA元基TVM 虚拟机需要一个切入点，ETL中文脚本编译机 恰好充当一个基础原型机。

于是这个项目开始了。

**3 适用**

3.1 该项目适用于所有并发的决策流内存计算场景。

3.2 该项目适用于编码能力薄弱的客户群体，非程序员群体。

3.3 该项目适用于各种复杂的工业基础体系, 如大数据计算类， 内存计算类，工业调度类，等。。

**4 逻辑**

4.1 PLSEARCH 包含可编程搜索命令 概念作者首发

将 德塔PLSQL 中非join table的命令拿出来 优化成适用于 内存计算的命令。

4.2 PLETL 包含可编程节点流操作 概念作者首发

将 ETL的节点流配置执行界面设计成

命令如下

PLETL:中节点|进行表格相交|主码为|ID|模式为|新增列;  
PLETL:中节点|进行表格相交|主码为|ID|模式为|叠加列;  
PLETL:中节点|进行表格相交|主码为|ID|模式为|有交集叠加列;  
PLETL:中节点|进行表格相交|主码为|ID|模式为|有交集新增列;  
PLETL:中节点|进行表格相交|主码为|ID|模式为|无交集新增列;

PLETL:中节点|进行表格剔除|主码为|ID|模式为|相交部分剔除;

PLETL:中节点|进行表格合并|主码为|ID|模式为|新增列;  
PLETL:中节点|进行表格合并|主码为|ID|模式为|叠加列;  
PLETL:中节点|进行表格合并|主码为|ID|模式为|有交集叠加列;  
PLETL:中节点|进行表格合并|主码为|ID|模式为|有交集新增列;  
PLETL:中节点|进行表格合并|主码为|ID|模式为|无交集新增列;

4.3 PLTCP 包含可编程网络请求 概念作者首发

4.4 PLSQL 可编程数据库操作 概念美国甲骨文公司首发

**5 使用方法**

5.1 指令集 已有中文命令分类 如下

操作

条件为

获取表名

获取表列名

进行分词

词性标注

词性显色

DNN

颜色标记为

红色

蓝色

黄色

绿色

进行字符排序

进行数字排序

从小到大

从大到小

行至

包含

改名为

过滤掉

不包含

进行选择

精度搜索

PLETL

中节点

进行表格相交

进行表格合并

进行表格剔除

主码为

ID

模式为

相交部分剔除

新增列  
叠加列  
有交集叠加列  
有交集新增列  
无交集新增列

其他非中文命令见 德塔PLSQL 文档

**语法为**

；一个shell句型分隔

：一个shell函数分隔

| 一个shell对象分隔

5.2 组合方式示例

5.2.1完整句型

获取表名:中药同源:进行选择;

条件为:和:功效|精度搜索|风热咳嗽|0;

条件为:和:中药名称|字符串长度大于|3;

条件为:或:功效|包含|清热解毒:功效|包含|利尿;

条件为:和:性味|不包含|温:脉络|包含|肺;

条件为:和:风险规避|过滤掉|毒:风险规避|过滤掉|孕;

获取表列名:功效:风险规避|改名为|风险:脉络:性味:中药名称|改名为|药名;

操作:0|行至|20;

操作:风险|颜色标记为|黄色;

操作:药名|颜色标记为|红色;

操作:功效|进行分词|DNN;

5.2.2 流句型 完整测试指令如下：

节点1

获取表名:中医诊断:进行选择;

条件为:和:笔记|包含|发热:笔记|包含|身重;

获取表列名ID病症;

操作:0|行至|30;

节点1->2

操作:病症|进行分词|词性显色;

节点1->3

操作:病症|进行分词|DNN;

节点（（1->2）+（1->3））->4

PLETL:中节点|进行表格合并|主码为|ID|模式为|新增列;

操作:ID|进行数字排序|从小到大;

操作:ID|颜色标记为|红色;

5.2.3 流并发句型

节点（（1->2）+（1->3））->4

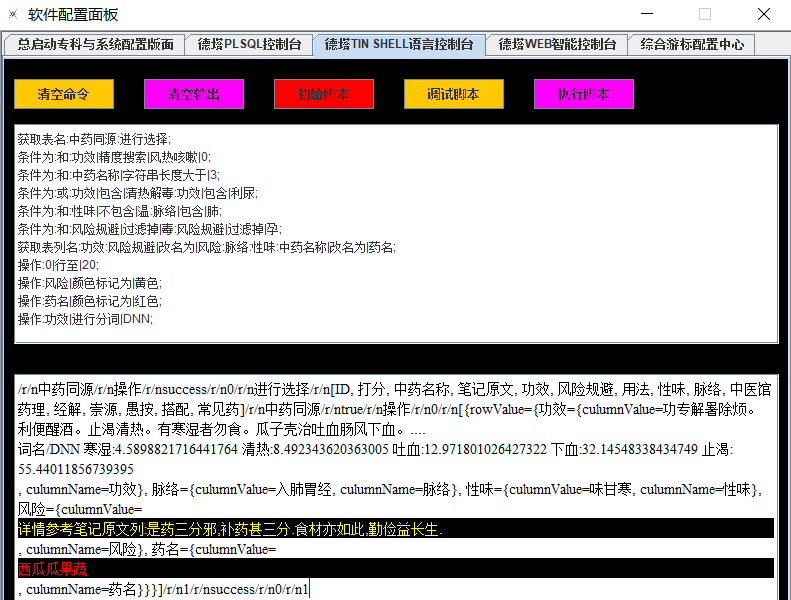
PLETL:中节点|进行表格合并|主码为|ID|模式为|新增列;

操作:ID|进行数字排序|从小到大;

操作:ID|颜色标记为|红色;

**6 展示**

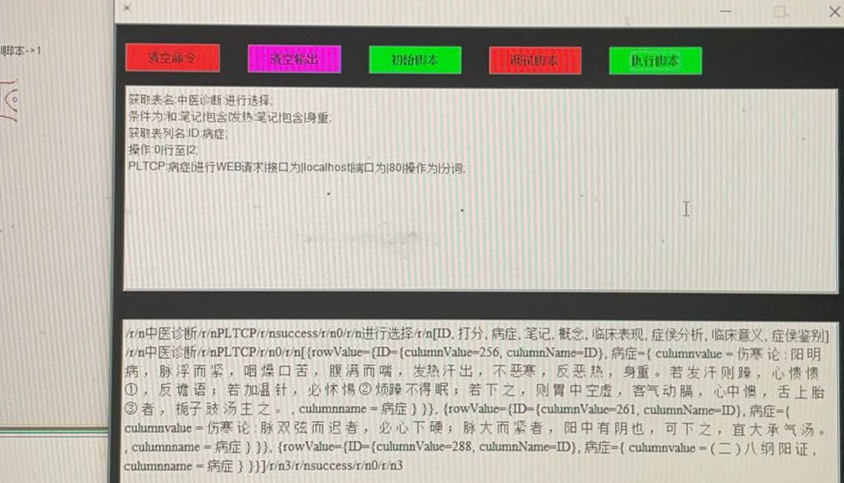
6.1 单一TinShell 执行PLSearch



6.2多节点Tinshell执行并发 PLETL



6.3节点Tinshell执行 PLTCP HTTP接口 请求



7 源码

package OSM.shell;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import OSA.shell.PL\_XA\_E;

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

public class E\_PLSearch\_E {

public static Map<String, Object> E\_PLSearch(String plSearch, boolean mod, Map<String, Object> output) throws Exception{

//working for here

//Map<String, Object> output= new ConcurrentHashMap<>();

//1make container

if(null= = output) {

output= new ConcurrentHashMap<>();

}

output.put("firstTime", "true");

output.put("start", "0");

output.put("countJoins", "0");

//2make line

String[] commands= plSearch.replace(" ", "").replace("\n", "").split(";");

String[] acknowledge= null;

for(String command:commands) {

acknowledge= command.split(":");

if(acknowledge[0].equals("setRoot")) {

PLSearchCommand\_E.P\_SetRoot(acknowledge, output);

}

if(acknowledge[0].equals("baseName")) {

PLSearchCommand\_E.P\_BaseName(acknowledge, output);

}

if(acknowledge[0].equals("获取表名")) {

PLSearchCommand\_E.P\_TableName(acknowledge, output);

}

if(acknowledge[0].equals("culumnName")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("changeCulumnName")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("culumnValue")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("join")) {

PLSearchCommand\_E.P\_Join(acknowledge, output);

}

if(acknowledge[0].equals("条件为")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("relation")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("操作")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("PLETL")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("获取表列名")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

if(acknowledge[0].equals("PLTCP")) {

PLSearchCommand\_E.P\_ListNeedStart(acknowledge, output);

}

output.put("newCommand", acknowledge[0]);

PLSearchCommand\_E.P\_E(acknowledge, output, mod);

output.put("lastCommand", output.get("newCommand"));

}

if(null!= acknowledge) {

if(output.get("start").toString().equals("1")) {

PLSearchCommand\_E.P\_E(acknowledge, output, mod);

}

}

System.out.println("1");

PLSearchCommand\_E.P\_Check(output.get("newCommand").toString(), output, mod);

return output;

}

public static Map<String, Object> E\_PLSearch(PL\_XA\_E orm, boolean b, Map<String, Object> output) throws Exception {

return E\_PLSearch(orm.getPLSearch(), true, output);

}

}

----------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.IOException;

import java.math.BigDecimal;

import java.net.URLEncoder;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import ME.APM.VSQ.HRJFrame;

import MSV.OSQ.sets.DetaDouble;

import OEU.LYG4DQS4D.LYG10DWCMSSort13D\_XCDX\_C\_A\_S;

import OEU.LYG4DQS4D.LYG9DWithDoubleTopSort4D;

import OEU.LYG4DQS4D.Quick\_7D\_luoyaoguang\_Sort;

import OSI.AOP.PCS.PP.port\_E.RestNLPPortImpl;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.TinMap;

import PEQ.AMV.ECS.test.ANNTest;

import PEQ.AMV.ECS.test.DNNTest;

import PEQ.AMV.ECS.test.SensingTest;

import PEU.P.table.TableSorterZYNK;

@SuppressWarnings({"unused"})

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

//pletl 先实现 上中下的 tinmap中指令合并

//设计了简答宾语补足语，有时间思考下 定语和slang 的模式 和成语模式 罗瑶光

public class P\_AggregationPLETL {

@SuppressWarnings({"unchecked"})

public static void P\_PletlLimitMap(String[] sets, List<Map<String, Object>> output, Map<String, Object> object)

throws InstantiationException, IllegalAccessException, IOException {

List<Map<String, Object>> outputTemp= new ArrayList<>();

//中节点|表格合并|主码|新增列|ID|。。

//上节点是main节点作为accumulator，模拟rna芯片计算容器，中下节点模拟神经元记忆接口。

//设计宾语补足语 今天改为如下 20211011 罗瑶光

//PLETL:中节点|进行表格合并|主码为|ID|模式为|新增列;

//PLETL:中节点|进行表格合并|主码为|ID|模式为|叠加列;

//PLETL:中节点|进行表格合并|主码为|ID|模式为|有交集叠加列;

//PLETL:中节点|进行表格合并|主码为|ID|模式为|有交集新增列;

//PLETL:中节点|进行表格合并|主码为|ID|模式为|无交集新增列;

if(sets[1].equalsIgnoreCase("进行表格合并")) {

TinMap mapShell= null;

String 列标识= null;

if(sets[0].equalsIgnoreCase("中节点")) {

mapShell= (TinMap)object.get("midShell");

列标识= "m\_";

}

if(sets[0].equalsIgnoreCase("下节点")) {

mapShell= (TinMap)object.get("downShell");

列标识= "d\_";

}

//将上面进行内外循环 颠倒rotation 如下

if(sets[2].equalsIgnoreCase("主码为")) {//先单一primary key， 之后再设计 forenge key 和 combination key

//To do。。。初始

Map<String, Object> tinShellETL= (Map<String, Object>)mapShell.get("TinShellETL");

List<Map<String, Object>> rowList= (List<Map<String, Object>>)tinShellETL.get("obj");

//主循环

List<Map<String, Object>> 主要输入轮训= (List<Map<String, Object>>)object.get("obj");

Iterator<Map<String, Object>> outputTempIterator= 主要输入轮训.iterator();

while(outputTempIterator.hasNext()) {

Map<String, Object> rowOutputTempIterator= outputTempIterator.next();

Map<String, Object> rowValueRowOutputTempIterator

= (Map<String, Object>)rowOutputTempIterator.get("rowValue");

if(0!= rowList.size()) {

//辅循环

Iterator<Map<String, Object>> iterator= rowList.iterator();

Here:

while(iterator.hasNext()) {//非主要输入轮训

Map<String, Object> row= iterator.next();

Map<String, Object> rowValue= (Map<String, Object>)row.get("rowValue");

Map<String, Object> culumnValue= (Map<String, Object>)rowValue.get(sets[3]);

//outputTemp

if(rowValueRowOutputTempIterator.containsKey(sets[3])) {

Map<String, Object> rowValueRowOutputTempIteratorCulumnValue

= (Map<String, Object>)rowValueRowOutputTempIterator.get(sets[3]);

//合并rowValueRowOutputTempIteratorCulumnValue 与 culumnValue

//合并方式，1 叠加列合并 2 新增列合并

//先实现简单的 新增列合并

//在执行前进行sets[3]相等检查

if(rowValueRowOutputTempIteratorCulumnValue.get("culumnValue").equals(culumnValue.get("culumnValue"))) {

if(sets[4].equalsIgnoreCase("模式为")) {

model(sets, rowValue, 列标识, rowValueRowOutputTempIterator);

}

}

//其他定状补语 函数

//。。。

//。。。

//。

}

rowOutputTempIterator.put("rowValue", rowValueRowOutputTempIterator);

}

}

outputTemp.add(rowOutputTempIterator);

}

}

//if(sets[2].equalsIgnoreCase("自由定义各种命令。。")) {

////To do。。。

// }

output.clear();

output.addAll(outputTemp);

}

//设计点 相交

//PLETL:中节点|进行表格相交|主码为|ID|模式为|新增列;

//PLETL:中节点|进行表格相交|主码为|ID|模式为|叠加列;

//PLETL:中节点|进行表格相交|主码为|ID|模式为|有交集叠加列;

//PLETL:中节点|进行表格相交|主码为|ID|模式为|有交集新增列;

//PLETL:中节点|进行表格相交|主码为|ID|模式为|无交集新增列;

if(sets[1].equalsIgnoreCase("进行表格相交")) {

TinMap mapShell= null;

String 列标识= null;

if(sets[0].equalsIgnoreCase("中节点")) {

mapShell= (TinMap)object.get("midShell");

列标识= "m\_";

}

if(sets[0].equalsIgnoreCase("下节点")) {

mapShell= (TinMap)object.get("downShell");

列标识= "d\_";

}

//将上面进行内外循环 颠倒rotation 如下

if(sets[2].equalsIgnoreCase("主码为")) {//先单一primary key， 之后再设计 forenge key 和 combination key

//To do。。。初始

Map<String, Object> tinShellETL= (Map<String, Object>)mapShell.get("TinShellETL");

List<Map<String, Object>> rowList= (List<Map<String, Object>>)tinShellETL.get("obj");

//主循环

List<Map<String, Object>> 主要输入轮训= (List<Map<String, Object>>)object.get("obj");

Iterator<Map<String, Object>> outputTempIterator= 主要输入轮训.iterator();

while(outputTempIterator.hasNext()) {

Map<String, Object> rowOutputTempIterator= outputTempIterator.next();

Map<String, Object> rowValueRowOutputTempIterator

= (Map<String, Object>)rowOutputTempIterator.get("rowValue");

boolean findConjunction= false;

if(0!= rowList.size()) {

//辅循环

Iterator<Map<String, Object>> iterator= rowList.iterator();

while(iterator.hasNext()) {//非主要输入轮训

Map<String, Object> row= iterator.next();

Map<String, Object> rowValue= (Map<String, Object>)row.get("rowValue");

Map<String, Object> culumnValue= (Map<String, Object>)rowValue.get(sets[3]);

//outputTemp

if(rowValueRowOutputTempIterator.containsKey(sets[3])) {

Map<String, Object> rowValueRowOutputTempIteratorCulumnValue

= (Map<String, Object>)rowValueRowOutputTempIterator.get(sets[3]);

//合并rowValueRowOutputTempIteratorCulumnValue 与 culumnValue

//合并方式，1 叠加列合并 2 新增列合并

//先实现简单的 新增列合并

//在执行前进行sets[3]相等检查

if(rowValueRowOutputTempIteratorCulumnValue.get("culumnValue").equals(culumnValue.get("culumnValue"))) {//以后命令多了优化

findConjunction= true;

if(sets[4].equalsIgnoreCase("模式为")) {

model(sets, rowValue, 列标识, rowValueRowOutputTempIterator);

}

}

//其他定状补语 函数

//。。。

//。。。

//。

}

rowOutputTempIterator.put("rowValue", rowValueRowOutputTempIterator);

}

}

if(true= = findConjunction) {//有交集的行才保留

outputTemp.add(rowOutputTempIterator);

}

}

}

//if(sets[2].equalsIgnoreCase("自由定义各种命令。。")) {

////To do。。。

// }

output.clear();

output.addAll(outputTemp);

}

//PLETL:中节点|进行表格剔除|主码为|ID|模式为|相交部分剔除;

if(sets[1].equalsIgnoreCase("进行表格剔除")) {

TinMap mapShell= null;

String 列标识= null;

if(sets[0].equalsIgnoreCase("中节点")) {

mapShell= (TinMap)object.get("midShell");

列标识= "m\_";

}

if(sets[0].equalsIgnoreCase("下节点")) {

mapShell= (TinMap)object.get("downShell");

列标识= "d\_";

}

//将上面进行内外循环 颠倒rotation 如下

if(sets[2].equalsIgnoreCase("主码为")) {//先单一primary key， 之后再设计 forenge key 和 combination key

//To do。。。初始

Map<String, Object> tinShellETL= (Map<String, Object>)mapShell.get("TinShellETL");

List<Map<String, Object>> rowList= (List<Map<String, Object>>)tinShellETL.get("obj");

//主循环

List<Map<String, Object>> 主要输入轮训= (List<Map<String, Object>>)object.get("obj");

Iterator<Map<String, Object>> outputTempIterator= 主要输入轮训.iterator();

while(outputTempIterator.hasNext()) {

Map<String, Object> rowOutputTempIterator= outputTempIterator.next();

Map<String, Object> rowValueRowOutputTempIterator

= (Map<String, Object>)rowOutputTempIterator.get("rowValue");

boolean findConjunction= false;

if(0!= rowList.size()) {

//辅循环

Iterator<Map<String, Object>> iterator= rowList.iterator();

while(iterator.hasNext()) {//非主要输入轮训

Map<String, Object> row= iterator.next();

Map<String, Object> rowValue= (Map<String, Object>)row.get("rowValue");

Map<String, Object> culumnValue= (Map<String, Object>)rowValue.get(sets[3]);

//outputTemp

if(rowValueRowOutputTempIterator.containsKey(sets[3])) {

Map<String, Object> rowValueRowOutputTempIteratorCulumnValue

= (Map<String, Object>)rowValueRowOutputTempIterator.get(sets[3]);

//合并rowValueRowOutputTempIteratorCulumnValue 与 culumnValue

//合并方式，1 叠加列合并 2 新增列合并

//先实现简单的 新增列合并

//在执行前进行sets[3]相等检查

if(rowValueRowOutputTempIteratorCulumnValue.get("culumnValue").equals(culumnValue.get("culumnValue"))) {//以后命令多了优化

findConjunction= true;

}

//其他定状补语 函数

//。。。

//。。。

//。

}

rowOutputTempIterator.put("rowValue", rowValueRowOutputTempIterator);

}

}

if(false= = findConjunction) {//无交集的行才保留

outputTemp.add(rowOutputTempIterator);

}

}

}

//if(sets[2].equalsIgnoreCase("自由定义各种命令。。")) {

////To do。。。

// }

output.clear();

output.addAll(outputTemp);

}

}

//之后这个定状补的函数我会分出去 结构化 罗瑶光 20211012

@SuppressWarnings("unchecked")

private static void model(String[] sets, Map<String, Object> rowValue, String 列标识

, Map<String, Object> rowValueRowOutputTempIterator) {

if(sets[5].equalsIgnoreCase("新增列")) {

Iterator<String> iteratorCulumnValue= rowValue.keySet().iterator();

while(iteratorCulumnValue.hasNext()) {

String string= iteratorCulumnValue.next();

Map<String, Object> culumnCell= (Map<String, Object>) rowValue.get(string);

culumnCell.put("culumnName", 列标识+ string);

rowValueRowOutputTempIterator.put(列标识+ string, culumnCell);

//先这样，测试下

}

}

//叠加列

if(sets[5].equalsIgnoreCase("叠加列")) {

//列遍历

Iterator<String> iteratorCulumnValue= rowValue.keySet().iterator();

while(iteratorCulumnValue.hasNext()) {

String string= iteratorCulumnValue.next();

//列操作

Map<String, Object> culumnCell= (Map<String, Object>) rowValue.get(string);

if(rowValueRowOutputTempIterator.containsKey(string)) {

//有就叠加

Map<String, Object> culumnCellMain

= (Map<String, Object>) rowValueRowOutputTempIterator.get(string);

culumnCellMain.put("culumnValue", culumnCellMain.get("culumnValue").toString()

+ culumnCell.get("culumnValue").toString() );

rowValueRowOutputTempIterator.put(string, culumnCellMain);

}else {

//没有就添加

culumnCell.put("culumnName", 列标识+ string);

rowValueRowOutputTempIterator.put(列标识+ string, culumnCell);

}

}

}

//有交集列 叠加

if(sets[5].equalsIgnoreCase("有交集叠加列")) {

//列遍历

Iterator<String> iteratorCulumnValue= rowValue.keySet().iterator();

while(iteratorCulumnValue.hasNext()) {

String string= iteratorCulumnValue.next();

//列操作

Map<String, Object> culumnCell= (Map<String, Object>) rowValue.get(string);

if(rowValueRowOutputTempIterator.containsKey(string)) {

//有就叠加

Map<String, Object> culumnCellMain

= (Map<String, Object>) rowValueRowOutputTempIterator.get(string);

culumnCellMain.put("culumnValue", culumnCellMain.get("culumnValue").toString()

+ culumnCell.get("culumnValue").toString());

rowValueRowOutputTempIterator.put(string, culumnCellMain);

}

}

}

//有交集列 新增

if(sets[5].contains("交集新增列")) {

//列遍历

Iterator<String> iteratorCulumnValue= rowValue.keySet().iterator();

while(iteratorCulumnValue.hasNext()) {

String string= iteratorCulumnValue.next();

//列操作

Map<String, Object> culumnCell= (Map<String, Object>) rowValue.get(string);

if(sets[5].equalsIgnoreCase("有交集新增列")) {

if(rowValueRowOutputTempIterator.containsKey(string)) {

culumnCell.put("culumnName", 列标识+ string);

rowValueRowOutputTempIterator.put(列标识+ string, culumnCell);

}

}else if(sets[5].equalsIgnoreCase("无交集新增列")) {

if(!rowValueRowOutputTempIterator.containsKey(string)) {

culumnCell.put("culumnName", 列标识+ string);

rowValueRowOutputTempIterator.put(列标识+ string, culumnCell);

}

}

}

}

//相交部分剔除

}

}

-------------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.IOException;

import java.math.BigDecimal;

import java.net.URLEncoder;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import ME.APM.VSQ.HRJFrame;

import MSV.OSQ.sets.DetaDouble;

import OEU.LYG4DQS4D.LYG10DWCMSSort13D\_XCDX\_C\_A\_S;

import OEU.LYG4DQS4D.LYG9DWithDoubleTopSort4D;

import OEU.LYG4DQS4D.Quick\_7D\_luoyaoguang\_Sort;

import OSI.AOP.PCS.PP.port\_E.RestNLPPortImpl;

import PEQ.AMV.ECS.test.ANNTest;

import PEQ.AMV.ECS.test.DNNTest;

import PEQ.AMV.ECS.test.SensingTest;

import PEU.P.table.TableSorterZYNK;

@SuppressWarnings({"unused"})

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

public class P\_AggregationPLSearch {

@SuppressWarnings({ "unchecked" })

public static void P\_AggregationLimitMap(String[] sets

, List<Map<String, Object>> output) throws InstantiationException, IllegalAccessException, IOException {

List<Map<String, Object>> outputTemp= new ArrayList<>();

if(sets[1].equalsIgnoreCase("sortNumber")) {

//outputTemp 是一个arraylist，已经具备了 排序的 模子。

//这里通常会有数字和字符串2种模式，

//于是设计sortNumber， sortString两个语法先

return;

}

if(sets[1].equalsIgnoreCase("进行字符排序")) {

//outputTemp 是一个 arraylist，已经具备了 排序的 模子。

//这里通常会有数字和字符串2种模式，

//于是设计sortNumber， sortString两个语法先

//outputTemp

//先把之前的文字拼音笔画排序接口拿过来，

//然后面向该接口进行封装适应这里的功能。

//看怎么改

outputTemp.addAll(output);

//1 list 存map

Map<String, Map<String, Object>> maps= new HashMap<>();

Iterator<Map<String, Object>> iterators= outputTemp.iterator();

String[] strings= new String[outputTemp.size()];

int index= 0;

while(iterators.hasNext()) {

Map<String, Object> map= iterators.next();

Map<String, Object> rowValue= (Map<String, Object>)map.get("rowValue");

Map<String, Object> culumnValue= (Map<String, Object>)rowValue.get(sets[0]);

maps.put(culumnValue.get("culumnValue").toString(), map);

strings[index++]= culumnValue.get("culumnValue").toString();

}

//2 list 去map名

//3 sort map名

SortStringDemo.initMap();

int returnInt= new LYG10DWCMSSort13D\_XCDX\_C\_A\_S()

.quick4DChineseStringArrayWithSmallInTwoChar3bihuaReturns(strings

, 0, strings.length- 1, 80, SortStringDemo.pinYin

, SortStringDemo.biHua, 7, 70);

//4 输出

outputTemp.clear();

if(sets[2].equalsIgnoreCase("从小到大")) {

for(int i= 0; i< strings.length; i++) {

outputTemp.add(maps.get(strings[i]));

}

}else if(sets[2].equalsIgnoreCase("从大到小")) {

for(int i= 0; i< strings.length; i++) {

outputTemp.add(maps.get(strings[strings.length- i- 1]));

}

}

output.clear();

output.addAll(outputTemp);

return;

}

if(sets[1].equalsIgnoreCase("进行数字排序")) {

//outputTemp 是一个 arraylist，已经具备了 排序的 模子。

//这里通常会有数字和字符串2种模式，

//于是设计sortNumber， sortString两个语法先

//outputTemp

//先把之前的文字拼音笔画排序接口拿过来，

//然后面向该接口进行封装适应这里的功能。

//看怎么改

outputTemp.addAll(output);

//1 list 存map

Map<String, Map<String, Object>> maps= new HashMap<>();

Iterator<Map<String, Object>> iterators= outputTemp.iterator();

double[] doubles= new double[outputTemp.size()];

int index= 0;

while(iterators.hasNext()) {

Map<String, Object> map= iterators.next();

Map<String, Object> rowValue= (Map<String, Object>)map.get("rowValue");

Map<String, Object> culumnValue= (Map<String, Object>)rowValue.get(sets[0]);

maps.put(culumnValue.get("culumnValue").toString(), map);

doubles[index++]= Double.valueOf(culumnValue.get("culumnValue").toString());

}

//2 list 去map名

//3 sort map名

//SortStringDemo.initMap();

//int returnInt= new LYG10DWCMSSort13D\_XCDX\_C\_A\_S()

// .quick4DChineseStringArrayWithSmallInTwoChar3bihuaReturns(strings

// , 0, strings.length- 1, 80, SortStringDemo.pinYin

// , SortStringDemo.biHua, 7, 70);

new LYG9DWithDoubleTopSort4D().sort(doubles, 7, 70);

//4 输出

outputTemp.clear();

if(sets[2].equalsIgnoreCase("从小到大")) {

for(int i= 0; i< doubles.length; i++) {

outputTemp.add(maps.get(""+ (int)doubles[i]));

}

}else if(sets[2].equalsIgnoreCase("从大到小")) {

for(int i= 0; i< doubles.length; i++) {

outputTemp.add(maps.get(""+ (int)doubles[doubles.length- i- 1]));

}

}

output.clear();

output.addAll(outputTemp);

return;

}

if(sets[1].equalsIgnoreCase("行至")) {

Iterator<Map<String, Object>> iterator= output.iterator();

int count= 0;

while(iterator.hasNext()) {

int rowid= count++;

Map<String, Object> row= iterator.next();

Map<String, Object> rowMap= new HashMap<>();

if(sets[1].equalsIgnoreCase("行至")) {

if(rowid >= new BigDecimal(sets[0]).doubleValue() && rowid

<= new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(row);

}

}

}

output.clear();

output.addAll(outputTemp);

return;

}

//稍后我会把这里 改成 contains 德塔DNN词汇，这样语言就自适应了。

//罗瑶光 20211003

if(sets[1].equalsIgnoreCase("颜色标记为")) {

Iterator<Map<String, Object>> iterator= output.iterator();

int count= 0;

while(iterator.hasNext()) {

int rowid= count++;

Map<String, Object> row= iterator.next();

Map<String, Object> rowMap= new HashMap<>();

if(sets[1].equalsIgnoreCase("颜色标记为")) {

Map<String, Object> map= (Map<String, Object>)row.get("rowValue");

Map<String, Object> mapCulumn= (Map<String, Object>)map.get(sets[0]);

String rowCellFromString= mapCulumn.get("culumnValue").toString();

if(sets[2].equals("红色")) {

sets[2]= "red";

}

if(sets[2].equals("黄色")) {

sets[2]= "yellow";

}

if(sets[2].equals("蓝色")) {

sets[2]= "blue";

}

if(sets[2].equals("绿色")) {

sets[2]= "green";

}

rowCellFromString= "<div style= \"background:black\"><font color= \""+ sets[2] +"\">"

+ rowCellFromString+ "</font></div>";

//更新

outputTemp.remove(row);

mapCulumn.put("culumnValue", rowCellFromString);

map.put(sets[0], mapCulumn);

row.put("rowValue", map);

outputTemp.add(row);

}

}

output.clear();

output.addAll(outputTemp);

return;

}

if(sets[1].equalsIgnoreCase("进行分词")) {

Iterator<Map<String, Object>> iterator= output.iterator();

int count= 0;

while(iterator.hasNext()) {

int rowid= count++;

Map<String, Object> row= iterator.next();

Map<String, Object> rowMap= new HashMap<>();

if(sets[2].equalsIgnoreCase("词性显色")) {

Map<String, Object> map= (Map<String, Object>)row.get("rowValue");

Map<String, Object> mapCulumn= (Map<String, Object>)map.get(sets[0]);

String rowCellFromString= mapCulumn.get("culumnValue").toString();

List<String> list= HRJFrame.NE.\_A.parserMixedString(rowCellFromString);

Map<String, String> nlp= HRJFrame.NE.\_A.getPosCnToCn();

Iterator<String> iterators= list.iterator();

rowCellFromString= "";

rowCellFromString+= "<div style= \"background:white\">";

while(iterators.hasNext()) {

String string= iterators.next();

if(nlp.containsKey(string)) {

rowCellFromString+= "<font color= \""+

(!nlp.get(string).contains("动")?!nlp.get(string).contains("名")?!nlp.get(string).contains("形")?

"black": "blue": "red": "green") +"\">"

+ string+ "</font>";

}

}

rowCellFromString+= "</div>";

//rowCellFromString= "<div style= \"background:white\"><font color= \""+ sets[2] +"\">"

//+ rowCellFromString+ "</font></div>";

//更新

outputTemp.remove(row);

mapCulumn.put("culumnValue", rowCellFromString);

map.put(sets[0], mapCulumn);

row.put("rowValue", map);

outputTemp.add(row);

}

if(sets[2].equalsIgnoreCase("词性标注")) {

Map<String, Object> map= (Map<String, Object>)row.get("rowValue");

Map<String, Object> mapCulumn= (Map<String, Object>)map.get(sets[0]);

String rowCellFromString= mapCulumn.get("culumnValue").toString();

List<String> list= HRJFrame.NE.\_A.parserMixedString(rowCellFromString);

Map<String, String> nlp= HRJFrame.NE.\_A.getPosCnToCn();

Iterator<String> iterators= list.iterator();

rowCellFromString= "";

rowCellFromString+= "<div style= \"background:white\">";

while(iterators.hasNext()) {

String string= iterators.next();

if(nlp.containsKey(string)) {

rowCellFromString+= string+ "("+ nlp.get(string)+ ") ";

}

}

rowCellFromString+= "</div>";

//rowCellFromString= "<div style= \"background:white\"><font color= \""+ sets[2] +"\">"

//+ rowCellFromString+ "</font></div>";

//更新

outputTemp.remove(row);

mapCulumn.put("culumnValue", rowCellFromString);

map.put(sets[0], mapCulumn);

row.put("rowValue", map);

outputTemp.add(row);

}

//之后我会把dataCG函数进行重新封装，去重。

if(sets[2].equalsIgnoreCase("DNN")) {

Map<String, Object> map= (Map<String, Object>)row.get("rowValue");

Map<String, Object> mapCulumn= (Map<String, Object>)map.get(sets[0]);

String rowCellFromString= mapCulumn.get("culumnValue").toString();

//

//System.out.printntln(string);

SensingTest sensingTest= HRJFrame.NE.\_A.getSensingTest();

DNNTest dNNTest= new DNNTest();

ANNTest aNNTest= new ANNTest();

String[][] ann= aNNTest.getANNMatrix(sensingTest, rowCellFromString, HRJFrame.NE.\_A);

String[][] dnn= dNNTest.getDNNMatrix(sensingTest, ann, HRJFrame.NE.\_A, rowCellFromString);

List<String> cigan= new LinkedList<>();

Here:

for(int i= 0; i<dnn.length; i++) {

double dnn\_lwa= 0;

if(null= = dnn[i][3]) {

continue Here;

}

dnn\_lwa= DetaDouble.parseDouble(dnn[i][3]);

if(dnn\_lwa>0) {

String line= "";

line+= ann[i][0] + ":";

line+= dnn[i][3] + ":";

cigan.add(line);

}

}

String[][] value= new String[cigan.size()][2];

Iterator<String> iterators= cigan.iterator();

int valueCount= 0;

while(iterators.hasNext()) {

String iteratorString= iterators.next();

value[valueCount][0]= iteratorString.split(":")[0];

value[valueCount++][1]= iteratorString.split(":")[1];

}

//value= new Quick\_6D\_luoyaoguang\_Sort().sort(value);

value= new Quick\_7D\_luoyaoguang\_Sort().sort(value);

String cg= "词名/DNN";

cg+= "\r\n";

for(int i= 0; i<value.length; i++) {

cg += value[i][0] + ":" + value[i][1] + "\r\n";

}

rowCellFromString+= "<div style= \"background:white\">";

rowCellFromString+= cg +"</div>";

//更新

outputTemp.remove(row);

mapCulumn.put("culumnValue", rowCellFromString);

map.put(sets[0], mapCulumn);

row.put("rowValue", map);

outputTemp.add(row);

}

}

output.clear();

output.addAll(outputTemp);

return;

}

// //操作:进行合并:列名:上中下

//

// if(sets[1].equalsIgnoreCase("进行合并")) {

//

// }

// //稍后把这里 行遍历 改成 命令遍历。提高计算速度

// //罗瑶光20211002

// Iterator<Map<String, Object>> iterator= output.iterator();

// int count= 0;

// while(iterator.hasNext()) {

//int rowid= count++;

//Map<String, Object> row= iterator.next();

//Map<String, Object> rowMap= new HashMap<>();

//if(sets[1].equalsIgnoreCase("行至")) {

// if(rowid >= new BigDecimal(sets[0]).doubleValue() && rowid

// <= new BigDecimal(sets[2]).doubleValue()) {

// outputTemp.add(row);

// }

//}

//

//if(sets[1].equalsIgnoreCase("颜色")) {

// Map<String, Object> map= (Map<String, Object>)row.get("rowValue");

// Map<String, Object> mapCulumn= (Map<String, Object>)map.get(sets[0]);

// String rowCellFromString= mapCulumn.get("culumnValue").toString();

// if(sets[2].equals("红色")) {

// sets[2]= "red";

// }

// if(sets[2].equals("黄色")) {

// sets[2]= "yellow";

// }

// if(sets[2].equals("蓝色")) {

// sets[2]= "blue";

// }

// if(sets[2].equals("绿色")) {

// sets[2]= "green";

// }

// rowCellFromString= "<div style= \"background:black\"><font color= \""+ sets[2] +"\">"

// + rowCellFromString+ "</font></div>";

// //更新

// outputTemp.remove(row);

// mapCulumn.put("culumnValue", rowCellFromString);

// map.put(sets[0], mapCulumn);

// row.put("rowValue", map);

// outputTemp.add(row);

//}

//

//if(sets[1].equalsIgnoreCase("分词")) {

// Map<String, Object> map= (Map<String, Object>)row.get("rowValue");

// Map<String, Object> mapCulumn= (Map<String, Object>)map.get(sets[0]);

// String rowCellFromString= mapCulumn.get("culumnValue").toString();

// List<String> list= HRJFrame.NE.\_A.parserMixedString(rowCellFromString);

// Map<String, String> nlp= HRJFrame.NE.\_A.getPosCnToCn();

// Iterator<String> iterators= list.iterator();

// rowCellFromString= "";

// rowCellFromString+= "<div style= \"background:white\">";

// while(iterators.hasNext()) {

// String string= iterators.next();

// if(nlp.containsKey(string)) {

// rowCellFromString+= "<font color= \""+

// (!nlp.get(string).contains("动")?!nlp.get(string).contains("名")?!nlp.get(string).contains("形")?

// "black": "blue": "red": "green") +"\">"

// + string+ "</font>";

// }

// }

// rowCellFromString+= "</div>";

// //rowCellFromString= "<div style= \"background:white\"><font color= \""+ sets[2] +"\">"

// //+ rowCellFromString+ "</font></div>";

// //更新

// outputTemp.remove(row);

// mapCulumn.put("culumnValue", rowCellFromString);

// map.put(sets[0], mapCulumn);

// row.put("rowValue", map);

// outputTemp.add(row);

//}

// }

output.clear();

output.addAll(outputTemp);

}

//分出去

public static void P\_PletlLimitMap(String[] sets, List<Map<String, Object>> obj) {

// TODO Auto-generated method stub

}

}

--------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.UnsupportedEncodingException;

import java.math.BigDecimal;

import java.net.HttpURLConnection;

import java.net.URL;

import java.net.URLEncoder;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import ESU.string.String\_ESU;

import ME.APM.VSQ.HRJFrame;

import MSU.AMS.VQS.SQV.SI.OSU.SMV.http.RestCall;

import MSU.AMS.VQS.SQV.SI.OSU.SMV.http.SessionValidation;

import MSV.OSQ.sets.DetaDouble;

import OEU.LYG4DQS4D.LYG10DWCMSSort13D\_XCDX\_C\_A\_S;

import OEU.LYG4DQS4D.LYG9DWithDoubleTopSort4D;

import OEU.LYG4DQS4D.Quick\_7D\_luoyaoguang\_Sort;

import OSI.AOP.PCS.PP.port\_E.RestNLPPortImpl;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.TinMap;

import PEQ.AMV.ECS.test.ANNTest;

import PEQ.AMV.ECS.test.DNNTest;

import PEQ.AMV.ECS.test.SensingTest;

import PEU.P.dna.Token;

import PEU.P.dna.TokenCerts;

import PEU.P.table.TableSorterZYNK;

@SuppressWarnings({"unused"})

//这个文件主要用来设计关于web的rest，server，http 请求

//+ "PLTCP:病症|进行WEB请求|接口为|localhost|端口为|8000|操作为|分词;" (正在设计）

//+ "PLTCP:病症|进行WEB请求|接口为|localhost|端口为|8000|操作为|DNN;" (正在设计）

//+ "PLTCP:病症|进行WEB请求|接口为|localhost|端口为|8000|操作为|POS;" (正在设计）

//罗瑶光 20211014

public class P\_AggregationPLTCP {

@SuppressWarnings("unchecked")

public static void P\_PltcpLimitMap(String[] sets, List<Map<String, Object>> output

, Map<String, Object> object) throws IOException {

List<Map<String, Object>> outputTemp= new ArrayList<>();

if(sets[1].equalsIgnoreCase("进行WEB请求")) {

//主循环

List<Map<String, Object>> 主要输入轮训= (List<Map<String, Object>>)object.get("obj");

Iterator<Map<String, Object>> outputTempIterator= 主要输入轮训.iterator();

while(outputTempIterator.hasNext()) {

Map<String, Object> rowOutputTempIterator= outputTempIterator.next();

Map<String, Object> rowValueRowOutputTempIterator

= (Map<String, Object>)rowOutputTempIterator.get("rowValue");

//先固定好主谓宾格式，以后再来设计宾补的格式，目前先按定语来。

String server= sets[3];//稍后设计 安全检测。

String port= sets[5];

if(sets[7].contains("分词")) {

String setOfi= rowValueRowOutputTempIterator.get(sets[0]).toString();

String string= String\_ESU.charsetSwap(setOfi, "GBK", "GBK");

String encode= String\_ESU.stringToURIencode(string, "UTF8");

//String response= RestCall.backEndRequest(encode);

//模拟加个测试账号: 313699483@QQ.COM, 密码: fengyue1985

String id= "313699483@QQ.COM";

String idString= String\_ESU.charsetSwap(id, "GBK", "GBK");

String idEncoder= String\_ESU.stringToURIencode(idString, "UTF8");

String password= "Fengyue1985!";

String lock= "AISD>\_<111111111111111>\_<11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111"

+ ">\_<1111111111111111>\_<10111011101101101110110110110110111011011011101110110110"

+ "1101101101101101110110110111011011011011011011011101101101101101101110110111011101";

String[] MD5dice\_DNA= lock.split(">\_<");

//DNA元基加密

SessionValidation sessionValidation= new SessionValidation();

TokenCerts tokenCerts= sessionValidation.sessionTokenCertsInitWithHumanWordsByDNA(password, true, MD5dice\_DNA[0]);

Token token= sessionValidation.sessionInitByTokenPDICertsDNA(tokenCerts);

String passwordString= String\_ESU.charsetSwap(token.getmPassword(), "GBK", "GBK");

String passwordEncoder= String\_ESU.stringToURIencode(passwordString, "UTF8");

System.out.println("pds--1>"+tokenCerts.getPds());

//这里的数据下面没有标识，准备写个tag来描述下先

//java 从没有url的长度限制，如果出问题就会不报错，欺骗方式下滑。

//所以准备写个tag。

//localhost 我会做个表来描述。

//罗瑶光 20210731

URL url= new URL("http://"+ server+ ":"+ port+ "/dataWS?message= "+ encode+

"&id= "+ idEncoder+

"&password= "+ password+

"&de= "+ MD5dice\_DNA[1] +

"&ds= "+ MD5dice\_DNA[2] +

"&ie= "+ MD5dice\_DNA[3] +

"&is= "+ MD5dice\_DNA[4] +

"&lock= "+ MD5dice\_DNA[0]);

HttpURLConnection conn= (HttpURLConnection) url.openConnection();

//这里是url 的httpconnection, 只能服务器网卡链接情况下使用,如果要本机断网调试, 请查阅网卡相关函数.

//与程序功能无关, 以后讨论.

//HttpConnection conn= (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

conn.setRequestProperty("Accept", "application/json");

if (conn.getResponseCode() != 200) {

throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());

}

BufferedReader br= new BufferedReader(new InputStreamReader((conn.getInputStream()), "GBK"));

String out= "";

String out1;

while ((out1= br.readLine()) != null) {

out += out1;

}

conn.disconnect();

//

String[] strings= out.split("\"");

out= strings.length> 3? strings[3]: "";

out= String\_ESU.uRIencodeToURIdecode(out, "UTF8");

//输出存储替换

rowValueRowOutputTempIterator.put(sets[0], out);

rowOutputTempIterator.put("rowValue", rowValueRowOutputTempIterator);

outputTemp.add(rowOutputTempIterator);

}

if(sets[7].contains("DNN")) {

String setOfi= rowValueRowOutputTempIterator.get(sets[0]).toString();

String string= String\_ESU.charsetSwap(setOfi, "GBK", "GBK");

String encode= String\_ESU.stringToURIencode(string, "UTF8");

//String response= RestCall.backEndRequest(encode);

//模拟加个测试账号: 313699483@QQ.COM, 密码: fengyue1985

String id= "313699483@QQ.COM";

String idString= String\_ESU.charsetSwap(id, "GBK", "GBK");

String idEncoder= String\_ESU.stringToURIencode(idString, "UTF8");

String password= "Fengyue1985!";

String lock= "AISD>\_<111111111111111>\_<11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111"

+ ">\_<1111111111111111>\_<10111011101101101110110110110110111011011011101110110110"

+ "1101101101101101110110110111011011011011011011011101101101101101101110110111011101";

String[] MD5dice\_DNA= lock.split(">\_<");

//DNA元基加密

SessionValidation sessionValidation= new SessionValidation();

TokenCerts tokenCerts= sessionValidation.sessionTokenCertsInitWithHumanWordsByDNA(password, true, MD5dice\_DNA[0]);

Token token= sessionValidation.sessionInitByTokenPDICertsDNA(tokenCerts);

String passwordString= String\_ESU.charsetSwap(token.getmPassword(), "GBK", "GBK");

String passwordEncoder= String\_ESU.stringToURIencode(passwordString, "UTF8");

System.out.println("pds--1>"+tokenCerts.getPds());

//这里的数据下面没有标识，准备写个tag来描述下先

//java 从没有url的长度限制，如果出问题就会不报错，欺骗方式下滑。

//所以准备写个tag。

//localhost 我会做个表来描述。

//罗瑶光 20210731

URL url= new URL("http://"+ server+ ":"+ port+ "/dataCG?message= "+ encode+

"&id= "+ idEncoder+

"&password= "+ password+

"&de= "+ MD5dice\_DNA[1] +

"&ds= "+ MD5dice\_DNA[2] +

"&ie= "+ MD5dice\_DNA[3] +

"&is= "+ MD5dice\_DNA[4] +

"&lock= "+ MD5dice\_DNA[0]);

HttpURLConnection conn= (HttpURLConnection) url.openConnection();

//这里是url 的httpconnection, 只能服务器网卡链接情况下使用,如果要本机断网调试, 请查阅网卡相关函数.

//与程序功能无关, 以后讨论.

//HttpConnection conn= (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

conn.setRequestProperty("Accept", "application/json");

if (conn.getResponseCode() != 200) {

throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());

}

BufferedReader br= new BufferedReader(new InputStreamReader((conn.getInputStream()), "GBK"));

String out= "";

String out1;

while ((out1= br.readLine()) != null) {

out += out1;

}

conn.disconnect();

//

String[] strings= out.split("\"");

out= strings.length> 3? strings[3]: "";

out= String\_ESU.uRIencodeToURIdecode(out, "UTF8");

//输出存储替换

rowValueRowOutputTempIterator.put(sets[0], out);

rowOutputTempIterator.put("rowValue", rowValueRowOutputTempIterator);

outputTemp.add(rowOutputTempIterator);

}

if(sets[7].contains("POS")) {

String setOfi= rowValueRowOutputTempIterator.get(sets[0]).toString();

String string= String\_ESU.charsetSwap(setOfi, "GBK", "GBK");

String encode= String\_ESU.stringToURIencode(string, "UTF8");

//String response= RestCall.backEndRequest(encode);

//模拟加个测试账号: 313699483@QQ.COM, 密码: fengyue1985

String id= "313699483@QQ.COM";

String idString= String\_ESU.charsetSwap(id, "GBK", "GBK");

String idEncoder= String\_ESU.stringToURIencode(idString, "UTF8");

String password= "Fengyue1985!";

String lock= "AISD>\_<111111111111111>\_<11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111"

+ ">\_<1111111111111111>\_<10111011101101101110110110110110111011011011101110110110"

+ "1101101101101101110110110111011011011011011011011101101101101101101110110111011101";

String[] MD5dice\_DNA= lock.split(">\_<");

//DNA元基加密

SessionValidation sessionValidation= new SessionValidation();

TokenCerts tokenCerts= sessionValidation.sessionTokenCertsInitWithHumanWordsByDNA(password, true, MD5dice\_DNA[0]);

Token token= sessionValidation.sessionInitByTokenPDICertsDNA(tokenCerts);

String passwordString= String\_ESU.charsetSwap(token.getmPassword(), "GBK", "GBK");

String passwordEncoder= String\_ESU.stringToURIencode(passwordString, "UTF8");

System.out.println("pds--1>"+tokenCerts.getPds());

//这里的数据下面没有标识，准备写个tag来描述下先

//java 从没有url的长度限制，如果出问题就会不报错，欺骗方式下滑。

//所以准备写个tag。

//localhost 我会做个表来描述。

//罗瑶光 20210731

URL url= new URL("http://"+ server+ ":"+ port+ "/dataCX?message= "+ encode+

"&id= "+ idEncoder+

"&password= "+ password+

"&de= "+ MD5dice\_DNA[1] +

"&ds= "+ MD5dice\_DNA[2] +

"&ie= "+ MD5dice\_DNA[3] +

"&is= "+ MD5dice\_DNA[4] +

"&lock= "+ MD5dice\_DNA[0]);

HttpURLConnection conn= (HttpURLConnection) url.openConnection();

//这里是url 的httpconnection, 只能服务器网卡链接情况下使用,如果要本机断网调试, 请查阅网卡相关函数.

//与程序功能无关, 以后讨论.

//HttpConnection conn= (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

conn.setRequestProperty("Accept", "application/json");

if (conn.getResponseCode() != 200) {

throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());

}

BufferedReader br= new BufferedReader(new InputStreamReader((conn.getInputStream()), "GBK"));

String out= "";

String out1;

while ((out1= br.readLine()) != null) {

out += out1;

}

conn.disconnect();

//

String[] strings= out.split("\"");

out= strings.length> 3? strings[3]: "";

out= String\_ESU.uRIencodeToURIdecode(out, "UTF8");

//输出存储替换

rowValueRowOutputTempIterator.put(sets[0], out);

rowOutputTempIterator.put("rowValue", rowValueRowOutputTempIterator);

outputTemp.add(rowOutputTempIterator);

}

}

}

output.clear();

output.addAll(outputTemp);

}

}

-------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import AVQ.ASQ.OVQ.OSQ.VSQ.obj.WordFrequency;

import ESU.list.List\_ESU;

import ESU.sort.Quick9DLYGWithString\_ESU;

import ME.APM.VSQ.HRJFrame;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache\_M;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Table;

import OSA.shell.SearchShellTable;

import OSA.shell.SearchShellTables;

@SuppressWarnings({"unused", "unchecked"})

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

public class P\_ConditionPLSearch\_XCDX\_Cache extends P\_ConditionPLSearch\_XCDX {

public static void P\_Cache(String[] sets, List<Map<String, Object>> output

, String tableName, Map<String, Object> object, String condition) {

//Table table= DetaDBBufferCache\_M.db.getBase(baseName).getTable(tableName);

//我没有将search shell 用DetaDBBufferCache\_M 内存机制来存储是因为 table操作是增删改查严谨操作。

//而search shell 操作是筛选查找 极速操作，严谨操作与极速操作 的应用环境有天壤之别。

//SearchShellTable table= SearchShellTables.searchShellTables.get(tableName);

//稍后把下面的 Table 替换成 SearchShellTable 即可。

//Table table= DetaDBBufferCache\_M.db.getBase(baseName).getTable(tableName);

//算了统一接口， 以后统一优化改。

List<Map<String, Object>> outputTemp= new ArrayList<>();

//创建一个table

SearchShellTable table= null;

outputTemp.addAll(output);

if(outputTemp.isEmpty()||condition.equals("或")||condition.equals("和")) { //因为改成了or map，所以只有原表重新load

table= SearchShellTables.searchShellTables.get(tableName);

}else if(!outputTemp.isEmpty()&& condition.equals("和")) {//这里不会再走了。成了伪函数

Row[] huaRuiJiJtableRows= new Row[outputTemp.size()];

for(int i= 0; i< outputTemp.size(); i++) {

huaRuiJiJtableRows[i]= P\_ConditionPLSearch\_XCDX\_Map.rowMapToRow(outputTemp.get(i));

}

table= new SearchShellTable();

table.setHuaRuiJiJtableRows(huaRuiJiJtableRows);

}

//修改下把output的逻辑重复利用 满足conditon的and 和or

//只拿前100行 以后改成分页

//稍后把这个函数片段移除这个文件，变成一个函数。

if(sets[1].equalsIgnoreCase("精度搜索")) {

//table to object

//稍后我的养疗经界面搜索release函数的 片段 这里也可以优化如下。

String key= sets[2];

if(null= = key|| key.equals("")) {

return;

}

String[] score= new String[table.huaRuiJiJtableRows.length];

int[] score\_code= new int[table.huaRuiJiJtableRows.length];

int []reg= new int[table.huaRuiJiJtableRows.length];

int count= 0;

Map<String, String> pos= HRJFrame.NE.\_A.getPosCnToCn();

Map<String, WordFrequency> mapSearchWithoutSort= null;

mapSearchWithoutSort= HRJFrame.NE.\_A.parserMixStringByReturnFrequencyMap(key);

//Iterator<String> iteratorForCopy= copy.iterator();

int copyCount= 0;

List<String> list= HRJFrame.NE.\_A.parserMixedString(key);

String[] string= List\_ESU.listToArray(list);

String[] stringReg= new String[key.length()/3];

for(int i= 0; i< stringReg.length; i++) {

stringReg[i]= key.substring(i\*3, (i\*3+ 3)<key.length()?(i\*3+ 3):key.length()-1);

}

Map<String, Row> map= new HashMap<>();

for(int i= 0; i< table.huaRuiJiJtableRows.length; i++) {

//while(iteratorForCopy.hasNext()) {

String temps= table.huaRuiJiJtableRows[i].getCell(sets[0]).getCellValue().toString();

// if(null= = temps) {

// temps= "";

// }

score[copyCount]= "i"+ i;//因为 不再有map key，所以就通用为map 内容。

map.put(score[copyCount], table.huaRuiJiJtableRows[i]);

//String iteratorForCopyString= iteratorForCopy.next();

//score[copyCount]= iteratorForCopyString;

//String temps= dic\_map.get(iteratorForCopyString).toString();

Iterator<String> iteratorWordFrequency= mapSearchWithoutSort.keySet().iterator();

Here:

while(iteratorWordFrequency.hasNext()) {

String mapSearchaAtII= iteratorWordFrequency.next();

WordFrequency wordFrequencySearch= mapSearchWithoutSort.get(mapSearchaAtII);

if(temps.contains(mapSearchaAtII)) {

if(reg[copyCount]= = 0){

count += 1;

}

//score[copyCount]= temps;//因为 不再有map key，

//所以就通用为map 内容。，还是需要map

// if(score[copyCount].contains(key.replace(" ", ""))) {

// reg[copyCount]+= 500;

// }

// if(key.contains(score[copyCount].replace(" ", ""))) {

// reg[copyCount]+= 500;

//}

if(temps.contains(key.replace(" ", ""))) {

reg[copyCount]+= 500;

}

if(key.contains(temps.replace(" ", ""))) {

reg[copyCount]+= 500;

}

if(!pos.containsKey(mapSearchaAtII)) {

reg[copyCount] += 1;

score\_code[copyCount] += 1 << mapSearchaAtII.length() << wordFrequencySearch.getFrequency() ;

continue Here;

}

if(pos.get(mapSearchaAtII).contains("名")||pos.get(mapSearchaAtII).contains("动")

||pos.get(mapSearchaAtII).contains("形")||pos.get(mapSearchaAtII).contains("谓")) {

reg[copyCount] += 2;

}

reg[copyCount] += 1;

score\_code[copyCount] += (temps.contains(mapSearchaAtII) ? 2 : 1)

\* (!pos.get(mapSearchaAtII).contains("名")

? pos.get(mapSearchaAtII).contains("动")? 45 : 1 : 50)

<< mapSearchaAtII.length() \* wordFrequencySearch.getFrequency();

continue Here;

}

if(mapSearchaAtII.length()>1) {

for(int j= 0;j<mapSearchaAtII.length();j++) {

if(temps.contains(String.valueOf(mapSearchaAtII.charAt(j)))) {

if(reg[copyCount]= = 0){

count += 1;

}

// score[copyCount]= temps;

score\_code[copyCount]+= 1;

if(pos.containsKey(String.valueOf(mapSearchaAtII.charAt(j)))&&(

pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("名")

||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("动")

||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("形")

||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("谓")

)) {

reg[copyCount] += 2;

}

reg[copyCount] += 1;

continue Here;

}

}

}

}

score\_code[copyCount]= score\_code[copyCount] \* reg[copyCount];

//词距

int code= 100;

int tempb= 0;

int tempa= score\_code[copyCount];

if(key.length()> 4) {

//全词

for(int j= 0; j< string.length; j++) {

if(temps.contains(string[j])) {

tempb+= code;

}

}

//断句

for(int j= 0; j< stringReg.length; j++) {

if(temps.contains(stringReg[j])) {

tempb+= code;

}

}

score\_code[copyCount]= (int) (tempa/Math.pow(HRJFrame.NE.lookrot+ 1, 4)

+ tempb\*Math.pow(Integer.valueOf(sets[3]), 2));

}

if(key.replace(" ", "").length()> 1&& key.replace(" ", "").length()< 5) {

if(temps.contains(key.replace(" ", ""))) {

tempb+= code<< 7;

}

score\_code[copyCount]= (int) (tempa/Math.pow(Integer.valueOf(sets[3])+ 1, 4)

+ tempb\*Math.pow(Integer.valueOf(sets[3]), 2));

}

copyCount++;

}

LABEL2:

new Quick9DLYGWithString\_ESU().sort(score\_code, score);

int max= score\_code[0];

Object[][] tableData= new Object[count][18];

int new\_count= 0;

//newTableModel.getDataVector().clear();

//if(null= = key|| key.equals("")) {

// return;

//}

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

if(null= = recordRows) {

recordRows= new HashMap<>();

}

recordRows.clear();

//recordRows 没有 值

//recordRows 有 值

Here:

for(int i= score.length- 1; i> 0; i--) {

if(score\_code[i]< 1){

continue Here;

}

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(map.get(score[i])));

recordRows.put(Integer.valueOf(score[i].replace("i", "")), true);

}

object.put("recordRows", recordRows);

return;

}

int max= 50;

//获取table的row

Here:

for(int i= 0; i< table.huaRuiJiJtableRows.length; i++ ) {

//if(i> max) {

// continue Here;

//}

//Object[] row= table.huaRuiJiJtable[count];

//还是要变成map，不然 命令的 key值查询 只能forloop， 效率减低

Row row= table.huaRuiJiJtableRows[i];

Cell cell= new Cell();

cell.I\_CellValue(i); //加id

//出现一个问题，我的table db是非线性map 结构， 自带表头key， 而data 是矩阵，

row.putCell("Index", cell);

if(sets[1].equalsIgnoreCase("<")|| sets[1].equalsIgnoreCase("-lt")) {

double rowCellFromBigDecimal= new BigDecimal(row.getCell(sets[0])

.getCellValue().toString()).doubleValue();

if(rowCellFromBigDecimal< new BigDecimal(sets[2]).doubleValue()

&& row.containsCell("is\_delete\_0")) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("<= ")||sets[1].equalsIgnoreCase("= <")

||sets[1].equalsIgnoreCase("-lte")) {

String set= sets[0];

Cell setCell= row.getCell(set);

String cellString= setCell.getCellValue().toString();

cellString= cellString.isEmpty()? "0": cellString;

double rowCellFromBigDecimal= new BigDecimal(cellString).doubleValue();

if(rowCellFromBigDecimal<= new BigDecimal(sets[2]).doubleValue()

&& row.containsCell("is\_delete\_0")) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("包含")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(rowCellFromString.contains(sets[2])) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

//字符串长度小于

if(sets[1].equalsIgnoreCase("字符串长度大于")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString().trim();

if(rowCellFromString.length()> new BigDecimal(sets[2]).doubleValue()) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

//字符串长度小于

if(sets[1].equalsIgnoreCase("字符串长度小于")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString().trim();

if(rowCellFromString.length()< new BigDecimal(sets[2]).doubleValue()) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("过滤")||sets[1].equalsIgnoreCase("不包含")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(!rowCellFromString.contains(sets[2])) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

//PLSEARCH 准备整体去掉plsql db的 is\_delete\_0 关键字

//罗瑶光 20211015

if(sets[1].equalsIgnoreCase("= = ")||sets[1].equalsIgnoreCase("= ")

||sets[1].equalsIgnoreCase("= = = ")) {

double rowCellFromBigDecimal= new BigDecimal(row.getCell(sets[0])

.getCellValue().toString()).doubleValue();

if(rowCellFromBigDecimal= = new BigDecimal(sets[2]).doubleValue()) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase(">= ")||sets[1].equalsIgnoreCase("= >")

||sets[1].equalsIgnoreCase("-gte")) {

double rowCellFromBigDecimal= new BigDecimal(row.getCell(sets[0])

.getCellValue().toString()).doubleValue();

if(rowCellFromBigDecimal >= new BigDecimal(sets[2]).doubleValue()) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase(">")||sets[1].equalsIgnoreCase("-gt")) {

double rowCellFromBigDecimal= new BigDecimal(row.getCell(sets[0])

.getCellValue().toString()).doubleValue();

if(rowCellFromBigDecimal > new BigDecimal(sets[2]).doubleValue() ) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("!= ")||sets[1].equalsIgnoreCase("= !")) {

double rowCellFromBigDecimal= new BigDecimal(row.getCell(sets[0])

.getCellValue().toString()).doubleValue();

if(rowCellFromBigDecimal != new BigDecimal(sets[2]).doubleValue()) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("equal")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(rowCellFromString.equalsIgnoreCase(sets[2])) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("!equal")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(!rowCellFromString.equalsIgnoreCase(sets[2])) {

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("in")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

String set= "," + sets[2] + ",";

if(set.contains("," + rowCellFromString + ",")){

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

if(sets[1].equalsIgnoreCase("!in")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

String set= "," + sets[2] + ",";

if(!set.contains("," + rowCellFromString + ",")){

if(!((Map<Integer, Boolean>)(object.get("recordRows"))).containsKey(i)) {

output.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");

recordRows.put(i, true);

object.put("recordRows", recordRows);

}

}

}

}

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.List;

import java.util.Map;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

//先替换所有涉及DMA的函数片段

public class P\_ConditionPLSearch\_XCDX\_Kernel extends P\_ConditionPLSearch\_XCDX {

//比较是否有数据取出列表到输出 检验中 罗瑶光 20210405

//这个走硬盘查询函数来标识下, 在我设计了数据缓存查询启动函数 后就没用过了, 时间点大概在2019年1月后, 我先调通下, 之后朔源.

//准备验算下20210406 罗瑶光

public static void P\_kernel(String temp, File readDBTableRowIndexCulumnFile, File readDBTableRowIndexFile

, BufferedReader reader, String DBTableRowIndexPath, List<Map<String, Object>> output, Row bufferRow

, Map<String, Object> rowMap) throws IOException {

String[] culumnList= readDBTableRowIndexFile.list(); //生成一个列表头名组

NextFile:

for(String culumn: culumnList) { //遍历头名

if(culumn.contains("is\_delete")) {//已删除文件逃逸

continue NextFile;

}

String DBTableCulumnIndexPath= readDBTableRowIndexFile + "/" + culumn;//开始取值

File readDBTableCulumnIndexPathFile= new File(DBTableCulumnIndexPath);

Cell cell= new Cell();

if (readDBTableCulumnIndexPathFile.isDirectory()) {

//似乎被动了手脚, 20210405 罗瑶光重新检查

reader= new BufferedReader(new FileReader(readDBTableCulumnIndexPathFile + "/" + "value.lyg"));

temp= "";

String tempString;

while ((tempString= reader.readLine()) != null) {

temp += tempString;

}

reader.close();

rowMap.put(culumn, temp); //储值

cell.I\_CellValue(temp); //数据库内存储值

bufferRow.putCell(culumn, cell);

}else {

rowMap.put(culumn, null);

cell.I\_CellValue(null);

bufferRow.putCell(culumn, cell);

}

}

output.add(rowMap);

}

//P\_kernel等比复制过来的search shell组件， 我要替换的是数据库储值， jtable表做output

//把jtable数据表的数据 变成数据库的db映射， 传入参数是dma的每一行文件的集合，

//DefaultTableModel 的 Object[][] huaRuiJiJtable 对应 DBTablePath

//readDBTableRowIndexFile对应 row id

//huaRuiJiJtableCulumns 对应 culumn

//jtable 太上层，用它的spec 速度会很慢。所以用object[][] 先

//罗瑶光 20210924

//出现一个问题，我的table db是非线性map 结构， 自带表头key， 而data 是矩阵，下面逻辑要全部改掉

public static void P\_kernel\_search(String temp, File readDBTableRowIndexFile, int rowId, Object[] huaRuiJiJtableCulumns

, Object[][] huaRuiJiJtable,List<Map<String, Object>> output, Row bufferRow, Map<String, Object> rowMap) throws IOException {

Object[] rowList= huaRuiJiJtable[rowId];

for(int i= 0; i< huaRuiJiJtableCulumns.length; i++) {

Cell cell= new Cell();

rowMap.put((String)huaRuiJiJtableCulumns[i], rowList[i]);

cell.I\_CellValue(rowList[i]);

bufferRow.putCell((String)huaRuiJiJtableCulumns[i], cell);

}

output.add(rowMap);

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import AVQ.ASQ.OVQ.OSQ.VSQ.obj.WordFrequency;

import ESU.list.List\_ESU;

import ESU.sort.Quick9DLYGWithString\_ESU;

import ME.APM.VSQ.HRJFrame;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache\_M;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Table;

import OSA.shell.SearchShellTable;

import OSA.shell.SearchShellTables;

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

@SuppressWarnings({ "unused"})

public class P\_ConditionPLSearch\_XCDX\_Map extends P\_ConditionPLSearch\_XCDX {

//以后优化成统一对象输出，不需要再转换。2019-1-15 tin

public static Map<String, Object> rowToRowMap(Row row) {

Map<String, Object> culumnMaps= new HashMap<>();

Map<String, Object> rowMap= new HashMap<>();

Iterator<String> iterator= row.getCells().keySet().iterator();

while(iterator.hasNext()) {

String cellName= iterator.next();

if(!cellName.contains("is\_delete")) {

Cell cell= row.getCell(cellName);

Map<String, Object> culumnMap= new HashMap<>();

culumnMap.put("culumnName", cellName);

culumnMap.put("culumnValue", cell.getCellValue().toString());

culumnMaps.put(cellName, culumnMap);

}

}

rowMap.put("rowValue", culumnMaps);

return rowMap;

}

//将rowToRowMap 进行逆向 RowMapToRow 一来验证，2来找最小计算模型，方便下一步表格编译计算。

//罗瑶光202109302339

@SuppressWarnings("unchecked")

public static Row rowMapToRow(Map<String, Object> map) {

Row row= new Row();

ConcurrentHashMap<String, Cell> cells= new ConcurrentHashMap<>();

row.I\_Cells(cells);

Iterator<String> iterator= ((Map<String, Object>)map.get("rowValue")).keySet().iterator();

while(iterator.hasNext()) {

String cellName= iterator.next();

if(!cellName.contains("is\_delete")) {

Cell cell= new Cell();

Map<String, Object> culumnMap

= (Map<String, Object>)((Map<String, Object>)map.get("rowValue")).get(cellName);

cell.I\_CellValue(culumnMap.get("culumnValue"));

row.putCell(cellName, cell);

}

}

return row;

}

//猫腻哥 把我pmap的output 都改了， 今天一查问题全出来了。20210927

//懒得管，把P\_Map 改成 shellP\_Map

public static void P\_Map(String[] sets, List<Map<String, Object>> output, String tableName

, Map<String, Object> object) {

//算了统一接口， 以后统一优化改。

List<Map<String, Object>> outputTemp= new ArrayList<>();

//创建一个table

SearchShellTable table;

//outputTemp.addAll(output);

if((output.isEmpty()||null= = output)&& object.get("firstTime").equals("true")) {

table= SearchShellTables.searchShellTables.get(tableName);

object.put("firstTime", "others");

}else {

Row[] huaRuiJiJtableRows= new Row[output.size()];

for(int i= 0; i< output.size(); i++) {

huaRuiJiJtableRows[i]= P\_ConditionPLSearch\_XCDX\_Map.rowMapToRow(output.get(i));

}

table= new SearchShellTable();

table.setHuaRuiJiJtableRows(huaRuiJiJtableRows);

}

//修改下把output的逻辑重复利用 满足conditon的and 和or

//只拿前50行 以后改成分页

//稍后把这个函数片段移除这个文件，变成一个函数。

if(sets[1].equalsIgnoreCase("精度搜索")) {

//table to object

//稍后我的养疗经界面搜索release函数的 片段 这里也可以优化如下。

String key= sets[2];

if(null= = key|| key.equals("")) {

return;

}

String[] score= new String[table.huaRuiJiJtableRows.length];

int[] score\_code= new int[table.huaRuiJiJtableRows.length];

int []reg= new int[table.huaRuiJiJtableRows.length];

int count= 0;

Map<String, String> pos= HRJFrame.NE.\_A.getPosCnToCn();

Map<String, WordFrequency> mapSearchWithoutSort= null;

mapSearchWithoutSort= HRJFrame.NE.\_A.parserMixStringByReturnFrequencyMap(key);

//Iterator<String> iteratorForCopy= copy.iterator();

int copyCount= 0;

List<String> list= HRJFrame.NE.\_A.parserMixedString(key);

String[] string= List\_ESU.listToArray(list);

String[] stringReg= new String[key.length()/3];

for(int i= 0; i< stringReg.length; i++) {

stringReg[i]= key.substring(i\*3, (i\*3+ 3)<key.length()?(i\*3+ 3):key.length()-1);

}

Map<String, Row> map= new HashMap<>();

for(int i= 0; i< table.huaRuiJiJtableRows.length; i++) {

//while(iteratorForCopy.hasNext()) {

String temps= table.huaRuiJiJtableRows[i].getCell(sets[0]).getCellValue().toString();

// if(null= = temps) {

// temps= "";

// }

score[copyCount]= "i"+ i;//因为 不再有map key，所以就通用为map 内容。

map.put(score[copyCount], table.huaRuiJiJtableRows[i]);

//String iteratorForCopyString= iteratorForCopy.next();

//score[copyCount]= iteratorForCopyString;

//String temps= dic\_map.get(iteratorForCopyString).toString();

Iterator<String> iteratorWordFrequency= mapSearchWithoutSort.keySet().iterator();

Here:

while(iteratorWordFrequency.hasNext()) {

String mapSearchaAtII= iteratorWordFrequency.next();

WordFrequency wordFrequencySearch= mapSearchWithoutSort.get(mapSearchaAtII);

if(temps.contains(mapSearchaAtII)) {

if(reg[copyCount]= = 0){

count += 1;

}

// score[copyCount]= temps;//因为 不再有map key，所以就通用为map 内容。，还是需要map

// if(score[copyCount].contains(key.replace(" ", ""))) {

// reg[copyCount]+= 500;

// }

// if(key.contains(score[copyCount].replace(" ", ""))) {

// reg[copyCount]+= 500;

// }

if(temps.contains(key.replace(" ", ""))) {

reg[copyCount]+= 500;

}

if(key.contains(temps.replace(" ", ""))) {

reg[copyCount]+= 500;

}

if(!pos.containsKey(mapSearchaAtII)) {

reg[copyCount] += 1;

score\_code[copyCount] += 1 << mapSearchaAtII.length() << wordFrequencySearch.getFrequency() ;

continue Here;

}

if(pos.get(mapSearchaAtII).contains("名")||pos.get(mapSearchaAtII).contains("动")

||pos.get(mapSearchaAtII).contains("形")||pos.get(mapSearchaAtII).contains("谓")) {

reg[copyCount] += 2;

}

reg[copyCount] += 1;

score\_code[copyCount] += (temps.contains(mapSearchaAtII) ? 2 : 1)

\* (!pos.get(mapSearchaAtII).contains("名") ? pos.get(mapSearchaAtII).contains("动")? 45 : 1 : 50)

<< mapSearchaAtII.length() \* wordFrequencySearch.getFrequency();

continue Here;

}

if(mapSearchaAtII.length()>1) {

for(int j= 0;j<mapSearchaAtII.length();j++) {

if(temps.contains(String.valueOf(mapSearchaAtII.charAt(j)))) {

if(reg[copyCount]= = 0){

count += 1;

}

// score[copyCount]= temps;

score\_code[copyCount]+= 1;

if(pos.containsKey(String.valueOf(mapSearchaAtII.charAt(j)))&&(

pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("名")

||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("动")

||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("形")

||pos.get(String.valueOf(mapSearchaAtII.charAt(j))).contains("谓")

)) {

reg[copyCount] += 2;

}

reg[copyCount] += 1;

continue Here;

}

}

}

}

score\_code[copyCount]= score\_code[copyCount] \* reg[copyCount];

//词距

int code= 100;

int tempb= 0;

int tempa= score\_code[copyCount];

if(key.length()> 4) {

//全词

for(int j= 0; j< string.length; j++) {

if(temps.contains(string[j])) {

tempb+= code;

}

}

//断句

for(int j= 0; j< stringReg.length; j++) {

if(temps.contains(stringReg[j])) {

tempb+= code;

}

}

score\_code[copyCount]= (int) (tempa/Math.pow(HRJFrame.NE.lookrot+ 1, 4)

+ tempb\*Math.pow(Integer.valueOf(sets[3]), 2));

}

if(key.replace(" ", "").length()> 1&& key.replace(" ", "").length()< 5) {

if(temps.contains(key.replace(" ", ""))) {

tempb+= code<< 7;

}

score\_code[copyCount]= (int) (tempa/Math.pow(Integer.valueOf(sets[3])+ 1, 4)

+ tempb\*Math.pow(Integer.valueOf(sets[3]), 2));

}

copyCount++;

}

LABEL2:

new Quick9DLYGWithString\_ESU().sort(score\_code, score);

int max= score\_code[0];

Object[][] tableData= new Object[count][18];

int new\_count= 0;

//newTableModel.getDataVector().clear();

//if(null= = key|| key.equals("")) {

// return;

//}

//recordRows 没有 值

//recordRows 有 值

Here:

for(int i= score.length- 1; i> 0; i--) {

if(score\_code[i]< 1){

continue Here;

}

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(map.get(score[i])));

}

output.clear();

output.addAll(outputTemp);

return;

}

int max= 50;

//获取table的row

Here:

for(int i= 0; i< table.huaRuiJiJtableRows.length; i++ ) {

//if(i> max) {

// continue Here;

//}

Row row= table.huaRuiJiJtableRows[i];

if(sets[1].equalsIgnoreCase("<")||sets[1].equalsIgnoreCase("-lt")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

//大家看见没， rowvalue是 db的 Row单例，这里竟然是output的iterator。2019年被动手脚了。

if(new BigDecimal(rowCellFromString).doubleValue() < new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("<= ")||sets[1].equalsIgnoreCase("= <")

||sets[1].equalsIgnoreCase("-lte")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(new BigDecimal(rowCellFromString).doubleValue() <= new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("= = ")||sets[1].equalsIgnoreCase("= ")||sets[1].equalsIgnoreCase("= = = ")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(new BigDecimal(rowCellFromString).doubleValue()= = new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase(">= ")||sets[1].equalsIgnoreCase("= >")

||sets[1].equalsIgnoreCase("-gte")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(new BigDecimal(rowCellFromString).doubleValue() >= new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase(">")||sets[1].equalsIgnoreCase("-gt")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(new BigDecimal(rowCellFromString).doubleValue() > new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("字符串长度大于")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(rowCellFromString.length()> new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("字符串长度小于")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(rowCellFromString.length()< new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("!= ")||sets[1].equalsIgnoreCase("= !")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(new BigDecimal(rowCellFromString).doubleValue() != new BigDecimal(sets[2]).doubleValue()) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("包含")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(rowCellFromString.contains(sets[2])) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("过滤掉")||sets[1].equalsIgnoreCase("不包含")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(!rowCellFromString.contains(sets[2])) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("equal")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(rowCellFromString.equalsIgnoreCase(sets[2])) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("!equal")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

if(!rowCellFromString.equalsIgnoreCase(sets[2])) {

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("in")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

String set= "," + sets[2] + ",";

if(set.contains("," + rowCellFromString + ",")){

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

if(sets[1].equalsIgnoreCase("!in")) {

String rowCellFromString= row.getCell(sets[0]).getCellValue().toString();

String set= "," + sets[2] + ",";

if(!set.contains("," + rowCellFromString + ",")){

outputTemp.add(P\_ConditionPLSearch\_XCDX\_Map.rowToRowMap(row));

}

}

}

output.clear();

output.addAll(outputTemp);

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache\_M;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Table;

@SuppressWarnings({ "unused"})

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

public class P\_ConditionPLSearch\_XCDX {

// public void P\_Cache(String[] sets, List<Map<String, Object>> output

// , String tableName, String baseName, Map<String, Object> object) {}

//

// //以后优化成统一对象输出，不需要再转换。2019-1-15 tin

// Map<String, Object> rowToRowMap(Row row) {}

//

// public void P\_Map(String[] sets, List<Map<String, Object>> output, String dBTablePath) {}

//

// //plsql 引擎函数获取表开始检查 罗瑶光 20210405 //奇怪了 这是一个没有读 缓存的plsql引擎,我准备对比下history

// //object 指令堆栈

// //output 数据行

// public void P\_Table(String[] sets, List<Map<String, Object>> output

// , String DBTablePath, Map<String, Object> object) throws IOException {}

//

// //比较是否有数据取出列表到输出 检验中 罗瑶光 20210405

// //这个走硬盘查询函数来标识下, 在我设计了数据缓存查询启动函数 后就没用过了, 时间点大概在2019年1月后, 我先调通下, 之后朔源.

// //准备验算下20210406 罗瑶光

// public void P\_kernel(String temp, File readDBTableRowIndexCulumnFile, File readDBTableRowIndexFile

// , BufferedReader reader, String DBTableRowIndexPath, List<Map<String, Object>> output, Row bufferRow

// , Map<String, Object> rowMap) throws IOException {}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

@SuppressWarnings({"unused", "unchecked"})

public class P\_GetCulumnsPLSearch {

public static Object getCulumnsMapWithAs(String[] sets, Map<String, Object> row) {

return row.get(sets[2]);

}

public static Object getCulumnsMap(String[] sets, Map<String, Object> row) {

return row.get(sets[0]);

}

public static Object P\_GetCulumnsMap(List<Map<String, Object>> obj, String[] getCulumnsValueArray) {

List<Map<String, Object>> newobj= new ArrayList<Map<String, Object>>();

Iterator<Map<String, Object>> iterator= obj.iterator();

int count= 0;

NextRow:

while(iterator.hasNext()) {

int rowId= count ++;

Map<String, Object> row= iterator.next();

Map<String, Object> newRow= new HashMap<>();

Map<String, Object> rowValue= new HashMap<>();

NextCell:

for(int i= 1; i < getCulumnsValueArray.length; i++) {

String[] sets= getCulumnsValueArray[i].split("\\|");

if(null != sets && ((Map<String, Object>)row.get("rowValue")).containsKey(sets[0])) {

Map<String, Object> cell

= (Map<String, Object>)((Map<String, Object>)row.get("rowValue")).get(sets[0]);

if(1= = sets.length) {

rowValue.put(sets[0], cell);

continue NextCell;

}

if(3= = sets.length && sets[1].equalsIgnoreCase("改名为")) {

cell.put("culumnName", sets[2]);

rowValue.put(sets[2], cell);

continue NextCell;

}

}

}

newRow.put("rowValue", rowValue);

newobj.add(newRow);

}

obj.clear();

return obj.addAll(newobj);

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

@SuppressWarnings({"unused","unchecked"})

public class P\_RelationPLSearch {

public static void P\_AndMap(String[] sets, List<Map<String, Object>> obj

, List<Map<String, Object>> joinObj

, Map<String, Object> object, List<Map<String, Object>> newObj) {

List<Map<String, Object>> newObjTemp= new ArrayList<>();

Iterator<Map<String, Object>> iterator= newObj.iterator();

int count= 0;

while(iterator.hasNext()) {

int objRowId= count++;

Map<String, Object> objRow= iterator.next();

if(objRow.containsKey(sets[0])&&objRow.containsKey(sets[2])) {

if(sets[1].equalsIgnoreCase("= = ") || sets[1].equalsIgnoreCase("= = = ")) {

if(new BigDecimal(objRow.get(sets[0]).toString()).doubleValue()

= = new BigDecimal(objRow.get(sets[2]).toString()).doubleValue()) {

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("!= ") || sets[1].equalsIgnoreCase("= !")

|| sets[1].equalsIgnoreCase("<>") || sets[1].equalsIgnoreCase("><")) {

if(new BigDecimal(objRow.get(sets[0]).toString()).doubleValue()

!= new BigDecimal(objRow.get(sets[2]).toString()).doubleValue()) {

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase(">= ") || sets[1].equalsIgnoreCase("= >")) {

if(new BigDecimal(objRow.get(sets[0]).toString()).doubleValue()

>= new BigDecimal(objRow.get(sets[2]).toString()).doubleValue()) {

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase(">")) {

if(new BigDecimal(objRow.get(sets[0]).toString()).doubleValue()

> new BigDecimal(objRow.get(sets[2]).toString()).doubleValue()) {

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("<")) {

if(new BigDecimal(objRow.get(sets[0]).toString()).doubleValue()

< new BigDecimal(objRow.get(sets[2]).toString()).doubleValue()) {

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("<= ") || sets[1].equalsIgnoreCase("<= ")) {

if(new BigDecimal(objRow.get(sets[0]).toString()).doubleValue()

<= new BigDecimal(objRow.get(sets[2]).toString()).doubleValue()) {

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("equal")) {

if(objRow.get(sets[0]).toString().equals(objRow.get(sets[2]).toString())){

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("!equal") || sets[1].equalsIgnoreCase("equal!")) {

if(!objRow.get(sets[0]).toString().equals(objRow.get(sets[2]).toString())){

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("in")) {

String set= "," + objRow.get(sets[2]).toString() + ",";

if(set.contains(objRow.get(sets[0]).toString())){

newObjTemp.add(objRow);

}

}

if(sets[1].equalsIgnoreCase("!in")) {

String set= "," + objRow.get(sets[2]).toString() + ",";

if(!set.contains(objRow.get(sets[0]).toString())){

newObjTemp.add(objRow);

}

}

}

}

}

public static void P\_OrMap(String[] sets, List<Map<String, Object>> obj

, List<Map<String, Object>> joinObj

, Map<String, Object> object, List<Map<String, Object>> newObj

, Map<String, Boolean> findinNewObj) {

Iterator<Map<String, Object>> iterator= obj.iterator();

int count= 0;

while(iterator.hasNext()) {

int objRowId= count++;

Map<String, Object> objRow= iterator.next();

Map<String, Object> row= (Map<String, Object>) objRow.get("rowValue");

Iterator<Map<String, Object>> iteratorJoin= joinObj.iterator();

int countJoin= 0;

while(iteratorJoin.hasNext()) {

int objJoinRowId= countJoin++;

Map<String, Object> objJoinRow= iteratorJoin.next();

Map<String, Object> joinRow= (Map<String, Object>) objJoinRow.get("rowValue");

Map<String, Object> cell= (Map<String, Object>) row.get(sets[0]);

Map<String, Object> cellJoin= (Map<String, Object>) joinRow.get(sets[2]);

if(sets[1].equalsIgnoreCase("= = ") || sets[1].equalsIgnoreCase("= = = ")) {

if(new BigDecimal(cell.get("culumnValue").toString()).doubleValue()

= = new BigDecimal(cellJoin.get("culumnValue").toString()).doubleValue()) {

if(!findinNewObj.containsKey(objRowId + ":" + objJoinRowId)) {

Map<String, Object> newObjRow= new HashMap<>();

Map<String, Object> newRow= new HashMap<>();

newRow.putAll((Map<? extends String, ? extends Object>) objJoinRow.get("rowValue"));

newRow.putAll((Map<? extends String, ? extends Object>) objRow.get("rowValue"));

newObjRow.put("rowValue", newRow);

newObj.add(newObjRow);

findinNewObj.put(objRowId + ":" + objJoinRowId, true);

}

}

}

if(sets[1].equalsIgnoreCase("equal")) {

if(cell.get("culumnValue").toString().equals(cellJoin.get("culumnValue").toString())) {

if(!findinNewObj.containsKey(objRowId + ":" + objJoinRowId)) {

Map<String, Object> newObjRow= new HashMap<>();

Map<String, Object> newRow= new HashMap<>();

newRow.putAll((Map<? extends String, ? extends Object>) objJoinRow.get("rowValue"));

newRow.putAll((Map<? extends String, ? extends Object>) objRow.get("rowValue"));

newObjRow.put("rowValue", newRow);

newObj.add(newObjRow);

findinNewObj.put(objRowId + ":" + objJoinRowId, true);

}

}

}

}

}

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import java.util.concurrent.CopyOnWriteArrayList;

import OSA.shell.SearchShellQ\_JoinRows\_E;

import OSA.shell.SearchShellQ\_Rows\_E;

import OSA.shell.SearchShellTable;

import OSA.shell.SearchShellTables;

import PEU.P.cache.\*;

@SuppressWarnings("unchecked")

public class PLSearchCommand\_E {

public static void P\_SetRoot(String[] acknowledge, Map<String, Object> output) throws Exception {

String dbPath= acknowledge[1];

for(int i= 2; i<acknowledge.length; i++) {

dbPath += ":" + acknowledge[i];

}

if(null != Cache\_M.getCacheInfo("DBPath")) {

File file= new File(dbPath);

if(!file.exists()) {

file.mkdirs();

Cache c= new Cache();

c.I\_Value(dbPath);

Cache\_M.putCache("DBPath", c);

}else if(file.isFile()) {

throw new Exception();

}else if(file.isDirectory()) {

Cache c= new Cache();

c.I\_Value(dbPath);

Cache\_M.putCache("DBPath", c);

}

}

}

public static void P\_BaseName(String[] acknowledge, Map<String, Object> object) {

object.put(acknowledge[0], acknowledge[1]);

}

public static void P\_TableName(String[] acknowledge, Map<String, Object> object) {

object.put(acknowledge[0], acknowledge[1]);

object.put("type", acknowledge[2]);

}

public static void P\_ListNeedStart(String[] acknowledge, Map<String, Object> object) {

object.put("start", "1");

if(object.containsKey(acknowledge[0])) {

List<String[]> relationValues= (List<String[]>) object.get(acknowledge[0]);

relationValues.add(acknowledge);

object.put(acknowledge[0], relationValues);

return;

}

List<String[]> relationValues= new CopyOnWriteArrayList<>();

relationValues.add(acknowledge);

object.put(acknowledge[0], relationValues);

}

public static void P\_Join(String[] acknowledge, Map<String, Object> object) {

if(object.get("countJoins").toString().equals("1")) {

object.put("countJoins", "n");

}

if(object.get("countJoins").toString().equals("0")) {

object.put("countJoins", "1");

}

object.put("joinBaseName", acknowledge[1]);

object.put("joinTableName", acknowledge[2]);

}

public static void P\_E(String[] acknowledge, Map<String, Object> object, boolean mod) throws Exception {

if(object.get("start").toString().equals("1")) {

if(!acknowledge[0].equalsIgnoreCase(object.get("lastCommand").toString())

&&(object.get("lastCommand").toString().contains("changeCulumnName")

||object.get("lastCommand").toString().contains("culumnValue")

||object.get("lastCommand").toString().contains("条件为")

||object.get("lastCommand").toString().contains("relation")

||object.get("lastCommand").toString().contains("操作")

||object.get("lastCommand").toString().contains("PLETL")

||object.get("lastCommand").toString().contains("PLTCP")

||object.get("lastCommand").toString().contains("获取表列名")

||object.get("lastCommand").toString().contains("culumnName")

||object.get("lastCommand").toString().contains("relation"))) {

P\_E\_Kernel(object, mod);

}

}

}

//处理机中心, 别急, 准备验证 罗瑶光

private static void P\_E\_Kernel(Map<String, Object> object, boolean mod) throws Exception{

if(object.get("type").toString().equalsIgnoreCase("进行选择") &&

(object.get("countJoins").toString().equalsIgnoreCase("0") ||

(object.get("countJoins").toString().equalsIgnoreCase("1") &&

object.get("newCommand").toString().equalsIgnoreCase("join")))){

if(object.containsKey("条件为")) {

object.put("obj", SearchShellQ\_Rows\_E.selectRowsByAttributesOfCondition(object));

}

if(object.containsKey("操作")) {

//plsearch 的筛选 在这里拓展。罗瑶光20210927

object.put("obj", SearchShellQ\_Rows\_E.selectRowsByAttributesOfAggregation(object));

}

if(object.containsKey("获取表列名")) {

object.put("obj", SearchShellQ\_Rows\_E.selectRowsByAttributesOfGetCulumns(object));

}

if(object.containsKey("PLETL")) {

object.put("obj", SearchShellQ\_Rows\_E.selectRowsByAttributesOfPLETL(object));

}

if(object.containsKey("PLTCP")) {

object.put("obj", SearchShellQ\_Rows\_E.selectRowsByAttributesOfPLTCP(object));

}

object.remove("recordRows");

}

if(object.get("type").toString().equalsIgnoreCase("进行选择") &&

(object.get("countJoins").toString().equalsIgnoreCase("n") ||

(object.get("countJoins").toString().equalsIgnoreCase("1") &&

!object.get("newCommand").toString().equalsIgnoreCase("join")))){

if(object.containsKey("condition")) {

object.put("joinObj", SearchShellQ\_JoinRows\_E.selectRowsByAttributesOfJoinCondition(object));

}

if(object.containsKey("relation")) {

object.put("obj", SearchShellQ\_JoinRows\_E.selectRowsByAttributesOfJoinRelation(object));

}

if(object.containsKey("aggregation")) {

//object.put("obj", SearchShellQ\_JoinRows\_E.selectRowsByAttributesOfJoinAggregation(object));

}

if(object.containsKey("getCulumns")) {

object.put("joinObj", SearchShellQ\_JoinRows\_E.selectRowsByAttributesOfJoinGetCulumns(object));

}

if(object.containsKey("PLETL")) {

// object.put("obj", SearchShellQ\_Rows\_E.selectRowsByAttributesOfPLETL(object));

}

object.remove("recordRows");

}

if(object.get("type").toString().equalsIgnoreCase("create")){

if(object.containsKey("culumnName")) {

//I\_Tables\_E.I\_Table(object, mod);

}

object.remove("recordRows");

}

//离散数学的conjuction变换 a^&&b^&&c \* kernel[]= (a^&&b^)^^&&c \* kernel[]= (a||b)^&&c \* kernel[]

if(object.get("type").toString().equalsIgnoreCase("update") &&

(object.get("countJoins").toString().equalsIgnoreCase("0") ||

(object.get("countJoins").toString().equalsIgnoreCase("1") &&

object.get("newCommand").toString().equalsIgnoreCase("join")))){

if(object.containsKey("condition")) {

//object.put("updateObj", U\_Rows\_E.updateRowsByAttributesOfCondition(object, mod));

}

if(object.containsKey("aggregation")) {

//object.put("updateObj", U\_Rows\_E.updateRowsByAttributesOfAggregation(object, mod));

}

if(object.containsKey("culumnValue")) {

//U\_Rows\_E.updateRowsByRecordConditions(object, mod);

}

object.remove("recordRows");

}

if(object.get("type").toString().equalsIgnoreCase("update") &&

(object.get("countJoins").toString().equalsIgnoreCase("n") ||

(object.get("countJoins").toString().equalsIgnoreCase("1") &&

!object.get("newCommand").toString().equalsIgnoreCase("join")))){

if(object.containsKey("condition")) {

//object.put("updateJoinObj", U\_JoinRows\_E.updateRowsByAttributesOfJoinCondition(object, mod));

}

if(object.containsKey("relation")) {

//object.put("updateObj", U\_JoinRows\_E.updateRowsByAttributesOfJoinRelation(object, mod));

}

if(object.containsKey("aggregation")) {

//object.put("updateObj", U\_JoinRows\_E.updateRowsByAttributesOfJoinAggregation(object, mod));

}

if(object.containsKey("culumnValue")) {

//U\_Rows\_E.updateRowsByRecordConditions(object, mod);

}

object.remove("recordRows");

}

if(object.get("type").toString().equalsIgnoreCase("insert")) {

if(object.containsKey("culumnValue")) {

//IU\_Rows\_E.IU\_RowByAttributes(object, mod);

}

}

if(object.get("type").toString().equalsIgnoreCase("delete")) {

if(object.containsKey("condition")) {

//D\_Rows\_E.D\_RowByAttributesOfCondition(object, mod);

}

}

object.remove("条件为");

object.remove("culumnName");

object.remove("changeCulumnName");

object.remove("relation");

object.remove("操作");

object.remove("获取表列名");

object.remove("PLETL");

object.remove("PLTCP");

object.put("start", "0");

}

//plsql函数执行指令 正在检查中 罗瑶光 20210405

public static void P\_Check(String acknowledge, Map<String, Object> object, boolean mod) throws Exception {

if(object.get("start").toString().equals("1")) {

P\_E\_Kernel(object, mod);

}

List<Map<String, Object>> obj= ((List<Map<String, Object>>)(object.get("obj")));

int totalPages= 0;

if(obj != null) {

totalPages= obj.size();

}

int rowBeginIndex= object.containsKey("pageBegin")? Integer.valueOf(object.get("pageBegin").toString()):0;

int rowEndIndex= object.containsKey("pageEnd")?Integer.valueOf(object.get("pageEnd").toString()):totalPages>15?15:totalPages;

object.put("pageBegin", rowBeginIndex);

object.put("pageEnd", rowEndIndex);

// String DBPath= Cache\_M.getCacheInfo("DBPath").getValue().toString() + "/" + object.get("baseName").toString();

// String DBTablePath= DBPath + "/" + object.get("tableName").toString();

object.put("tablePath", object.get("获取表名").toString());

object.put("returnResult", "success");

object.put("totalPages", totalPages);

object.put("loginInfo", "success");

List<Object> spec= new ArrayList<>();

// Iterator<String> iterator= new ArrayList<String>().iterator();

// if(obj= = null || obj.size()< 1) {

////Base base= DetaDBBufferCache\_M.db.getBase(object.get("baseName").toString());

//SearchShellTable table= SearchShellTables.searchShellTables.get(object.get("tableName").toString());

//Object[] specs= table.getHuaRuiJiJtableCulumns();

// }else {//进行map 验证检测 罗瑶光 20210405

//Map<String, Object> map= obj.get(0);

//Map<String, Object> objectInMap= (Map<String, Object>)map.get("rowValue");

//iterator= null= = objectInMap? null:objectInMap.keySet().iterator();

// }

SearchShellTable table= SearchShellTables.searchShellTables.get(object.get("获取表名").toString());

Object[] specs= table.getHuaRuiJiJtableCulumns();

for(Object specS: specs) {

spec.add(specS);

}

object.put("spec", spec);

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSM.shell;

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.Map;

import OEU.LYG4DQS4D.LYG10DWCMSSort13D\_XCDX\_C\_A\_S;

import PEU.P.table.TableSorterZYNK;

import PEU.S.verbal.VerbalSource;

import SVQ.stable.StableFile;

public class SortStringDemo{

public static Map<String, String> pinYin= null;

public static Map<String, Integer> biHua= null;

public static void initMap() {

try {

if(null!= pinYin|| null!= biHua) {

return;

}

InputStream inputStreamp= new VerbalSource().getClass().getResourceAsStream(StableFile.PinYinCN\_lyg);

BufferedReader cReaderp= new BufferedReader(new InputStreamReader(inputStreamp, "GBK"));

//index

String cInputStringp;

Map<String, String> map= new HashMap<>();

biHua= new HashMap<>();

while ((cInputStringp= cReaderp.readLine())!= null) {

String[] words= cInputStringp.split("->");

if(words.length>1) {

map.put(words[0], words[1]);

}

}

cReaderp.close();

InputStream inputStreamb= new VerbalSource().getClass().getResourceAsStream(StableFile.BiHuaCN\_lyg);

BufferedReader cReaderb= new BufferedReader(new InputStreamReader(inputStreamb, "GBK"));

//index

String cInputStringb;

while ((cInputStringb= cReaderb.readLine())!= null) {

String[] words= cInputStringb.split("->");

if(words.length>1) {

biHua.put(words[0], Integer.valueOf(words[1]));

}

}

pinYin= map;

cReaderb.close();

}catch(Exception e) {

}

}

@SuppressWarnings("unused")

public static void main(String[] argv) {

initMap();

TableSorterZYNK tableSorterZYNK= new TableSorterZYNK();

String[] strings= new String[10];

strings[0]= "luoy罗瑶光uang";

strings[1]= "罗瑶光";

strings[2]= "瑶光";

strings[3]= "罗瑶";

strings[4]= "yaoguang";

strings[5]= "y瑶光g";

strings[6]= "yaog瑶光ng";

strings[7]= "y瑶光guang";

strings[8]= "ya罗瑶光ang";

strings[9]= "yaoguang";

int returnInt= new LYG10DWCMSSort13D\_XCDX\_C\_A\_S()

.quick4DChineseStringArrayWithSmallInTwoChar3bihuaReturns(strings

, 0, 9, 30, pinYin, biHua, 7, 70);

for(String string:strings){

System.out.println(string);

}

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSA.shell;

import java.util.Map;

//稍后将DMA文件与内存操作替换成 jtable表内存操作 罗瑶光

public interface PL\_XA\_C{

public String getPLSearch();

public void I\_PLSearch(String pLSearch);

public PL\_XA\_C withTableCreate(String tableName);

public PL\_XA\_C withTableDelete(String tableName);

public PL\_XA\_C withTableInsert(String tableName);

public PL\_XA\_C withTableUpdate(String tableName);

public PL\_XA\_C withTableSelect(String tableName);

public PL\_XA\_C getCulumns();

public PL\_XA\_C startAtRootDir(String rootAddress);

public PL\_XA\_C withBaseName(String baseName);

public PL\_XA\_C withCondition(String conditionType);

public PL\_XA\_C let(String leftSet);

public PL\_XA\_C lessThanAndEqualTo(String compareSet);

public PL\_XA\_C equalTo(String compareSet);

public PL\_XA\_C lessThan(String compareSet);

public PL\_XA\_C greatThan(String compareSet);

public PL\_XA\_C greatThanAndEqualTo(String compareSet);

public PL\_XA\_C notEqualTo(String compareSet) ;

public PL\_XA\_C in(String compareSet) ;

public PL\_XA\_C notIn(String compareSet) ;

public PL\_XA\_C equals(String compareSet);

public PL\_XA\_C notEquals(String compareSet);

public PL\_XA\_C innerJoinWithTable(String baseName, String tableName);

public PL\_XA\_C withRelation(String relationType) ;

public PL\_XA\_C as(String compareSet) ;

public PL\_XA\_C upTo(String compareSet);

public PL\_XA\_C withAggregation(String aggregationType);

public PL\_XA\_C changeCulumnName(String newCulumnName, String oldCulumnName);

public PL\_XA\_C withCulumnName(String culumnName, String dataType);

public PL\_XA\_C withCulumnValue(String culumnName, String culumnValue);

public PL\_XA\_C checkErrors(String string);

public PL\_XA\_C fixErrors(String string);

public PL\_XA\_C finalE(boolean b) throws Exception;

public Map<String, Object> returnAsMap();

public PL\_XA\_C checkAndFixPlSearchGrammarErrors();

public PL\_XA\_C checkAndFixSystemEnvironmentErrors();

public PL\_XA\_C withTableDrop(String tabKey);

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSA.shell;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import javax.swing.table.DefaultTableModel;

import ME.APM.VSQ.App;

import OSI.AOP.MEC.SIQ.plorm.Const;

import OSM.shell.E\_PLSearch\_E;

//1 我的逻辑很简单， 仅仅按照PLORM 进行 PLSearch ，将deta数据库操作 用在

//养疗经的界面表操作上。先不设计改和写操作。

//2 用XA元基 来代替search 词汇。

// 准备用DefaultTableModel 来做输出对象， 因为养疗经的内存表是这个容器。

// 罗瑶光

@SuppressWarnings("unused")

public class PL\_XA\_E implements PL\_XA\_C{

private DefaultTableModel defaultTableModel;

private Object[][] tableData\_old;

private App app;

private String PLSearch= "";

private String[] PLSearchArray;

private Map<String, Object> map;

public String getPLSearch() {

return PLSearch;

}

public void I\_PLSearch(String pLSearch) {

PLSearch= pLSearch;

}

public PL\_XA\_E startAtRootDir(String rootAddress) {

PLSearch= Const.SET\_ROOT+ Const.COLON+ rootAddress

+ Const.SEMICOLON;

return this;

}

public PL\_XA\_E withBaseName(String baseName) {

PLSearch+= Const.SEMICOLON+ Const.BASE\_NAME+ Const.COLON

+ baseName;

return this;

}

//

public PL\_XA\_E withTableSelect (String tableName) {

PLSearch+= Const.SEMICOLON+ Const.TABLE\_NAME+ Const.COLON

+ tableName

+ Const.COLON+ Const.SELECT;

return this;

}

public PL\_XA\_E withTableCreate(String tableName) {

PLSearch+= Const.SEMICOLON+ Const.TABLE\_NAME+ Const.COLON

+ tableName

+ Const.COLON+ Const.CREATE;

return this;

}

public PL\_XA\_C withTableDrop(String tableName) {

PLSearch+= Const.SEMICOLON+ Const.TABLE\_NAME+ Const.COLON

+ tableName

+ Const.COLON+ Const.DROP;

return this;

}

public PL\_XA\_E withTableDelete(String tableName) {

PLSearch+= Const.SEMICOLON+ Const.TABLE\_NAME+ Const.COLON

+ tableName

+ Const.COLON+ Const.DELETE;

return this;

}

public PL\_XA\_E withTableInsert(String tableName) {

PLSearch+= Const.SEMICOLON+ Const.TABLE\_NAME+ Const.COLON

+ tableName

+ Const.COLON+ Const.INSERT;

return this;

}

public PL\_XA\_E withTableUpdate(String tableName) {

PLSearch+= Const.SEMICOLON+ Const.TABLE\_NAME+ Const.COLON

+ tableName

+ Const.COLON+ Const.UPDATE;

return this;

}

public PL\_XA\_E withCondition(String conditionType) {

PLSearch+= Const.SEMICOLON+ Const.CONDITION+ Const.COLON

+ conditionType;

return this;

}

public PL\_XA\_E let(String leftSet) {

PLSearch+= Const.COLON+ leftSet;

return this;

}

public PL\_XA\_E lessThanAndEqualTo(String compareSet) {

PLSearch+= Const.LESS\_THAN\_AND\_EQUAL\_TO+ compareSet;

return this;

}

public PL\_XA\_E equalTo(String compareSet) {

PLSearch+= Const.EQUAL\_TO+ compareSet;

return this;

}

public PL\_XA\_E lessThan(String compareSet) {

PLSearch+= Const.LESS\_THAN+ compareSet;

return this;

}

public PL\_XA\_E greatThan(String compareSet) {

PLSearch+= Const.GREAT\_THAN+ compareSet;

return this;

}

public PL\_XA\_E greatThanAndEqualTo(String compareSet) {

PLSearch+= Const.GREAT\_THAN\_AND\_EQUAL\_TO+ compareSet;

return this;

}

public PL\_XA\_E notEqualTo(String compareSet) {

PLSearch+= Const.NOT\_EQUAL\_TO+ compareSet;

return this;

}

public PL\_XA\_E in(String compareSet) {

PLSearch+= Const.IN+ compareSet;

return this;

}

public PL\_XA\_E notIn(String compareSet) {

PLSearch+= Const.NOT\_IN+ compareSet;

return this;

}

public PL\_XA\_E equals(String compareSet) {

PLSearch+= Const.EQUALS+ compareSet;

return this;

}

public PL\_XA\_E notEquals(String compareSet) {

PLSearch+= Const.NOT\_EQUALS+ compareSet;

return this;

}

public PL\_XA\_E innerJoinWithTable(String baseName, String tableName) {

PLSearch+= Const.SEMICOLON+ Const.JOIN+ Const.COLON+ baseName

+ Const.COLON+ tableName;

return this;

}

public PL\_XA\_E withRelation(String relationType) {

PLSearch+= Const.SEMICOLON+ Const.RELATION+ Const.COLON

+ relationType;

return this;

}

public PL\_XA\_E as(String compareSet) {

PLSearch+= Const.AS+ compareSet;

return this;

}

public PL\_XA\_E upTo(String compareSet) {

PLSearch+= Const.UP\_TO+ compareSet;

return this;

}

public PL\_XA\_E withAggregation(String aggregationType) {

PLSearch+= Const.SEMICOLON+ Const.WITH\_AGGREGATION

+ Const.COLON+ aggregationType;

return this;

}

public PL\_XA\_E getCulumns() {

PLSearch+= Const.SEMICOLON+ Const.GET\_CULUMNS;

return this;

}

public PL\_XA\_E changeCulumnName(String newCulumnName, String oldCulumnName) {

PLSearch+= Const.SEMICOLON+ Const.CHANGES\_CULUMN\_NAME+ Const.COLON

+ newCulumnName+ Const.COLON+ oldCulumnName;

return this;

}

public PL\_XA\_E withCulumnName(String culumnName, String dataType) {

PLSearch+= Const.SEMICOLON+ Const.CULUMN\_NAME+ Const.COLON+ culumnName

+ Const.COLON+ dataType;

return this;

}

public PL\_XA\_E withCulumnValue(String culumnName, String culumnValue) {

PLSearch+= Const.SEMICOLON+ Const.CULUMN\_VALUE+ Const.COLON+ culumnName

+ Const.COLON+ culumnValue;

return this;

}

public PL\_XA\_C exec(boolean b) throws Exception {

//map= E\_PLSearch\_E.E\_PLORM(this, true);

return this;

}

@Override

public PL\_XA\_C checkErrors(String string) {

return this;

}

@Override

public PL\_XA\_C fixErrors(String string) {

return this;

}

@Override

public PL\_XA\_C finalE(boolean b) throws Exception {

map= E\_PLSearch\_E.E\_PLSearch(this, true, new ConcurrentHashMap<>());

//这里需要 把数据库的 编译机器也重设计成执行内存操作的模式。

return this;

}

@Override

public Map<String, Object> returnAsMap() {

return this.map;

}

@Override

public PL\_XA\_C checkAndFixPlSearchGrammarErrors() {

//string to array

this.PLSearchArray= PLSearch.split(Const.SEMICOLON);

//条件检查 1 过滤 2 修改 3 语义检测

//1

for(int i= 1; i< PLSearchArray.length; i++) {

//1.1 过滤相同句型

//1.2 过滤无效字符

//1.3 过滤攻击代码

if(PLSearchArray[i].equalsIgnoreCase(PLSearchArray[i- 1])) {

PLSearchArray[i]= "";

}

PLSearchArray[i]= PLSearchArray[i].replaceAll(">+", ">");

PLSearchArray[i]= PLSearchArray[i].replaceAll("<+", "<");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\!+", "!");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\~+", "~");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\@+", "@");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\&&+", "&&");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\|\\|+", "||");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\[+", "[");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\]+", "]");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\:+", ":");

PLSearchArray[i]= PLSearchArray[i].replaceAll("\\s+", "");

}

//2

//2.1 修改错误比较符号

//2.2 修改错误语法关键字

//2.3 修改错误标注符号

//3

//3.1 检测是否有关键字前后句段混乱

//3.2 检测是否有关键字 格式 倒置

//3.3 检测是否有关键字 句型 倒置

//rerturn

String string= "";

for(int i= 0; i< PLSearchArray.length; i++) {

string+= PLSearchArray[i]+ Const.SEMICOLON;

}

PLSearch= string;

return this;

}

@Override

public PL\_XA\_C checkAndFixSystemEnvironmentErrors() {

return this;

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSA.shell;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import MS.OP.SM.AOP.MEC.SIQ.cache.DetaDBBufferCache\_M;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Spec;

import OSM.shell.P\_AggregationPLSearch;

import OSM.shell.P\_ConditionPLSearch\_XCDX\_Cache;

import OSM.shell.P\_ConditionPLSearch\_XCDX\_Map;

import OSM.shell.P\_GetCulumnsPLSearch;

import OSM.shell.P\_RelationPLSearch;

//整体 plsql 替换成 plsearch，稍后测试验证 罗瑶光20210927

@SuppressWarnings({"unused", "unchecked"})

public class SearchShellQ\_JoinRows\_E {

public static Object selectRowsByAttributesOfJoinCondition(Map<String, Object> object)

throws IOException {

if(!object.containsKey("recordRows")) {

Map<String, Boolean> recordRows= new ConcurrentHashMap<>();

object.put("recordRows", recordRows);

}

List<Map<String, Object>> output= new ArrayList<>();

List<String[]> conditionValues= (List<String[]>) object.get("condition");

Iterator<String[]> iterator= conditionValues.iterator();

while(iterator.hasNext()) {

boolean overMap= output.size()= = 0? false: true;

String[] conditionValueArray= iterator.next();

String type= conditionValueArray[1];

boolean andMap= type.equalsIgnoreCase("and")?true:false;

for(int i= 2; i< conditionValueArray.length; i++) {

String[] sets= conditionValueArray[i].split("\\|");

if(overMap&& andMap) {

P\_ConditionPLSearch\_XCDX\_Map.P\_Map(sets, output, object.get("joinBaseName").toString(), object);//1

}else {

P\_ConditionPLSearch\_XCDX\_Cache.P\_Cache(sets, output

, object.get("joinTableName").toString()

, object, type);//1

}//SHELL 无 DMA

}

}

return output;

}

public static Object selectRowsByAttributesOfJoinAggregation(Map<String, Object> object) throws InstantiationException, IllegalAccessException, IOException {

if(!object.containsKey("joinObj")) {

return new ArrayList<>();

}

List<Map<String, Object>> obj= ((List<Map<String, Object>>)(object.get("obj")));

List<String[]> aggregationValues= (List<String[]>) object.get("aggregation");

Iterator<String[]> iterator= aggregationValues.iterator();

while(iterator.hasNext()) {

boolean overMap= obj.size()= = 0? false: true;

String[] aggregationValueArray= iterator.next();

String type= aggregationValueArray[1];

boolean limitMap= type.equalsIgnoreCase("limit")?true:false;

for(int i= 2; i < aggregationValueArray.length; i++) {

String[] sets= aggregationValueArray[i].split("\\|");

if(limitMap) {

P\_AggregationPLSearch.P\_AggregationLimitMap(sets, obj);

}

//基于sort key 前序treeMap 之后排序功能设计

//基于sort key 后序treeMap

}

}

return obj;

}

public static Object selectRowsByAttributesOfJoinGetCulumns(Map<String, Object> object) {

if(!object.containsKey("joinObj")) {

return new ArrayList<>();

}

List<Map<String, Object>> obj= ((List<Map<String, Object>>)(object.get("joinObj")));

List<String[]> getCulumnsValues= (List<String[]>) object.get("getCulumns");

Iterator<String[]> iterator= getCulumnsValues.iterator();

while(iterator.hasNext()) {

boolean overMap= obj.size()= = 0? false: true;

String[] getCulumnsValueArray= iterator.next();

if(overMap) {

P\_GetCulumnsPLSearch.P\_GetCulumnsMap(obj, getCulumnsValueArray);

}

}

return obj;

}

public static Object selectRowsByAttributesOfJoinRelation(Map<String, Object> object) {

if(!object.containsKey("obj")||!object.containsKey("joinObj")) {

return new ArrayList<>();

}

Map<String,Boolean> findinNewObj= new HashMap<>();

List<Map<String, Object>> newObj= new ArrayList<Map<String, Object>>();

List<Map<String, Object>> obj= ((List<Map<String, Object>>)(object.get("obj")));

List<Map<String, Object>> joinObj= ((List<Map<String, Object>>)(object.get("joinObj")));

List<String[]> relationValues= (List<String[]>) object.get("relation");

Iterator<String[]> iterator= relationValues.iterator();

while(iterator.hasNext()) {

boolean overObjMap= obj.size()= = 0? false: true;

boolean overJoinObjMap= joinObj.size()= = 0? false: true;

String[] getRelationValueArray= iterator.next();

String type= getRelationValueArray[1];

boolean andMap= type.equalsIgnoreCase("and")?true:false;

for(int i= 2; i< getRelationValueArray.length; i++) {

String[] sets= getRelationValueArray[i].split("\\|");

if(overObjMap&& overJoinObjMap&&andMap && i>2) {

P\_RelationPLSearch.P\_AndMap(sets, obj, joinObj,object, newObj);

}else {

P\_RelationPLSearch.P\_OrMap(sets, obj, joinObj, object

, newObj, findinNewObj);

}

}

}

return newObj;

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSA.shell;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;

//还是要变成map，不然 命令的 key值查询 只能forloop， 效率减低

//罗瑶光

public class SearchShellTable{

public Object[] getHuaRuiJiJtableCulumns() {

return huaRuiJiJtableCulumns;

}

public void setHuaRuiJiJtableCulumns(Object[] huaRuiJiJtableCulumns) {

this.huaRuiJiJtableCulumns= huaRuiJiJtableCulumns;

}

public Object[][] getHuaRuiJiJtable() {

return huaRuiJiJtable;

}

public void setHuaRuiJiJtable(Object[][] huaRuiJiJtable) {

this.huaRuiJiJtable= huaRuiJiJtable;

}

public Object getHuaRuiJiJtableName() {

return huaRuiJiJtableName;

}

public void setHuaRuiJiJtableName(Object huaRuiJiJtableName) {

this.huaRuiJiJtableName= huaRuiJiJtableName;

}

public Row[] getHuaRuiJiJtableRows() {

return huaRuiJiJtableRows;

}

public void setHuaRuiJiJtableRows(Row[] huaRuiJiJtableRows) {

this.huaRuiJiJtableRows= huaRuiJiJtableRows;

}

public Object[] huaRuiJiJtableCulumns;

public Object[][] huaRuiJiJtable;

public Row[] huaRuiJiJtableRows;

public Object huaRuiJiJtableName;

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSA.shell;

import java.util.HashMap;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import ME.APM.VSQ.App;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Cell;

import OP.SM.AOP.MEC.SIQ.SM.reflection.Row;

//这个函数我将 txt 和 db 表文件 到jtable 进行映射成object[][] 表。jtable 太上层，我准备设计成VPCS 结构，把jtable的 object[][]标记脱离出来，

//映射函数框架完成， 稍后开始应用， 先在 控制台做个shell命令输入框，然后开始修改 之前plsql orm 对应的 searchShell 计算文件

//罗瑶光

public class SearchShellTables{

public static Map<String, SearchShellTable> searchShellTables= new HashMap<>();

//先把接口做足，罗瑶光20210925

//将表变成表映射

public static boolean addNewSearchShellTable(String tableName, JTable jtable) {

//..

TableModel defaultTableModel= jtable.getModel();

addNewSearchShellTable(tableName, (DefaultTableModel)defaultTableModel);

return true;

}

//按sonar方式重复函数分离

public static Object[] getNewSearchShellTableSpecFromDefaultTableModel(DefaultTableModel defaultTableModel) {

//..

Object[] jtableSpec= new Object[defaultTableModel.getColumnCount()];

for(int i= 0; i< defaultTableModel.getColumnCount(); i++) {

jtableSpec[i]= defaultTableModel.getColumnName(i);

}

return jtableSpec;

}

//按sonar方式重复函数分离

public static Object[][] getNewSearchShellTableDataFromDefaultTableModel(DefaultTableModel defaultTableModel) {

//..

Object[][] jtableData= new Object[defaultTableModel.getColumnCount()][defaultTableModel.getRowCount()];

for(int i= 0; i< defaultTableModel.getColumnCount(); i++) {

for(int j= 0; j< defaultTableModel.getRowCount(); j++) {

jtableData[i][j]= defaultTableModel.getValueAt(i, j);

}

}

return jtableData;

}

//设计个row的结合表map内存结构 用于shell的表头搜索。

public static Row[] getNewSearchShellTableRowsFromDefaultTableModel(Object[] spec, DefaultTableModel defaultTableModel) {

//..

Row[] rows= new Row[defaultTableModel.getRowCount()];

for(int i= 0; i< defaultTableModel.getColumnCount(); i++) {

rows[i]= new Row();

for(int j= 0; j< defaultTableModel.getRowCount(); j++) {

Cell cell= new Cell();

cell.I\_CellValue(defaultTableModel.getValueAt(i, j));

rows[i].putCell(""+ spec[i], cell);

}

}

return rows;

}

//设计个row的结合表map内存结构 用于shell的表头搜索。

public static Row[] getNewSearchShellTableRowsFromDefaultTableModel(Object[] spec, Object[][] tableData) {

//..

Row[] rows= new Row[tableData.length];

for(int i= 0; i< tableData.length; i++) {

rows[i]= new Row();

rows[i].I\_Cells(new ConcurrentHashMap<String, Cell>());//init

for(int j= 0; j< tableData[0].length; j++) {

Cell cell= new Cell();

cell.I\_CellValue(tableData[i][j]);

rows[i].putCell(""+ spec[j], cell);

}

}

return rows;

}

//将表映射变成shell映射,接口VPCS 多样化，稍后做新陈代谢用。

public static boolean addNewSearchShellTable(String tableName, DefaultTableModel defaultTableModel) {

//..

Object[] jtableSpec= getNewSearchShellTableSpecFromDefaultTableModel(defaultTableModel);

Object[][] jtableData= getNewSearchShellTableDataFromDefaultTableModel(defaultTableModel);

addNewSearchShellTable(tableName, jtableData, jtableSpec);

return true;

}

//将表映射变成shell映射,接口VPCS 多样化，稍后做新陈代谢用。 object data 稍后准备 用 \_S\_ 元基替换。

public static boolean addNewSearchShellTableWithObjectData(String tableName, DefaultTableModel defaultTableModel

, Object[][] defaultTableData) {

//..

Object[] jtableSpec= getNewSearchShellTableSpecFromDefaultTableModel(defaultTableModel);

addNewSearchShellTable(tableName, defaultTableData, jtableSpec);

return true;

}

//shell映射封装

public static boolean addNewSearchShellTable(String tableName, Object[][] defaultTableData

, Object[] defaultTableDataSpec) {

//..

SearchShellTable searchShellTable= new SearchShellTable();

searchShellTable.setHuaRuiJiJtableCulumns(defaultTableDataSpec);

searchShellTable.setHuaRuiJiJtableRows(getNewSearchShellTableRowsFromDefaultTableModel(defaultTableDataSpec, defaultTableData));

searchShellTable.setHuaRuiJiJtable(defaultTableData);

searchShellTable.setHuaRuiJiJtableName(tableName);

searchShellTables.put(tableName, searchShellTable);

return true;

}

//Reflection map 表头方式存储

//将表映射变成shell映射,接口VPCS 多样化，稍后做新陈代谢用。 object data 稍后准备 用 \_S\_ 元基替换。

public static boolean addNewSearchShellTableWithObjectDataReflectionDBRows(String tableName, DefaultTableModel defaultTableModel

, Object[][] defaultTableData) {

//..

Object[] jtableSpec= getNewSearchShellTableSpecFromDefaultTableModel(defaultTableModel);

//defaultTableModel TO ROWS

Row[] rows= getNewSearchShellTableRowsFromDefaultTableModel(jtableSpec, defaultTableModel);

addNewSearchShellTableReflectionDBRows(tableName, rows, jtableSpec);

return true;

}

//Reflection map 表头方式存储

//shell映射封装

public static boolean addNewSearchShellTableReflectionDBRows(String tableName, Row[] rows

, Object[] defaultTableDataSpec ) {

//..

SearchShellTable searchShellTable= new SearchShellTable();

searchShellTable.setHuaRuiJiJtableCulumns(defaultTableDataSpec);

searchShellTable.setHuaRuiJiJtableRows(rows);

searchShellTable.setHuaRuiJiJtableName(tableName);

searchShellTables.put(tableName, searchShellTable);

return true;

}

//然后所有养疗经的jtable表全部基于这个文件函数 进行shell 封装。与数据库的内存映射分离。

public static boolean addInitSearchShellTable(App app) {

//把养疗经的表都在这里初始化映射成 SearchShellTables 内存先。

addNewSearchShellTable("西医内科", app.xynkPage.tableData\_old, app.xynkPage.columnTitle);

addNewSearchShellTable("中医方剂", app.zynkxPage.tableData\_old, app.zynkxPage.columnTitle);

addNewSearchShellTable("中医诊断", app.zyzdxPage.tableData\_old, app.zyzdxPage.columnTitle);

addNewSearchShellTable("古籍经典", app.fyydPage.tableData\_old, app.fyydPage.columnTitle);

addNewSearchShellTable("中医生殖", app.fqzPage.tableData\_old, app.fqzPage.columnTitle);

addNewSearchShellTable("妇产科学", app.fckxPage.tableData\_old, app.fckxPage.columnTitle);

addNewSearchShellTable("急诊科学", app.jzkxPage.tableData\_old, app.jzkxPage.columnTitle);

addNewSearchShellTable("西医外科", app.wkxPage.tableData\_old, app.wkxPage.columnTitle);

addNewSearchShellTable("中医外伤", app.wskxPage.tableData\_old, app.wskxPage.columnTitle);

addNewSearchShellTable("西药手册", app.xyscPage.tableData\_old, app.xyscPage.columnTitle);

addNewSearchShellTable("中药同源", app.tableData\_old, app.columnTitle);

addNewSearchShellTable("哈里森", app.cecil.tableData\_old, app.cecil.columnTitle);

//上面是主页面，

//节点添加导入的数据表页面 我稍后也会做个 扩充函数。

return true;

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package ME.APM.VSQ.OPE.config;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

//import java.beans.Beans;

//import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import javax.swing.JCheckBox;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextPane;

import ME.APM.VSQ.App;

import ME.APM.VSQ.HRJFrame;

import OSA.shell.SearchShellTables;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.AddTinShellView;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.TinMap;

import OSM.shell.E\_PLSearch\_E;

public class ShellJPanel extends JPanel implements MouseListener, KeyListener, ActionListener{

/\*\*

\* 稍后进行优化成 申请版权的格式。

\* 罗瑶光

\*/

private static final long serialVersionUID= 1L;

public JCheckBox jlabel\_box[];

public boolean[] tabNamesHook= new boolean[30];

public boolean isConfig= true;

public JTextPane jTextPane;

public JTextPane outputjTextPane;

public String plsearch;

public Map<String, Object> output;//准备做文章流计算的内存 罗瑶光20211008

@SuppressWarnings("unused")

private App appInThisClass;

@SuppressWarnings("unused")

private JCheckBox jlabel\_peizhi\_di2515;

public ShellJPanel(App app, AddTinShellView sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ

, TinMap topOutput, TinMap midOutput, TinMap downOutput){

appInThisClass= app;

jlabel\_box= new JCheckBox[30];

this.setLayout(null);

this.setBounds(0, 0, 800, 600);

this.setBackground(Color.BLACK);

//copy tab

MVQ.button.DetaButton jlabel\_button= new MVQ.button.DetaButton("清空命令");

jlabel\_button.setBounds(10, 20, 100, 30);

jlabel\_button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// tabNamesHook[0]= true;

// isConfig= false;

jTextPane.setText("");

jTextPane.updateUI();

app.jTabbedpane.validate();

app.validate();

}

});

this.add(jlabel\_button);

MVQ.button.DetaButton jlabel\_button\_clear= new MVQ.button.DetaButton("清空输出");

jlabel\_button\_clear.setBounds(10+1\*(100+30), 20, 100, 30);

jlabel\_button\_clear.addActionListener(new ActionListener() {

@SuppressWarnings("unchecked")

public void actionPerformed(ActionEvent e) {

// tabNamesHook[0]= true;

// isConfig= false;

//清空的时候避免output 重叠计算

//sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.remove("TinShellETL");

//if(null!= topOutput) {

// 将原来的

//outputout

//tinsheletl midshell downshell

// 结构改为

//outputout

//tinsheletl

//midshell downshell

try {

if(null!= topOutput) {

sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut= topOutput.clone();

}

Map<String, Object> map

= (Map<String, Object>)sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.get("TinShellETL");

if(null!= midOutput&& null!= map) {

map.put("midShell", midOutput.clone());

}

if(null!= downOutput&& null!= map) {

map.put("downShell", downOutput.clone());

}

if(null!= map) {

sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put("TinShellETL", map);

}

} catch (CloneNotSupportedException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

outputjTextPane.setText("\"正在使用 养疗经 1.8.8.8.0 Tin Shell系统(8.8.8.0) . . .\"");

outputjTextPane.updateUI();

// app.jTabbedpane.validate();

app.validate();

}

});

this.add(jlabel\_button\_clear);

MVQ.button.DetaButton jlabel\_init\_button= new MVQ.button.DetaButton("初始脚本");

jlabel\_init\_button.setBounds(10+2\*(100+30), 20, 100, 30);

jlabel\_init\_button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(null!= app) {

app.jTabbedpane.validate();

app.validate();

//检测脚本综合分类、

//分类执行脚本编译机

//

if(null!= HRJFrame.NE) {

SearchShellTables.addInitSearchShellTable(HRJFrame.NE);

}

outputjTextPane.setText("已经初始脚本数据。。");

outputjTextPane.updateUI();

app.jTabbedpane.validate();

app.validate();

}

}

});

this.add(jlabel\_init\_button);

MVQ.button.DetaButton jlabel\_debug\_button= new MVQ.button.DetaButton("调试脚本");

jlabel\_debug\_button.setBounds(10+3\*(100+30), 20, 100, 30);

jlabel\_debug\_button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(null!= app) {

app.jTabbedpane.validate();

app.validate();

//检测脚本综合分类、

//分类执行脚本编译机

}

}

});

this.add(jlabel\_debug\_button);

MVQ.button.DetaButton jlabel\_flush\_button= new MVQ.button.DetaButton("执行脚本");

jlabel\_flush\_button.setBounds(10+ 4\*(100+ 30), 20, 100, 30);

jlabel\_flush\_button.addActionListener(new ActionListener() {

@SuppressWarnings("unchecked")

public void actionPerformed(ActionEvent e) {

if(null!= app) {

app.jTabbedpane.validate();

app.validate();

//检测脚本综合分类、

//分类执行脚本编译机

//

//执行shell

String plSearch= jTextPane.getText();

try {

if(!sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.containsKey("TinShellETL")) {

sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put("TinShellETL", new ConcurrentHashMap<String, Object>());

}

output= E\_PLSearch\_E.E\_PLSearch(plSearch.replace("\r\n", "")

, false, (Map<String, Object>)sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.get("TinShellETL"));

//开始涉及PLETL，于是 上中下都要，就把outputOut 完整代入VPCS函数。

//更新

//计算完后去除output的 mid down 部分

if(output.containsKey("midShell")) {

output.remove("midShell");

}

if(output.containsKey("downShell")) {

output.remove("downShell");

}

sQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put("TinShellETL", output);

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

//输出 检测

outputjTextPane.setContentType("text/html");

System.out.println("end:"+output.size());

Iterator<String> iterator= output.keySet().iterator();

StringBuilder stringBuilder= new StringBuilder();

int max= 50;

int i= 0;

while(iterator.hasNext()){

if(i++> max) {

break;

}

String string= iterator.next();

System.out.println(output.get(string));

stringBuilder.append("/r/n"+output.get(string).toString());

}

//稍后涉及分页20211001

stringBuilder= stringBuilder.length()>300000? stringBuilder.delete(300000, stringBuilder.length()):stringBuilder;

outputjTextPane.setText(stringBuilder.toString());

outputjTextPane.validate();

}

}

});

this.add(jlabel\_flush\_button);

jTextPane= new JTextPane();

JScrollPane jsp\_jTextPane= new JScrollPane(jTextPane);

jsp\_jTextPane.setBounds(10+ 0\* 150, 20+ 1\* 15+ 30, 765, 220);

// jTextPane.setText("tableName:中药同源:select;\r\n"

// + "condition:or:功效|contains|清热:功效|contains|解毒;\r\n"

// + "condition:and:性味|!contains|热:脉络|contains|肺;\r\n"

// + "condition:and:风险规避|fliter|毒:风险规避|fliter|孕;\r\n"

// + "getCulumns:功效:风险规避|as|风险:脉络:性味:中药名称|as|药名;\r\n"

// + "aggregation:风险|color|yellow;\r\n"

// + "aggregation:药名|color|red;\r\n"

// + "aggregation:功效|parser|pos;\r\n"

// + "aggregation:0|limit|20;\r\n"

// + "aggregation:药名|sortString|increment;");

// jTextPane.setText("表名:中药同源:选择;\r\n"

// + "条件:或:功效|包含|清热:功效|包含|解毒;\r\n"

// + "条件:和:性味|不包含|热:脉络|包含|肺;\r\n"

// + "条件:和:风险规避|过滤|毒:风险规避|过滤|孕;\r\n"

// + "获取列名:功效:风险规避|改名|风险:脉络:性味:中药名称|改名|药名;\r\n"

// + "操作:风险|颜色|黄色;\r\n"

// + "操作:药名|颜色|红色;\r\n"

// + "操作:功效|分词|词性;\r\n"

// + "操作:0|行至|20;\r\n"

// + "操作:药名|字符排序|从小到大;");

//稍后改名

// jTextPane.setText("获取表名:中药同源:进行选择;

// 条件为:和:功效|精度搜索|风热咳嗽|0;

// 条件为:和:中药名称|字符串长度大于|3;

// 条件为:或:功效|包含|清热解毒:功效|包含|利尿;

// 条件为:和:性味|不包含|温:脉络|包含|肺;

// 条件为:和:风险规避|过滤掉|毒:风险规避|过滤掉|孕;

// 获取表列名:功效:风险规避|改名为|风险:脉络:性味:中药名称|改名为|药名;

// 操作:0|行至|20;

// 操作:风险|颜色标记为|黄色;

// 操作:药名|颜色标记为|红色;

// 操作:功效|进行分词|DNN;

// + "操作:药名|进行字符排序|从小到大;");

//结果出西瓜

jTextPane.setText("获取表名:中医诊断:进行选择;\r\n"

+ "条件为:和:笔记|包含|发热:笔记|包含|身重;\r\n"

+ "获取表列名:ID:病症;\r\n"

+ "操作:0|行至|30;\r\n"

+ "操作:病症|进行分词|词性显色;\r\n"

+ "操作:ID|进行数字排序|从小到大;"

+ "操作:ID|颜色标记为|红色;");

// + "操作:药名|进行字符排序|从小到大;");

// + "PLETL:中节点|进行表格合并|主码为|ID|模式为|新增列;"

// 下面这个五个命令 rest命令 首先符号冲突，

// + "定义:变量1|://localhost......;" (正在设计)//稍后。

// + "PLTCP:病症|进行WEB请求|接口为|localhost|端口为|8000|操作为|分词;"

// + "PLTCP:病症|进行WEB请求|接口为|localhost|端口为|8000|操作为|DNN;"

// + "PLTCP:病症|进行WEB请求|接口为|localhost|端口为|8000|操作为|POS;"

// + "PLETL:该节点|进行输出|模式为|打印;" (正在设计)

// + "PLETL:该节点|进行保存|模式为|文件|路径为|F盘|巴拉/巴拉小魔仙/。。。。.lyg;" (正在设计)

// + "PLETL:文档|进行执行|时间为|时间戳|路径为|D盘|巴拉巴拉小魔仙/。。。。.etl;" (正在设计)

this.add(jsp\_jTextPane);

outputjTextPane= new JTextPane();

JScrollPane jsp\_outputjTextPane= new JScrollPane(outputjTextPane);

jsp\_outputjTextPane.setBounds(10 + 0\* 150, 20+ 1\* 15+ 30+ 250, 765, 350);

outputjTextPane.setText("\"正在使用 养疗经 1.8.8.8.0 Tin Shell系统(8.8.8.0) . . .\"");

this.add(jsp\_outputjTextPane);

//jTextPane.setText("正在使用 养疗经 1.8.8.8.0 Tin Shell系统(8.8.8.0) . . .");

}

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void keyPressed(KeyEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void keyReleased(KeyEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void keyTyped(KeyEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseClicked(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseEntered(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent arg0) {

// TODO Auto-generated method stub

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSI.OPE.SI.MCI.OEI.OVU.PQE.extOSGI;

import OSI.OPE.OVU.MVU.OVU.PQE.nodeEdit.LinkNode;

import OSI.OPE.OVU.MVU.OVU.PQE.nodeEdit.Sort;

public class OSGI\_chansfer {

//增加tin shell output map， 罗瑶光20211008

public OSGI\_chansfer(LinkNode node, LinkNode first){

first= Sort.sort(first);

LinkNode linkNode= new LinkNode();

linkNode= first;

//节点只有上中下3个input，于是优化成 max= 3；

int max= 0;

while(null!= linkNode){

if(node.tBeconnect

//&&node.tBeconnectID= = linkNode.ID

&&node.tBeconnetName.equals(linkNode.name)

&& (node.tBeconnectPrimaryKey.equalsIgnoreCase(linkNode.primaryKey))){

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.toptablein

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.tableout;

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.topgin

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.gout;

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.topOutput

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut;

//return; // 涉及多个节点测试

max++ ;

}

if(node.mBeconnect

//&&node.mBeconnectID= = linkNode.ID

&& node.mBeconnetName.equals(linkNode.name)

&& (node.mBeconnectPrimaryKey.equalsIgnoreCase(linkNode.primaryKey))){

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.midtablein

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.tableout;

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.midgin

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.gout;

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.midOutput

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut;

//return;

max++ ;

}

if(node.dBeconnect

//&&node.dBeconnectID= = linkNode.ID

&& node.dBeconnetName.equals(linkNode.name)

&& (node.dBeconnectPrimaryKey.equalsIgnoreCase(linkNode.primaryKey))){

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.downtablein

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.tableout;

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.downgin

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.gout;

node.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI.downOutput

= linkNode.thisFace.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut;

//return;

max++ ;

}

if(null= = linkNode.next|| 3= = max){//以后节点类型多了就重新设计。20211011 罗瑶光

break;

}

linkNode= linkNode.next;

}

}

}

-----------------------------------------------------------------------------------------------------------------------------------

注册表

package OSI.OPE.SI.MCI.OEI.OVU.PQE.extOSGI;

import java.io.IOException;

import java.util.Map;

import javax.swing.JTextPane;

import ME.APM.VSQ.App;

import OCI.ME.analysis.C.A;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.fFT.FFTFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.fft2DFilter.Ft2DFilterInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.freqCount.FreqCountNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.guassianWav2DFilter.GuassianWav2DFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.houghWavFilter.HoughWavFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.laplacianFilter.LaplacianFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.logFFT.LogFFTInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.logFFTcount.LogFFTcountInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.lygFilter.LygFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.lygSlaveFilter.LygSlaveFilterInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.maxMiniFilter.MaxMiniFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.medianFilter.MedianFilterNodeInterface;

import OSI.OEU.OSU.MSQ.OSU.AVU.OSQ.wavRead.WavReadNodeInterface;

import OSI.OPE.OEQ.MCQ.OVU.PQE.osgi.\*;

import OSI.OSU.MSQ.ASU.OSU.PSU.MSU.AVQ.ASQ.OPE.xlsReaderNode.XlsReaderNodeInterface;

import OSI.OSU.PSI.OSU.MSQ.VQ.SQ.lygWrite.LYGWriteNodeInterface;

import OSI.OSU.PSU.OSU.MSQ.VQ.SQ.aviToLyg.AVItoLYGNodeInterface;

import OSI.OSU.PSU.OSU.MSQ.VQ.SQ.movieTransfer.MovieTransferNodeInterface;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.AddTinShellNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI;

import OSI.OSU.SI.OVI.OSI.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.addPGSearchPage.AddPGSearchPageNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI;

//import OSI.OSU.SI.OVI.OSI.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.addZNSZPage.AddZNSZPageNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI;

import OSI.OSU.VSQ.OSU.MSQ.VQ.SQ.lygPlayer.LYGPlayerNodeInterface;

import OSI.OSU.VSQ.OSU.MSQ.VQ.SQ.lygRead.LYGReadNodeInterface;

import OSI.OVI.OSU.MSQ.MV.SQ.imageRead.ImageReadNodeInterface;

import OSI.OVQ.OSU.MSQ.MV.SQ.findColorB.FindColorBInterface;

import OSI.OVQ.OSU.MSQ.MV.SQ.findColorG.FindColorGInterface;

import OSI.OVQ.OSU.MSQ.MV.SQ.findColorR.FindColorRInterface;

import OSI.OVQ.OSU.MSQ.MV.SQ.show3D.Show3DInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.embossFilter.EmbossFilterInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.grayFilter.GrayFilterNodeInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.guassianFilter.GuassianFilterInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.houghTransform.HoughTransformInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.imageStrech.ImageStrechNodeInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.laplacianFilter.LaplacianFilterInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.medianImageFilter.MedianImageNodeInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.morphologyFilter.MorphologyFilterInterface;

import OSI.OVU.OSU.MSQ.MV.SQ.sobelFilter.SobelFilterNodeInterface;

import OSI.OVU.OSU.MSQ.OSU.AVU.OSQ.butterworthFilter.ButterworthFilterNodeInterface;

import OSI.PEQ.OSU.MSQ.OSU.AVU.OSQ.wavePlay.WavePlayNodeInterface;

public class OSI\_OSU\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI\_register{

JTextPane text;

Object[][] tableData\_old;

public App u;

public A \_A;

public Map<String, String> pos;

public OSI\_OSU\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI\_register(Object[][] tableData\_old, JTextPane text, App u

, A \_A, Map<String, String> pos){

this.text= text;

this.tableData\_old= tableData\_old;

this.u= u;

this.\_A= \_A;

this.pos= pos;

}

public NodeOSGI Rigester(NodeOSGI first, LinkOSGI link) throws IOException{

//注册

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI XlsReadernode= new XlsReaderNodeInterface();

first= link.addNode(first, XlsReadernode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI imageReadNode= new ImageReadNodeInterface();

first= link.addNode(first, imageReadNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI imageStrechNode= new ImageStrechNodeInterface();

first= link.addNode(first, imageStrechNode);

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI arffTransferNode= new arffTransferNodeInterface();

// first= link.addNode(first, arffTransferNode);

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI WekaPilot2DNode= new WekaPilot2DNodeInterface();

// first= link.addNode(first, WekaPilot2DNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI MedianImageNode= new MedianImageNodeInterface();

first= link.addNode(first, MedianImageNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI GrayFilterNode= new GrayFilterNodeInterface();

first= link.addNode(first, GrayFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI GuassianFilterNode= new GuassianFilterInterface();

first= link.addNode(first, GuassianFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI FindColorRNode= new FindColorRInterface();

first= link.addNode(first, FindColorRNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI FindColorGNode= new FindColorGInterface();

first= link.addNode(first, FindColorGNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI FindColorBNode= new FindColorBInterface();

first= link.addNode(first, FindColorBNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI SobelFilterNode= new SobelFilterNodeInterface();

first= link.addNode(first, SobelFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI EmbossFilterNode= new EmbossFilterInterface();

first= link.addNode(first, EmbossFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI LaplacianFilterNode= new LaplacianFilterInterface();

first= link.addNode(first, LaplacianFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI HoughTransformNode= new HoughTransformInterface();

first= link.addNode(first, HoughTransformNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI WavReadNode= new WavReadNodeInterface();

first= link.addNode(first, WavReadNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI MedianFilterNode= new MedianFilterNodeInterface();

first= link.addNode(first, MedianFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI ButterworthFilterNode= new ButterworthFilterNodeInterface();

first= link.addNode(first, ButterworthFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI LaplacianWaveFilterNode= new LaplacianFilterNodeInterface();

first= link.addNode(first, LaplacianWaveFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI HoughWavFilterNode= new HoughWavFilterNodeInterface();

first= link.addNode(first, HoughWavFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI GuassianWav2DFilterNode= new GuassianWav2DFilterNodeInterface();

first= link.addNode(first, GuassianWav2DFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI MaxMiniFilterNode= new MaxMiniFilterNodeInterface();

first= link.addNode(first, MaxMiniFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI wavePlayNode= new WavePlayNodeInterface();

first= link.addNode(first, wavePlayNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI Show3DNode= new Show3DInterface();

first= link.addNode(first, Show3DNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI MorphologyFilter= new MorphologyFilterInterface();

first= link.addNode(first, MorphologyFilter);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI LYGReadNode= new LYGReadNodeInterface();

first= link.addNode(first, LYGReadNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI LYGWriteNode= new LYGWriteNodeInterface();

first= link.addNode(first, LYGWriteNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI MovieTransferNode= new MovieTransferNodeInterface();

first= link.addNode(first, MovieTransferNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI AVItoImagesNode= new AVItoLYGNodeInterface();

first= link.addNode(first, AVItoImagesNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI LYGPlayerNode= new LYGPlayerNodeInterface();

first= link.addNode(first, LYGPlayerNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI FFTFilterNode= new FFTFilterNodeInterface();

first= link.addNode(first, FFTFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI freqCountNode= new FreqCountNodeInterface();

first= link.addNode(first, freqCountNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI lygFilterNode= new LygFilterNodeInterface();

first= link.addNode(first, lygFilterNode);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI lygFilterComp= new Ft2DFilterInterface();

first= link.addNode(first, lygFilterComp);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI lygSlave= new LygSlaveFilterInterface();

first= link.addNode(first, lygSlave);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI logFFT= new LogFFTInterface();

first= link.addNode(first, logFFT);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI logFFTcount= new LogFFTcountInterface();

first= link.addNode(first, logFFTcount);

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI dNA3DShow= new dNA3DShowNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI();

// first= link.addNode(first, dNA3DShow);

//

//医学图片页添加

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI searchPG= new AddPGSearchPageNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI();

searchPG.pageName= "图片搜索";

//first= link.addNode(first, searchPG);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI

= (OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI) searchPG;

u.gUISample.nodeReflection.put(OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_SQ\_VPC\_PCS, null);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.register(u.gUISample.tableData\_old, u.gUISample.text

, u.gUISample.u, u.gUISample.\_A, u.gUISample.pos);

try {

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.IMP\_PSU();

} catch (IOException e1) {

e1.printStackTrace();

}

// u.gUISample.nodeView.first= u.gUISample.nodeView.link.addNode(u.gUISample.nodeView.first

// , OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

first= link.addNode(first, OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

u.searchList.add(searchPG);

//add extp//////

// //声诊断,该接口已经2年停止开源研发。

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI soundCheck= new AddZNSZPageNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI();

// soundCheck.pageName= "智能声诊";

// //first= link.addNode(first, searchPG);

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI

// = (OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI) soundCheck;

// u.gUISample.nodeReflection.put(oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_SQ\_VPC\_PCS, null);

// oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.register(u.gUISample.tableData\_old, u.gUISample.text

// , u.gUISample.u, u.gUISample.\_A, u.gUISample.pos);

// try {

//oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.IMP\_PSU();

// } catch (IOException e1) {

//e1.printStackTrace();

// }

//// u.gUISample.nodeView.first= u.gUISample.nodeView.link.addNode(u.gUISample.nodeView.first

//// , OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

// first= link.addNode(first, oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

// u.searchList.add(soundCheck);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI tinShell= new AddTinShellNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI();

tinShell.pageName= "Tin语言";

//first= link.addNode(first, searchPG);

OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI

= (OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI) tinShell;

u.gUISample.nodeReflection.put(oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_SQ\_VPC\_PCS, null);

oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.register(u.gUISample.tableData\_old, u.gUISample.text

, u.gUISample.u, u.gUISample.\_A, u.gUISample.pos);

try {

oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI.IMP\_PSU();

} catch (IOException e1) {

e1.printStackTrace();

}

// u.gUISample.nodeView.first= u.gUISample.nodeView.link.addNode(u.gUISample.nodeView.first

// , OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

first= link.addNode(first, oSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

u.searchList.add(tinShell);

//ddPGSearchPageNodeASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI

////////////////

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI editPanelReader= new EditPanelReaderNodeInterface(this.text);

// first= link.addNode(first, editPanelReader);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI editPanelReaderH= new EditPanelReaderHNodeInterface(this.text);

// first= link.addNode(first, editPanelReaderH);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI addChuFangAttributeH= new AddChuFangAttributeHNodeInterface(this.tableData\_old

// , this.text);

// first= link.addNode(first,addChuFangAttributeH);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI filterChuFangJinJiAttributeH=

// new filterChuFangJinJiAttributeHNodeInterface(this.tableData\_old, this.text);

// first= link.addNode(first, filterChuFangJinJiAttributeH);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI chuFangWuXingShowHInterface=

// new ChuFangWuXingShowHNodeInterface(this.tableData\_old, this.text);

// first= link.addNode(first,chuFangWuXingShowHInterface);

// //扫描jar、、添加jar

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI filterChuFangXingWeiKeyWordsAttributeH=

// new filterChuFangXingWeiKeyWordsAttributeHNodeInterface(this.tableData\_old, this.text);

// first= link.addNode(first, filterChuFangXingWeiKeyWordsAttributeH);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI filterChuFangJinJiKeyWordsAttributeH=

// new filterChuFangJinJiKeyWordsAttributeHNodeInterface(this.tableData\_old, this.text);

// first= link.addNode(first, filterChuFangJinJiKeyWordsAttributeH);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI updateToEditPane=

// new updateToEditPaneNodeInterface(this.tableData\_old, this.text);

// first= link.addNode(first, updateToEditPane);

//

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI readNodeInterface=

// new ReadNodeInterface(this.tableData\_old, this.text);

// first= link.addNode(first, readNodeInterface);

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI DNN3DInterface=

// new ChuFangDNN3DShowNodeInterface(this.tableData\_old, this.text, this.u, this.\_A, this.pos);

// first= link.addNode(first, DNN3DInterface);

// try {

////1扫描

////1.1设计一个文件夹

////扫描文件夹下面资源包录入

//String jarCategoryLink= "";

//FileDialog filedialog= new FileDialog(new Frame(), StableData.ATTENSION\_LOAD\_HISTORY

// , FileDialog.LOAD);

//filedialog.setFilenameFilter(new TXTFilter(StableData.FILE\_FORMAT\_ETL));

//filedialog.setVisible(true);

//jarCategoryLink= filedialog.getDirectory();

////System.out.println(jarCategoryLink);

//if(null= = jarCategoryLink|| jarCategoryLink.isEmpty()|| jarCategoryLink.contains

// (StableData.FILE\_FORMAT\_JAR)) {

// System.out.println(StableData.ATTENSION\_RECHOICE);

// return first;

//}

//File file= new File(jarCategoryLink);

//if(file.isFile()) {

// System.out.println(StableData.ATTENSION\_FILE\_CHOICE);

// return first;

//}

//File[] files= file.listFiles();

//for(int i= 0; i< files.length; i++) {

// @SuppressWarnings({ "deprecation", "resource" })

// URLClassLoader loader= new URLClassLoader(new URL[]{ files[i].toURL() });

// String filename= files[i].getName().replace(StableData.FILE\_FORMAT\_JAR, StableData.STRING\_EMPTY);

// String[] columns= filename.split("\\.");

// //如下注释2行代码 refer https://www.cnblogs.com/chinaxin/p/3678442.html 这小伙子以后有前途。哈哈

// //Class<?> myclass= loader.loadClass("hand.java.loadjar.TestClass");

// //Gene new object

// //Object myobject= myclass.newInstance();

// Class<?> myclass= null;

// try {

// myclass= loader.loadClass(filename+ "."+ columns[columns.length- 1]

// + StableData.NODE\_NODE\_INTERFACE);

// } catch (ClassNotFoundException e) {

// // TODO Auto-generated catch block

// e.printStackTrace();

// }

// Object myobject= null;

// try {

// myobject= myclass.newInstance();

// } catch (InstantiationException | IllegalAccessException e) {

// // TODO Auto-generated catch block

// e.printStackTrace();

// }

// //我准备之后设计成病毒式热插拔，因为绕过虚拟机的思想涉及情报学特工和计算机病毒领域

// //，害怕国家相关安全体系管控，暂时不研发。

// OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI= (OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI) myobject;

// first= link.addNode(first, OSU\_AVQ\_ASQ\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI);

//}

// }catch(Exception e) {

//

// }

return first;

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell;

import java.awt.ScrollPane;

import java.io.IOException;

//import java.util.HashMap;

//import java.util.Iterator;

//import java.util.concurrent.ConcurrentHashMap;

import java.util.Map;

import javax.swing.JFrame;

import ME.APM.VSQ.HRJFrame;

import ME.APM.VSQ.OPE.config.ShellJPanel;

import OSI.OPE.OEQ.MCQ.OVU.PQE.osgi.\*;

//midshell downshell, PLETL的时代开始了。稍后增加pletl的mid down 计算命令集合。

public class AddTinShellRun extends OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI{

private static final long serialVersionUID= 1L;

public int value= 0;

public String filepath;

public AddTinShellRun() throws IOException{

super();

}

//把SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut的地址位剥离出来，避免计算重叠，

//罗瑶光20211009

@SuppressWarnings("unchecked")

public void run(final AddTinShellView SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ) throws IOException, CloneNotSupportedException{

SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.tableout= this.toptablein;

//SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut= this.topOutput;

//if(null= = SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut) {

// SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut= new HashMap<String, Object>();

//}

SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut= new TinMap();

// if(null!= this.topOutput) {//我先设置成top为mainshell mid和down为附加shell，这样就可以设计 附加shell的命令了。

//SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut= this.topOutput.clone();

// }

// if(null!= this.midOutput) {

//SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put("midShell", this.midOutput.clone());

// }

// if(null!= this.downOutput) {

//SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put("downShell", this.downOutput.clone());

// }

if(null!= topOutput) {

SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut= topOutput.clone();

}

Map<String, Object> map= (Map<String, Object>)SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.get("TinShellETL");

if(null!= midOutput&& null!= map) {

map.put("midShell", midOutput.clone());

}

if(null!= downOutput&& null!= map) {

map.put("downShell", downOutput.clone());

}

if(null!= map) {

SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put("TinShellETL", map);

}

// if(null!= this.topOutput) {

//Iterator<String> iterator= this.topOutput.keySet().iterator();

//while(iterator.hasNext()) {

// String string= iterator.next();

// ConcurrentHashMap<String, Object> newMap= new ConcurrentHashMap<>();

// ConcurrentHashMap<String, Object> map= (ConcurrentHashMap<String, Object>)this.topOutput.get(string);

// Iterator<String> iterators= map.keySet().iterator();

// while(iterators.hasNext()) {

// String strings= iterators.next();

// newMap.put(strings, map.get(strings));

// }

// SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut.put(string, newMap);

//}

// }

//先设计一种 只有上链接的模式，以后在设计三种的

JFrame jframe= new JFrame();

//把SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ.outputOut的地址位剥离出来，避免计算重叠，

ShellJPanel ShellJPanel= new ShellJPanel(HRJFrame.NE, SQ\_OSU\_MSQ\_OSU\_AVQ\_ASQ\_AVQ\_ASQ\_OVQ\_OSQ\_VSQ

, null= = this.topOutput? new TinMap(): this.topOutput, this.midOutput, this.downOutput);

//之前统一节点界面是300\*300，因为这个改成800\*750，不太好就干脆分开来

ScrollPane scrollPane= new ScrollPane();

scrollPane.setSize(810, 760);

scrollPane.add(ShellJPanel);

jframe.setLayout(null);

jframe.add(scrollPane);

jframe.setSize(810,760);

jframe.setIconImage(HRJFrame.NE.logo.getImage());

jframe.setResizable(false);

jframe.setVisible(true);

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.concurrent.ConcurrentHashMap;

public class TinMap extends ConcurrentHashMap<String, Object> implements Cloneable{

/\*\*

\*因为map 下面的指令集比较复杂，先用一个clone 代替，如果不行就再完整设计一个copy函数

\*测试了下不能复制，不知道是不是这个jdk版本问题，于是重新设计 map copy clone 函数。

\*本可以用jsonString，什么都解决了， 因为涉及著作权申请，能不用第三方就不用。

\*罗瑶光 20211009

\*/

private static final long serialVersionUID= 1L;

@SuppressWarnings({ "unchecked", "rawtypes" })

@Override

public TinMap clone() throws CloneNotSupportedException {

TinMap newTinMap= new TinMap();

if(null!= this) {

Iterator<String> iterator= this.keySet().iterator();

while(iterator.hasNext()) {

String string= iterator.next();

ConcurrentHashMap<String, Object> newMap= new ConcurrentHashMap<>();

ConcurrentHashMap<String, Object> map= (ConcurrentHashMap<String, Object>)this.get(string);

Iterator<String> iterators= map.keySet().iterator();

while(iterators.hasNext()) {

String strings= iterators.next();

if(strings.contains("obj")) {

//arraylist<hashmap>

ArrayList<HashMap<String, HashMap<String, HashMap<String, String>>>> arrayListnew= new ArrayList();

ArrayList<HashMap<String, HashMap<String, HashMap<String, String>>>> arrayList= (ArrayList)map.get(strings);

//

Iterator<HashMap<String, HashMap<String, HashMap<String, String>>>> iteratormap= arrayList.iterator();

while(iteratormap.hasNext()) {

HashMap<String, HashMap<String, HashMap<String, String>>> hashmapNew= new HashMap<>();

HashMap<String, HashMap<String, HashMap<String, String>>> hashmap= iteratormap.next();

Iterator<String> iteratormapIterator= hashmap.keySet().iterator();

while(iteratormapIterator.hasNext()) {

String iteratormapIteratorString= iteratormapIterator.next();

HashMap<String, HashMap<String, String>> hashMapsNew= new HashMap<>();

HashMap<String, HashMap<String, String>> hashMaps= hashmap.get(iteratormapIteratorString);

Iterator<String> iteratormapIteratorHashMaps= hashMaps.keySet().iterator();

while(iteratormapIteratorHashMaps.hasNext()) {

String iteratormapIteratorHashMapsString= iteratormapIteratorHashMaps.next();

HashMap<String, String> iteratormapIteratorHashMapsStringHashMapsNew= new HashMap<>();

HashMap<String, String> iteratormapIteratorHashMapsStringHashMaps= hashMaps.get(iteratormapIteratorHashMapsString);

Iterator<String> iteratormapIteratorHashMapsStringHashMapsIterator= iteratormapIteratorHashMapsStringHashMaps.keySet().iterator();

while(iteratormapIteratorHashMapsStringHashMapsIterator.hasNext()) {

String stringCell= iteratormapIteratorHashMapsStringHashMapsIterator.next(); iteratormapIteratorHashMapsStringHashMapsNew.put(stringCell,iteratormapIteratorHashMapsStringHashMaps.get(stringCell).toString());

}

hashMapsNew.put(iteratormapIteratorHashMapsString, iteratormapIteratorHashMapsStringHashMapsNew);

}

hashmapNew.put(iteratormapIteratorString, hashMapsNew);

}

arrayListnew.add(hashmapNew);

}

newMap.put(strings, arrayListnew);

//object row

}else if(strings.contains("spec")) {

List<String> list= new ArrayList<>();

List<String> object= (ArrayList)map.get(strings);

Iterator<String> iteratorString= object.iterator();

while(iteratorString.hasNext()) {

list.add(iteratorString.next().toString());

}

newMap.put(strings, list);

//array

}else if(strings.contains("hashmap")) {

//map

//newMap.put(strings, map.get(strings));

}else {

//字符串

newMap.put(strings, map.get(strings).toString());

}

// Object object= map.get(strings);

// Iterator<String> iteratorss= maps.keySet().iterator();

// while(iteratorss.hasNext()) {

// String stringss= iteratorss.next();

// //array

// Object object= maps.get(stringss);

//// if(object.getType().equals("ArrayList")) {

////

//// }

// //map

//

// //object

// }

//newMap.put(strings, map.get(strings));

}

newTinMap.put(string, newMap);

}

}

return newTinMap;

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package ME.APM.VSQ;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.ScrollPane;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import javax.swing.ImageIcon;

import javax.swing.JTabbedPane;

import ME.APM.VSQ.OPE.config.SectionJPanel;

import ME.APM.VSQ.OPE.config.ShellJPanel;

import OPM.ESU.admin.PLSQLSectionPanel;

import OPM.ESU.admin.YouBiaoSectionPanel;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.AddTinShellView;

import OSI.OSU.SI.ASQ.OSD.AVI.AEI.ACI.ASI.OVI.OEI.OCI.OSI.PVI.PEI.PCI.PSI.tinShell.TinMap;

import OPM.ESU.admin.VPCSRestPanel;

import SVQ.stable.StableFile;

public class AppConfig extends ScrollPane implements MouseListener, KeyListener, ActionListener{

private static final long serialVersionUID= 1L;

public boolean isConfig= true;

public SectionJPanel SectionJPanel;

public PLSQLSectionPanel pLSQLJPanel;

public YouBiaoSectionPanel youBiaoJPanel;

public ShellJPanel pLShellJPanel;

public VPCSRestPanel vPCSRestPanel;

public void IV\_(App app){

StableFile.DNA\_PDN.put(this.getClass().getCanonicalName(), true);

JTabbedPane jTabbedpane= new JTabbedPane();

Container SectionJPanelContainer= new Container();

SectionJPanel= new SectionJPanel(app);

SectionJPanel.setLayout(null);

SectionJPanel.setBounds(0, 0, 800, 750);

SectionJPanelContainer.add(SectionJPanel);

jTabbedpane.addTab("总启动专科与系统配置版面", new ImageIcon(), SectionJPanelContainer

, "总启动专科与系统配置版面");

jTabbedpane.setMnemonicAt(0, KeyEvent.VK\_0);

Container pLSQLJPanelContainer= new Container();

pLSQLJPanel= new PLSQLSectionPanel(app);

pLSQLJPanel.setLayout(null);

pLSQLJPanel.setBounds(0, 0, 800, 750);

pLSQLJPanelContainer.add(pLSQLJPanel);

jTabbedpane.addTab("德塔PLSQL控制台", new ImageIcon(), pLSQLJPanelContainer

, "德塔PLSQL控制台");

jTabbedpane.setMnemonicAt(1, KeyEvent.VK\_1);

Container pLShellJPanelContainer= new Container();

pLShellJPanel= new ShellJPanel(app, new AddTinShellView(), new TinMap(), new TinMap(), new TinMap());

pLShellJPanel.setLayout(null);

pLShellJPanel.setBounds(0, 0, 800, 750);

pLShellJPanelContainer.add(pLShellJPanel);

jTabbedpane.addTab("德塔TIN SHELL语言控制台", new ImageIcon(), pLShellJPanelContainer

, "德塔TIN SHELL语言控制台");

jTabbedpane.setMnemonicAt(2, KeyEvent.VK\_2);

Container vPCSRestPanelContainer= new Container();

vPCSRestPanel= new VPCSRestPanel(app);

vPCSRestPanel.setLayout(null);

vPCSRestPanel.setBounds(0, 0, 800, 750);

vPCSRestPanelContainer.add(vPCSRestPanel);

jTabbedpane.addTab("德塔WEB智能控制台", new ImageIcon(), vPCSRestPanelContainer,

"德塔WEB智能控制台");

jTabbedpane.setMnemonicAt(3, KeyEvent.VK\_3);

Container zongHeJPanelContainer= new Container();

youBiaoJPanel= new YouBiaoSectionPanel(app);

youBiaoJPanel.setLayout(null);

youBiaoJPanel.setBounds(0, 0, 800, 750);

zongHeJPanelContainer.add(youBiaoJPanel);

jTabbedpane.addTab("综合游标配置中心", new ImageIcon(), zongHeJPanelContainer

, "综合游标配置中心");

jTabbedpane.setMnemonicAt(4, KeyEvent.VK\_4);

//this.setLayout(null);

this.setPreferredSize(new Dimension(800, 750));

//jTabbedpane.setBounds(0, 0, 805, 505);

this.add(jTabbedpane);

this.setBounds(0, 0, 793, 753);

this.setVisible(true);

this.validate();

}

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void keyPressed(KeyEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void keyReleased(KeyEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void keyTyped(KeyEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseClicked(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseEntered(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent arg0) {

// TODO Auto-generated method stub

}

}

-----------------------------------------------------------------------------------------------------------------------------------

package OEU.LYG4DQS4D;

//20200314 集成了最新的小高峰过滤催化排序5代思想。

//20200818 集成了最新的小高峰过滤催化排序9代思想。

//增加同拼音同笔画的字按char的int大小区分20210529

//罗瑶光

//今天将新陈代谢技术应用到 中文拼音笔画分词上.

//罗瑶光

public class LYG10DWCMSSort15D\_XCDX\_C\_U\_A extends LYG10DWCMSSort13D\_XCDX\_C\_A implements LYG10DWCMSSort13D\_XCDX\_C\_U\_A\_C {

public void processKernel(String[] kernel, int leftPosition

, int rightPosition, int scale, int point) {

int rightPositionReflection= rightPosition;

if(point> scale) {

return;

}

processQS4DLYG9D(kernel, leftPosition, rightPosition, scale, point, 0);

int i;

for(i= leftPosition; i<= rightPosition; i++) {

if(!(kernel[i].length()<= point|| kernel[leftPosition].length()<= point)) {

if(kernel[i].charAt(point)!= kernel[leftPosition].charAt(point)){

rightPositionReflection= i- 1;

processKernel(kernel, leftPosition, rightPositionReflection, scale, point+ 1);

leftPosition= i;

}

}

}

if(leftPosition!= rightPosition) {

processKernel(kernel, leftPosition, i- 1, scale, point+ 1);

}

}

public void processSort(String[] kernel, int leftPosition

, int rightPosition, int scale, int point) {

if(point> scale) {

return;

}

for(int i= leftPosition; i<= rightPosition; i++) {

Here:

for(int j= i; j<= rightPosition; j++) {

if(i= = j) {

continue Here;

}

if(kernel[i].length()<= point|| kernel[j].length()<= point) {

if(kernel[i].length()< kernel[j].length()) {

for(int p= 0; p< scale; p++) {

if(!(kernel[i].length()<= p|| kernel[j].length()<= p)) {

if(kernel[i].charAt(p)!= kernel[j].charAt(p)) {

continue Here;

}

}

}

String temp= kernel[i].toString();;

kernel[i]= kernel[j].toString();;

kernel[j]= temp;

}

continue Here;

}else {

boolean hasXi= pinyin.containsKey(""+ kernel[i].charAt(point));

boolean hasXj= pinyin.containsKey(""+ kernel[j].charAt(point));

boolean hasBi= bihua.containsKey(""+ kernel[i].charAt(point));

boolean hasBj= bihua.containsKey(""+ kernel[j].charAt(point));

if(!(!hasXi|| !hasXj)){//都有拼音

String[] js= new String[2];

js[0]= this.pinyin.get(""+ kernel[i].charAt(point));

js[1]= this.pinyin.get(""+ kernel[j].charAt(point));

if(js[0].equalsIgnoreCase(js[1])) {

if(!(!hasBi|| !hasBj)){//都有笔画

if(this.bihua.get(""+ kernel[i].charAt(point))

> this.bihua.get(""+ kernel[j].charAt(point))) {

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

continue Here;

}else if(this.bihua.get(""+ kernel[i].charAt(point))

= = this.bihua.get(""+ kernel[j].charAt(point))) {

int asci= kernel[i].charAt(point);

int ascj= kernel[j].charAt(point);

if(asci< ascj) {//根据前面select的sort定义来规范,盲目改成大于会出错.

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

continue Here;

}

}

}

}

boolean change= processSortpinyin(js, 3);

if(!(!change|| i>= j)) {

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

}

continue Here;

}else if(!(hasXi|| !hasXj)){//其中一个有拼音

if(i< j) {

if(!(i= = rightPosition+1 || j= = rightPosition+1)) {

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

}

}

continue Here;

}else if(!(!hasXi|| hasXj)){

if(i> j) {

if(!(i= = rightPosition+1 || j= = rightPosition+1)) {

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

}

}

continue Here;

}else if(!(hasXi|| hasXj)){//都没有拼音

if(kernel[i].toLowerCase().charAt(point)

> kernel[j].toLowerCase().charAt(point)) {

if(i< j) {

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

}

continue Here;

}

if(kernel[i].toLowerCase().charAt(point)

= = kernel[j].toLowerCase().charAt(point)) {

if(kernel[i].charAt(point)> kernel[j].charAt(point)) {

if(i< j) {

String temp= kernel[i].toString();

kernel[i]= kernel[j].toString();

kernel[j]= temp;

}

}

continue Here;

}

}

}

}

}

}

public void processQS4DLYG9D(String[] kernel, int leftPosition

, int rightPosition, int scale, int point, int deep) {

if(leftPosition< rightPosition){

int c= rightPosition- leftPosition+ 1;

if(!(c< this.range|| deep> this.deeps)) {//增加了deep

int pos= partition(kernel, leftPosition, rightPosition, scale, point);

if(leftPosition< pos- 1) {

processQS4DLYG9D(kernel, leftPosition, pos- 1, scale, point, deep+ 1);

}

if(pos+ 1< rightPosition) {

processQS4DLYG9D(kernel, pos+ 1, rightPosition, scale, point, deep+ 1);

}

return;

}

processSort(kernel, leftPosition, rightPosition, scale, point);

return;

}

}

public int partition(String[] array, int leftPosition, int rightPosition, int scale, int point) {

String x= findSmall(array, scale, point, leftPosition, rightPosition, rightPosition)

? array[rightPosition]: array[leftPosition];

int leftPositionReflection= leftPosition;

while(leftPositionReflection< rightPosition) {

while(!(findSmallWithTwoChar(array[leftPositionReflection]

, x, scale, point)|| leftPositionReflection++ >= rightPosition)) {}

while(findSmallWithTwoChar(array[rightPosition--], x, scale, point)){}

if(leftPositionReflection< ++rightPosition){

String temp= array[rightPosition].toString();;

array[rightPosition]= array[leftPositionReflection].toString();;

array[leftPositionReflection]= temp;

}

}

array[leftPosition]= array[rightPosition].toString();

array[rightPosition]= x.toString();

return rightPosition;

}

}

-----------------------------------------------------------------------------------------------------------------------------------

7.1 文件名目录

public class E\_PLSearch\_E {

public class P\_AggregationPLETL

public class P\_AggregationPLSearch {

public class P\_AggregationPLTCP {

public class P\_ConditionPLSearch\_XCDX\_Cache extends P\_ConditionPLSearch\_XCDX {

public class P\_ConditionPLSearch\_XCDX\_Kernel extends P\_ConditionPLSearch\_XCDX {

public class P\_ConditionPLSearch\_XCDX\_Map extends P\_ConditionPLSearch\_XCDX {

public class P\_ConditionPLSearch\_XCDX {

public class P\_GetCulumnsPLSearch {

public class P\_RelationPLSearch {

public class PLSearchCommand\_E {

public class SortStringDemo{

public interface PL\_XA\_C{

public class PL\_XA\_E implements PL\_XA\_C{

public class SearchShellQ\_JoinRows\_E {

public class SearchShellTable{

public class SearchShellTables{

public class ShellJPanel extends JPanel implements MouseListener, KeyListener, ActionListener{

public class OSGI\_chansfer {

public class OSI\_OSU\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI\_register{

public class AddTinShellRun extends OSU\_AVQ\_ASQ\_OPE\_OPC\_ECI{

public class TinMap extends ConcurrentHashMap<String, Object> implements Cloneable{

public class AppConfig extends ScrollPane implements MouseListener, KeyListener, ActionListener{

public class LYG10DWCMSSort15D\_XCDX\_C\_U\_A extends LYG10DWCMSSort13D\_XCDX\_C\_A implements LYG10DWCMSSort13D\_XCDX\_C\_U\_A\_C {

7.2 文件内容

DNA元基编码索引

SEARCH= XA

CONDITION= CO

AGGREGATION= AO

E\_PLSearch\_E= E\_PL\_XA\_E

P\_AggregationPLETL= P\_AO\_PLETL

P\_AggregationPLSearch= P\_AO\_PL\_XA

P\_AggregationPLTCP= P\_AO\_PLTCP

P\_ConditionPLSearch\_XCDX\_Cache= P\_CO\_PL\_XA\_XCDX\_Cache

P\_ConditionPLSearch\_XCDX\_Kernel = P\_CO\_PL\_XA\_XCDX\_Kernel

P\_ConditionPLSearch\_XCDX\_Map= P\_CO\_PL\_XA\_XCDX\_Map

P\_ConditionPLSearch\_XCDX= P\_CO\_PL\_XA\_XCDX

P\_GetCulumnsPLSearch= P\_I\_CulumnsPL\_XA

P\_RelationPLSearch= P\_RelationPL\_XA

PLSearchCommand\_E= PL\_XA\_Command\_E

SortStringDemo= SortStringDemo

PL\_XA\_C= PL\_XA\_C

PL\_XA\_E= PL\_XA\_E

SearchShellQ\_JoinRows\_E= XA\_ShellQ\_JoinRows\_E

SearchShellTable= XA\_ShellTable

SearchShellTables= XA\_ShellTables

ShellJPanel= ShellJPanel

OSGI\_chansfer= OSGI\_chansfer

OSI\_OSU\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI\_register= OSI\_OSU\_ASQ\_OCQ\_OSI\_PCI\_PCU\_MCI\_MCU\_MSI\_register

AddTinShellRun = I\_TinShellRun

TinMap = TinMap

AppConfig = App\_CM

LYG10DWCMSSort15D\_XCDX\_C\_U\_A= LYG10DWCMSSort15D\_XCDX\_C\_U\_A

LYG10DWCMSSort13D\_XCDX\_C\_U\_A\_C= LYG10DWCMSSort13D\_XCDX\_C\_U\_A\_C

8 引用

Deta Socket流可编程数据库系统

DNA元基编码与肽计算 第三修订版

9 注意

9.1（Programmable Language SQL）PLSQL 第一次提出这个关键词概念 为美国甲骨文公司。