esProc

SPL assists mongoDB calculation

Issued by Raqsoft

Contents

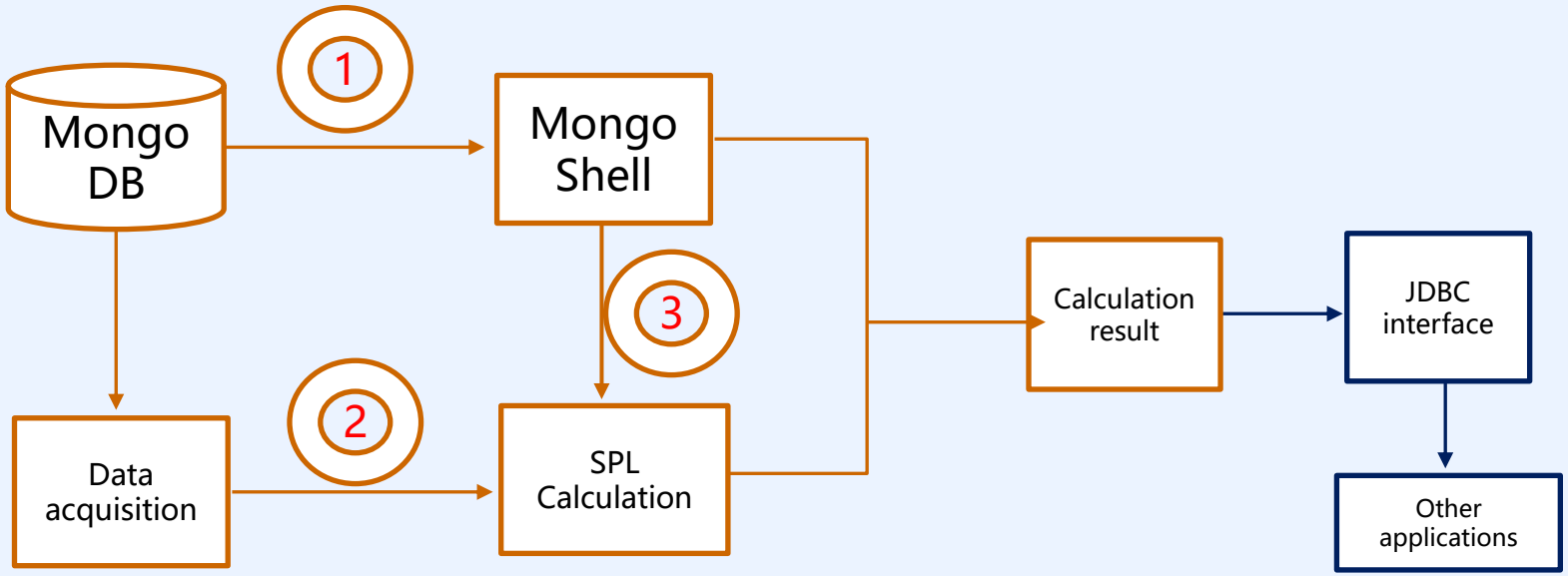
1 Preface

MongoDB is a very popular NoSQL database, because it has high concurrency, high performance, flexible data model, supports distributed query, is suitable for the needs of Internet agile development, and is popular in all professions and trades.

MongoDB has a rich query language interface, which is similar to the object-oriented query language. It can realize most of the functions similar to single table query of relational database. However, the script language has a certain degree of complexity, and it will face some difficulties to realize some calculations. When it is necessary to treat the hierarchical data in mongoDB as relational data for calculation from a specific perspective, the esProc SPL can use mongoDB as an external data source for enhanced SQL processing, so that the requirements of different applications are relatively easy to implement.

Let's learn from the following aspects how esProc can assist mongoDB calculation.

2 Structural relationship



There are three ways to implement mongoDB calculation in esProc:
1. Pure script mode: Directly calculate the calculation result through the Mongo shell script.
2. SPL mode: Take mongoDB as the data source and use SPL language to calculate the calculation result.。
3. Mixed mode: Output based on the Mongo shell script calculation result and SPL language processing.

Of course, SPL also provides external JDBC interface as a data source for other applications to access and facilitate integration.

3 SQL style calculation

The basic association table uses A.F=B.F internal and external key join query, which is realized by $lookup function. In this paper, the multi key join is discussed.

A JOIN

Set A and B join by multi key, such as A.F1 = B.F1 and A.F2 = B.F2.

Set C1:

user1	user2	income
1	2	0.56
1	3	0.26

Set C2:

user1	user2	output
1	2	0.3
1	3	0.4
2	3	0.5

Expected result:

user1	user2	income	output
1	2	0.56	0.3
1	3	0.26	0.4

A. JOIN

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"c1.find()").fetch()
3	=mongo_shell(A1,"c2.find()").fetch()
4	=A2.join(user1:user2,A3:user1:user2,output)
5	>A1.close()

Join table key relationship:
c1.user1=c2.user1 and c1.user2 = c2.user2

Select the required fields from the associated table and combine them into a new table.

Calculation result:

user1	user2	income	output
1	2	0.56	0.3
1	3	0.26	0.4

B. Subquery

A subquery is a query nested in the main query, that is, where A.F in(B.F). It is implemented by the select function of esProc SPL. Subquery includes single table operation and join calculation.

B1. Subquery single table operation

Course set, column name is student number, course number, grade.

Sno	Cno	Grade
200810121	1	96
200810121	2	93
200810121	3	85
200810122	2	88
200810122	3	90
200810123	4	56
200810123	2	76

It is required to find out the course number and grade of each student that exceeds his / her average elective grade. Written as SQL:

```
select Sno,Cno,Grade from course x where Grade >=( select avg(Grade) from course y where y.sno=x.sno);
```

B1. Subquery single table operation

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"course.find(,{_id:0})")
3	=A2.group(Sno).((avg = ~.avg(Grade), ~.select(Grade>avg))).conj()
4	>A1.close()

Find out the record that is greater than the mean value after calculating the average grade by student number grouping.

Calculation result:

Sno	Cno	Grade
200810121	1.0	96.0
200810121	2.0	93.0
200810122	3.0	90.0
200810123	2.0	76.0

B2. Subquery join calculation

Set orders is order data，and set employee is employee data，the data is as follows：

Order set：

ORDERID	CLIENT	SELLERID	AMOUNT	ORDERDATE
1	UJRNP	17	392	2008/11/2 15:28
2	SJCH	6	4802	2008/11/9 15:28
3	UJRNP	16	13500	2008/11/5 15:28
4	PWQ	9	26100	2008/11/5 15:28
…				

Employee set：

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974-11-20	2005-03-11	R&D	7000
2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000
3	Rachel	Johnson	F	New Mexico	1970-12-17	2010-12-01	Sales	9000
…								

B2. Subquery join calculation

Find out the order information, where the sellerid of the order is the employee Eid of state = California in the employee set. Written as SQL:

Select * from orders where orders.sellerid in (select eid from employee where employee.state='California')

	A
1	=mongo_open("mongodb://localhost:27017/test?user=test&password=test")
2	=mongo_shell(A1,"orders.find(,{_id:0})")
3	=mongo_shell(A1,"employee.find({STATE:'California'},{_id:0})").fetch()
4	=A3.(EID).sort()
5	=A2.select(A4.pos@b(SELLERID)).fetch()
6	>mongo_close(A1)

Find out the records of sellerid in employee table

ORDERID	CLIENT	SELLERID	AMOUNT	ORDERDATE
2	SJCH	6	4802.0	2008/11/9 15:28
3	UJRNP	6	13500.0	2008/11/5 15:28
17	VILJX	3	20100.0	2008/10/3 15:28
20	EGU	8	11800	2008/11/21 15:28

C. Join table statistics

Count () the joined table.
Set progress records the relationship between users and courses. Its courseid field is a foreign key, which points to the _id field of set course. You need to count the number of people in each course, and the course name needs to be displayed using the title field of course.

Set Progress：

userId	courseid	timespent	score
u01	c01	6000	99
u02	c01	6000	99
u03	c01	6000	99
u04	c01	6000	99
U05	c01	6000	99
u01	c02	6000	99
u02	c02	6000	99
u03	c03	6000	99

Set Course：

_id	title	description	category
c01	Japanese159	Japanese base	language
c02	Chinese200	Chinese middle	language
c03	Political science 280	Political middle	politics
c04	EE490	electronic engineering hign	Electronic

C. Join table statistics

	A
1	=mongo_open("mongodb://localhost:27017/local?user=test&password=test")
2	=mongo_shell(A1,"Progress.aggregate([{$group: {_id: {'primary': '$courseid'}, 'popularityCount': {$sum: 1}}}, {$sort:{'popularityCount':-1}},{$project:{_id:0,'courseid':'$_id.primary','popularityCount':1}}])")
3	=mongo_shell(A1,"Course.find(,{title:1})").fetch()
4	=A2.switch(courseid,A3:_id)
5	=A4.new(popularityCount,courseid.title)
6	=mongo_close(A1)

Generate table data on demand

popularityCount	title
5	Japanese159
2	Chinese200
1	Political science 280

Switch to record data

popularityCount	courseid
5	[c01, Japanese159]
2	[c02, Chinese200]
1	[c03, Political science 280]

D. Join array lookup

Find the qualified records from the record array of the join table, and combine them into a new table with the given fields, A.F in (B.F).

Set users:

_id	Name	workouts
1000	xxx	[2,4,6]
1002	yyy	[1,3,5]

Set workouts:

_id	Date	Book
1	1/1/2001	Othello
2	2/2/2001	A Midsummer Night's Dream
3	3/3/2001	The Old Man and the Sea
4	4/4/2001	GULLIVER'S TRAVELS
5	5/5/2001	Pickwick Papers
6	6/6/2001	The Red and the Black

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"users.find()").fetch()
3	=mongo_shell(A1,"workouts.find()").fetch()
4	=A2.conj(A3.select(A2.workouts^~.array(_id)!=[]).derive(A2.name))
5	>A1.close()

Find the record where the _id field of A3 exist in the workouts array of A2, and append the name field. Return the merged sequence table.

Calculation result:

Name	_id	Date	Book
xxx	2	2/2/2001	A Midsummer Night's Dream
xxx	4	4/4/2001	GULLIVER'S TRAVELS
xxx	6	6/6/2001	The Red and the Black
yyy	1	1/1/2001	Othello
yyy	3	3/3/2001	The Old Man and the Sea
yyy	5	5/5/2001	Pickwick Papers

4. Set intersect, union and minus

A. Single table union all

Get the data of a key value in the nested structure and shows which source the value comes from.

Netmap set:

_id	linkedIn. people	Twitter. people
1	{ name : 'Fred' }, { name : 'Matilda'}	{ name : 'Hanna' }, { name : 'Walter' }
2	{ name : 'Jack' }, { name : 'Carl'}	{ name : 'Tom' }, { name : 'Jam' }

Expected result:

name	source
Fred	linkedIn
Matilda	linkedIn
Hanna	Twitter
Walter	Twitter
Jack	linkedIn
Carl	linkedIn
Tom	Twitter
Jam	Twitter

A. Single table union all

	A	B	C
1	=mongo_open("mongodb://localhost:27017/user?user=test&password=test")		
2	=mongo_shell(A1, "netmap.find(, {_id:0})").fetch()		
3	for A2	=A3.(linkedIn).people	=A3.(twitter).people
4		=B3.new(name, "linked": source)	=C3.new(name,"twitter": source)
5		=@|B4|C4	
6	>mongo_close(A1)		

Data record append and merge

Name merged with record source

Calculation result:

name	source
Fred	linkedIn
Matilda	linkedIn
Hanna	Twitter
Walter	Twitter
Jack	linkedIn
Carl	linkedIn
Tom	Twitter
Jam	Twitter

name	source
Hanna	Twitter
Walter	Twitter

B. union all

Multiple sets of the same structure are combined for calculation, and mongodb does not provide the corresponding interface for implementation.
The calculation includes：UNION ALL， UNION(data deduplication)，INTERSECT，MINUS.

Set emp1：

_id	NAME	STATE	HIREDATE	DEPT	SALARY
1	Ashley	New York	2008-03-16	Finance	11000
2	Rachel	Michigan	2001-04-16	Sales	9000
3	Emily	New York	2011-07-11	HR	8800
4	Matthew	Texas	2003-03-06	R&D	8000
5	Alexis	Illinois	2008-03-10	Sale	14000

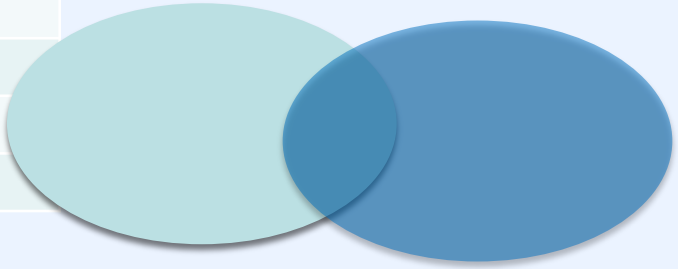
Set emp2：

_id	NAME	STATE	HIREDATE	DEPT	SALARY
10	Jacob	New York	2009-03-14	Sales	13000
12	Jessica	Florida	2011-04-19	Sales	9500
13	Daniel	New York	2001-02-11	HR	7800
14	Alyssa	Montana	2013-09-06	R&D	8000
15	Hannah	Florida	2015-06-10	Sales	12500
1	Ashley	New York	2008-03-16	Finance	11000

There are two kinds of operations of union, intersection and minus, one is full line comparison, the other is only keyword comparison.

B. union all

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"emp1.find()").fetch()
3	=mongo_shell(A1,"emp2.find()").fetch()
4	=[A2,A3].conj()
5	>A1.close()



Set **union all** , Return all query records.

Data after consolidation:

_id	NAME	STATE	HIREDATE	DEPT	SALARY
1	Ashley	New York	2008-03-16	Finance	11000
2	Rachel	Michigan	2001-04-16	Sales	9000
3	Emily	New York	2011-07-11	HR	8800
4	Matthew	Texas	2003-03-06	R&D	8000
5	Alexis	Illinois	2008-03-10	Sale	14000
10	Jacob	New York	2009-03-14	Sales	13000
12	Jessica	Florida	2011-04-19	Sales	9500
13	Daniel	New York	2001-02-11	HR	7800
14	Alyssa	Montana	2013-09-06	R&D	8000
15	Hannah	Florida	2015-06-10	Sales	12500
1	Ashley	New York	2008-03-16	Finance	11000

C. Union

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"emp1.find()").fetch()
3	=mongo_shell(A1,"emp2.find()").fetch()
4	=[A2,A3].merge@ou()
5	=[A2,A3].merge@ou(_id, NAME)
6	>A1.close()

Full line comparison and union

Key value comparison and union

Set union, return the query result after removing duplicate records.

Data after consolidation:

_id	NAME	STATE	HIREDATE	DEPT	SALARY
1	Ashley	New York	2008-03-16	Finance	11000
2	Rachel	Michigan	2001-04-16	Sales	9000
3	Emily	New York	2011-07-11	HR	8800
4	Matthew	Texas	2003-03-06	R&D	8000
5	Alexis	Illinois	2008-03-10	Sale	14000
10	Jacob	New York	2009-03-14	Sales	13000
12	Jessica	Florida	2011-04-19	Sales	9500
13	Daniel	New York	2001-02-11	HR	7800
14	Alyssa	Montana	2013-09-06	R&D	8000
15	Hannah	Florida	2015-06-10	Sales	12500

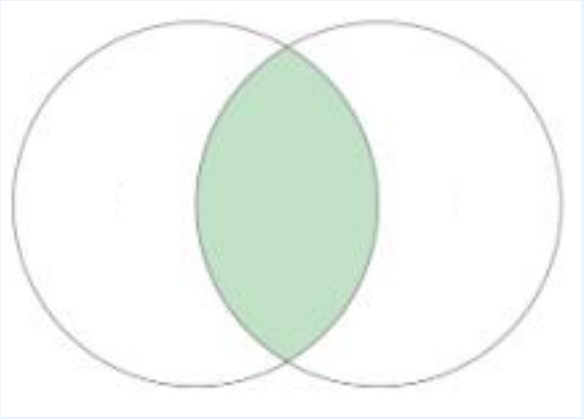
D. Intersect

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"emp1.find()").fetch()
3	=mongo_shell(A1,"emp2.find()").fetch()
4	=[A2,A3].merge@oi()
5	=[A2,A3].merge@oi(_id, NAME)
6	>A1.close()

Full line comparison and intersect

Key value comparison and intersect

Set intersection, return the same part of the query result.



Data after consolidation:

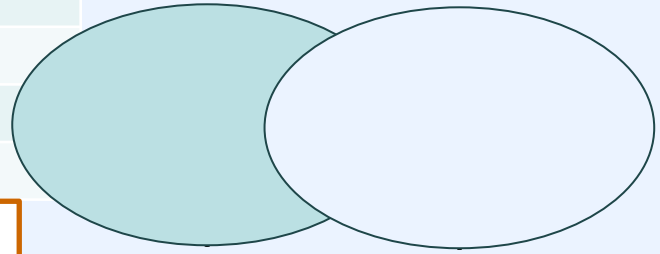
_id	NAME	STATE	HIREDATE	DEPT	SALARY
1	Ashley	New York	2008-03-16	Finance	11000

E. Minus

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"emp1.find()").fetch()
3	=mongo_shell(A1,"emp2.find()").fetch()
4	=[A2,A3].merge@od()
5	=[A2,A3].merge@od(_id, NAME)
6	>A1.close()

Full line comparison and minus

Key value comparison and minus



Set minus, return data included in emp1 result but not included in emp2 result.

Data after consolidation:

_id	NAME	STATE	HIREDATE	DEPT	SALARY
2	Rachel	Michigan	2001-04-16	Sales	9000
3	Emily	New York	2011-07-11	HR	8800
4	Matthew	Texas	2003-03-06	R&D	8000
5	Alexis	Illinois	2008-03-10	Sale	14000

5 Subset operation

A. Look for array index

Query the index by the value of the element to know where it is located.
In the set users, names and friends (array) are saved. The names of people in the friends array are saved in order of ranking.

name	friends
jim	["tom", "jack", "luke", "rose", "james", "sam", "peter"]
jack	["blan", "bill", "Carl", " Colin"]

Mongodb finds the name of the person in the specified ranking, for example, among Jim's friends, the first person in the ranking:

```
> db.b.find({"name":"jim"},{"friends":{"$slice":[0,1]},_id:0})
{
        "name" : "jim", "friends" : [ "tom" ]
}
```

But we can't find the rank value of "Luke" among Jim's friends.

A. Look for array index

	A
1	=mongo_open("mongodb://localhost:27017/local?user=test&password=test")
2	=mongo_shell(A1,"users.find({name:'jim'},{name:1,friends:1,_id:0})")
3	=A2.fetch()
4	=A3.friends.pos("luke")
5	=mongo_close(A1)

Value
3

Get the friends array according to the filter conditions, and the POS function gets the location of Luke.

B Array filter and search

Finds the records that satisfy the conditions from the specified array. For example, if the array is: ["Chemical", "Biology", "Math"] , query the students whose elective courses contain ["chemical", "biology", "math"].

_id	Name	Lesson
1	jacker	[English, Chemical，Math, Physics]
2	tom	[Chinese, Chemical，Math, Biology]
3	Mint	[Chinese, History]

	A
1	[Chemical, Biology, Math]
2	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
3	=mongo_shell(A2,"student.find()").fetch()
4	=A3.select(lesson^A1!=[])
5	=A4.new(_id, name, ~.lesson^A1:lession)
6	>A2.close()

Query the qualified records in the array

Calculation result:

_id	Name	Lesson
1	Jacker	[Chemical,Math]
2	Tom	[Chemical,Math,Biology]

C. topN sorting

Statistical requirements: Sort a large number of objects, and then only take the top n as the data of the ranking list.
Set last3 has two fields: variable and timestamp. First, group by variable, then select the latest three timestamps from each group, and finally find the earliest timestamp from these documents.

_id	variable	timestamp
54f69645e4b077ed8d997857	A	1995-01-01T00:00:00Z
54f69645e4b077ed8d997856	A	1995-01-02T00:00:00Z
54f69645e4b077ed8d997855	A	1995-01-03T00:00:00Z
54f69645e4b077ed8d997854	B	1995-01-02T00:00:00Z
54f69645e4b077ed8d997853	B	1995-01-01T00:00:00Z
54f69645e4b077ed8d997852	B	1994-01-03T00:00:00Z
54f69645e4b077ed8d997851	C	1994-01-03T00:00:00Z
54f69645e4b077ed8d997850	C	1994-01-02T00:00:00Z
54f69645e4b077ed8d997858	C	1994-01-01T00:00:00Z
54f69645e4b077ed8d997859	C	1993-01-01T00:00:00Z

C. topN sorting

	A	B
1	=mongo_open("mongodb://localhost:27017/local?user=test&password=test")	
2	=mongo_shell(A1,"last3.find(,{_id:0};{variable:1})")	
3	for A2;variable	=A3.top(3;-timestamp)
4		=@|B3
5	=B4.minp(~.timestamp)	
6	=mongo_close(A1)	

Add B3 selected documents with the latest three timestamps to B4 continuously

Select the earliest document in B4

variable	timestamp
A	Tue Jan 03 08:00:00 CST 1995
A	Mon Jan 02 08:00:00 CST 1995
A	Sun Jan 01 08:00:00 CST 1995
B	Mon Jan 02 08:00:00 CST 1995
B	Sun Jan 01 08:00:00 CST 1995
B	Mon Jan 03 08:00:00 CST 1994
C	Mon Jan 03 08:00:00 CST 1994
C	Sun Jan 02 08:00:00 CST 1994
C	Sat Jan 01 08:00:00 CST 1994

variable	timestamp
C	Sat Jan 01 08:00:00 CST 1994

6. Nested query

Gets subdocument data from an embedded data structure object.

A. Aggregation query

Aggregate queries against data with nested structures.
Calculate the sum of income and output of each record.

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"computer.find()").fetch()
3	=A2.new(_id:ID,income.array().sum():INCOME,output.array().sum():OUTPUT)
4	>A1.close()

Sum after the value under input and output is converted to the sequence table.

ID	INCOME	OUTPUT
1	1600.0	1720.0
2	3550.0	1800.0

Set computer:

_id	income	output
1	{"cpu":1000, "mem":500, "mouse": "100"}	{"cpu":1000, "mem":600 , "mouse":"120"}
2	{"cpu":2000, "mem":1000, "mouse":50, "mainboard“ :500 }	{"cpu":1500, "mem":300 }

B. Same structure subdocument

The key value(field value) of set storage is the subdocument of the same structure.

```
{
    "_id": "alpha",
    "name": "Storage Alpha",
    "items": [
        {
            "category": "food",
            "name": "apple"
        },
        {
            "category": "food",
            "name": "banana"
        },
        {
            "category": "tool",
            "name": "hammer"
        },
        {
            "category": "furniture",
            "name": "couch"
        }
    ]
}
```

Search records with category== “food” .

	A
1	=mongo_open("mongodb://localhost:27017/local?user=test&password=test")
2	=mongo_shell(A1,"storage.find()").fetch()
3	=A2.(items).conj(~.select(category=="food"))
4	=mongo_close(A1)

Merge the qualified records into a sequence table to return.

category	name
food	apple
food	banana

C. Subdocument with multi attributes

Merge a field under multiple documents to form a new record.
Set C1 data is as follows:

id	acls	NAME
1	"append": {"users" : [ObjectId("54f5bfb0336a15084785c393")], "groups" : [ ] }, "edit" : { "groups" : [ ], "users" : [ObjectId("54f5bfb0336a15084785c392")]}, “fullControl” : { "users" : [ ], "groups" : [ ]}, "read" : {"users" : [ObjectId("54f5bfb0336a15084785c392"), ObjectId("54f5bfb0336a15084785c398")], "groups" : [ ] }	ABC
2	"append": {"users" : [ObjectId("54f5bfb0336a15084785c365")], "groups" : [ ] }, "edit" : { "groups" : [ ], "users" : [ObjectId("54f5bfb0336a15084785c392")] }, “fullControl” : { "users" : [ ], "groups" : [ ] }, "read" : {"users" : [ObjectId("54f5bfb0336a15084785c392"), ObjectId("54f5bfb0336a15084785c370")], "groups" : [ ] }	ABC
…		

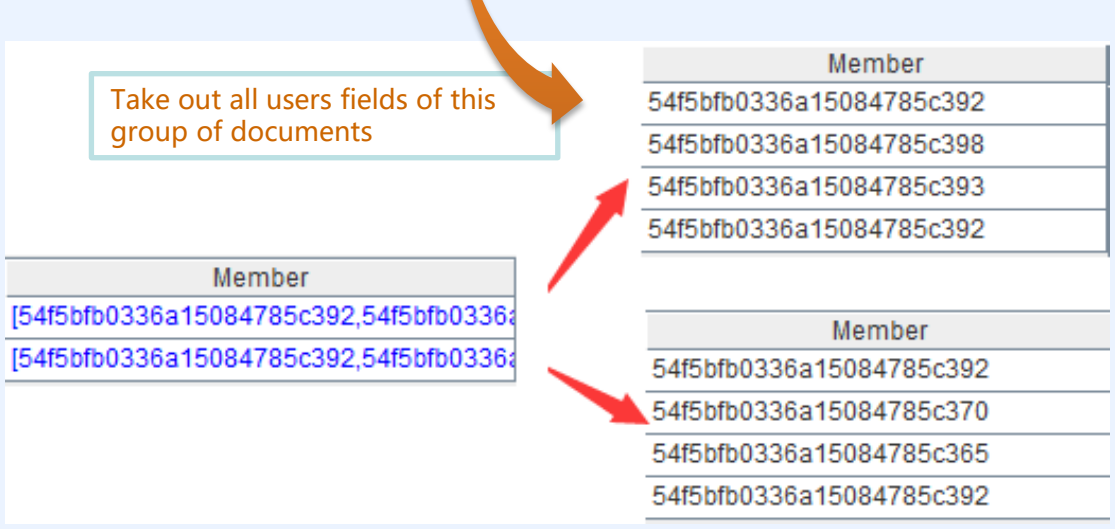
C. Subdocument with multi attributes

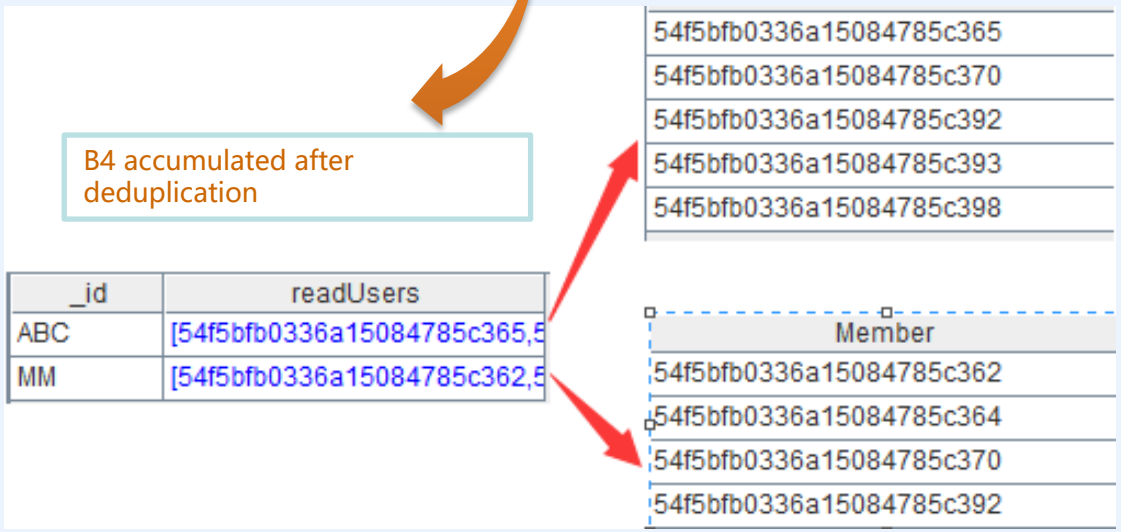
It is required to group by name. Each group of data is the users field in the subdocument corresponding to the same name, and the data cannot be duplicate. The final calculation result is similar to the following:

```
{
   result : [
            {
                     _id: "ABC",
                     readUsers : [
                              ObjectId("54f5bfb0336a15084785c393"),
                              ObjectId("54f5bfb0336a15084785c392"),
                              ObjectId("54f5bfb0336a15084785c398"),
                              ObjectId("54f5bfb0336a15084785c365"),
                              ObjectId("54f5bfb0336a15084785c370")
            ]
            }
   ]
}
```

C. Subdocument with multi attributes

	A	B
1	=mongo_open("mongodb://localhost:27017/local?user=test&password=test")	
2	=mongo_shell(A1,"c1.find(,{_id:0};{name:1})")	
3	for A2;name	=A3.(acls.read.users|acls.append.users|acls.edit.users|acls.fullControl.users)
4		=B3.new(A3.name:_id,B3.union().id():readUsers)
5		=@|B4.group@1(~._id,~.readUsers)
6	=mongo_close(A1)	





D. List subdocument

The set Cbettwen contains multi-level subdocuments, where datalist is of list type and contains multiple strings, each string is composed of multiple numbers.
Find strings that match the following conditions: the first number is greater than 6154 and less than or equal to 6155.


```
{
        "_id" : ObjectId("54f6a766bf4436333edcd6a2"),
        "_class" : "com.abc.core.bo.obj.Objs",
        "objList" : [
                {
                 "name" : "ABB-09",
                 "uid" : "ABB-09",
                 "data" : {
                     "dataId" : NumberLong(0),
                     "dataList" : [
                             "6150,32.9,1.475,,1.434",                "6150.5,43,,1.529,1.402",
                             "6151,31.8,1.506,1.447,1.453",           "6151.5,33.6,1.481,1.456,1.521",
                             "6152,30.9,1.465,1.472,1.547",           "6152.5,39.5,1.404,1.425,1.485",
                             "6153,43.2,1.406,1.446,1.481",           "6153.5,39.5,1.433,1.468,1.488",
                             "6154,32.7,1.459,1.477,1.427",           "6154.5,37.9,1.529,1.429,1.429",
                             "6155,30.4,1.505,1.532,1.543",           "6155.5,37.3,1.49,1.436,1.462",
                             "6156,35.3,1.538,1.45,1.488",            "6156.5,37.3,1.517,1.535,1.473",
                             "6157,32.7,1.401,1.405,1.497",           "6157.5,38.9,1.488,1.468,1.499",
                             "6158,35.4,1.526,1.422,1.452",           "6158.5,43.3,1.516,1.433,1.491",
                             "6159,34.6,1.519,1.442,1.478",           "6159.5,42.7,1.426,1.514,1.428",
                             "6160,32.7,1.451,1.5,1.516"
                    ] }
                }
        ] }
```

D. List subdocument

	A
1	=mongo_open("mongodb:// localhost:27017/local?user=test&password=test")
2	=mongo_shell(A1,"Cbettwen.find(,{_id:0})")
3	=A2.conj((t=~.objList.data.dataList.new(~),t.select((s=float(#1.split@c()(1)),s>6154 && s<=6155))))
4	=A3.fetch()
5	=mongo_close(A1)

Split each record in the datalist to get the value of the first string, and filter the records whose value meets the conditions.

Value
6154.5,37.9,1.529,1.429,1.429
6155,30.4,1.505,1.532,1.543

7 Complex grouping

A. Cross summary

Cross summary is a kind of practical classified statistics in data statistics. Generally, we use a certain variable as a row after grouping, and then use other variables or combinations of variables as columns to form a database table for statistical analysis. For example, the following table structure:

		Score				
School	Subject	1	2	3	4	5
A	Sub1	Number of people	Number of people	…	…	…
	Sub2	Number of people	…			
B	Sub1	Number of people				
	Sub2	Number of people				

If subjects and scores are combined, they can be further evolved into:

	Subject-score					
School	sub1-1	sub1-2	…	sub1-5	sub2-1	…
A	Number of people	…				
B	Number of people					

A. Cross summary

Mongodb can store similar data clearly and naturally, but it is difficult to realize cross summary.
The student set records the school, student's name, subject and score. The data is as follows:

school	sname	sub1	sub2
school1	Sean	4	5
school1	chris	4	3
school1	becky	5	4
school1	sam	5	4
shool2	dustin	2	2
shool2	greg	3	4
shool2	peter	5	1
shool2	brad	2	2
shool2	liz	3	null

Expected aggregation result:

school	sub1-5	sub1-4	sub1-3	sub1-2	Sub1-1	Sub2-5	Sub2-4	Sub2-3	Sub2-2	Sub2-1
school1	2	2	0	0	0	1	2	1	0	0
School2	1	0	2	2	0	0	1	0	2	1

A. Cross summary

	A
1	=mongo_open("mongodb://localhost:27017/local?user=test&password=test")
2	=mongo_shell(A1,"student.find()").fetch()
3	=A2.group(school)
4	=A3.new(school:school,~.align@a(5,sub1).(~.len()):sub1,~.align@a(5,sub2).(~.len()):sub2)
5	=A4.new(school,sub1(5):sub1-5,sub1(4):sub1-4,sub1(3):sub1-3,sub1(2):sub1-2,sub1(1):sub1-1,sub2(5):sub2-5,sub2(4):sub2-4,sub2(3):sub2-3,sub2(2):sub2-2,sub2(1):sub2-1)
6	=mongo_close(A1)

The scores of each group are aligned according to the sequence of [1,2,3,4,5], and the sequence length of each group is calculated.

school	sub1	sub2
school1	[0,0,0,...]	[0,0,1,...]
school2	[0,2,2,...]	[1,2,0,...]

Combine subject and score to columns for storage

school	sub1-5	sub1-4	sub1-3	sub1-2	Sub1-1	Sub2-5	Sub2-4	Sub2-3	Sub2-2	Sub2-1
school1	2	2	0	0	0	1	2	1	0	0
School2	1	0	2	2	0	0	1	0	2	1

B. Grouping by sections

Group statistics by the specified segment.
Count the number of records in each segment. Segment according to the sales volume, and count the data in each segment. The data is as follows:

_id	NAME	STATE	SALES
1	Ashley	New York	11000
2	Rachel	Montana	9000
3	Emily	New York	8800
4	Matthew	Texas	8000
5	Alexis	Illinois	14000

Segment method: 0-3000;3000-5000;5000-7500;7500-10000;10000+.

Calculation result:

Segment	number
3	3
4	2

	A
1	[3000,5000,7500,10000,15000]
2	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
3	=mongo_shell(A2,"sales.find()").fetch()
4	=A3.groups(A1.pseg(int(~.SALES)):Segment;count(1): number)
5	>A2.close()

Group and count the number of employees according to sales interval.

C. Group by categories

Count the total number and number of sub items under each classification.
Classification method: count the total number of books and the number of different books according to the addr classification.

addr	book
address1	book1
address2	book1
address1	book5
address3	book9
address2	book5
address2	book1
address1	book1
address15	book1
address4	book3
address5	book1
address7	book11
address1	book1

	A
1	=mongo_open("mongodb://127.0.0.1:27017/raqdb")
2	=mongo_shell(A1,"books.find()")
3	=A2.groups(addr,book;count(book): Count)
4	=A3.groups(addr;sum(Count):Total)
5	=A3.join(addr,A4:addr,Total)
6	>A1.close()

Join the total in A4 according to the key value of addr and merge it into the sequence table.

Calculation result:

_id	Total	books	Count
address1	4	book1	3
address1	4	book5	1
address15	1	book1	1
address2	3	book1	2
address2	3	book5	1
address3	1	book9	1
address4	1	book3	1
address5	1	book1	1
address7	1	book11	1

8 Import and export

The unstructured mongodb data can be converted into structured data, exported to CSV file or imported into database, and the data from other data sources can also be imported into mongodb.

A. Export to CSV

	cars	
id	name	car
No1	Putin	["porche", "bmw"]
No2	jack	["Toyota", "Jetta", "Audi"]
.....		

	A
1	=mongo_open("mongodb://localhost:27017/raqdb")
2	=mongo_shell(A1,"carInfo.find(,{_id:0})")
3	=A2.conj((t=~,~.cars.car.new(t.id:id, t.cars.name:name, ~:car)))
4	=file("D:\\data.csv").export@t(A3;",")
5	>mongo_close(A1)

id	name	car
No1	Putin	porche
No1	Putin	bmw
No2	jack	Toyota
No2	jack	Jetta
No2	jack	Audi
No3	Lis	Sienna
No3	Lis	bmw
No3	Lis	infiniti
.....		

The car field is split into rows to form a sequence table. The function conj merges the sequence table data and exports it to generate a CSV file.

B. import into database

Import mongodb data into the associated database.
Import the set course data in mongodb into the MySQL database table course 2.

Set course:

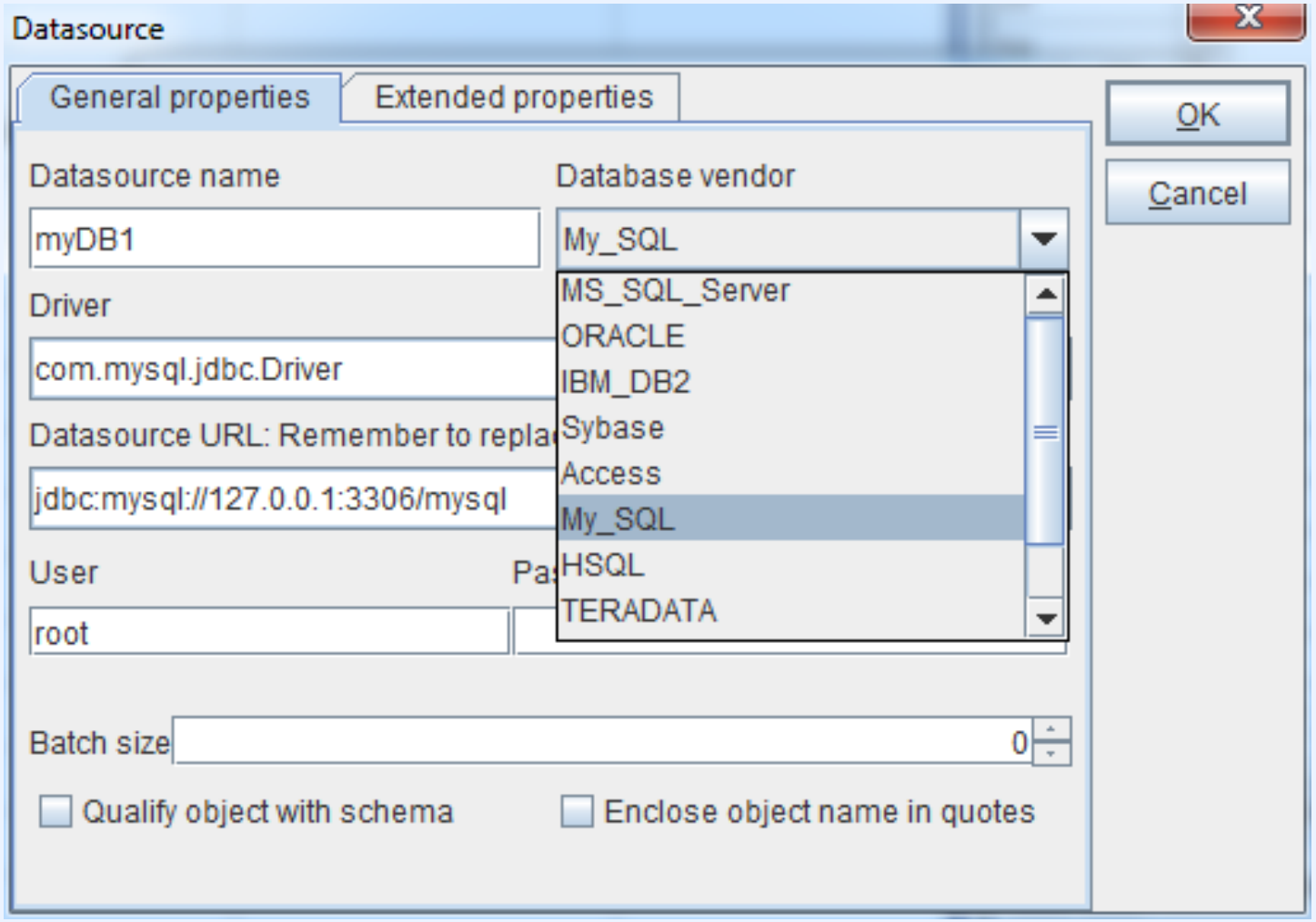
Sno	Cno	Grade
200810121	1	96
200810121	2	93
200810121	3	85
200810122	2	88
200810122	3	90
200810123	4	56
200810123	2	76

	A
1	=mongo_open("mongodb://localhost:27017/raqdb?user=test&password=test")
2	=mongo_shell(A1,"course.find(,{_id:0})").fetch()
3	=connect("myDB1")
4	=A3.query("select * from course2").keys(Sno, Cno)
5	=A3.update(A2:A4, course2, Sno, Cno, Grade; Sno,Cno)

The function update supports inserting records if they do not exist and updating records if they already exist.

B. import into database

Where mydb1 is the data source name, and the JDBC configuration interface is as follows:



The data source uses a JDBC connection, which can support any database. JDBC data source can be connected / closed automatically or manually like mongodb. The former is used here.

C. Import into mongodb

Import data from other data sources into mongodb.
Import the data of MySQL database table Course2 into mongodb set course.

Table course2:

Sno	Cno	Grade
200810121	1	96
200810121	2	93
200810121	3	85
200810122	2	88
200810122	3	90
200810123	4	56
200810123	2	76

	A
1	=connect("mysql")
2	=A1.query("select * from course2")
3	=mongo_open("mongodb://localhost:27017/raqdb?user=test&password=test")
4	=mongo_insert(A3, "course",A2)
5	>mongo_close(A3)

Import the data of MySQL database table Course2 into mongodb set course.

9 Multi source mixed calculation

esProc SPL is a strong computing engine for structured processing. It supports diverse data sources and is easy to integrate. It can help different reporting tools to easily implement such requirements. The following is an example of mongodb + mysql.

EMP is the set of mongodb and cities is the table of MySQL. The field cityid in EMP is logically equivalent to a foreign key, pointing to the cityid field of cities, which has fields such as cityid and cityname. Now you need to query the employees in EMP by time period and display the cityind as cityname.

Set emp:

EID	Dept	CityID	Name	Gender	Salary	Birthday
10	R&D	199	Ryan	M	13000	1976-03-12
100	Sales		Jacob	M	5000	1978-02-12
101	Sales	56	Michael	M	6500	1984-03-29
102	Sales	46	Christian	M	12000	1972-07-25
103	Marketing	34	Madison	F	5000	1976-07-11
104	Marketing	6	Sarah	F	8000	1982-11-17
105	Marketing	4	Tyler	M	6500	1978-04-08
106	Marketing	6	Emily	F	7000	1975-12-05
107	Marketing	2	Madison	F	5000	1981-09-29

Table cities:

CityID	CityName	Population	StateId
1	New York	8084316	32
2	Los Angeles	3798981	5
3	Chicago	2886251	13
4	Houston	2009834	43
5	Philadelphia	1492231	38
6	Phoenix	1371960	3
7	San Diego	1259532	5

9 Multi source mixed calculation

	A
1	=mongo_open("mongodb://localhost:27017/test?user=test&password=test")
2	=mongo_shell(A1,"emp.find({'$and':[{'Birthday':{'$gte':'"+string(begin)+"'}},{'Birthday':{'$lte':'" +string(end)+"'}}]},{_id:0})").fetch()
3	=mongo_close(A1)
4	=myDB1.query("select * from cities")
5	=A2.switch(CityID,A4: CityID)
6	=A5.new(EID,Dept,CityID.CityName:CityName,Name,Gender)
7	return A6

Begin and end in the query criteria are external parameters from the report, representing the start time and end time of birthday respectively.

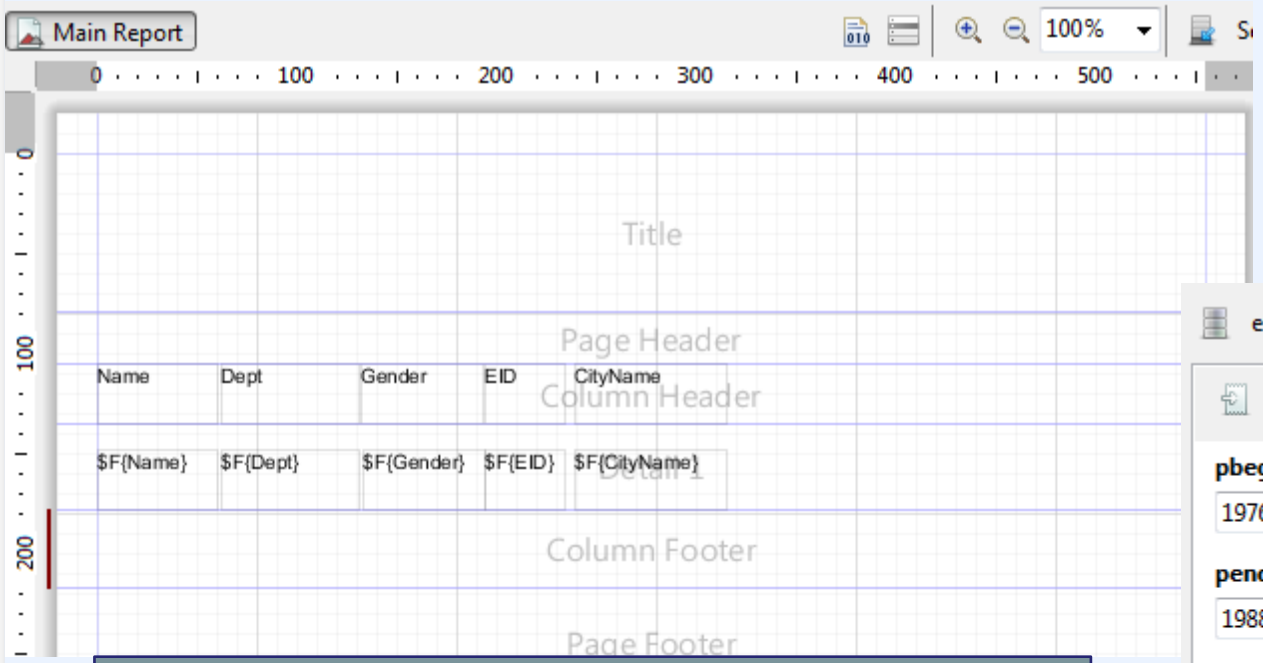
Mongodb and MySQL mixed calculation

Return result:

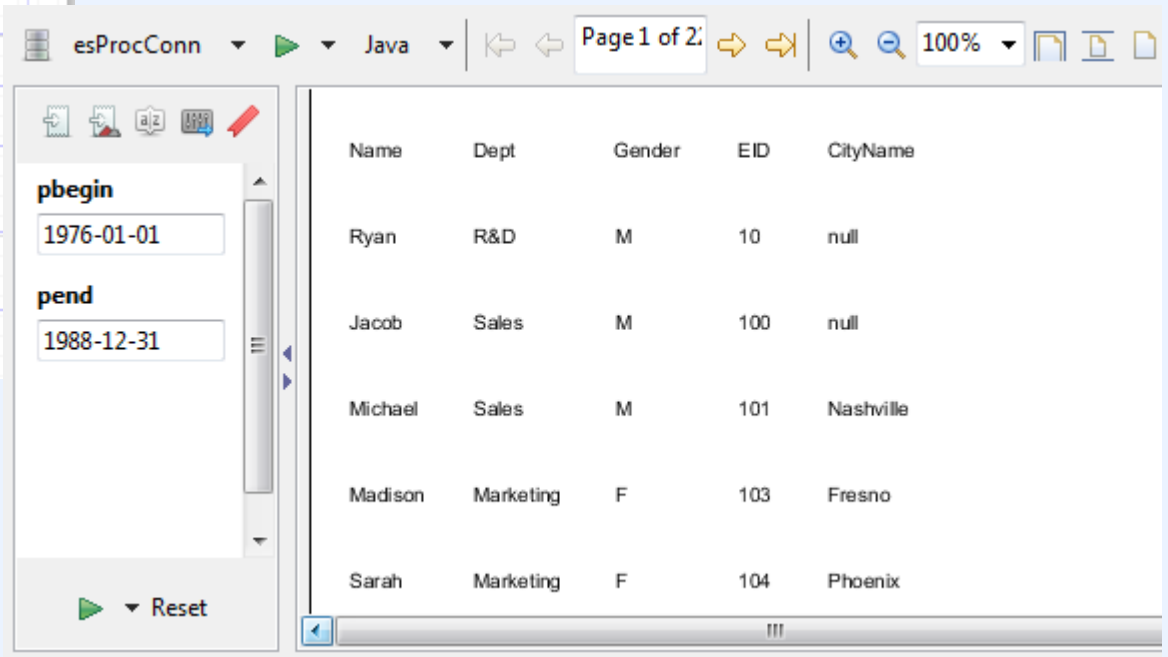
EID	Dept	CityName	Name	Gender
10	R&D		Ryan	M
100	Sales		Jacob	M
101	Sales	Nashville	Michael	M
103	Marketing	Fresno	Madison	F
104	Marketing	Phoenix	Sarah	F
105	Marketing	Houston	Tyler	M
107	Marketing	Los Angeles	Madison	F
108	Marketing	Phoenix	William	M

9 Multi source mixed calculation

esProc provides JDBC interface, and the reporting tool will recognize esProc as an ordinary database. Please refer to the relevant documents for the integration solution.
Take JasperReport as an example to design the report. The sample table is as follows:



You need to define two report parameters, pbegin and pend, which correspond to two parameters in SPL respectively. After preview, you can see the report results:



The method of report calling SPL is the same as that of calling stored procedure. In this case, the script can be saved as mongodbjoin2.dfx. In the SQL designer of jasperreport, mongodbjoin2 $p {pbegin}, $p {pend} can be used to call SPL.

10 Summary

As can be seen from the previous examples of different types, the data structure of mongodb storage is more complex and flexible than that of relational database. If mongodb script is used to realize these functions, there are many functions that need to be familiar with, and it is difficult to master how to combine functions, so it is not easy to skillfully apply it.

Because of the strong and easy-to-use function of esProc SPL, using mongodb as an external data source for SQL or enhanced SQL processing can make up for the shortage of mongodb. It effectively reduces the learning cost of mongodb and the difficulty and complexity in the use and maintenance of mongodb, so that the function of mongodb can be fully demonstrated.

At the same time, mongodb and SPL are integrated with each other and complement each other's advantages, so they will have a broader development space in all professions and trades and big data fields.