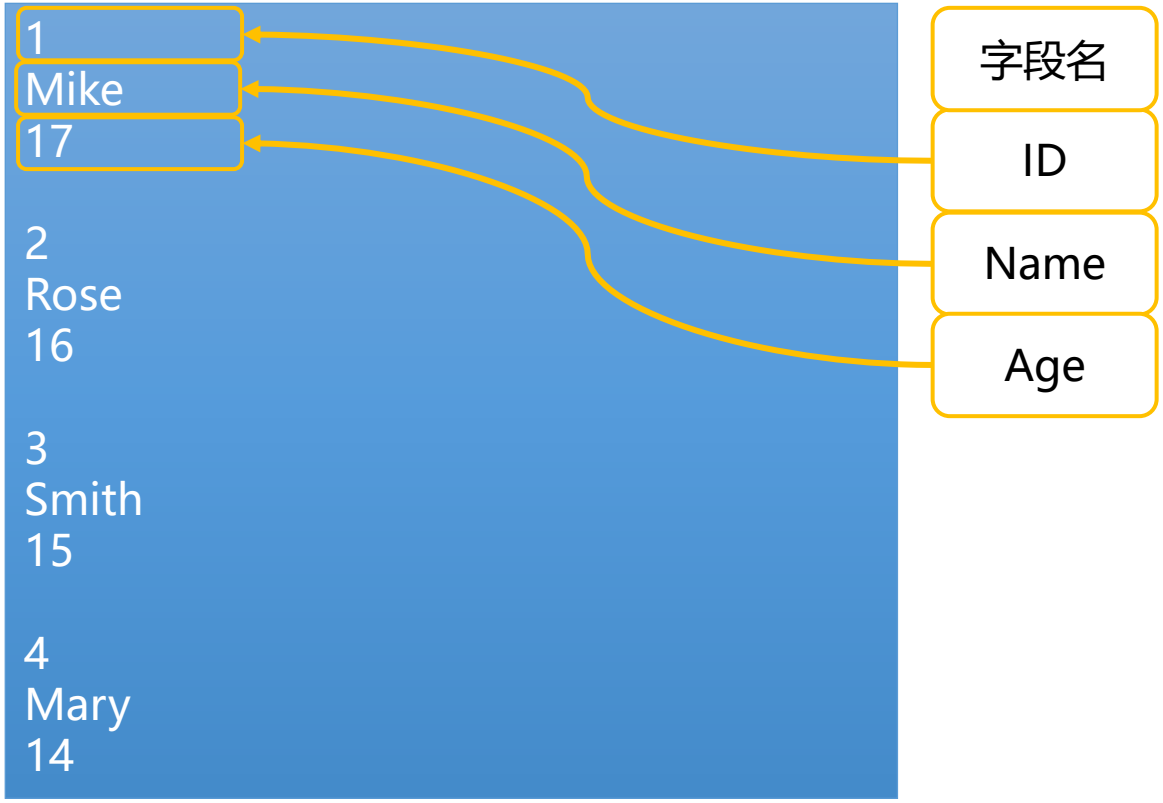
SPL Base

未结构化文本计算

CONTENTS

硬结构化—固定行

如下为每3行一条记录，然后空一行隔开的学生信息(student.txt)。



对于固定行结构的文本，只需将其读入为序列，去掉多余的空行后，再依次填充到表结构。

	A	B
1	=file("D:\\student.txt")	//打开学生文件
2	=A1.read@n()	//将数据按行读进序列
3	=A2.select(~!="")	//去掉记录间的空行
4	=create(ID,Name,Age)	//构造表结构
5	=A4.record(A3)	//将A3序列填充到表结构

缺省会将整个文本读成一个大文本串，这里用n选项表示按行读成序列，每一行一个成员。

硬结构化—固定行-游标

当文件很大（内存装不下）时，需要用游标来分块处理。

游标缺省将数据处理为带结构的序表，但该文本总共只有一列，此处用i选项将只有一列的序表转为序列来处理，方便后续对序列的计算。

注意游标数据跟上一节读行的区别，游标总是按结构化处理数据，且会自动将数据解析为相应的数据类型，所以空行被解析为null数据了。

循环分块来处理数据，这里要注意的是每次取数的块必须为3的倍数，因为当前表结构为3列，否则数据填充会错位。

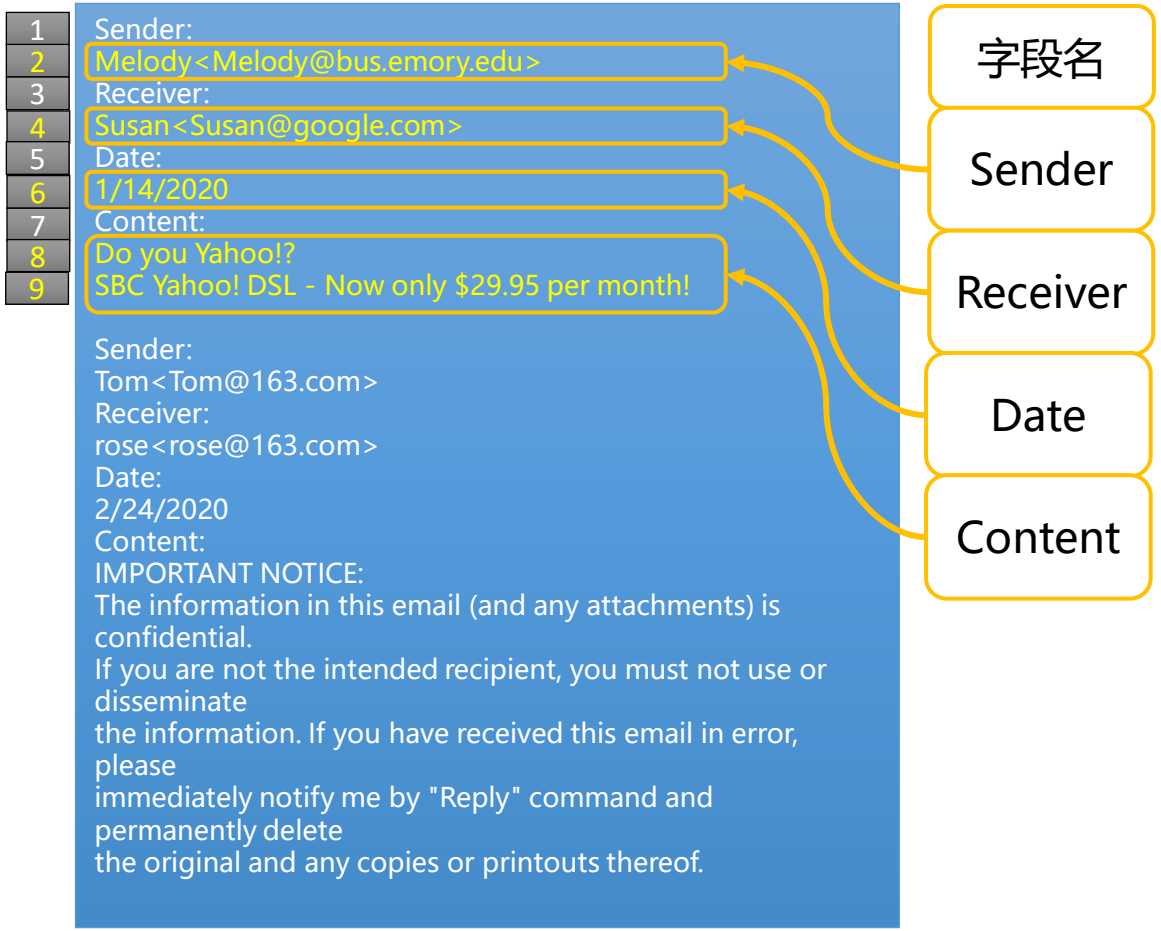
结构化的数据被计算后，要注意清空，防止内存溢出。

此时不能将文件一次性读入内存，而需将文件创建为游标，并附加上游标的延迟函数(select)后，一块块地处理数据。

	A	B
1	=file("D:\\student.txt")	//打开学生文件
2	=A1.cursor@i()	//创建文件游标
3	=A2.select(~!=null)	//去掉记录之间的空行
4	=create(ID,Name,Age)	//构造表结构
5	for A3,300	>A4.reset()
6		=A4.record(A5)

硬结构化—不定行

如下为一些邮件信息(mail.txt)，邮件Content占用行数不定。



对于不定行的文本，将其导入序列后，需要将不定行合并为一行后，再依次填充到表结构。

	A	B
1	=file("D:\\mail.txt")	//打开邮件文件
2	=A1.read@n().select(~!="")	//导入序列并去掉空行
3	=A2.group@i(~=="Sender:")	//每一个Sender:开头以及后续行为一组
4	=A3.new(~(2):Sender,~(4):Receiver,~(6):Date,**~.to(8,).string()**:Content)	//摘出记录值，合并正文，创建新的结构表
5		

~.to(8,)表示从当前分组的第8行开始，逗号后面省略参数表示取后面的所有行。.string()则将当前的行序列合并为一行。

硬结构化—不定行-游标

当文件很大（内存装不下）时，需要用游标来分块处理。

同样需要注意用i选项返回序列后，去掉空行时要用null值。

对构造好的表取数，注意实际应用中要分块取数，防止内存溢出。

	A	B
1	=file("D:\\mail.txt")	//打开邮件文件
2	=A1.cursor@i().select(~!=null)	//产生游标序列并去掉空行
3	=A2.group@i(~=="Sender:")	//按照记录分组
4	=A3.new(~(2):Sender,~(4):Receiver,~(6):Date,~.to(8,).string():Content)	//摘出记录值，合并正文，创建新的结构表
5	=A4.fetch()	//游标取出全部数据

硬结构化—单行正则拆分

如下为某视频软件的启动日志(QQLive.log)。

```
[18-08-13 13:50:21][13104]-
[0ms][QQLiveMainModule.dll][CQQLiveModule::Pars
eCommandLine] cmd="C:\Program Files
(x86)\Tencent\QQLive\QQLive.exe"
[18-08-13 13:50:21][13104]-
[78ms][QQLiveMainModule.dll]QQLiveDaemon:Reg
AllHotKey:Default Bosskey Registered.
[18-08-13 13:50:21][13104]-
[78ms][QQLiveMainModule.dll]ctrl + alt + shift + 5
Registered
[18-08-13 13:50:21][13104]-
[78ms][QQLiveMainModule.dll]ctrl + alt + shift + 6
Registered
[18-08-13 13:50:21][13104]-
[78ms][QQLiveMainModule.dll]ctrl + alt + shift + 7
Registered
```

该日志为一行对应一条记录，左边示例自动折行了，图示黄色部分为一行数据(后面所有行相同)。

此时没法用简单分割符来拆分，而且内容包含冗余的括号([])以及减号(-)，字符(ms)等。

这种情形可以用正则表达式来拆分。

	A	B
1	\[(.*)\]\[(.*)\]- \[(.*)ms\]\[(.*)\]\[(.*)\](.*)	//定义正则表达式
2	=file("D:/QQLive.log").read@n()	//打开日志文件，按行将内容读成序列
3	=A2.regex(A1)	//用序列的正则分析函数regex，拆解出字段

硬结构化—多行正则拆分

如下为某软件的监控日志(raq.log)。

```
[2018-05-14 09:20:20]
SEVERE: Load library module.dll failed.

[2018-05-14 09:20:21]
DEBUG: Temporary file directory is:
D:\temp\esProc\nodes\127.0.0.1_8281\temp.
Files in temporary directory will be deleted on
every 12 hours.

[2018-05-14 09:20:21]
INFO: Server is succeed :started.
```

该日志为多行，而且是不定行对应一条记录。

此时需要先确定记录边界，该文本以左中括号开头来判断，然后按记录分成组。将记录合并为串后，再用正则表达式来拆分 。

	A	B
1	=file("D:/raq.log").read@n()	//打开日志文件，按行将内容读成序列
2	=A1.select(~!="")	//过滤掉空行
3	=A2.group@i(like(~,"[*"))	//按记录内容分组
4	\[(.*)\] ([A-Z]+):(.*)	//定义正则表达式
5	=A3.regex(A4)	//执行正则分析

硬结构化—交叉表还原

如下为CSV格式的一张成绩交叉表(scores.csv)。

```
ID,Name,Math,Physics,Chemistry
1,Mike,67,87,72
2,Rose,80,90,84
3,Smith,90,88,76
4,Mary,88,67,77
5,Tod,55,70,87
6,Melody,40,90,55
7,David,90,65,80
8,Snoopy,100,90,85
9,Michale,70,78,55
10,Nikita,66,88,70
```

该文件本身为规整的结构化数据。只是内容为学生的各门功课的成绩交叉表。现在需要将各门功课还原到字段名为Subject和Score的记录。

	A	B
1	=file("D:/scores.csv")	//打开学生成绩表
2	=A1.import@tc()	//导入数据
3	=A1.pivot@r(ID,Name;Subject,Score)	//将各门功课转置到字段名为Subject和Score的记录行
4		

硬结构化—综合

如下为windows下的HPUpdate.exe.log。

```
1,"fusion","GAC",0
1,"WinRT","NotApp",1
3,"System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089","C:\windows\assembly\
NativeImages_v4.0.30319_64\System\01a3608d87251d7ea99
342a88d079c23\System.ni.dll",0
3,"System.Core, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089","C:\windows\assembly\
NativeImages_v4.0.30319_64\System.Core\2a6c39230fef9dfaf
c7ede45f99ec776\System.Core.ni.dll",0
3,"WindowsBase, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35","C:\windows\assembly\
NativeImages_v4.0.30319_64\WindowsBase\996cd1a75c20ce
6e697aad199323707b\WindowsBase.ni.dll",0
3,"PresentationCore, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35","C:\windows\assembly\
NativeImages_v4.0.30319_64\PresentationCore\6228d402fde
bfae866e84fdfe08773bf\PresentationCore.ni.dll",0
```

该日志文件为一行对应一条记录，左边图示用间隔颜色来区分不同的行，对应4列取名为Type，Desc，File，Status。

拆分要点有：

1：不能简单用逗号拆分，要考虑引号的配对分节。

2：Desc包含子字段，且是用分节串来描述。

3：各Type的Desc没对齐，Type为1时没有Version等子字段。

4：Desc不是规则的分节串，其中第一节为Name，不含节值。

硬结构化—综合

拆分步骤：

首先按照逗号，且注意引号配对拆成基础表，并将缺省字段重命名：

序号	Type	Desc	File	Status
1	1	fusion	GAC	0
2	1	WinRT	NotApp	1
3	3	System, Version=4.0.0.0, Cult...	C:\windows\as...	0
4	3	System.Core, Version=4.0.0.0...	C:\windows\as...	0
5	3	WindowsBase, Version=4.0.0...	C:\windows\as...	0
6	3	PresentationCore, Version=4....	C:\windows\as...	0
7	3	PresentationFramework, Vers...	C:\windows\as...	0
8	3	System.Xaml, Version=4.0.0.0...	C:\windows\as...	0
9	3	System.Configuration, Versio...	C:\windows\as...	0
10	3	System.Xml, Version=4.0.0.0, ...	C:\windows\as...	0
11	3	System.Web, Version=4.0.0.0,...	C:\windows\as...	0
12	3	System.Web.Extensions, Vers...	C:\windows\as...	0
13	3	System.Security, Version=4.0....	C:\windows\as...	0
14	3	PresentationFramework.Aero...	C:\windows\as...	0
15	3	WindowsFormsIntegration, V...	C:\windows\as...	0
16	3	System.Drawing, Version=4.0...	C:\windows\as...	0
17	3	System.Windows.Forms, Ver...	C:\windows\as...	0

然后将Desc字段内容拆为name，value键值对的序表

序号	name	value
1	Culture	neutral
2	System	
3	PublicKeyToken	b77a5c561934e089
4	Version	4.0.0.0

硬结构化—综合

拆分步骤：

从Desc的键值对序表中，将value为空的键提出为Name列：

序号	Type	Desc	File	Status	Name
1	1	[[fusion,]]	GAC	0	fusion
2	1	[[WinRT,]]	NotApp	1	WinRT
3	3	[[Culture,ne...	C:\windows\assembly\...	0	System
4	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Core
5	3	[[Culture,ne...	C:\windows\assembly\...	0	WindowsB...
6	3	[[Culture,ne...	C:\windows\assembly\...	0	Presentatio...
7	3	[[Culture,ne...	C:\windows\assembly\...	0	Presentatio...
8	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Xaml
9	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Co...
10	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Xml
11	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Web
12	3	[[Culture,ne...	C:\windows\assembly\...	0	System.We...
13	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Se...
14	3	[[Culture,ne...	C:\windows\assembly\...	0	Presentatio...
15	3	[[Culture,ne...	C:\windows\assembly\...	0	WindowsF...
16	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Dr...
17	3	[[Culture,ne...	C:\windows\assembly\...	0	System.Wi...

然后删掉键值对中的Name行

序号	name	value
1	Culture	neutral
2	System	
3	PublicKeyToken	b77a5c561934e089
4	Version	4.0.0.0

序号	name	value
1	Culture	neutral
2	PublicKeyToken	b77a5c561934e089
3	Version	4.0.0.0

硬结构化—综合

拆分步骤：

将键值对表转置，方便将子表字段合并到主表：

序号	name	value
1	Culture	neutral
2	PublicKeyToken	b77a5c561934e089
3	Version	4.0.0.0

序号	Culture	PublicKeyToken	Version
1	neutral	b77a5c561934e089	4.0.0.0

转置后的子表，合并到主表：

序号	Type	Name	Culture	PublicKeyT...	Version	File	Status
1	1	fusion	(null)	(null)	(null)	GAC	0
2	1	WinRT	(null)	(null)	(null)	NotApp	1
3	3	System	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
4	3	System.Core	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
5	3	WindowsBase	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
6	3	PresentationCore	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
7	3	PresentationFra...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
8	3	System.Xaml	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
9	3	System.Configur...	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
10	3	System.Xml	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
11	3	System.Web	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
12	3	System.Web.Exte...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
13	3	System.Security	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
14	3	PresentationFra...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
15	3	WindowsFormsIn...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
16	3	System.Drawing	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
17	3	System.Windows...	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0

硬结构化—综合

SPL脚本实现：

使用q选项去掉分节串的引号；o表示引号为转义符，否则路径符号的斜杠会被当成缺省转义符；c表示用逗号分隔。

合并到主表时，由于Type为1的行没有键值对，用1选项表示即使键值对为空时，也会连接到主表。否则就会丢失键值对为空的记录。

	A	B
1	=file("D:/HPUpdate.exe.log").import@qoc()	//打开日志并导入文件
2	=A1.rename(_1:Type,_2:Desc,_3:File,_4:Status)	//将缺省字段重命名
3	=A2.run(Desc=Desc.property@c())	//拆分Desc分节串为键值对表
4	=A3.derive(Desc.select(value=="").name:Name)	//从键值对中提出Name列
5	=A4.run(Desc.delete(Desc.pselect(value:"")))	//再删掉键值对中的name
6	=A5.run(Desc=Desc.pivot(;name,value))	//将键值对转置
7	=A6.news@1(Desc;Type,Name,Culture,PublicKeyToken,Version,File,Status)	//合并子表到主表

CONTENTS

文件检索—搜索

类似GREP命令，给定根目录后，搜索当前目录下所有文本文件中包含关键词的行。

定义两个入口参数，path为搜索根目录，key为需要搜索的关键词。

run为循环执行函数，对根目录下的所有文件遍历执行。后面还有一个run则是对文件内容遍历搜索。

output的逻辑很简单，找到后，打印出行内容，行号等信息，这里要注意的是SPL中，整数类型的行号跟字符串拼接时，要用/，不能用+。

	A	B
1	=directory@ps(path+"/*.txt")	//列出搜索目录下的所有文本文件（包含子目录）
2	=A1.run(file(~).read@n().run(if(pos(~,key),output(A1.~/"　　第"/#/"行: "/~))))	//读入每个文件内容，并逐行比较关键词，输出相应信息
3		

文件检索—搜索-游标

当文件很大（内存装不下）时，需要用游标来分块处理。

使用游标时，需要分块处理，所以文件列表的run，用for代码块，方便书写多个语句。

用s选项将数据按行处理，i选项将返回的单字段序表结构转为序列来处理，从而跟上一节的表达式一致。

对游标B2循环取数，每次取数记录数为1000行，直到全部取完。取数过程中会自动执行前面定义的run函数。

	A	B	C
1	=directory@ps(path+"/*.txt")	//列出搜索目录下的所有文本文件（包含子目录）	
2	for A1	=file(A2).cursor@is().run(if(pos(~,key),output(A2/" 第"/#/"行: "/~)))	//构建文件游标
3		for B2,1000	//分块取数
4			

文件检索—替换

给定根目录，替换路径下所有文本文件中的指定文本。

定义三个入口参数，path为根目录，source为需要替换的源串，target为要替换的目标串。

与搜索不同，替换后还需要将内容保存到文件，所以不方便在一个语句中完成整套动作。用for代码块拆写为读入文件，执行替换，再保存到文件。

	A	B	C
1	=directory@ps(path+"/*.txt")	//列出path目录下的所有文本文件（包含子目录）	
2	for A1	=file(A2).read@n()	//循环处理目录下的所有文件
3		=B2.run(~=replace(~,source,target))	//每个文件读入的内容执行替换动作
4		=file(A2).write(B3)	//再将替换后的内容写出到原文件

文件检索—替换-游标

当文件很大（内存装不下）时，需要用游标来分块处理。

采用游标替换时，由于同时进行源文件读取和替换后的写出，所以需要写入到一个新的文件。

循环取出的数据用追加方式写到文件B4。

	A	B	C
1	=directory@ps(path+"/*.txt")	//列出path目录下的所有文本文件（包含子目录）	
2	for A1	=file(A2).cursor@is()	//循环目录下的所有文件并创建游标
3		=B2.run(~=replace(~,source,target))	//对游标定义替换计算
4		=file(filename@d(A2)+"\\"+filename@n(A2)+"_2."+filename@e(A2))	//在源文件同路径下定义新的输出文件
5		=movefile(B4)	//删除同名的新文件，防止累加到同名文件
6		for B3,1000	=B4.write@a(B6)

CONTENTS

单词统计—单词计数

实现WORDCOUNT算法，给定文本文件，统计出文本中每个单词的数量。

首先拆分出文本中的所有单词，然后只需按单词分组，并计数每组的数量即可。

需要定义一个入口参数，filePath为需要统计的带路径文件名。

	A	B
1	=file(filePath).read()	//读入给定文件的文本内容
2	=A1. words()	//将内容拆分为单词序列
3	=A2.groups(lower(~):Word;count(~):Count)	//将单词转为小写后，分组并计数

单词统计—单词计数-游标

当文件很大（内存装不下）时，需要用游标来处理。

仍然是is选项产生只有一列的游标，返回序列。

	A	B
1	=file(filePath).cursor@is()	//创建文件游标
2	=A1.run(~=~.words()).conj()	//定义延迟计算，每行数据拆分为单词序列后，最后合并为大序列
3	=A2.groups(~:Word;count(~):Count)	//对游标数据汇总统计

单词统计—字母计数

实现WORDCOUNT算法，给定文本文件，统计出文本中每个字母的数量。

跟单词拆分类似，只是拆分为字母用到split函数。

	A	B
1	=file(filePath).read()	//读入给定文件的文本内容
2	=A1. split()	//将内容拆分为单词序列
3	=A2.groups(~:Char;count(~):Count)	//对字符分组并计数

单词统计—字母计数-游标

当文件很大（内存装不下）时，需要用游标来处理。

仍然是is选项产生只有一列的游标，返回序列。

	A	B
1	=file(filePath).cursor@is()	//创建文件游标
2	=A1.run(~=~.split()).conj()	//定义延迟计算，每行数据拆分为字母序列后，最后合并为大序列
3	=A2.groups(~:Char;count(~):Count)	//对游标数据汇总统计

CONTENTS

按行去重—文本

如下为某同学收集的一些常用网络地址(urls.txt)，需要整理去掉重复的网址。

```
https://123.sogou.com/
https://www.sogou.com/
https://stackoverflow.com/
https://123.sogou.com/
http://www.raqsoft.com.cn/
https://www.baidu.com/
https://www.sogou.com/
https://123.sogou.com/
https://stackoverflow.com/
http://www.raqsoft.com.cn/
```

按行读取文件内容，然后按行分组，重复的行只取首行。

	A	B
1	=file("d:/urls.txt"))	//打开指定文件
2	=A1.read@n()	//按行读取文件内容为序列
3	=A2.group@1()	//按序列成员分组
4	=A1.write(A3)	//获得所有行号序列

使用1选项，重复的行只返回首行，便得到了去重后的内容。

按行去重—文本-游标

当文件很大（内存装不下）时，需要用游标来分块处理。

跟上一节稍有区别，用游标处理分组时，必须指定分组的参数，且返回值都是序表，所以宜用序表来计算。

只用s选项产生只有一列的游标，没有i选项，取数时返回缺省列名的序表。

注意跟上一节的区别，上一节的取数是序列，用write写出。本节的取数是序表，用export导出。a选项仍然表示追加方式。

	A	B
1	d:/urls.txt	//指定文件路径
2	=file(A1).cursor@s()	//打开文件并构造按行内容的游标
3	=A2.groupx(_1)	//按缺省字段名_1分组
4	=file(filename@d(A1)+"\\"+filename@n(A1)+"_2."+filename@e(A1))	//构造同路径下输出文件
5	for A3,1000	=A4.export@a(A5)

按行去重—文章

从网上帖子复制下来的小说(novel.txt)，时不时有重复的段落。

按文章去重，不同于文本，去重后的内容不能打乱。给每一行添加行号Row，用于分组后，再按Row恢复原序。

在导入的序表后增加计算列Row，值为行号。

导出时，只导出包含内容的_1字段，缺省会导出所有字段。

	A	B
1	=file("d:/novel.txt"))	//打开指定文件
2	=A1.import@s()	//按行导入所有内容
3	=A2.derive(#:Row)	//增加行号计算列
4	=A3.group@1(_1)	//按缺省列名_1分组
5	=A4.sort(Row)	//重新按行号排序
6	=A1.export(A5,_1)	//将排序后的内容的导出

按行去重—文章-游标

当文件很大（内存装不下）时，需要用游标来分块处理。

注意序号表达式为seq(),跟序表中的#不同。#为序表中的顺序号；游标中会分块取数，后续取的块#又是从1开始，所以此时要用函数seq()来获取正确的连续序号。

获取每一组的唯一行时，注意区别：
1：序列用group@1()，有1选项，可以没参数。
2：序表用group@1(_1)，有1选项，有字段参数。
3：序表的游标用groupx(_1;min(Row):Row)，没选项，有分组字段，有输出行聚合函数。

	A	B
1	d:/novel.txt	//指定文件路径
2	=file(A1).cursor@s()	//打开文件并构造按行内容的游标
3	=A2.derive(seq():Row)	//增加行号计算列
4	=A3.groupx(_1;min(Row):Row)	//根据字段_1分组，保留重复行中的最小行号
5	=A4.sortx(Row)	//重新按行号排序
6	=file(filename@d(A1)+"\\"+filename@n(A1)+"_2."+filename@e(A1))	//构造同路径下输出文件
7	for A5,1000	=A6.export@a(A7,_1)

THANKS

感谢观看