

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_pl_XA_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<>();
    //I make 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")) {
            Pl_XA_Command_E.P_SetRoot(acknowledge, output);
        }
        if(acknowledge[0].equals("baseName")) {
            Pl_XA_Command_E.P_BaseName(acknowledge, output);
        }
        if(acknowledge[0].equals("获取表名")) {
            Pl_XA_Command_E.P_TableName(acknowledge, output);
        }
        if(acknowledge[0].equals("columnName")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("changeCulumnName")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("culumnValue")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("join")) {
            Pl_XA_Command_E.P_Join(acknowledge, output);
        }
        if(acknowledge[0].equals("条件为")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("relation")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("操作")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("PLETL")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("获取表列名")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        if(acknowledge[0].equals("PLTCP")) {
            Pl_XA_Command_E.P_ListNeedStart(acknowledge, output);
        }
        output.put("newCommand", acknowledge[0]);
    }
}
```

```

        Pl_XA_Command_E.P_E(acknowledge, output, mod);
        output.put("lastCommand", output.get("newCommand"));
    }
    if(null!= acknowledge) {
        if(output.get("start").toString().equals("1")) {
            Pl_XA_Command_E.P_E(acknowledge, output, mod);
        }
    }
    System.out.println("1");
    Pl_XA_Command_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.LYG10DWCMSort13D_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.ANNTTest;
import PEQ.AMV.ECS.test.DNNTTest;
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_AO_PLETL {
    @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
                    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.LYG10DWCMSort13D_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.ANNTTest;
import PEQ.AMV.ECS.test.DNNTTest;
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_AO_pl_XA {
    @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 模式。
            //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 模式。
    //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> columnValue= (Map<String, Object>)rowValue.get(sets[0]);
        maps.put(columnValue.get("columnValue").toString(), map);
        strings[index++]= columnValue.get("columnValue").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 模式。
    //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> columnValue= (Map<String, Object>)rowValue.get(sets[0]);
    }
}

```



```

        maps.put(columnValue.get("columnValue").toString(), map);
        doubles[index++]= Double.valueOf(columnValue.get("columnValue").toString());
    }
    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> mapColumn= (Map<String, Object>)map.get(sets[0]);
            String rowCellFromString= mapColumn.get("columnValue").toString();
            if(sets[2].equalsIgnoreCase("红色")) {
                sets[2]= "red";
            }
            if(sets[2].equalsIgnoreCase("黄色")) {
                sets[2]= "yellow";
            }
            if(sets[2].equalsIgnoreCase("蓝色")) {
                sets[2]= "blue";
            }
            if(sets[2].equalsIgnoreCase("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.println(string);
    SensingTest sensingTest= HRJFrame.NE._A.getSensingTest();
    DNNTest dNNTest= new DNNTest();
    ANNTTest aNNTest= new ANNTTest();
    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;
}
output.clear();
output.addAll(outputTemp);
}
//分出去
public static void P_PletLimitMap(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.LYG10DWCMSort13D_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_AO_PLTCP {
    @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

```

[illegible]

[illegible]

```

        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("\n");
    out= strings.length> 3? strings[3]: "";
    out= String_ ESU.uRlencodeToURldecode(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.XA_ShellTable;
import OSA.shell.XA_ShellTables;
@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_CO_pl_XA_XCDX_Cache extends P_CO_pl_XA_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。
        //XA_ShellTable table= XA_ShellTables.XA_ShellTables.get(tableName);
        //稍后把下面的 Table 替换成 XA_ShellTable 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
        XA_ShellTable 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= XA_ShellTables.XA_ShellTables.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_CO_pl_XA_XCDX_Map.rowMapToRow(outputTemp.get(i));
        }
        table= new XA_ShellTable();
        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 mapSearchAtII= iteratorWordFrequency.next();
                WordFrequency wordFrequencySearch= mapSearchWithoutSort.get(mapSearchAtII);
                if(temps.contains(mapSearchAtII)) {
                    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(mapSearchAtII)) {
//        reg[copyCount] += 1;
//        score_code[copyCount] += 1 << mapSearchAtII.length() <<
wordFrequencySearch.getFrequency() ;
        continue Here;
    }
    if(pos.get(mapSearchAtII).contains("名")||pos.get(mapSearchAtII).contains("动")
        ||pos.get(mapSearchAtII).contains("形")||pos.get(mapSearchAtII).contains("谓"))
{
        reg[copyCount] += 2;
    }
    reg[copyCount] += 1;
    score_code[copyCount] += (temps.contains(mapSearchAtII) ? 2 : 1)
        * (!pos.get(mapSearchAtII).contains("名")
            ? pos.get(mapSearchAtII).contains("动")? 45 : 1 : 50)
        << mapSearchAtII.length() * wordFrequencySearch.getFrequency();
    continue Here;
}
if(mapSearchAtII.length() > 1) {
    for(int j = 0; j < mapSearchAtII.length(); j++) {
        if(temps.contains(String.valueOf(mapSearchAtII.charAt(j)))) {
            if(reg[copyCount] == 0) {
                count += 1;
            }
            //                score[copyCount] = temps;
            score_code[copyCount] += 1;
            if(pos.containsKey(String.valueOf(mapSearchAtII.charAt(j))) && (
                pos.get(String.valueOf(mapSearchAtII.charAt(j))).contains("名")
                ||pos.get(String.valueOf(mapSearchAtII.charAt(j))).contains("动")
                ||pos.get(String.valueOf(mapSearchAtII.charAt(j))).contains("形")
                ||pos.get(String.valueOf(mapSearchAtII.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_CO_pl_XA_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++ ) {
        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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_XCDX_Map.rowToRowMap(row));
            Map<Integer, Boolean> recordRows= (Map<Integer, Boolean>) object.get("recordRows");
            recordRows.put(i, true);
        }
    }
}

```



```

String[] culumnList= readDBTableRowIndexFile.list(); //生成一个列表头名组
NextFile:
    for(String column: culumnList) { //遍历头名
        if(column.contains("is_delete")) { //已删除文件逃逸
            continue NextFile;
        }
        String DBTableCulumnIndexPath= readDBTableRowIndexFile + "/" + column; //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(column, temp); //储值
            cell.I_CellValue(temp); //数据库内存储值
            bufferRow.putCell(column, cell);
        } else {
            rowMap.put(column, null);
            cell.I_CellValue(null);
            bufferRow.putCell(column, 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
//huaRuiJiJtableCulums 对应 column
//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[] huaRuiJiJtableCulums
    , Object[][] huaRuiJiJtable, List<Map<String, Object>> output, Row bufferRow, Map<String, Object> rowMap)
throws IOException {
    Object[] rowList= huaRuiJiJtable[rowId];
    for(int i= 0; i< huaRuiJiJtableCulums.length; i++) {
        Cell cell= new Cell();
        rowMap.put((String)huaRuiJiJtableCulums[i], rowList[i]);
        cell.I_CellValue(rowList[i]);
        bufferRow.putCell((String)huaRuiJiJtableCulums[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.XA_ShellTable;
import OSA.shell.XA_ShellTables;
//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_CO_pl_XA_XCDX_Map extends P_CO_pl_XA_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
    XA_ShellTable table;
    //outputTemp.addAll(output);

```



```

if((output.isEmpty()||null== output)&& object.get("firstTime").equals("true")) {
    table= XA_ShellTables.XA_ShellTables.get(tableName);
    object.put("firstTime", "others");
}else {
    Row[] huaRuiJiJtableRows= new Row[output.size()];
    for(int i= 0; i< output.size(); i++) {
        huaRuiJiJtableRows[i]= P_CO_pl_XA_XCDX_Map.rowMapToRow(output.get(i));
    }
    table= new XA_ShellTable();
    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 mapSearchAtII= iteratorWordFrequency.next();
            WordFrequency wordFrequencySearch= mapSearchWithoutSort.get(mapSearchAtII);
            if(temps.contains(mapSearchAtII)) {
                if(reg[copyCount]= 0){
                    count += 1;
                }
                if(temps.contains(key.replace(" ", ""))) {
                    reg[copyCount]+= 500;
                }
                if(key.contains(temps.replace(" ", ""))) {
                    reg[copyCount]+= 500;
                }
            }
        }
    }
}

```

```

        if(!pos.containsKey(mapSearchAtII)) {
            reg[copyCount] += 1;
            score_code[copyCount] += 1 << mapSearchAtII.length() <<
wordFrequencySearch.getFrequency() ;
            continue Here;
        }
        if(pos.get(mapSearchAtII).contains("名")||pos.get(mapSearchAtII).contains("动")
            ||pos.get(mapSearchAtII).contains("形")||pos.get(mapSearchAtII).contains("谓"))
{
            reg[copyCount] += 2;
        }
        reg[copyCount] += 1;
        score_code[copyCount] += (temps.containsKey(mapSearchAtII) ? 2 : 1)
* (!pos.get(mapSearchAtII).contains("名") ? pos.get(mapSearchAtII).contains("动")? 45 : 1 : 50)
        << mapSearchAtII.length() * wordFrequencySearch.getFrequency();
        continue Here;
    }
    if(mapSearchAtII.length()>1) {
        for(int j= 0;j<mapSearchAtII.length();j++) {
            if(temps.containsKey(String.valueOf(mapSearchAtII.charAt(j)))) {
                if(reg[copyCount]== 0){
                    count += 1;
                }
                // score[copyCount]= temps;
                score_code[copyCount]+= 1;
                if(pos.containsKey(String.valueOf(mapSearchAtII.charAt(j)))&&(
                    pos.get(String.valueOf(mapSearchAtII.charAt(j))).contains("名")
                    ||pos.get(String.valueOf(mapSearchAtII.charAt(j))).contains("动")
                    ||pos.get(String.valueOf(mapSearchAtII.charAt(j))).contains("形")
                    ||pos.get(String.valueOf(mapSearchAtII.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.containsKey(string[j])) {
            tempb+= code;
        }
    }
    //断句
    for(int j= 0; j< stringReg.length; j++) {
        if(temps.containsKey(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.containsKey(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;
Here:
    for(int i= score.length- 1; i> 0; i--) {
        if(score_code[i]< 1){
            continue Here;
        }
        outputTemp.add(P_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_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_CO_pl_XA_XCDX {
}

```

```

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_I_CulumnsPL_XA {
    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("columnName", 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_RelationPL_XA {
    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("columnValue").toString()).doubleValue()
                    == new BigDecimal(cellJoin.get("columnValue").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("columnValue").toString().equals(cellJoin.get("columnValue").toString())) {
            if(!findinNewObj.containsKey(objRowId + ":" + objJoinRowId)) {
                Map<String, Object> newObjRow= new HashMap<>();

```



```

        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.selectRowsByAttributesOfGetCulums(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", XA_ShellQ_JoinRows_E.selectRowsByAttributesOfJoinCondition(object));
            }
        }
    }

```

```

        if(object.containsKey("relation")) {
            object.put("obj", XA_ShellQ_JoinRows_E.selectRowsByAttributesOfJoinRelation(object));
        }
        if(object.containsKey("aggregation")) {
            //object.put("obj", XA_ShellQ_JoinRows_E.selectRowsByAttributesOfJoinAggregation(object));
        }
        if(object.containsKey("getCulums")) {
            object.put("joinObj", XA_ShellQ_JoinRows_E.selectRowsByAttributesOfJoinGetCulums(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 conjunction 变换  $a \wedge b \wedge c * kernel[] = (a \wedge b)^{\wedge c} * kernel[] = (a \vee b)^{\wedge c} * kernel[]$ 
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("columnName");
    object.remove("changeColumnName");
    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());
    //XA_ShellTable table= XA_ShellTables.XA_ShellTables.get(object.get("tableName").toString());
    //Object[] specs= table.getHuaRuiJiJtableCulums();
    // }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();
    // }
    XA_ShellTable table= XA_ShellTables.XA_ShellTables.get(object.get("获取表名").toString());
    Object[] specs= table.getHuaRuiJiJtableCulums();
    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.LYG10DWCMSort13D_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 LYG10DWCMSort13D_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 getCulmns();
    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_pl_XA_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 getCumulus() {

```

```

        PLSearch+= Const.SEMICOLON+ Const.GET_COLUMNS;
        return this;
    }
    public Pl_XA_E changeColumnName(String newColumnName, String oldColumnName) {
        PLSearch+= Const.SEMICOLON+ Const.CHANGES_COLUMN_NAME+ Const.COLON
            + newColumnName+ Const.COLON+ oldColumnName;
        return this;
    }
    public Pl_XA_E withColumnName(String columnName, String dataType) {
        PLSearch+= Const.SEMICOLON+ Const.COLUMN_NAME+ Const.COLON+ columnName
            + Const.COLON+ dataType;
        return this;
    }
    public Pl_XA_E withColumnValue(String columnName, String columnValue) {
        PLSearch+= Const.SEMICOLON+ Const.COLUMN_VALUE+ Const.COLON+ columnName
            + Const.COLON+ columnValue;
        return this;
    }
    public Pl_XA_C exec(boolean b) throws Exception {
        //map= E_pl_XA_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_pl_XA_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
    //return
    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_AO_pl_XA;
import OSM.shell.P_CO_pl_XA__XCDX_Cache;
import OSM.shell.P_CO_pl_XA_XCDX_Map;
import OSM.shell.P_I_CulumnsPL_XA;
import OSM.shell.P_RelationPL_XA;
//整体 plsql 替换成 plsearch, 稍后测试验证 罗瑶光 20210927
@SuppressWarnings({"unused", "unchecked"})
public class XA_ShellQ_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_CO_pl_XA_XCDX_Map.P_Map(sets, output, object.get("joinBaseName").toString(), object);//1
    }else {
        P_CO_pl_XA_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_AO_pl_XA.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_I_CulumnsPL_XA.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_RelationPL_XA.P_AndMap(sets, obj, joinObj,object, newObj);
    }else {
        P_RelationPL_XA.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 XA_ShellTable{
    public Object[] getHuaRuiJiJtableCumlums() {
        return huaRuiJiJtableCumlums;
    }
    public void setHuaRuiJiJtableCumlums(Object[] huaRuiJiJtableCumlums) {
        this.huaRuiJiJtableCumlums= huaRuiJiJtableCumlums;
    }
    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[] huaRuiJiJtableCumlums;
    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 XA_ShellTables{
    public static Map<String, XA_ShellTable> XA_ShellTables= new HashMap<>();
    //先把接口做足， 罗瑶光 20210925
    //将表变成表映射
    public static boolean addNewXA_ShellTable(String tableName, JTable jTable) {
        //..
        TableModel defaultTableModel= jTable.getModel();
        addNewXA_ShellTable(tableName, (DefaultTableModel)defaultTableModel);
        return true;
    }
    //按 sonar 方式重复函数分离
    public static Object[] getNewXA_ShellTableSpecFromDefaultTableModel(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[][] getNewXA_ShellTableDataFromDefaultTableModel(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[] getNewXA_ShellTableRowsFromDefaultTableModel(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[] getNewXA_ShellTableRowsFromDefaultTableModel(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 addNewXA_ShellTable(String tableName, DefaultTableModel defaultTableModel) {
        //..
        Object[] jTableSpec= getNewXA_ShellTableSpecFromDefaultTableModel(defaultTableModel);
        Object[][] jTableData= getNewXA_ShellTableDataFromDefaultTableModel(defaultTableModel);
        addNewXA_ShellTable(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 addNewXA_ShellTableWithObjectData(String tableName, DefaultTableModel defaultTableModel
        , Object[][] defaultTableData) {
        //..
        Object[] jTableSpec= getNewXA_ShellTableSpecFromDefaultTableModel(defaultTableModel);
        addNewXA_ShellTable(tableName, defaultTableData, jTableSpec);
        return true;
    }
    //shell 映射封装
    public static boolean addNewXA_ShellTable(String tableName, Object[][] defaultTableData
        , Object[] defaultTableDataSpec) {
        //..
        XA_ShellTable XA_ShellTable= new XA_ShellTable();
        XA_ShellTable.setHuaRuiJiJtableCulumns(defaultTableDataSpec);
        XA_ShellTable.setHuaRuiJiJtableRows(getNewXA_ShellTableRowsFromDefaultTableModel(defaultTableDataSpec,
defaultTableData));
        XA_ShellTable.setHuaRuiJiJtable(defaultTableData);
        XA_ShellTable.setHuaRuiJiJtableName(tableName);
        XA_ShellTables.put(tableName, XA_ShellTable);
        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 addNewXA_ShellTableWithObjectDataReflectionDBRows(String tableName, DefaultTableModel
defaultTableModel
        , Object[][] defaultTableData) {
        //..
        Object[] jTableSpec= getNewXA_ShellTableSpecFromDefaultTableModel(defaultTableModel);
        //defaultTableModel TO ROWS
        Row[] rows= getNewXA_ShellTableRowsFromDefaultTableModel(jTableSpec, defaultTableModel);
        addNewXA_ShellTableReflectionDBRows(tableName, rows, jTableSpec);
        return true;
    }
    //Reflection map 表头方式存储
    //shell 映射封装
    public static boolean addNewXA_ShellTableReflectionDBRows(String tableName, Row[] rows
        , Object[] defaultTableDataSpec ) {
        //..
        XA_ShellTable XA_ShellTable= new XA_ShellTable();
        XA_ShellTable.setHuaRuiJiJtableCulumns(defaultTableDataSpec);
        XA_ShellTable.setHuaRuiJiJtableRows(rows);
        XA_ShellTable.setHuaRuiJiJtableName(tableName);
        XA_ShellTables.put(tableName, XA_ShellTable);
        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 addInitXA_ShellTable(App app) {
        //把养疗经的表都在这里初始化映射成 XA_ShellTables 内存先。
        addNewXA_ShellTable("西医内科", app.xynkPage.tableData_old, app.xynkPage.columnTitle);
        addNewXA_ShellTable("中医方剂", app.zynkxPage.tableData_old, app.zynkxPage.columnTitle);
        addNewXA_ShellTable("中医诊断", app.zydxPage.tableData_old, app.zydxPage.columnTitle);
        addNewXA_ShellTable("古籍经典", app.fyydPage.tableData_old, app.fyydPage.columnTitle);
    }

```

```

        addNewXA_ShellTable("中医生殖", app.fqzPage.tableData_old, app.fqzPage.columnTitle);
        addNewXA_ShellTable("妇产科学", app.fckxPage.tableData_old, app.fckxPage.columnTitle);
        addNewXA_ShellTable("急诊科学", app.jzcxPage.tableData_old, app.jzcxPage.columnTitle);
        addNewXA_ShellTable("西医外科", app.wkxPage.tableData_old, app.wkxPage.columnTitle);
        addNewXA_ShellTable("中医外伤", app.wskxPage.tableData_old, app.wskxPage.columnTitle);
        addNewXA_ShellTable("西药手册", app.xyscPage.tableData_old, app.xyscPage.columnTitle);
        addNewXA_ShellTable("中药同源", app.tableData_old, app.columnTitle);
        addNewXA_ShellTable("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.XA_ShellTables;
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_pl_XA_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("\n正在使用 养疗经 1.8.8.8.0 Tin Shell 系统(8.8.8.0) . . .\n");
        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、
            //分类执行脚本编译器
            //
            if(null!= HRJFrame.NE) {
                XA_ShellTables.addInitXA_ShellTable(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、
            //分类执行脚本编译器
        }
    }
});
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、
            //分类执行脚本编译器
            //
            //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_pl_XA_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("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;");
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.tBeconnectName.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.mBeconnectName.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;
}
}
}
}
}
-----}
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 I_TinShellRun extends OSU_AVQ_ASQ_OPE OPC_ECI{
private static final long serialVersionUID= 1L;
public int value= 0;
public String filepath;
public I_TinShellRun() 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());
                }
            }
        }
    }
}

```

```

        }
        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 App_CM 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 LYG10DWCMSort15D_XCDX_C_U_A extends LYG10DWCMSort13D_XCDX_C_A implements
LYG10DWCMSort13D_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;
}
}

```