润乾
RAQSOFT

敏捷计算编程语言

润乾软件出品

www.raqsoft.com.cn

目录 CONTENTS

01 集算器是什么

02 结构化数据计算的广泛性

03 为什么是SPL

05 SPL语言理念

06 集算器特色功能

目录 CONTENTS

01

集算器是什么

集算器

- 一种面向结构化数据的程序计算语言
- 集算器又称：SPL（Structured Process Language）
- 敏捷计算是集算器的主要特征

敏捷计算编程语言

SPL

敏捷计算

目录 CONTENTS

结构化数据计算的广泛性

结构化数据计算无处不在

- 结构化数据无处不在
- 结构化数据计算一直是数据处理的主流

表格数据

数据库

文件数据

WEB数据

绝大部分日常数据处理都是面向结构化数据的

编程的必要性

分析处理结构化数据的三种方式

BI工具	数据工具	编程
• 只能实施规整计算 • 无法完成数据准备	• 只能处理模式化的计算 • 无法完成灵活的复杂计算	• 可以处理所有情况

八成以上数据处理和计算的需求无法用可视化工具完成

结论

编程是处理结构化数据最有效的方式

目前仍然是必不可少的方式

目录 CONTENTS

02

为什么是SPL

现有编程语言的不足

SQL

SQL作为最常使用的结构化数据处理语言存在如下问题

难写难调试

SQL不提倡过程，难以实现复杂计算，经常要写得很长，难写难调试

过于封闭

SQL强依赖数据库，无法计算文件等库外数据，很难跨库计算

语法问题

*扩展阅读：SQL像英语是个善意的错误

SQL与SPL对比

计算目标：某支股票最长连续涨了多少交易日

```
select max(continuousDays)-1
from (select count(*) continuousDays
        from (select sum(changeSign) over(order by tradeDate) unRiseDays
                from (select tradeDate,
                        case when closePrice>lag(closePrice) over(order by tradeDate)
                        then 0 else 1 end changeSign
                    from stock) )
        group by unRiseDays)
```

SQL解法

- SQL在使用窗口函数的情况下嵌套三层完成；
- 前面读懂了吗？

	A
1	=stock.sort(tradeDate)
2	=0
3	=A1.max(A2=if(closePrice>closePrice[-1],A2+1,0))

SPL解法

其实这个计算很简单，按照自然思维：先按交易日排序（行1）,然后比较当天收盘价比前一天高就+1，否则就清零，最后求个最大值（行3）

现有编程语言的不足

Python

Python相对SQL更加灵活开放，但仍存在如下问题

难以实现复杂计算

pandas不是专业结构化数据计算包，在处理分组有序等复杂运算时较为繁琐

使用门槛太高

pandas等计算包只有专业程序员才能玩得转

*扩展阅读： Python 并不适合职场编程

语法问题

Python与SPL对比

计算每年同月份增长比

```python
import numpy as np
import pandas as pd
s = time.time()
sales = pd.read_csv("C:\\music_project_data\\sales.csv",sep='\t')
sales['ORDERDATE']=pd.to_datetime(sales['ORDERDATE'])
sales['y']=sales['ORDERDATE'].dt.year
sales['m']=sales['ORDERDATE'].dt.month
sales_g = sales[['y','m','AMOUNT']].groupby(by=['y','m'],as_index=False)
amount_df = sales_g.sum().sort_values(['m','y'])
yoy = np.zeros(amount_df.values.shape[0])
yoy=(amount_df['AMOUNT']-amount_df['AMOUNT'].shift(1))/amount_df['AMOUNT'].shift(1)
yoy[amount_df['m'].shift(1)!=amount_df['m']]=np.nan
amount_df['yoy']=yoy
print(amount_df)
```

Python解法

- Python通过多步算出结果，实现起来比SQL更容易
- 但Python代码比较繁琐，写出来并不容易

	A
1	=file("C:\\sales.csv").import@t()
2	=A1.groups(year(ORDERDATE):y,month(ORDERDATE):m;sum(AMOUNT):x)
3	=A2.sort(m)
4	=A3.derive(if(m==m[-1],x/x[-1]-1,null):yoy)

SPL解法

- SPL的分步代码更为简洁，实现更为简单

现有编程语言的不足

一些场景下还会使用JAVA、VBA实施结构化数据计算

```java
public static void groupCount(){
    groupBy mainJava = new groupBy();
    List<MainBean> list = mainJava.initList();
    List<MainBean> mainList = new ArrayList<MainBean>();
    Map<String, MainBean> map = new HashMap<String, MainBean>();
    MainBean totalBean = null;
    String employeeId = null;
    for(MainBean mainBean : list) {
        employeeId = mainBean.getEmployeeId();
        if (!map.containsKey(employeeId)) {
            totalBean = new MainBean();
            totalBean.setEmployeeId(employeeId);
            totalBean.setBuyNum(mainBean.getBuyNum());
            totalBean.setGoodsPrice(mainBean.getGoodsPrice());
            totalBean.setTotalBuyNum(mainBean.getTotalBuyNum());
            totalBean.setFreight(mainBean.getFreight());
            totalBean.setTotalGoodsPrice(mainBean.getGoodsPrice() * mainBean.getBuyNum());
            totalBean.setTotalPrice(mainBean.getGoodsPrice() * mainBean.getBuyNum() + mainBean.getFreight());
            mainList.add(totalBean);
            map.put(employeeId, totalBean);
        } else {
            totalBean = map.get(employeeId);
            totalBean.setTotalBuyNum(totalBean.getTotalBuyNum() + mainBean.getTotalBuyNum());
            totalBean.setFreight(totalBean.getFreight() + mainBean.getFreight());
            totalBean.setTotalGoodsPrice(totalBean.getTotalGoodsPrice() + mainBean.getGoodsPrice() * mainBean.getBuyNum());
            totalBean.setTotalPrice(totalBean.getTotalPrice() +
                    mainBean.getGoodsPrice() * mainBean.getBuyNum() + mainBean.getFreight());
        }
    }
```

这么长的JAVA代码仅仅实现了单个字段分组单个字段汇总的功能

JAVA和VBA都不是为结构化数据计算设计的

缺少集合化计算类库，编码过于复杂

为什么是SPL

相对以上编程语言

SPL

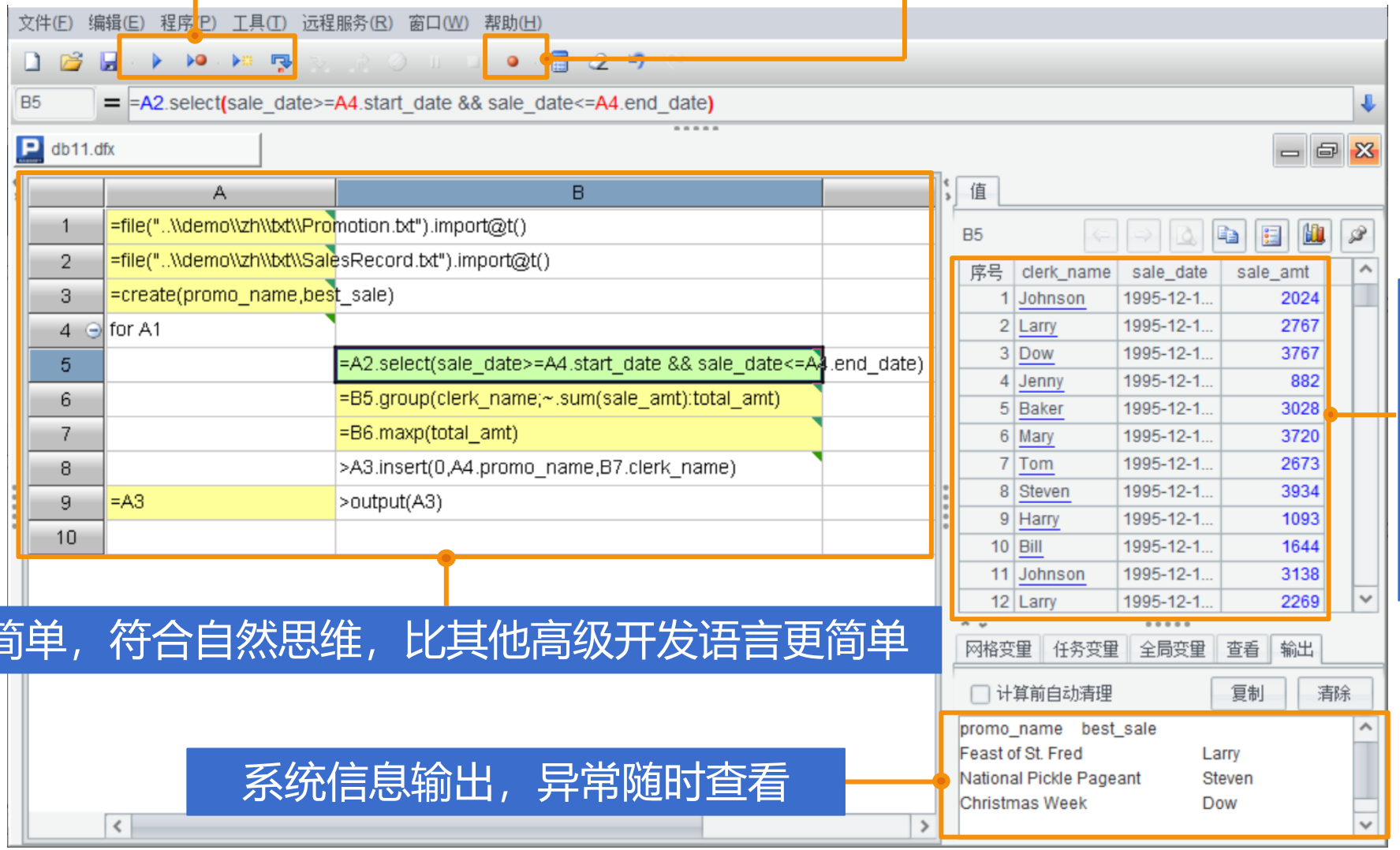
才是处理结构化数据最简洁有效的编程语言

结构化数据计算神器

简洁易用的开发环境

执行、调试执行、执行到光标、单步执行

设置断点



网格结果所见即所得，易于调试；方便引用中间结果

语法简单，符合自然思维，比其他高级开发语言更简单

系统信息输出，异常随时查看

专门设计的语法体系

SPL特别适合复杂过程运算

	A	B	C	D	E	F
1	=esProc.query("SELECT 订单ID AS 合同,订购日期 AS 日期,客户 ,订单金额 AS 金额,员工ID AS 销售 FROM 销售记录表 WHERE year(订购日期)=? OR year(订购日					
2	=esProc.query("select * from 员工表")					
3	>A1.run(销售=A2.select@1(编号:A1.销售))		/字段值 是记录			
4	=A1.group(销售)					
5	=create(销售,今年销售额,去年销售额,增长率,客户数,大客户数,大客户占比)					
6	for A4	=A6(1).销售.姓名				
7		=A6.select(year(日期)==年份).sum(金额)		/今年销售额		
8		=A6.select(year(日期)==年份-1).sum(金额)		/去年销售额		
9		=B8/B7-1	/增长率			
10		=A6.group(客户).(~.sum(金额))				
11		=B10.count()				
12		=B10.count(~>=10000)	/大客户数			
13		=B12/B11				
14		>A5.insert(0,B6,B7,B8,B9,B11,B12,B13)				
15	result A5					

天然分步、层次清晰、直接引用单元格名无需定义变量

丰富的运算类库

专门针对结构化数据表设计，类库丰富，远超SQL和Python

	A	B	C
1	=esProc.query("SELECT 订单ID AS 合同,订购日期 AS		/读取销售记录表
2	=A1.group(销售)		
3	=create(销售,今年销售额,去年销售额,客户数,大客户数)		
4	for A2	=A4(1).销售	
5		=A4.select(year(日期)==年份).sum(金额)	
6		=A4.select(year(日期)==年份-1).sum(金额)	
7		=A4.group(year(日期)).sum(金额)	
8		=B7.count()	
9		=B7.count(~>=10000)	

分组、循环

	A	B	C
1	=esProc.query("select * from 员工表")		
2	=A1.select(性别=="男")		
3	=A1.select(出生日期>=date("1970-01-01"))		
4	=A2^A3	/交运算，统计晚于1970年出生的男员工	
5	=A2&A3	/并运算，统计男员工或者晚于1970年出生的员工	
6	=A2\A3	/运算，统计早于1970年出生的男员工	
7	=A4.sum(工资)		
8	=A5.avg(年龄)		
9	=A6.sort(出生日期)		
10	/集合作为基本数据类型		
11			

集合运算

	A	B	C
1	=file("交易记录.txt").import@t()		
2	=A1.sort(客户编码,交易日期)		
3	=A2.select(车辆型号=="捷达" || 车辆型号=="迈腾").dup@t()		
4	=A3.derive(interval(交易日期[-1],交易日期):间隔)		
5	=A4.select(车辆型号[-1]=="捷达" && 车辆型号=="迈腾" &&客户编码==客户编码[-1])		
6	=A5.avg(间隔)		
7			
8			
9			

排序、过滤

	A	B	C
1	=esProc.query("select * from 员工表")		
2	=A1.sort(入职日期)		
3	=A2.pmin(出生日期)	/出生最早的员工的记录序号	
4	=A2(to(A3-1))	/直接用序号访问成员	
5	=esProc.query("select * from 股价表 where 股票代码='000062")		
6	=A5.sort(交易日期)		
7	=A6.pmax(收盘价)	/收盘价最高的那条记录的序号	
8	=A6.calc(A7,收盘价/收盘价[-1]-1)		
9			
10	/直接用序号访问成员		
11			

有序集合

更多对比

更多

SQL对比
http://www.raqsoft.com.cn/jsq/jsq-vs-1

Python对比
http://www.raqsoft.com.cn/jsq/spl-vs-python-1

职场人员该学什么程序语言
http://c.raqsoft.com.cn/article/1600999865254

桌面轻量级数据处理脚本
http://c.raqsoft.com.cn/article/1594178872147

辅助 Excel 的数据计算 add-ins
http://c.raqsoft.com.cn/article/1595320538131

查询计算大文件的桌面程序工具
http://c.raqsoft.com.cn/article/1593574900501

目录 CONTENTS

04

SPL语言理念

集合化

集合化

结构化数据总是批量形式

集合运算库 • lambda语法 • 动态数据结构

SQL是集合化的语言，但集合化不够彻底

无法描述集合的集合

字段取值不能再是集合

离散性

集合成员可以游离在集合外存在，可方便单独引用

独立运算或及其它游离成员再组合新集合运算

SQL缺乏离散性

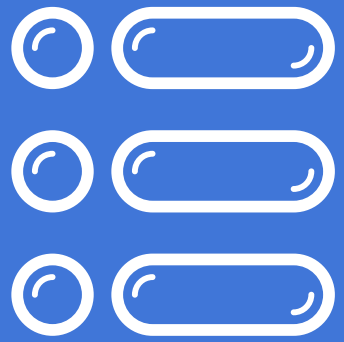
没有天然序号，引用集合指定成员很麻烦

只有单行记录的表，没有游离记录

离散性支持下的集合化

彻底的集合化需要离散性的支持

允许游离成员组成新集合



有序计算是集合化与离散性的结合物

运算不仅与数据本身有关，还和数据所在位置有关

彻底的集合化

- 显式集合数据类型
- 集合运算
- 成员再构成集合进行运算

	A
1	=人员表.group(生日).select(~.count(1)>1).conj()

	A
1	=V.len()
2	=V.to(A1\4+1,A1*3\4).run(value-=V(A1\2).value)

	A	B
1	...	
2	=A1.select(height>=1.7)	=A1.select(sex=="Female")
3	=A2^B2	=A1.select(height>=1.7 && sex=="Female")
4	=A2\B2	=A1.select(height>=1.7 && sex!="Female")

强Lambda语法解决有序运算

- 本值引用~
- 序号引用#
- 相邻成员与集合引用

	A	B
1	...	
2	=A1.select(sex=="Male")	=A1.select(sex=="Female")
3	=A2.conj(B2.([A2.~,~]))	=A3.minp@a(abs(~(1).height-~(2).height))

	A
1	...
2	=A1.group((#+2)\3)
2	=A1.group(#%3+1)

	A
...	...
3	=A2.derive(price-price[-1]:gain)
4	=A2.derive(price[-1:1].avg():mavg)

举例：返回集合的聚合

计算任务：列出每个用户最近一次登录间隔

```
WITH TT AS
        (SELECT RANK() OVER(PARTITION BY uid ORDER BY logtime DESC) rk, T.*  FROM 登录表 T)
SELECT uid,(SELECT TT.logtime FROM TT where TT.uid=TTT.uid and TT.rk=1)
            -(SELET TT.logtim FROM TT WHERE TT.uid=TTT.uid and TT.rk=2) 间隔
FROM 登录表 TTT GROUP BY uid
```

聚合函数返回值不一定是单值，也可以返回一个集合

彻底的集合化后很容易针对分组子集实施返回集合的聚合运算

	A	
1	=登录表.groups(uid;top(2,-logtime))	最后2个登录记录
2	=A1.new(uid,#2(1).logtime-#2(2).logtime:间隔)	计算间隔

举例：跨行引用

计算任务：计算每月前后各一个月的销售额移动平均值

```
WITH B AS
        (SELECT LAG(销售额) OVER (ORDER BY 月份) f1, LEAD(销售额) OVER (ORDER BY 月份) f2,   A.* FROM 销售表 A)
SELECT 月份,销售额,
        (NVL(f1,0)+NVL(f2,0)+销售额)/(DECODE(f1,NULLl,0,1)+DECODE(f2,NULL,0,1)+1) 移动平均
FROM B
```

窗口函数只有简单的跨行引用，涉及集合要用成员去拼

有序集合上可提供跨行集合引用方案

	A
1	=销售表.sort(月份).derive(销售额[-1,1].avg()):移动平均)

举例：有序分组

计算任务：一支股票最长连续上涨了多少天

```
SELECT max(连续日数) FROM
        (SELECT count(*) 连续日数 FROM
                (SELECT SUM(涨跌标志) OVER ( ORDER BY 交易日) 不涨日数 FROM
                ( SELECT 交易日,
                        CASE WHEN  收盘价>LAG(收盘价) OVER( ORDER BY 交易日 THEN 0 ELSE 1 END 涨跌标志
                FROM 股票 ))
        GROUP BY 不涨日数)
```

另一种和次序有关的分组，条件成立时产生新组

	A
1	=股票.sort(交易日).group@i(收盘价<收盘价[-1]).max(~.len())

举例：分组子集与有序分组混合

计算任务：找出连续上涨三天的股票

```
WITH A AS
    (SELECT 代码,交易日, 收盘价-LAG(收盘价) OVER (PARITITION BY 代码 ORDER BY 涨幅) FROM 股票)
B AS
    (SELECT 代码,
        CASE WHEN 涨幅>0 AND
            LAG(涨幅) OVER (PARTITION BY 代码 ORDER BY 交易日) >0  AND
            LAG(涨幅,2) OVER PARTITION BY 代码 ORDER BY 交易日) >0
            THEN 1 ELSE 0 END 三天连涨标志  FROM A)
SELECT distinct 代码 FROM B WHERE 三天连涨标志=1
```

分组子集与有序计算的组合

	A
1	=股票.sort(交易日).group(代码)
2	=A1.select((a=0,~.pselect(a=if(收盘价>收盘价[-1],a+1,0):3))>0).(代码)

举例：位置利用

计算任务：某股票股价最高的那三天的平均涨幅

```
SELECT AVG(涨幅) FROM
    ( SELECT 交易日, 收盘价-LAG(收盘价) OVER ( ORDER BY 交易日) 涨幅 FROM 股价表
        WHERE 交易日 IN
            (SELECT 交易日 FROM
                (SELECT 交易日, ROW_NUMBER() OVER(ORDER BY 收盘价 DESC) 排名 FROM 股票)
    WHERE 排名<=3 )
```

无序集合不能利用位置访问相邻成员，计算量增大

有序集合可以提供丰富的按位置访问机制

	A
1	=股票.sort(交易日)
2	=A1.calc(A1.ptop(3,-收盘价),收盘价-收盘价[-1]).avg()

离散性支持下的集合运算

- 过滤
- 计算列
- 位置引用
- 相邻引用
- 排序/排名
- 列上的宏运算

- 分组汇总/分组子集
- 枚举分组/序号分组
- 有序分组/条件分组
- 转置与扩展

- 集合交并差
- 归并
- 外键关联
- 同维主子关联

- …

合并汇总与拆分

- 列相同的表格合并
- 合并时去除重复的列
- 合并时汇总重复的列
- 横向拼接列不相同的表
- 行列方向都合并时去除重复项
- ……

查找与筛选

- 查找某个值的位置
- 使用位置筛选
- 找到第一个或最后一个
- 找到最大最小值
- 选出前N名/后N名
- ……

离散性支持下的集合运算

格值与汇总值计算

- 使用相邻行和区间计算
- 可能提前终止的累计
- 同类数据连续时使用同类相邻行计算
- 同类数据不连续时使用同类相邻行计算
- 使用同类数据的汇总信息
- .....

集合运算和从属判断

- 简单成员的交并差
- 行式数据求交并差
- 不确定数量的集合求交并差
- 集合相等与从属判断
- 次序无关的集合相等与从属判断
- .....

重复判断、计数与去除

- 判断是否有与自己重复的数据
- 统计重复次数
- 不确定多列一起统计重复次数
- 行式数据去重
- 简单数据去重
- .....

排序与排名

- 按指定次序对齐排列
- 指定次序有重复值的对齐
- 将并列排名的成员拼接起来
- 在相同分类内排序
- 分类下的排名
- .....

离散性支持下的集合运算

特殊分类和汇总方法

- 每N个成员分成一组
- 使用相邻数据作为分组条件
- 碰到空行或非空行时分组
- 按数据值的间隔分组
- 将分类内的数据拼接成文字
- ……

关联与比对

- 关联表引用
- 区间关联
- 二维关联表
- 使用区间范围倒查关联表
- 关联多行数据
- ……

行列转换

- 固定列的行转列
- 行式表与交叉表互换
- 行列的高层分类互转
- 分类内数据横向拼入列
- 分类数据拼入列时要再分类或排序
- ……

扩展与补齐

- 生成连续的区间
- 根据数值将一行扩展出多行
- 拆分文字后扩展成多行
- 在连续值中补足缺失部分
- 每隔N行补足若干空行
- ……

目录 CONTENTS

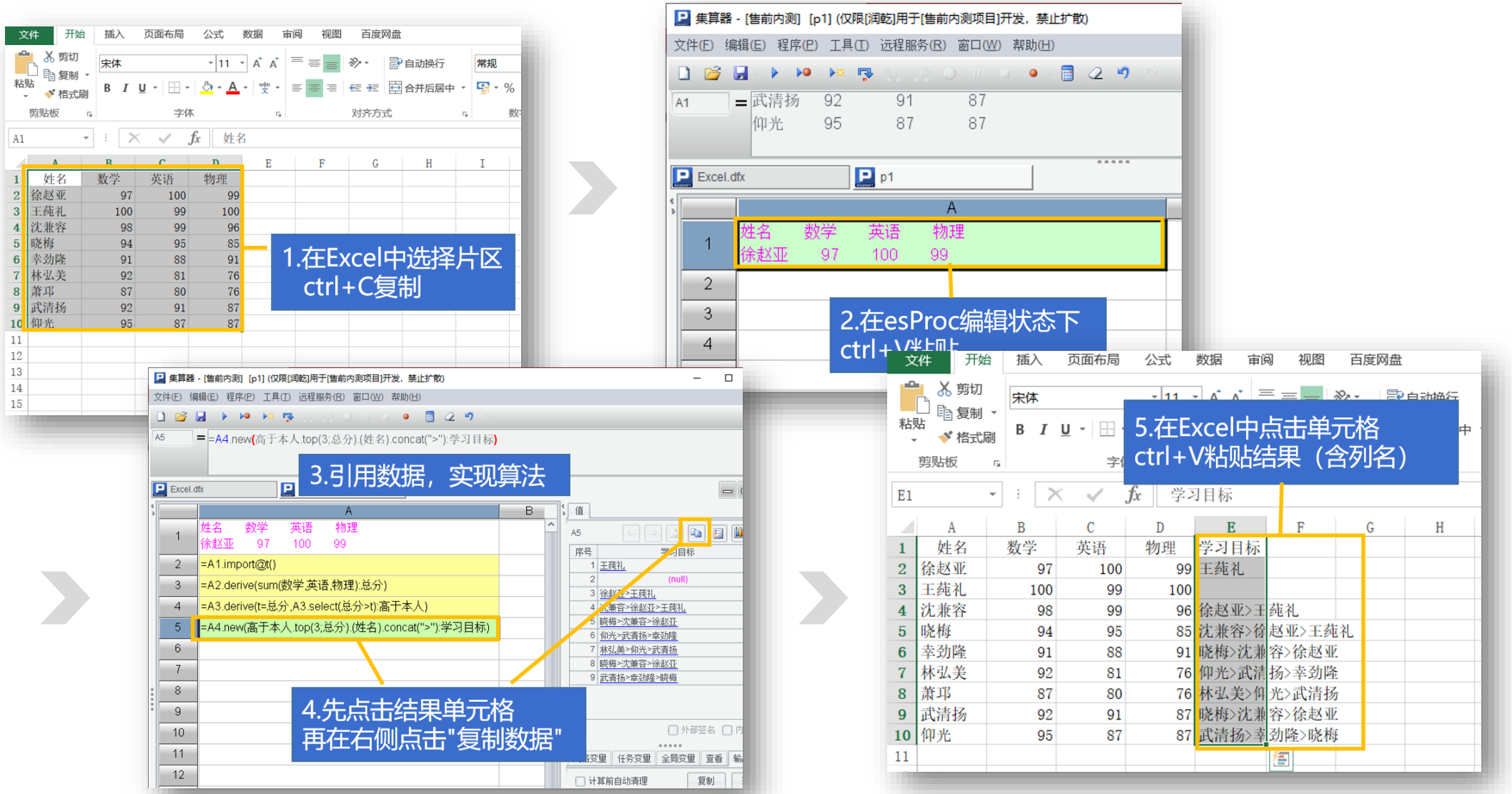
05

集算器特色功能

Excel配合

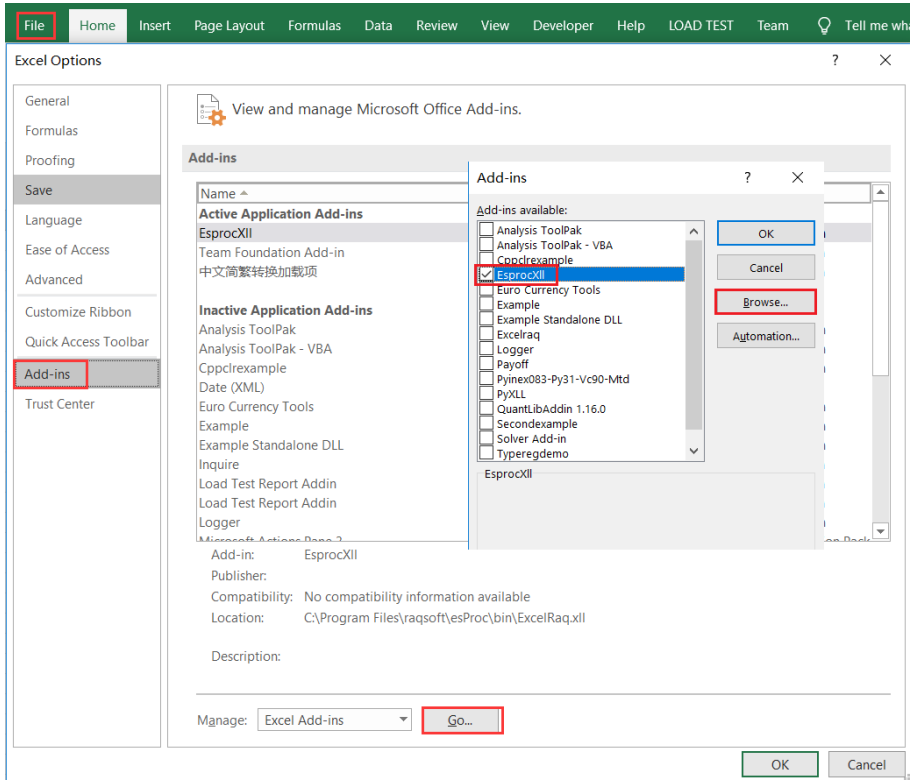
SPL与Excel配合可以增强Excel计算能力，降低计算实施难度

使用剪贴板



Excel配合

通过SPL的Excel插件可以在Excel中使用SPL函数进行计算，也可以在VBA中调用SPL脚本

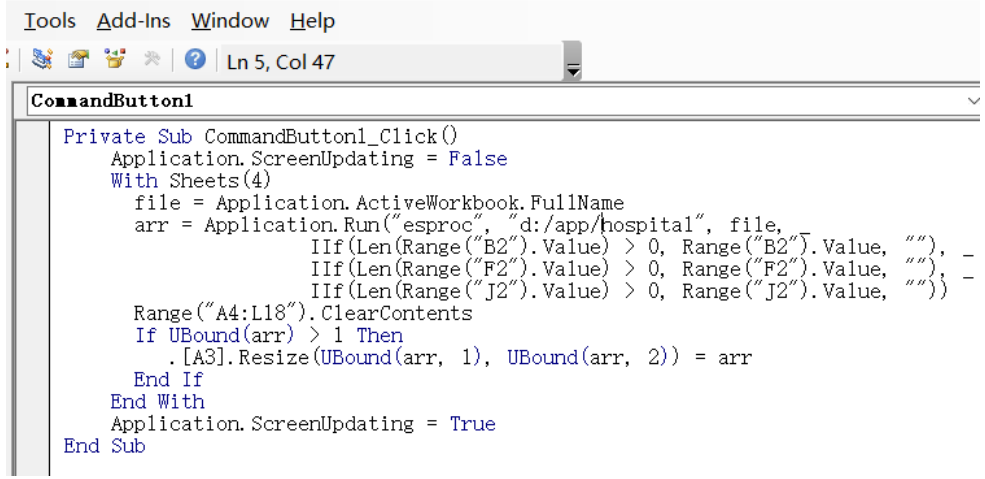


Excel加载SPL插件

Excel中使用SPL函数实施计算

```
=esproc("=T(""name.xlsx"").count(NAME==?)",A2)
```

VBA中调用SPL脚本完成计算




```
Private Sub CommandButton1_Click()
    Application.ScreenUpdating = False
    With Sheets(4)
      file = Application.ActiveWorkbook.FullName
      arr = Application.Run("esproc", "d:/app/hospital", file, _
                    IIf(Len(Range("B2").Value) > 0, Range("B2").Value, ""), _
                    IIf(Len(Range("F2").Value) > 0, Range("F2").Value, ""), _
                    IIf(Len(Range("J2").Value) > 0, Range("J2").Value, ""))
      Range("A4:L18").ClearContents
      If UBound(arr) > 1 Then
         .[A3].Resize(UBound(arr, 1), UBound(arr, 2)) = arr
      End If
    End With
    Application.ScreenUpdating = True
End Sub
```

报表工具配合

SPL可以调用报表工具生成、输出报表

报表模板文件（orders_group.rpx）

	A	B	C	D	E
1	Orders Statistical Table				
2	Area	Company	ID	Orderdate	Amount
3	=ds1.group(	=ds1.group(Company;	=ds1.select(	=ds1.Orderdate	=ds1.Amount
4				Total Amount	=sum(E3{})
5				Total Amount	=sum(E3{})

	A	B
1	=file("e:/txt/orders.txt":"UTF-8").import@t()	
2	>report_config("e:/cfg/raqsoftConfig.xml")	
3	=report_open("orders_group.rpx")	/调用报表模板
4	=report_run(A3;A1:"ds1")	/将A1数据传给报表
5	=report_exportXls@x(A3,"e:/excel/orders.xlsx")	/生成报表

SPL脚本，调用报表模板

	A	B	C	D	E
1	Orders Statistical Table				
2	Area	Company	ID	Orderdate	Amount
3	Center	Haotian	10254	2012-07-11	545.40
4				Total Amount	545.40
5				Total Amount	545.40
6	East	Dongdiwang	10249	2012-07-05	1842.00
7				Total Amount	1842.00
8		Qiangu	10251	2012-07-08	624.95
9				Total Amount	624.95
10				Total Amount	2466.95
11	North	Shantai	10248	2012-07-04	428.00
12				Total Amount	428.00
13		Shiyi	10250	2012-07-08	1523.50
14			10253	2012-07-10	1428.00
15				Total Amount	2951.50
16		Yongda	10255	2012-07-12	2450.00
17				Total Amount	2450.00
18				Total Amount	5829.50

根据报表模板样式生成的报表（Excel）

大数据计算

SPL提供了外存游标、并行计算等机制实现大数据文件计算

Python未提供大数据支持，需要手写代码，非常繁琐

计算目标：将大文件按照订单编号排序

```
45 def   merge_sort(directory,ofile,by,ascending=True,sep=","):
46
47 with open(ofile,'w') as   outfile:
48
49         file_list = os.listdir(directory)
50
51         file_chunk = [open(directory+"/"+file,'r')   for file in file_list]
52
53         k_row = [file_chunk[i].readline()for   i in range(len(file_chunk))]
54
55         by = pos_by(by,k_row[0],sep)
56
57         outfile.write(k_row[0])
58
59       k_row = [file_chunk[i].readline()for i in range(len(file_chunk))]
60
61 k_by =   [parse_type(k_row[i].split(sep)[by].strip())for i in range(len(file_chunk))]
62
63
64 with open(ofile,'a') as   outfile:
65
66         while True:
67
68             for i in range(len(k_by)):
69
70                 if i >= len(k_by):
71
72                     break
```

部分Python代码（共67行）

	A	B
1	=file(file_path).cursor@tc()	/创建大文件游标
2	=A1.sortx(key)	/游标排序
3	=file(out_file).export@tc(A2)	/结果流式输出到文件

SPL通过3行就可以完成大文件排序

文件SQL查询

通过SPL可以针对文本、Excel等文件使用SQL进行查询

提供不依赖于数据库的SQL查询能力

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97

Students_scores.txt

分组汇总

	A
1	$select CLASS,min(English),max(Chinese),sum(Math) from E:/txt/students_scores.txt group by CLASS

关联查询

	A
1	$select sum(S.quantity*P.Price) as totalfrom E:/txt/Sales.txt as S join E:/txt/Products.txt as P on S.productid=P.ID where S.quantity<=10

更多资料

SPL程序设计图书

http://c.raqsoft.com.cn/article/1607914978450

SPL 程序设计课程

http://www.raqsoft.com.cn/wx/SPL-programming.html

日常数据处理例程

http://c.raqsoft.com.cn/article/1611022287740

润乾
RAQSOFT

THANKS

创新技术 推动应用进步

www.raqsoft.com.cn