SPL Base

esProc Tutorial

Set operations

目录

CONTENTS

01 When members are basic data types

1. Concatenation
2. Intersection
3. Union
4. Difference
5. XOR
6. Operations on more than two sets

02 When members are records

1. Reference eligible records directly
2. Merge sets by certain fields
3. Merge sets by the primary key
4. Merge sets by all fields
5. When records are note ordered by the key

03 Big data operations

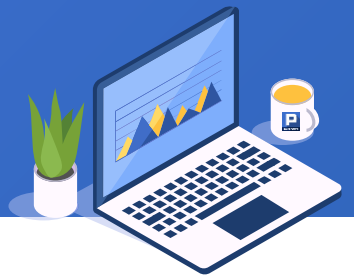
1. Concatenation of sets
2. Merge sets by column values

CONTENTS

1. Concatenation
2. Intersection
3. Union
4. Difference
5. XOR
6. Operations on more than two sets

When members are basic data types

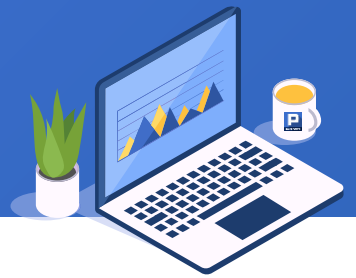
1. Concatenation



S2014.txt and *S2015.txt* store sale records in 2014 and 2015 respectively. They are of same structure. **Task**: Find how many times each customer order a product during the two years.

ID	Customer	Date	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
…	…	…	…

1. Concatenation

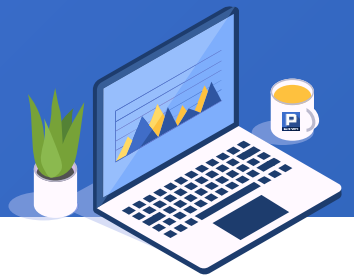


The SPL script uses "|" to calculate the concatenation:

	A	B
1	=file("S2014.txt").import@t(Customer)	/Import customers of 2014
2	=file("S2015.txt").import@t(Customer)	/Import customers of 2015
3	=A1 | A2	/Use "|" to concatenate customers, including the duplicate ones, of the two years.
4	=A3.groups(Customer; count(~):Count)	/Count the times each customer order a product

A4	Product	Count
	ANATR	5
	ANTON	6
	…	…

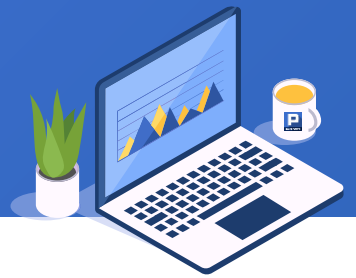
2. Intersection



Task: Find the students who enroll in both the painting class and dancing class. The table structure is as follows:

ID	StudentID	Subject
1	2	Painting
2	4	Dance
3	3	Robot
4	2	Dance
5	5	Writing
…	…	…

2. Intersection

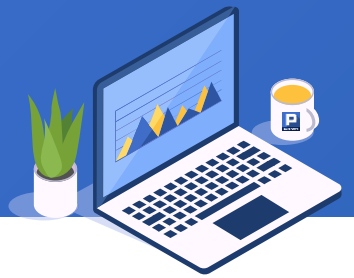


The SPL script uses “^” to get the intersection:

	A	B
1	=file("Interest.txt").import@t()	/Import the text file
2	=A1.select(Subject:"Painting")	/Get records of painting
3	=A1.select(Subject:"Dance")	/Get records of dancing
4	=A2.(StudentID) ^ A3.(StudentID)	/Use “^” to get intersection of students who are going to learn painting and dancing

A4	Member
	2
	8
	11
	…

3. Union

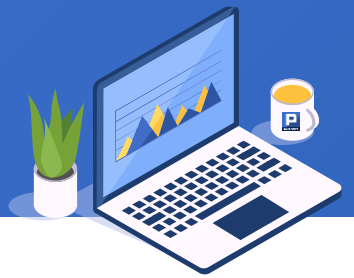


Task: Get records of students who enroll in painting and dancing.

The table structure is as follows:

ID	StudentID	Subject
1	2	Painting
2	4	Dance
3	3	Robot
4	2	Dance
5	5	Writing
…	…	…

3. Union

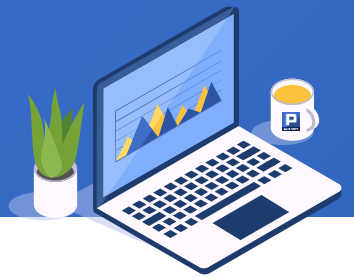


The SPL script uses “&” to get union:

	A	B
1	=file("Interest.txt").import@t()	/Import the text file
2	=A1.select(Subject:"Painting")	/Get records of painting
3	=A1.select(Subject:"Dance")	/Get records of dancing
4	=A2.(StudentID) & A3.(StudentID)	/Use “&” to get students who enroll in painting and dancing

A4	Member
	2
	4
	8
	…

4. Difference

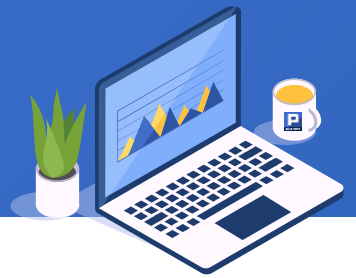


Task: Find the new customers in 2014 according to *Sales* table and *Customer* table, that is, the customers that are not included in the *Customer* table.

Sales
ID
Customer
OrderDate
…

Customer
ID
Name
City
…

4. Difference



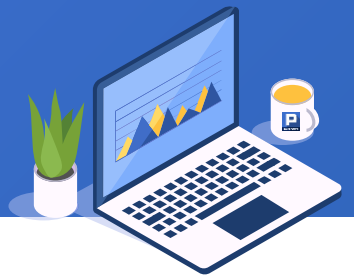
The SPL script uses “\” to get difference:

	A	B
1	=connect("db")	/Connect to the database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Get sales records of 2014
3	=A1.query("select * from Customer")	/Get records from *Customer* table
4	=A2.id(Customer)	/Use **id** function to remove duplicate sales records to get a sequence of unique customers
5	=A3.(ID)	/Get the sequence of customer IDs from *Customer* table
A6		/Use “\” to get the difference

Members
DOS
HUN
URL

Note: This example is for explaining how to perform a difference operation. Actually it’s more convenient to get same result using A.switch@d()/A.join@d(), which perform a join and filtering.

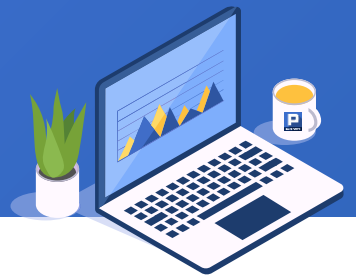
5. XOR



Student scores are stored in different files by semesters. Task: find the student IDs whose total scores rank in top 10 only once in both the first and second semesters.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
…	…	…	…

5. XOR



The SPL script uses “%” to get XOR.

	A	B
1	=file("Scores1.csv").import@ct()	/Import scores of the first semester
2	=file("Scores2.csv").import@ct()	/Import scores of the first semester
3	=A1.groups(STUDENTID; sum(SCORE):Score)	/Group by students and sum their total scores in the first semester
4	=A2.groups(STUDENTID; sum(SCORE):Score)	/Group by students and sum their total scores in the second semester
5	=A3.top(-10;Score).(STUDENTID)	/Get student IDs whose total scores rank in top 10 in the first semester
6	=A4.top(-10;Score).(STUDENTID)	/Get student IDs whose total scores rank in top 10 in the second semester
7	=A5%A6	/Get unique student IDs from A5 and A6

A5

Member
2
9
4
10
…

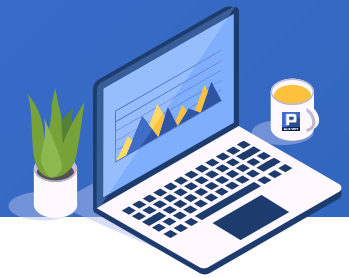
A6

Member
12
1
8
4
…

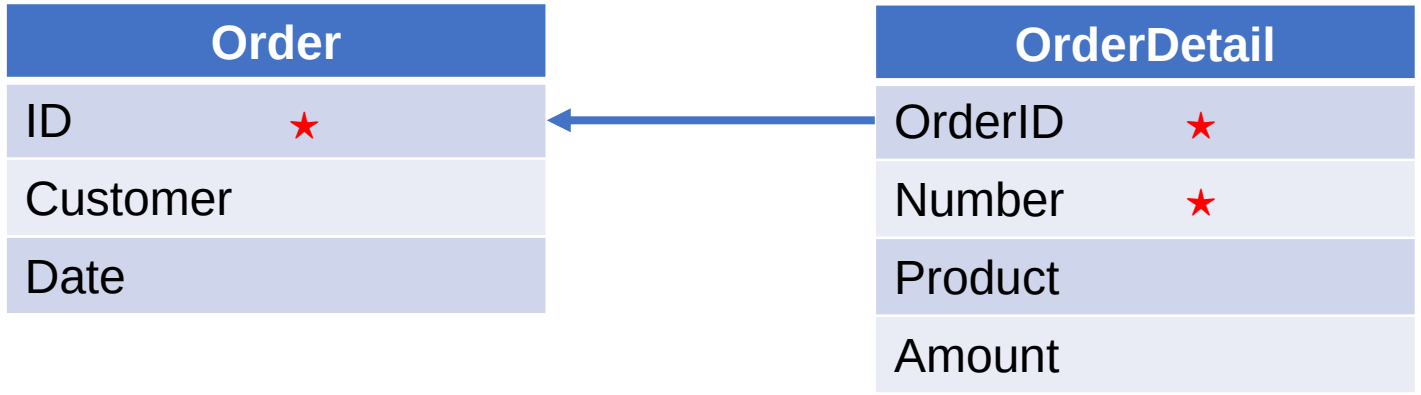
A7

Member
2
9
10
7
…

6. Operation on more than two sets: Concatenation



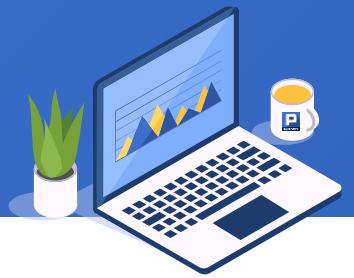
The relationship of Order table and *OrderDetail* table are that of main table and subtable. Each Order record corresponds to multiple OrderDetail records.



The OrderDetail records vary in length. **Task**: to get the following table:

ID	Customer	Date	Product1	Amount1	Product2	Amount2	Product3	Amount3
1	3	20190101	Apple	5	Milk	3	Salt	1
2	5	20190102	Beef	2	Pork	4		
3	2	20190102	Pizza	3				

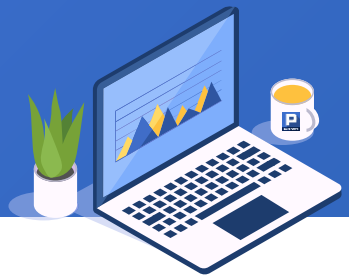
6. Operation on more than two sets: Concatenation



The SPL script uses A.conj() function to concatenate members of sets:

	A	B
1	=connect("db")	/Connect to the database
2	=A1.query("select * from OrderDetail left join Order on Order.ID=OrderDetail.OrderID")	/Import the two tables and left join *Order* table by order IDs
3	=A2.group(ID)	/Group retrieved records by order ID
4	=A3.max(~.count()).("Product"+string(~)+","+"Amount"+string(~)).concat@c()	/Get the group having the most members and define the data structure for the result table
5	=create(ID,Customer,Date,${A4})	/Create a table sequence according to the defined data structure
6	>A3.run(A5.record([ID,Customer,Date]|~.([Product,Amount]).conj()))	/Loop through the groups to piece members together into a sequence and concatenate Product and Amount from these groups using conj() function, and then insert the complete records into A5’ s table sequence

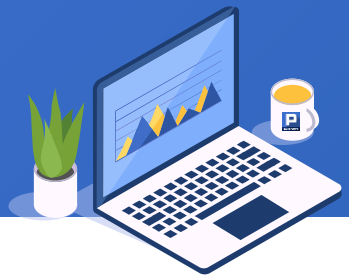
6. Operation on more than two sets: Concatenation



Below is JSON data recording the number of confirmed cases worldwide at a specific time point. **Task**: Calculate the total confirmed cases worldwide.

```
[
{Region:"China",Confirmed:[
        {Region:"Hubei",Confirmed:[
                {Region:"Wuhan",Confirmed:51986},
                {Region:"Xiaogan",Confirmed:3009},
                {Region:"Huanggang",Confirmed:3791},
                ...]
        },
        {Region:"Taiwan",Confirmed:18},
        ...]
},
{Region:"Thailand",Confirmed:33},
...]
```

6. Operation on more than two sets: Concatenation



The SPL script uses A.conj@r() function to concatenate members of sequences recursively:

	A	B
1	=json(file("COVID-19.json").read())	/Read in the JSON data
2	=A1.field@r("Confirmed")	/Use A.field@r() to get all Confirmed fields recursively
3	=A2.conj@r()	/Use A.conj@r() to perform recursive concatenation
4	=A3.sum()	/Sum the number of confirmed cases

A2

Member
[[51986,3009,3791],[58,…]
251
33
28
19
16
…

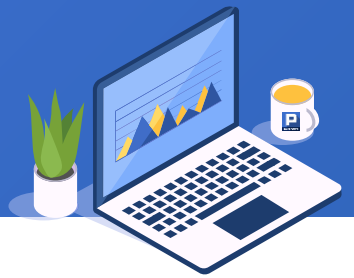
A3

Member
51986
3009
3791
1447
1206
1125
…

A4

Member
64438

6. Operation on more than two sets: Union & Difference

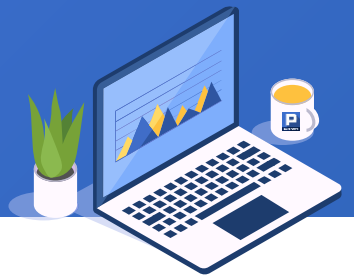


Here *are Course* table and *SelectCourse* table. The selected courses can be multiple that separated by comma. Task: Find courses that are not selected by any students.

Course		
ID	NAME	TEACHERID
1	Environmental protection and …	5
2	Mental health of College Students	1
3	Computer language Matlab	8
4	Electromechanical basic practice	7
5	Introduction to modern life science	3
6	Modern wireless communication system	14
…	…	…

SelectCourse		
ID	STUDENTID	COURSE
1	59	2,7
2	43	1,8
3	52	2,7,10
4	44	1,10
5	37	5,6
6	57	3
…	…	…

6. Operation on more than two sets: Union & Difference



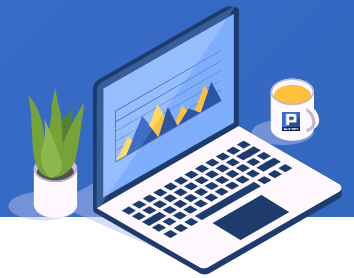
The SPL script uses A.union() function to get the union of sequences whose members are sequences, and A.diff() function to get their difference:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course")	/Query the *Course* table
3	=A1.query("select * from SelectCourse")	/Query the *SelectCourse* table
4	=A3.union(COURSE.split@cp())	/Split selected courses in *SelectCourse* table by comma and get union of *Course* records using union() function
5	=A2.(ID)	/Get course IDs from the *Course* table
6	=A2(A5.pos([A5,A4].diff()))	/Get difference of course IDs in the two tables, the courses that no students select, find their positions in A5 and get them from A2

A6

ID	NAME	TEACHERID
1	Fundamentals of economic management	21

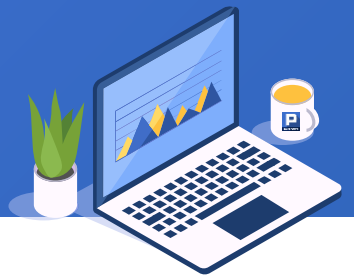
6. Operation on more than two sets: Intersection



Below is part of the *sales* table. **Task**: Find the customers whose order amounts rank top 20 in each month of 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

6. Operation on more than two sets: Intersection

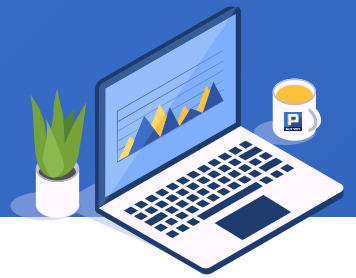


The SPL script uses A.isect() function to get intersection of the sets:

	A	B
1	=connect("db").query("select * from sales")	/Connect to data source to query the *sales* table
2	=A1.select(year(OrderDate)==2014)	/Select records of 2014
3	=A2.group(month(OrderDate))	/Use group() function to group records of 2014 by month
4	=A3.(~.group(Customer))	/Group the groups by Customer
5	=A4.(~.top(-20;sum(Amount)))	/Loop through records of each month to find customers whose order amounts rank top 20
6	=A5.(~.(Customer))	/List the eligible customers
7	=A6.isect()	/Get intersection of groups using isect() function

A7	Member
	HANAR
	SAVEA

6. Operation on more than two sets: Intersection



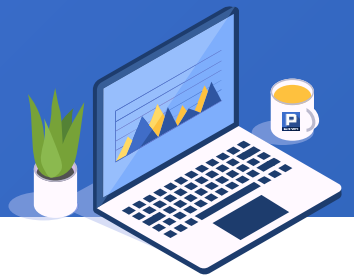
You can use A.isect(x) function to get intersection of sets whose members are calculated with expression x.

	A	B
1	=connect("db").query("select * from sales")	/Connect to data source to query the *sales* table
2	=A1.select(year(OrderDate)==2014)	/Select records of 2014
3	=A2.group(month(OrderDate))	/Use group() function to group records of 2014 by month
4	=A3.(~.group(Customer))	/Group the groups by Customer
5	=A4.(~.top(-20;sum(Amount)))	/Loop through records of each month to find customers whose order amounts rank top 20
6	=A5.isect(~.(Customer))	/Get the eligible customers from each group and calculate intersection of them using isect() function

A6

Member
HANAR
SAVEA

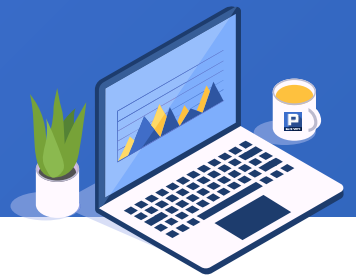
6. Operation on more than two sets: XOR



Task: Find customers whose monthly order amounts rank top 3 only once in 2014 according to the following *Sales* table:

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

6. Operation on more than two sets: XOR



The SPL script uses A.xunion() function to union unique members of sequences in a bigger sequence:

	A	B
1	=file("Sales.csv").import@ct()	/Import *Sales* file
2	=A1.select(year(OrderDate)==2014).group(month(OrderDate))	/Get records of 2014 and group them by month
3	=A2.(~.groups(Customer; sum(Amount):Amount))	/Group each group by Customer and sum each customer’ s total order amount
4	=A3.(~.top(-3;Amount).(Customer))	/Get customers whose order amount rank top 3 per month
5	=A4.xunion()	/Use xunion() function to get customers appearing only once per month

A5

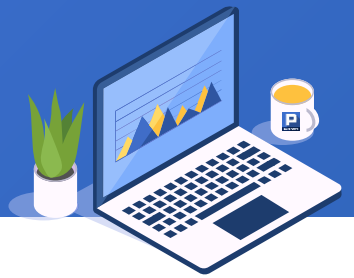
Member
KOENE
HANAR
RATTC
BOTTM
…

CONTENTS

1. Reference eligible records directly
2. Merge sets by certain fields
3. Merge sets by the primary key
4. Merge sets by all fields
5. When records are not ordered by the key

When members are records

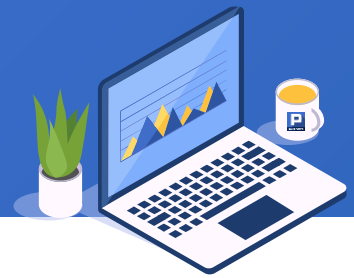
1. Reference eligible records directly



Task: Below is part of the Sales table. Find the records of 2014 where the single amounts rank top 3 per month.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

1. Reference eligible records directly

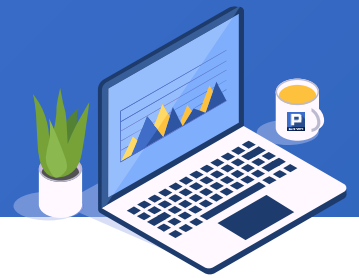


The SPL script uses A.conj() function to concatenate records from table sequences into one table sequence:

	A	B
1	=connect("db")	/Connect to data source
2	=A1.query("select * from Sales")	/Query *Sale*s table
3	=A2.select(year(OrderDate)==2014)	/Get records of 2014
4	=A3.groups(month(OrderDate):Month; top(-3;Amount):Top3)	/Group records by month and get records where the order amounts rank top 3 per month
5	=A4.conj(Top3)	/Use conj() to concatenate eligible records into a table sequence and return it

A5	OrderID	Customer	SellerId	OrderDate	Amount
	10424	MEREP	7	2014/01/23	11493.2
	10417	SIMOB	4	2014/01/16	11283.2
	10430	ERNSH	4	2014/01/30	5796.0
	…	…	…	…	…

1. Reference eligible records directly

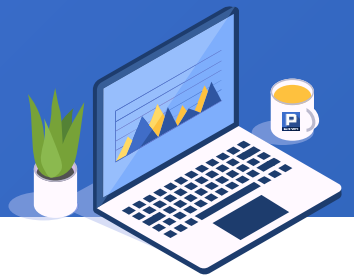


You can also use A.merge@o() to concatenate records from table sequence into one table sequence. It works out same result as A.conj() when @u/@i/@d/@x options are absent.

	A	B
1	=connect("db")	/Connect to data source
2	=A1.query("select * from Sales")	/Query *Sale*s table
3	=A2.select(year(OrderDate)==2014)	/Get records of 2014
4	=A3.groups(month(OrderDate):Month; top(-3;Amount):Top3)	/Group records by month and get records where the order amounts rank top 3 per month
5	=A4.merge@o(Top3)	//Use A.merge@o() to concatenate eligible records into a table sequence and return it

A5	OrderID	Customer	SellerId	OrderDate	Amount
	10424	MEREP	7	2014/01/23	11493.2
	10417	SIMOB	4	2014/01/16	11283.2
	10430	ERNSH	4	2014/01/30	5796.0
	…	…	…	…	…

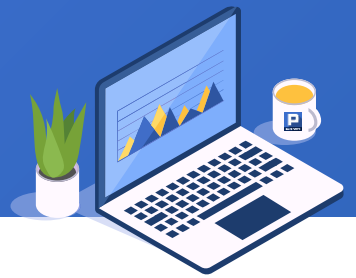
1. Reference eligible records directly



A company is planning a training session for employees younger than 30 and those have been on board less than 3 years. **Task**: Find the records of those employees according to the following *Employee* table.

ID	NAME	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	1974/11/20	2005/03/11	R&D
2	Ashley	1980/07/19	2008/03/16	Finance
3	Rachel	1970/12/17	2010/12/01	Sales
4	Emily	1985/03/07	2006/08/15	HR
…	…	…	…	…

1. Reference eligible records directly



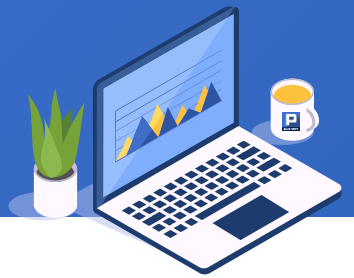
The SPL script uses A.union() function to get union of eligible records from different table sequences and return a record sequence:

	A	B
1	=connect("db")	/Connect to data source
2	=A1.query("select * from Employee")	/Query Employee table
3	=A2.select(age(BIRTHDAY) < 30)	/Get employees younger than 30
4	=A2.select(age(HIREDATE) < 3)	/Get employees who have been in less than 3 years
5	=[A3,A4].union()	/union() unions eligible records and return them as a table sequence

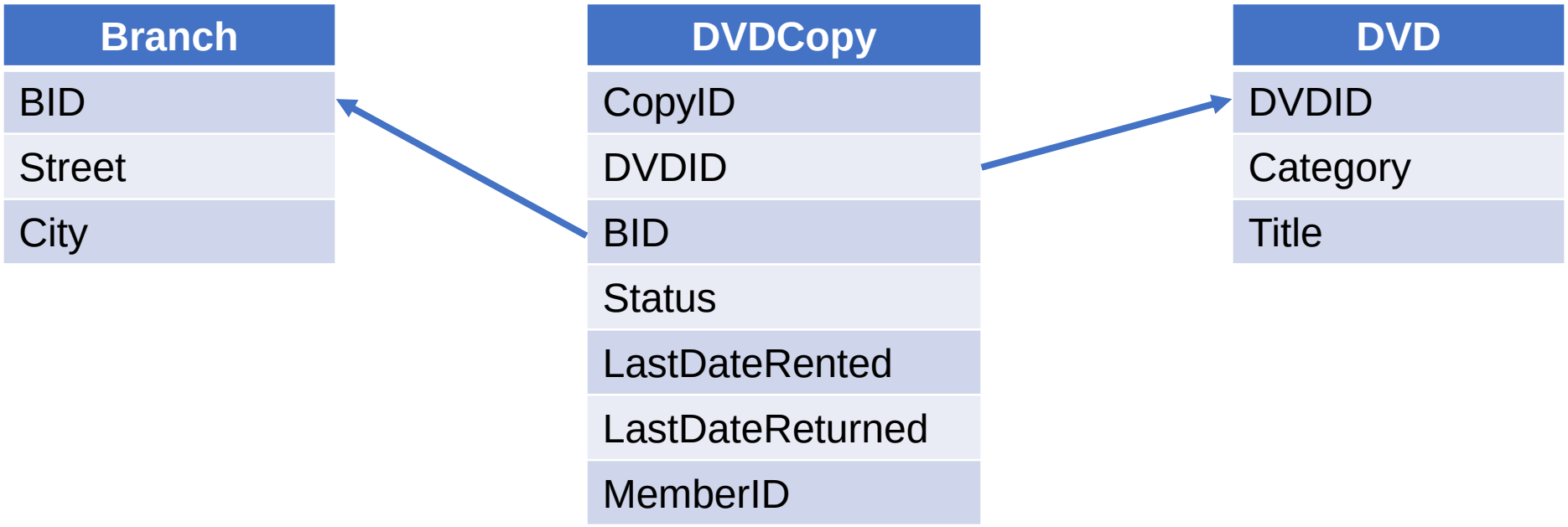
A5

ID	NAME	BIRTHDAY	HIREDATE	DEPT
89	Emily	1990/12/09	2017/02/01	Technology
241	Samantha	1991/12/04	2016/01/01	Finance
393	Hannah	1990/09/06	2016/01/01	Sales
…	…	…	…	…

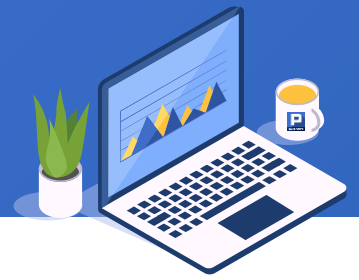
1. Reference eligible records directly



Branch stores information of DVD branch stores; *DVD* stores DVD titles and categories; *DVDCopy* stores information of DVD copies, which are physically owned by branch stores. **Task**: Find the branch stores that have less than 4 categories of DVD copies.



1. Reference eligible records directly



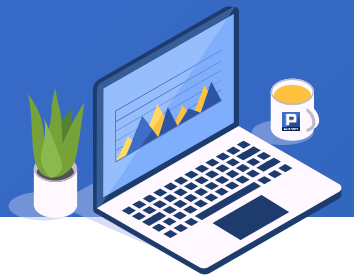
The SPL script uses "|" to get the sequence of concatenation, and "\" to get the sequence of difference:

	A	B
1	=connect("db")	/Connect to data source
2	=Branch=A1.query("select * from Branch")	/Query *Branch* table and define the result as a variable named *Branch*
3	=DVD=A1.query("select * from DVD")	/Query *DVD* table and define the result as a variable named *DVD*
4	=DVDCopy=A1.query("select * from DVDCopy")	/Query *DVDCopy* table and define the result as a variable named *DVDCopy*
5	=DVDCopy.switch(DVDID,DVD:DVDID; BID,Branch:BID)	/Replace DVDCopy.DVDID with corresponding records in *DVD* table
6	=DVDCopy.select(STATUS!="Miss" && LASTDATERETURNED!=null)	/Select the lost and unreturned DVD copies
7	=A6.group(BID)	/Group the filtered records by BID
8	=A7.select(~.icount(DVDID.CATEGORY)<4)	/Find branches having less than 4 categories of DVD copies
9	=A8.(BID) | (Branch \ A7.(BID))	/All desired branches. A8.(BID) are those having less than 4 categories of copies; Branch \ A7.(BID) are those that don't have certain copies.

A9

BID	STREET	CITY
B002	Street2	Houston
B003	Street3	LA
B004	Street4	Lincoln

2. Merge sets by certain fields



The Math scores and English scores are stored respectively in *Math.txt* and *English.txt*. The two files are of same structure. **Task**: Calculate the total score for each student.

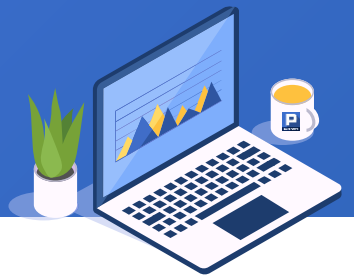
Math:

CLASS	STUDENTID	SUBJECT	SCORE
1	1	Math	77
1	2	Math	80
…	…	…	…

English:

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	2	English	81
…	…	…	…

2. Merge sets by certain fields



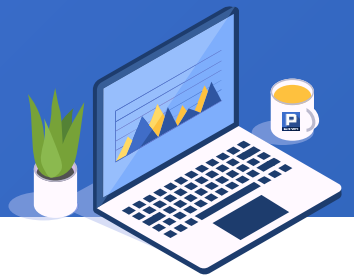
The SPL script uses A.merge(xi, ...) function to concatenate table sequences by expressions xi, ... :

	A	B
1	=file("Math.txt").import@t()	/Import *Math*.txt
2	=file("English.txt").import@t()	/Import *English*.txt
3	=A1.sort(CLASS,STUDENTID)	/Sort *Math* table by CLASS and STUDENTID
4	=A2.sort(CLASS,STUDENTID)	/Sort *English* table by CLASS and STUDENTID
5	=[A3,A4].merge(CLASS,STUDENTID)	/merge() to concatenate records by CLASS and STUDENTID
6	=A5.groups@o(CLASS,STUDENTID; ~.sum(SCORE):TOTALSCORE)	/Use groups@o() to group records, which creates a new group whenever the value changes, and sum scores for each student

A6

CLASS	STUDENTID	TOTALSCORE
1	1	161
1	2	161
1	3	159
...	...	...

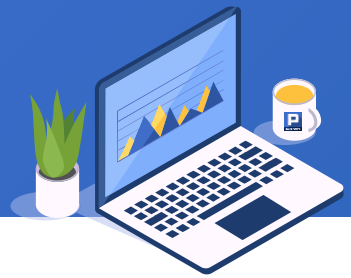
2. Merge sets by certain fields



Sales records are stored in *Online* table and *Store* table according to distribution channels. They are of same structure. Records during promotion periods of both channels are stored in both tables. **Task**: Calculate the actual total sales.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

2. Merge sets by certain fields

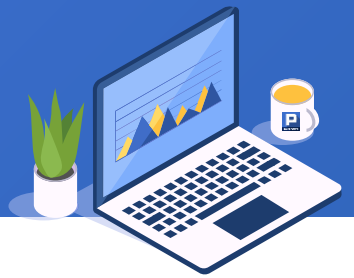


The SPL script uses A.merge@u(xi, ...) function to remove duplicate records during the order-based merge:

	A	B
1	=file("Online.txt").import@t()	/Import *Online*.txt
2	=file("Store.txt").import@t()	/Import *Store*.txt
3	=A1.sort(OrderID)	/Sort *Online* table by OrderID
4	=A2.sort(OrderID)	/Sort *Store* table by OrderID
5	=[A3,A4].merge@u(OrderID)	/merge@u() merges two tables by OrderID and delete duplicates at the same time
6	=A5.sum(Amount)	/Sum the sales amounts

A6	Member
	678756.41

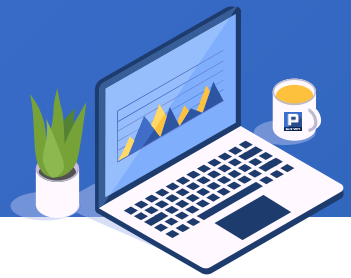
2. Merge sets by certain fields



Task: According to the previous files, we want to find the number of sales records that are stored in both tables.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

2. Merge sets by certain fields

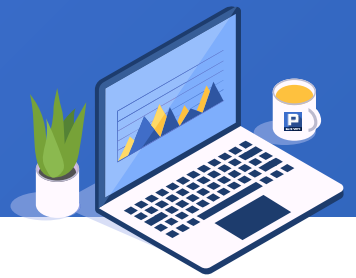


The SPL script uses A.merge(xi, ...)@i to get a table sequence consisting of common members of A(i)...:

	A	B
1	=file("Online.txt").import@t()	/Import *Online*.txt
2	=file("Store.txt").import@t()	/Import *Store*.txt
3	=A1.sort(OrderID)	/Sort *Online* table by OrderID
4	=A2.sort(OrderID)	/Sort *Store* table by OrderID
5	=[A3,A4].merge@i(OrderID)	/merge@i() merges two tables by OrderID to return a table sequence of their common records
6	=A5.count()	/Count the common records

A6	Member
	70

2. Merge sets by certain fields



The transaction records in March, 2015 are stored in *old*.csv and *new*.csv. Both use UserName and Date as their logical primary keys. **Task**: Find the newly-added, deleted and modified records.

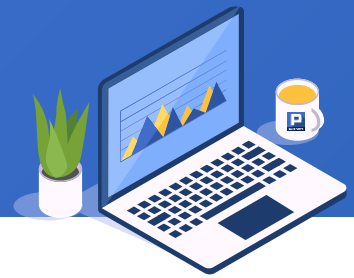
old.csv

UserName	Date	SaleValue	SaleCount
Rachel	2015-03-01	4500	9
Rachel	2015-03-03	8700	4
Tom	2015-03-02	3000	8
Tom	2015-03-03	5000	7
Tom	2015-03-04	6000	12
John	2015-03-02	4000	3
John	2015-03-02	4300	9
John	2015-03-04	4800	4

new.csv

UserName	Date	SaleValue	SaleCount
Rachel	2015-03-01	4500	9
Rachel	2015-03-02	5000	5
Ashley	2015-03-01	6000	5
Rachel	2015-03-03	11700	4
Tom	2015-03-03	5000	7
Tom	2015-03-04	6000	12
John	2015-03-02	4000	3
John	2015-03-02	4300	9
John	2015-03-04	4800	4

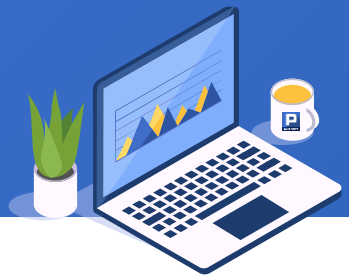
2. Merge sets by certain fields



The SPL script uses A.merge@d(xi, ...) function to remove members of A(2) &...A(n) from A(1) to generate a new table sequence:

	A	B
1	=file("old.csv").import@ct()	/Import *old*.csv
2	=file("new.csv").import@ct()	/Import *new*.csv
3	=A1.sort(UserName,Date)	/Sort *old* table by UserName and Date
4	=A2.sort(UserName,Date)	/Sort *new* table by UserName and Date
5	=new=[A4,A3].merge@d(UserName,Date)	/merge@d() deletes records of A3 from A4 while performing order-based merge to generate a table sequence of new records
6	=delete=[A3,A4].merge@d(UserName,Date)	/merge@d() deletes records of A4 from A3 while performing order-based merge to generate a table sequence of deleted records
7	=diff=[A4,A3].merge@d(UserName,Date,SaleValue,SaleCount)	/merge@d() deletes records of A3 where the specified field values change from A4 while performing order-based merge
8	=update=[diff,new].merge@d(UserName,Date)	/merge@d() deletes new from updated records while performing order-based merge to generate a table sequence of updated records
9	return [new, delete, update]	/Return a sequence of new, deleted and updated records

2. Merge sets by certain fields



A9
Members
[[Ashley,2015-03-01,6000,5], …]
[[Tom,2015-03-02,3000,8]]
[[Rachel,2015-03-03,11700,4]]

new

UserName	Date	SaleValue	SaleCount
Ashley	2015-03-01	6000	5
Rachel	2015-03-02	5000	5

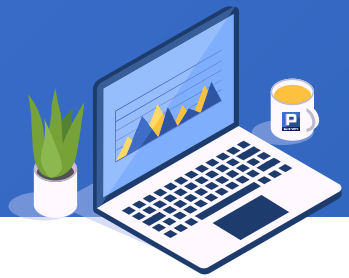
delete

UserName	Date	SaleValue	SaleCount
Tom	2015-03-02	3000	8

update

UserName	Date	SaleValue	SaleCount
Rachel	2015-03-03	11700	4

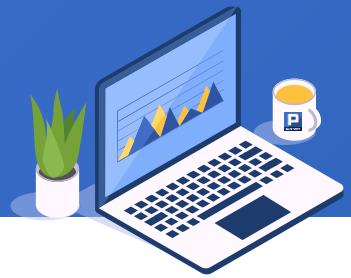
2 Merge sets by certain fields



Below are same-structure files generated by random samplings. **Task**: Count the unique IDs selected by the two files.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.009460545997473379	0.0
23	0.024176791871681664	0.0
…	…	…

2. Merge sets by certain fields

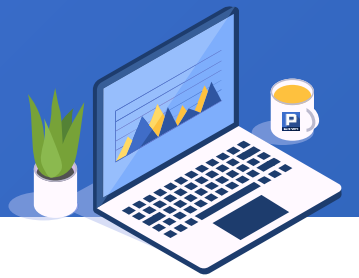


The SPL script uses A.merge @x(xi, …) function to return a new table sequence by removing common members of A(i)…:

	A	B
1	=file("p1.txt").import@t()	/Import the first sampling file *p1*
2	=file("p2.txt").import@t()	/Import the second sampling file *p2*
3	=A1.sort(ID)	/Sort *p1* by ID
4	=A2.sort(ID)	/Sort *p2* by ID
5	=[A3,A4].merge@x(ID)	/merge@x() performs an order-based merge by ID and return as sequence of records with different IDs
6	=A5.len()	/Count the different IDs

A6	Member
	458

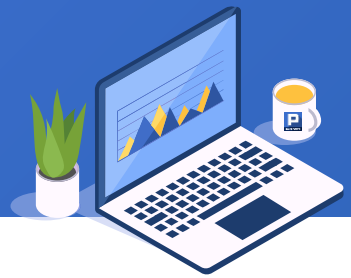
3. Merge sets by the primary key



There are a series of same-structure body temperature files named after dates, such as 601.txt for June 1. **Task**: Find the students who have a fever for at least 3 days consecutively.

StudentID	Name	Fever
10	Ryan	0
5	Ashley	0
13	Daniel	1
19	Samantha	0
1	Rebecca	0
…	…	…

3. Merge sets by the primary key

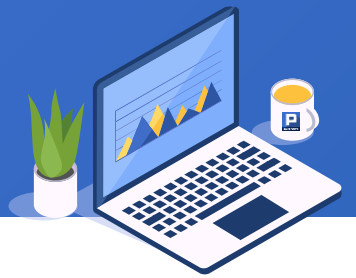


The SPL script uses A.merge() function to perform an order-based merge by the primary key as long as the primary key is set for A(i):

	A	B
1	=to(601, 620)	/Create a sequence of file names
2	=A1.(file(string(~)+".txt").import@t())	/Import files from June 1 to June 20
3	=A2.(~.keys(StudentID).sort(StudentID))	/Set StudentID as the primary key and sort the files by the key
4	=A3.merge()	/merge() compares the primary key values to perform the order-based merge
5	=A4.group@o(StudentID,Fever)	/group@o() creates a new group whenever the key value changes
6	=A5.select(~.Fever==1 && ~.len()>=3).id(Name)	/Get students who have had a fever for at least 3 days consecutively

A6	Name
	Ashley
	Rachel

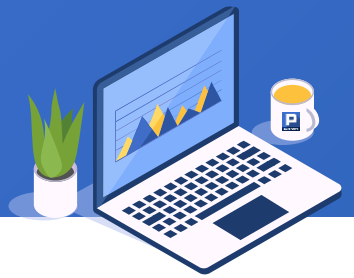
4. Merge sets by all fields



There are two same-structure files *p1.csv* and p2.csv. **Task**: Count the different records between them.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.00946054599747337 9	0.0
23	0.024176791871681664	0.0
…	…	…

4. Merge sets by all fields

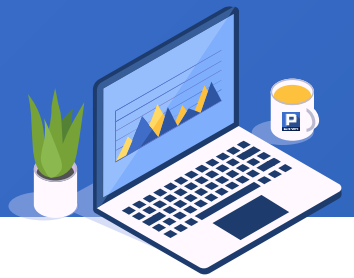


SPL script uses A.merge() function to compare all fields to perform the order-based merge when no primary key is set for A(i):

	A	B
1	=file("p1.txt").import@t()	/Import the first sampling file *p1*
2	=file("p2.txt").import@t()	/Import the second sampling file *p2*
3	=[A1,A2].merge@x()	/merge() compares all fields to perform the order-based merge. @x option returns a sequence of different IDs, that is, the records with different IDs
4	=A3.len()	/Return the number of different records

A4	Member
	458

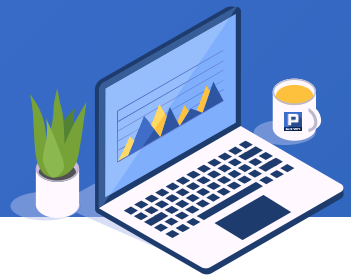
5. When records are not ordered by the key



Sales data is stored in two databases, the old in db1 and the new in db2. Both have same structures. **Task**: Calculate the total sales in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10426	GALED	4	2014/01/27	338.2
10676	TORTU	2	2014/09/22	534.85
10390	ERNSH	6	2013/12/23	2275.2
10400	EASTC	1	2014/01/01	3063.0
10464	FURIB	4	2014/03/04	1848.0
…	…	…	…	…

5. . When records are not ordered by the key



The SPL script uses A.merge @o(xi, ...) function to perform the merge when A(i) is not ordered by [xi,...]:

	A	B
1	=connect("db1").query("select * from Sales")	/Query *Sales* table from db1
2	=connect("db2").query("select * from Sales")	/Query *Sales* table from db2
3	=[A1,A2].merge@ou(OrderID)	/merge() performs the order-based merge by OrderID. @o option indicates that the records are not necessarily ordered by OrderID; @u option removes records with duplicate IDs
4	=A3.select(year(OrderDate)==2014)	/Get records of 2014
5	=A4.sum(Amount)	/Calculate the total sales in 2014

A5	Member
	723388.75

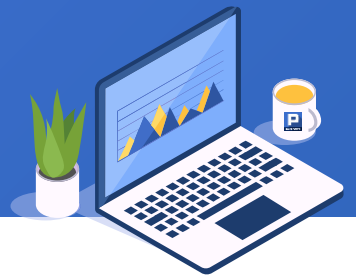
Note: A.merge@o() works similarly to A.conj(). Yet it is more common to use the @o option with @u/@i/@d/@x options, as this example shows.

CONTENTS

1. Concatenation of sets
2. Merge sets by column values

Big data operations

1. Concatenation of sets

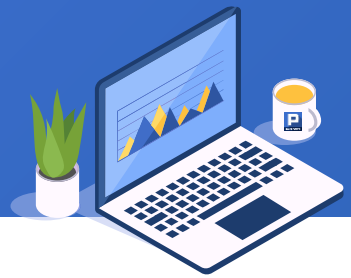


Task: Find the record that having the largest sales amount in each month.

The *Sales* table is too large to be wholly loaded into the memory.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

1. Concatenation of sets

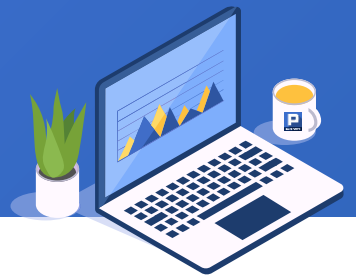


The SPL script uses cs.group(x, ...) to group records of the cursor by comparing neigboring records and return the grouped cursor:

	A	B
1	=connect("db").query("select * from Sales order by OrderDate")	/Query *Sales* table in the database and sort it by OrderDate
2	=A1.group(month(OrderDate))	/cs.group() groups records by comparing neighboring months
3	=A2.(~.maxp(Amount))	/Find the record with the largest sales in each month
4	=A3.conj()	/列Return the concatenation of records with the largest sales in each month
5	=A4.fetch()	/Fetch data from the cursor to get a relatively small result set

A5	OrderID	Customer	SellerId	OrderDate	Amount
	10267	FRANK	4	2013/07/29	4031.0
	10286	QUICK	8	2013/08/21	3016.0
	...	...	...	...	...

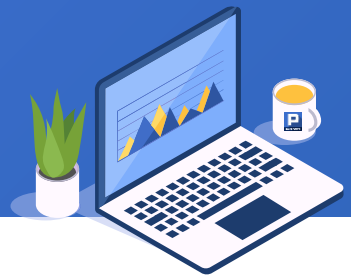
1. Concatenation of sets



The sales records of 2014 and 2015 are stored in same-structure tables *S2014* and *S2015* respectively. Both are too big to be loaded into the memory at once. **Task**: Find the customers whose order amounts rank top 3 in both years.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

1. Concatenation of sets

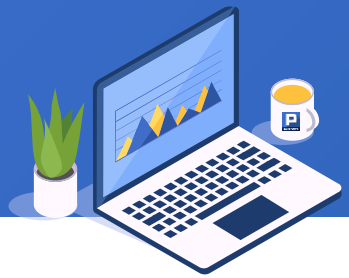


The SPL script uses CS.conjx() function to combine cursors vertically, which is the concatenation of records in the cursors:

	A	B
1	=connect("db")	/Connect to the database
2	=A1. cursor("select * from S2014")	/Get cursor of *S2014* table
3	=A1. cursor("select * from S2015")	/Get cursor of *S2015* table
4	=[A2,A3].conjx()	/CS.joinx() concatenates the two cursors together
5	=A4.groups(Customer; sum(Amount):Amount)	/Group and summarize the concatenation result to sum the sales amounts for each customer
6	=A5.top(-3;Amount)	/

A6	Customer	Amount
	SAVEA	177478.89
	QUICK	102764.99
	ERNSH	94066.28

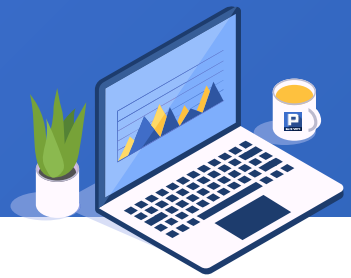
2. Merge sets by column values



The sales data is stored in old database db1 and new database db2. The two database tables are of same structure and too large to be loaded into the memory at a time. **Task**: Calculate the sales amount in each month of 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

2. Merge sets by column values

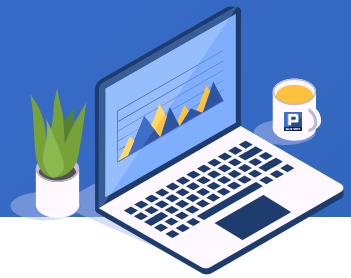


The SPL script uses CS.mergex(xi, …) function to merge sequences of records in cursors:

	A	B
1	=connect("db1").cursor("select * from Sales order by OrderDate")	/Query *Sales* table in db1 and sort it by OrderDate
2	=connect("db2").cursor("select * from Sales order by OrderDate")	/Query *Sales* table in db2 and sort it by OrderDate
3	=[A1,A2].mergex(OrderDate)	/mergex() merges the two cursors by OrderDate
4	=A3.select(year(OrderDate)==2014)	/Get records of 2014
5	=A4.groups@o(month(OrderDate):Month; count(~):Count)	/groups() groups and summarize sales amount for each month. @o option creates a new groups whenever the month changes

A5	Month	Count
	1	33
	2	29
	…	…

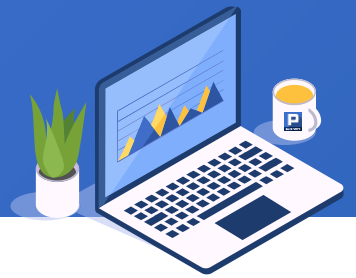
2. Merge sets by column values



With the same tables, assume that they have duplicate records. **Task**: Calculate total order amount of each customer in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
…	…	…	…	…

2. Merge sets by column values



CS.mergex(xi, ...) can work with @u/@i/@d/@x options that work similarly to options for A.merge(). Below is the SPL script:

	A	B
1	=connect("db1").cursor("select * from Sales order by OrderID")	/Query *Sales* table in db1 and sort it by OrderDate
2	=connect("db2").cursor("select * from Sales order by OrderID")	/Query *Sales* table in db2 and sort it by OrderDate
3	=[A1,A2].mergex@u(OrderID)	/mergex@u() removes duplicate records while merging the cursors by OrderID
4	=A3.select(year(OrderDate)==2014)	/Get records of 2014
5	=A4.groups(Customer; sum(Amount):Amount)	/groups() groups and summarize each customer’ s sales amount

A5

Customer	Amount
ANATR	1129.75
ANTON	6452.15
…	…

THANKS

for watching