SPL Base

Unstructured text computing

CONTENTS

1 Hard structured

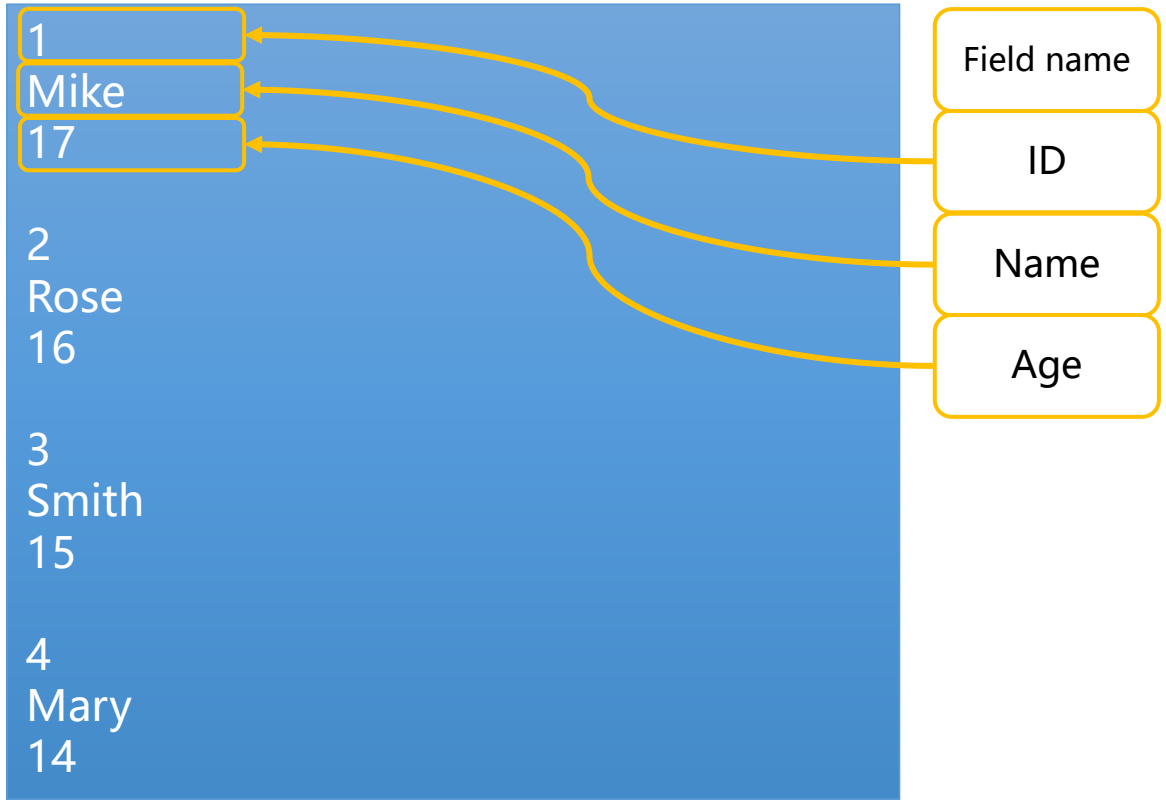
2 File retrieval

3 Word statistics

4 Deduplicate by row

Hard structured - fixed rows

The following is student information (student.txt): a record for every 3 lines, separated by a blank line.



For the text with fixed row structure, just read it as a sequence, remove the redundant empty rows, and then fill in the table structure one by one.

	A	B
1	=file("D:\\student.txt")	//Open the file
2	=A1.read@n()	//Read data into sequence by line
3	=A2.select(~!="")	//Remove blank lines between records
4	=create(ID,Name,Age)	// Create table structure
5	=A4.record(A3)	//Fill the A3 sequence into the table structure

By default, the whole text will be read as a large text string. The n option is used here to read the text as a sequence by line, with one member for each line.

Hard structured - fixed rows - cursor

When the file is large (can’ t be loaded into memory), it needs to be processed in blocks with cursor.

The file cannot be read into memory at one time. Instead, you need to create the file as a cursor and attach the cursor's delay function (select) to process the data block by block.

By default, the cursor processes the data as a sequence table with structure, but the text has only one column. Here, i option is used to convert the sequence table with only one column to a sequence, which is convenient for subsequent calculation of the sequence.

Note the difference between the cursor data and the row read in the last section. The cursor always processes the data according to the structure, and automatically resolves the data to the corresponding data type, so the empty row is resolved to null data.

The data is processed in loop and blocks. It should be noted that the block of each fetching must be a multiple of 3, because the current table structure is 3 columns, otherwise the data filling will be misplaced.

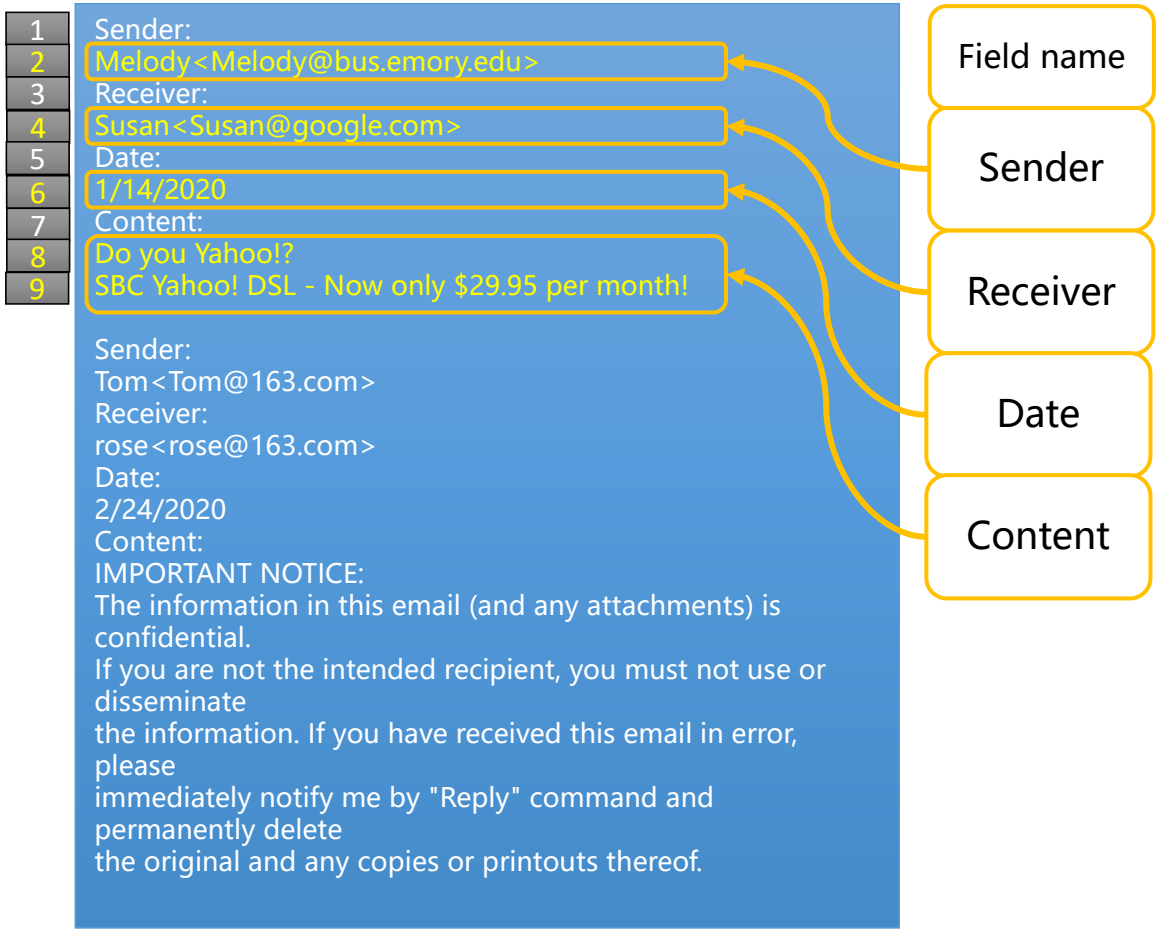
After the structured data is calculated, pay attention to reset to prevent memory overflow.

	A	B
1	=file("D:\\student.txt")	//Open the file
2	=A1.cursor@i()	//Create file cursor
3	=A2.select(~!=null)	//Remove blank lines between records
4	=create(ID,Name,Age)	//Create table structure
5	for A3,300	>A4.reset()
6		=A4.record(A5)

Hard structured - indefinite rows

Here are some email messages（mail.txt）, the number of lines occupied by the message content is variable.

For indefinite row text, after importing it into the sequence, you need to merge indefinite rows into one row, and then fill in the table structure one by one.



	A	B
1	=file("D:\\mail.txt")	//Open the file
2	=A1.read@n().select(~!="")	//Import sequence and remove blank lines
3	=A2.group@i(~=="Sender:")	//Each line starting with Sender: and subsequent lines is a group
4	=A3.new(~(2):Sender,~(4):Receiver,~(6):Date,**~.to(8,).string()**:Content)	//Extract record value, merge body, create new structure table
5		

~.to(8,) means starting from line 8 of the current group, and omitting parameters after comma means taking all subsequent lines. .string() merges the current row sequence into one row.

Hard structured - indefinite rows - cursor

When the file is large (can’ t be loaded into memory), it needs to be processed in blocks with cursor.

Also note that after returning the sequence with the i option, null values should be used when blank rows are removed.

When fetching data for structured table, pay attention to fetching in blocks to prevent memory overflow.

	A	B
1	=file("D:\\mail.txt")	//Open the file
2	=A1.cursor@i().select(~!=null)	//Create cursor sequence and remove empty lines
3	=A2.group@i(~=="Sender:")	//Group by record
4	=A3.new(~(2):Sender,~(4):Receiver,~(6):Date,~.to(8,).string():Content)	//Extract record value, merge body, create new structure table
5	=A4.fetch()	//Cursor fetches all data

Hard structured — Single row regular split

The following is the startup log of a video software （ 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
```

The log is one line corresponding to one record. The example on the left automatically turns the line. The yellow part of the figure shows one line of data (all the following lines are the same).

It is impossible to use simple separator to split, and the content includes redundant brackets ([]), minus sign (-), character (ms), etc.

In this case, regular expressions can be used to split.

	A	B
1	\ [(. *) \ ] \ [(. *) \ ] - \[(.*)ms\]\[(.*)\]\[(.*)\](.*)	//Define regular expression
2	=file("D:/QQLive.log").read@n()	//Open the log file, and read the contents into sequence by line
3	=A2.regex(A1)	//Using regex, the regular analysis function of sequence, the field is split

Hard structured — Multiple rows regular split

The following is the monitoring log of a software （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.
```

The log is multiple lines, and the indefinite lines correspond to one record.

The record boundary needs to be determined first, and it is judged by starting with the left bracket in the text, and then divided into groups according to the records. After merging records into strings, regular expressions are used to split them.

	A	B
1	=file("D:/raq.log").read@n()	//Open the log file, and read the contents into sequence by line
2	=A1.select(~!="")	//Filter out blank lines
3	=A2.group@i(like(~,"[*"))	//Group by record content
4	\[(.*)\] ([A-Z]+):(.*)	//Define regular expression
5	=A3.regex(A4)	//Perform regular analysis

Hard structured — Crosstab restore

The following is a score cross table in CSV format （ 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
```

The file itself is structured data. It's just a cross table of students' scores in each subject. Now you need to restore each subject to a record with the field names subject and score.

	A	B
1	=file("D:/scores.csv")	//Open the file
2	=A1.import@tc()	//Import data
3	=A1.pivot@r(ID,Name;Subject,Score)	//Transpose each course to the record row with field names subject and score
4		

Hard structured — Comprehensive

The following is HPUpdate.exe.log for windows.

```
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
```

The log file is a line corresponding to a record. The left figure uses the interval color to distinguish different lines. The corresponding 4 columns are named type, desc, file and status.

The main points of splitting are as follows:

1： It can't be simply separated by commas. Consider the pairing of quotation marks.

2：Desc contains subfields and is described as a segmented string.

3：Desc of each type is not aligned. There is no subfield such as version when the type is 1.

4：Desc is not a regular segmented string, where the first section is name without section value.

Hard structured — Comprehensive

Split steps:

First, follow the comma, and note that the quotation mark pairs are split into the basic table, and rename the default field:

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

Then, the desc field contents are divided into the sequence table of name and value key pairs.

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

Hard structured — Comprehensive

Split steps:

From the sequence table of Desc, extract the key with empty value as the name column:

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...

Then delete the name line in the key pair.

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

Hard structured — Comprehensive

Split steps:

Transpose the key pair table to facilitate merging the sub table fields into the main table:

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

The transposed sub tables are merged into the main table:

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

Hard structured — Comprehensive

SPL script:

Use the Q option to remove the quotation mark of the segmented string; O means that the quotation mark is an escape character, otherwise the slash of the path symbol will be regarded as the default escape character; C means that it is separated by commas.

When merging to the main table, since the row with type 1 has no key pair, the 1 option indicates that even if the key pair is empty, it will be joined to the main table. Otherwise, records with null key pairs will be lost.

	A	B
1	=file("D:/HPUpdate.exe.log").import@qoc()	//Open log and import file
2	=A1.rename(_1:Type,_2:Desc,_3:File,_4:Status)	//Rename the default field
3	=A2.run(Desc=Desc.property@c())	//Split desc into key pairs
4	=A3.derive(Desc.select(value=="").name:Name)	//Extract name column from key pairs
5	=A4.run(Desc.delete(Desc.pselect(value:"")))	//Delete the name in the key pair
6	=A5.run(Desc=Desc.pivot(;name,value))	//Transpose the key pair table
7	=A6.news@1(Desc;Type,Name,Culture,PublicKeyToken,Version,File,Status)	//Merge sub table to main table

CONTENTS

1 Hard structured

2 File retrieval

3 Word statistics

4 Deduplicate by row

File retrieval - Search

Similar to the grep command, after the root directory is given, search the lines containing keyword in all text files in the current directory.

Two entry parameters are defined. Path is the search root directory and key is the keyword to be searched.

Run is a loop execution function, which traverses and executes all files under the root directory. The next run is to traverse and search the file content.

The logic of output is very simple. After finding it, print the content of the line and line number. Note that in SPL, when the line number of integer type is concatenated with string, use /, not +.

	A	B
1	=directory@ps(path+"/*.txt")	//Lists all text files in the search directory(including subdirectories)
2	=A1.run(file(~).read@n().run(if(pos(~,key),output(A1.~/" No"/#/ "Row: "/~))))	//Read in the contents of each file, compare keyword line by line, and output corresponding information
3		

File retrieval – Search - Cursor

When the file is large (can’ t be loaded into memory), it needs to be processed in blocks with cursor.

When using cursors, they need to be processed in blocks. The for code block is used here, and it is convenient to write multiple statements.

Use the s option to process the data by rows, and the i option to convert the returned single field sequence table to sequence, so as to make the expression consistent with the last section.

Fetching data in loop for cursor B2, the number of fetched records is 1000 lines each time until the data is all fetched. The run function defined earlier will be executed automatically during the fetching process.

	A	B	C
1	=directory@ps(path+"/*.txt")	//Lists all text files in the search directory(including subdirectories)	
2	for A1	=file(A2).cursor@is().run(if(pos(~,key),output(A2/"No"/#/ “Row: "/~)))	//Create file cursor
3		for B2,1000	//Fetch data in blocks
4			

File retrieval - Replace

Given the root directory, replace the specified text in all text files under the path.

Unlike search, you need to save the content to a file after replacement, so it is not convenient to complete the whole action in one statement. Use the for code block to disassemble the action to read in file, execute replacement, and then save to file.

Three entry parameters are defined. Path is the root directory, source is the source string to be replaced, and target is the target string to be replaced.

	A	B	C
1	=directory@ps(path+"/*.txt")	//List all text files in the path directory (including subdirectories)	
2	for A1	=file(A2).read@n()	//Loop through all files in the directory
3		=B2.run(~=replace(~,source,target))	//Replace the content in each file
4		=file(A2).write(B3)	//Write out the replaced content to the original file

File retrieval – Replace - Cursor

When the file is large (can’ t be loaded into memory), it needs to be processed in blocks with cursor.

When cursor replacement is used, because the reading of source file and writing after replacement is carried out at the same time, we need to write to a new file.

The data retrieved in loop is written to the file B4 by appending.

	A	B	C
1	=directory@ps(path+"/*.txt")	//List all text files in the path directory(including subdirectories)	
2	for A1	=file(A2).cursor@is()	//Loop through all files in the directory and create cursor
3		=B2.run(~=replace(~,source,target))	//Define replacement calculation for cursor
4		=file(filename@d(A2)+"\\"+filename@n(A2)+"_2."+filename@e(A2))	//Define a new output file under the same path as the source file
5		=movefile(B4)	//Delete a new file with the same name to prevent adding to file with the same name
6		for B3,1000	=B4.write@a(B6)

CONTENTS

1 Hard structured

2 File retrieval

3 Word statistics

4 Deduplicate by row

Word statistics - Word count

Implement the wordcount algorithm. Given the text file, count the number of each word in the text.

First, all the words in the text are split. Then, just group by words and count the number of each group.

An entry parameter needs to be defined, and filepath is the file name with path that needs to be counted.

	A	B
1	=file(filePath).read()	//Read in the text content of the given file
2	=A1. words()	//Split content into word sequence
3	=A2.groups(lower(~):Word;count(~):Count)	//After transferring words to lowercase, group and count

Word statistics - Word count - Cursor

When the file is large (can’ t be loaded into memory), it needs to be processed with a cursor.

It is still the is option that produces a cursor with only one column and returns the sequence.

	A	B
1	=file(filePath).cursor@is()	//Create file cursor
2	=A1.run(~=~.words()).conj()	//Define delayed calculation. After each row of data is split into word sequence, it is finally merged into large sequence.
3	=A2.groups(~:Word;count(~):Count)	//Summary statistics of cursor data

Word statistics - letter count

Implement the wordcount algorithm. Given the text file, count the number of each letter in the text.

It's similar to word splitting, except that split function is used to split letters.

	A	B
1	=file(filePath).read()	//Read in the text content of the given file
2	=A1. split()	//Split content into word sequence
3	=A2.groups(~:Char;count(~):Count)	//Group and count characters

Word statistics - letter count - Cursor

When the file is large (can’ t be loaded into memory), it needs to be processed with a cursor.

It is still the is option that produces a cursor with only one column and returns the sequence.

	A	B
1	=file(filePath).cursor@is()	//Create file cursor
2	=A1.run(~=~.split()).conj()	//Define delayed calculation. After each row of data is split into letter sequence, it is finally merged into large sequence.
3	=A2.groups(~:Char;count(~):Count)	//Summary statistics of cursor data

CONTENTS

1 Hard structured

2 File retrieval

3 Word statistics

4 Deduplicate by row

Deduplicate by row - text

Here are some common network addresses collected by a student（urls.txt）. Need to sort out duplicate URLs.

```
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/
```

Read the contents of the file by row, and then group them by row. Only the first row is taken for the duplicate rows.

	A	B
1	=file("d:/urls.txt"))	//Open the specified file
2	=A1.read@n()	//Read file content by row as sequence
3	=A2.group@1()	//Group by sequence member
4	=A1.write(A3)	//Get all row number sequence

Using the 1 option, the repeated rows only returns the first row, and the content after deduplication is obtained.

Deduplicate by row - text - Cursor

When the file is large (can’ t be loaded into memory), it needs to be processed in blocks with cursor.

There is a slight difference compared with the last section. When using cursor to handle grouping, you must specify the parameter of the grouping, and the return values are all sequence tables. Therefore, sequence tables should be used for calculation.

Only the s option is used to generate a cursor with only one column. Without the I option, when fetching, the default column name sequence table is returned.

Pay attention to the difference with the last section. The last section is a sequence, which is written out with write. The fetching in this section is a sequence table, which is exported with export. The a option still indicates the append method.

	A	B
1	d:/urls.txt	//Specify the path to a file
2	=file(A1).cursor@s()	//Open the file and create a cursor by row
3	=A2.groupx(_1)	//Group by default field name _ 1
4	=file(filename@d(A1)+"\\"+filename@n(A1)+"_2."+filename@e(A1))	//Construct output file under the same path
5	for A3,1000	=A4.export@a(A5)

Deduplicate by row - article

Novels copied from online posts（ novel.txt ） often have repeated paragraph.

Deduplication for an article is different from the text, and the content after the deduplication can not be disrupted. Add row number to each row, which is used to restore the original order after grouping.

Add calculation column(Row) to the imported sequence table, the value of which is row number.

Only the_ 1 field is exported. All fields will be exported by default.

	A	B
1	=file("d:/novel.txt"))	//Open the specified file
2	=A1.import@s()	//Import content by row
3	=A2.derive(#:Row)	//Add row number calculation column
4	=A3.group@1(_1)	//Group by default column name _ 1
5	=A4.sort(Row)	//Sort by row number
6	=A1.export(A5,_1)	//Export sorted content

Deduplicate by row - article - Cursor

When the file is large (can’ t be loaded into memory), it needs to be processed in blocks with cursor.

Note that the sequence number expression is seq(), which is different from the # in the sequence table. # is the sequence number in the sequence table; the cursor will fetch data by blocks, and the # of subsequent blocks will start from 1, so you need to use the function seq() to get the correct consecutive sequence number.

When obtaining the unique row of each group, pay attention to the difference:
1：With sequence, group@1() is used. With 1 option, can have no parameter
2：With sequence table, group@1(_1) is used. With 1 option， and with filed parameter.
3：With sequence table cursor, groupx(_1;min(Row):Row) is used. Without option, with grouping filed, and with output row aggregation function.

	A	B
1	d:/novel.txt	//Specify the path to a file
2	=file(A1).cursor@s()	//Open the file and create a cursor by row
3	=A2.derive(seq():Row)	//Add row number calculation column
4	=A3.groupx(_1;min(Row):Row)	//Group by filed _1, keep the minimum row number in duplicate rows
5	=A4.sortx(Row)	//Sort by row number
6	=file(filename@d(A1)+"\\"+filename@n(A1)+"_2."+filename@e(A1))	//Construct output file under the same path
7	for A5,1000	=A6.export@a(A7,_1)

THANKS