集算器

辅助MongoDB计算

润乾软件出品

主要内容

1、序言
2、结构关系
3、SQL式计算
    A、JOIN
    B、子查询
    C、关联表统计
    D、关联数组查找
4、集合并交差
    A、单表union all
    B、union all
    C、并集
    D、交集
    E、差集
5、子集运算
    A、数组查下标
    B、数组过滤查找
    C、topN排序
6、嵌套查询
    A、聚合查询
    B、同结构子文档
    C、多属性子文档
    D、List子文档
7、复杂分组:
    A、交叉汇总
    B、分段分组
    C、分类分组
8、导入导出
    A、导出成csv
    B、导入数据库
    C、导入MongoDB
9、多源混算
10、最后总结

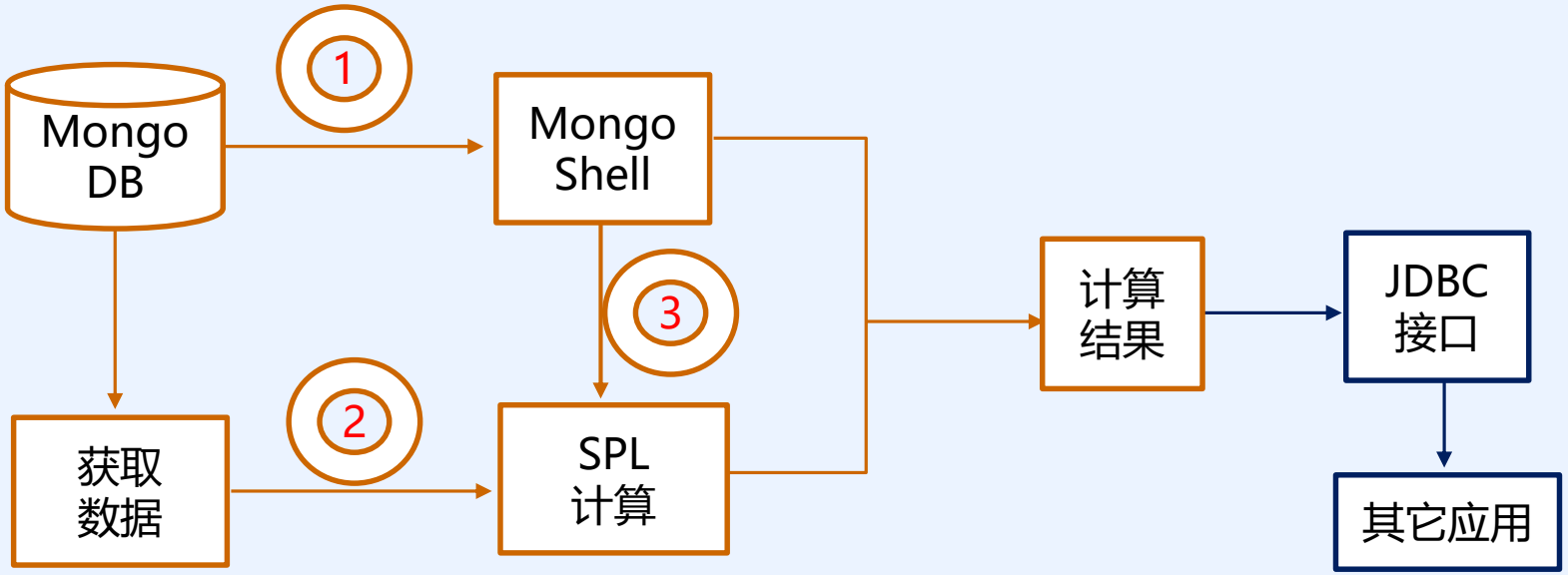
1、序言

MongoDB作为当今非常流行的NoSQL数据库，因为它具有高并发、高性能、灵活数据模型、支持分布式查询，适用于互联网敏捷开发的需求，深受各行各业的欢迎。

MongoDB查询语言接口丰富，有点类似面向对象的查询语言，可以实现类似关系数据库单表查询的绝大部分功能。但是，用其脚本语言有一定复杂度，实现某些计算会面临一些困难。当需要把MongoDB中层次数据从特定角度看作关系型数据进行计算时，集算器SPL可以把MongoDB作为外部数据源进行增强SQL式处理，这样不同应用的需求实现就相对容易多了。

我们从以下几个方面来了解，集算器如何辅助MongoDB计算的。

2、结构关系



在集算器下，实现MongoDB计算主要有三种方式：
1、纯脚本模式：直接通过Mongo shell脚本进行操作求计算结果。
2、SPL模式：将MongoDB当成数据源，利用SPL语言求计算结果。
3、混合模式：在Mongo shell脚本计算结果基础上再经过SPL语言处理后输出。

当然，SPL也对外提供JDBC接口，将其当作数据源，供其它应用访问，方便集成。

3、SQL式计算

基本的关联表用A.F=B.F内外键关联查询，通过$lookup函数实现，在此讨论多键关联情况。

A、JOIN

集合A、B多键值关联，如A.F1 = B.F1 and A.F2 = B.F2。

集合C1：

user1	user2	income
1	2	0.56
1	3	0.26

集合C2：

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

期待结果：

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()

关联表键值关系：
c1.user1=c2.user1 and c1.user2 = c2.user2

从关联表中选择所需要的字段组合成新表

计算结果：

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

B、子查询

子查询就是嵌套在主查询中的查询，即where A.F in(B.F)。用集算器SPL的select函数实现。子查询包括单表运算及关联计算。

B1、子查询单表运算

course集合, 列名为学号，课程号，成绩。

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

要求查找每个学生超过自己选修平均成绩的课程号,成绩。写成SQL是：

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

B1、子查询单表运算

	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()

按学号分组求平均成绩后，查出大于均值的记录

计算结果：

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

B2、子查询关联计算

集合orders为订单数据，employee为员工数据，数据如下：

orader集合：

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集合：

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、子查询关联计算

查出订单信息，其中订单的SELLERID是employee集合中STATE= California的员工eid。写成SQL是：
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)

找出SELLERID在employee中存在的记录

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、关联表统计

对关联表进行count()统计计算。
集合 Progress记录着用户和课程的关系，其courseid字段是外键，指向集合Course的_id字段。需要统计出每门课的人数，其中课程名称需要使用Course的title字段进行显示。

集合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

集合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、关联表统计

	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)

按需生成表数据

popularityCount	title
5	Japanese159
2	Chinese200
1	Political science 280

切换为记录数据

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

D、关联数组查找

从关联表记录数组中查找符合条件的记录，用给定的字段组合成新表，A.F in (B.F)。

集合users：

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

集合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()

查找序表A3的_id值存在于A2中workouts数组的记录，并追加name字段。返回合并的序表

计算结果：

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、集合交并差

A、单表union all

获取嵌套结构中的某键值数据，同时显示此值来自哪个文档。

netmap集合：

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

期待结果：

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

A、单表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)		

数据记录追加合并

计算结果：

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

Name与所在记录来源合并

name	source
Hanna	Twitter
Walter	Twitter

B、union all

多个同结构集合进行并归计算，MongoDB没有提供相应的接口来实现。
并归计算主要包括：UNION ALL， UNION(并集，数据去重)，INTERSECT(交集)，MINUS(差集)。

集合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

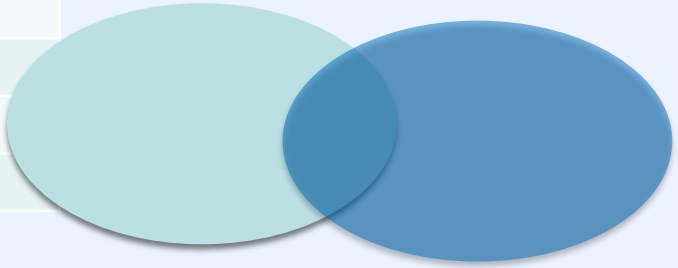
集合emp2：

_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

并、交、差运算有两种，
一种是全行对比的，
一种是只比关键字的。

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()



集合求**union all**， 返回所有的查询记录

合并后数据：

_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、并集

	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()

全行对比求并集

或键值对比求并集

集合求并集，返回去除重复记录后的查询结果

合并后数据：

_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

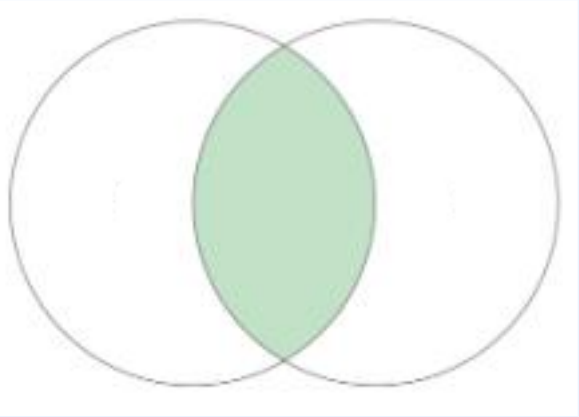
D、交集

	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()

全行对比求交集

或键值对比求交集

集合求交集，返回查询结果中相同的部分



合并后数据：

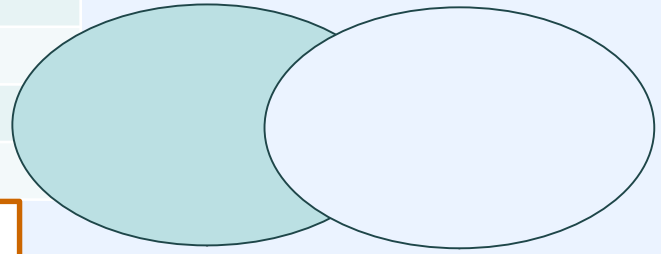
_id	NAME	STATE	HIREDATE	DEPT	SALARY
1	Ashley	New York	2008-03-16	Finance	11000

E、差集

	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()

全行对比求差集

或键值对比求差集



集合求差集，返回在emp1结果中包含但emp2结果中不包含的数据

合并后数据：

_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、子集运算

A、数组查下标

通过元素的值来查询下标，知道排列在第几个位置。
集合users中，保存了姓名和朋友（数组）。朋友数组中的人名是按照排名顺序保存的。

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

mongodb查找指定排名的人名，例如查找jim的朋友当中，排名第一的人名：

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

但是无法查找jim的朋友当中，“luke”的排名数值。

A、数组查下标

	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

根据过滤条件获取到friends数组，pos函数获取luke所在的位置。

B、数组过滤查找

从指定的数组中查找符合条件的记录。
如数组为：["Chemical", "Biology", "Math"]，查询选修课包含["Chemical", "Biology", "Math"]的同学。

_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()

查询数组中符合条件的记录

计算结果：

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

C、topN排序

统计需求：将大量的对象进行排序，然后只需要取出前N名作为排行榜的数据。
集合last3有两个字段：variable和timestamp，这里首先按variable分组，然后在每组文档中选出timestamp最晚的3个，最后再从这些文档中找到timestamp最早的1个。

_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排序

	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)	

选出B4中
最早的文档

将B3选出文档中
timestamp最晚的3个
不断地追加到B4中

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

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

6、嵌套查询

从内嵌数据结构对象中获取子文档数据。

A、聚合查询

针对带有嵌套结构的数据进行聚合查询。
下面要统计每条记录的income，output的数量和。

	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()

Income, output下的值转换成序表后求和

ID	INCOME	OUTPUT
1	1600.0	1720.0
2	3550.0	1800.0

集合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 }

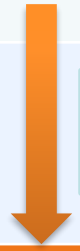
B、同结构子文档

集合storage的文档下键值(字段值)是相同结构的子文档

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

查找category==“food”的记录

	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)



将符合条件的记录合并成序表返回

category	name
food	apple
food	banana

C、多属性子文档

将多个document下的某个字段合并，形成新的记录。
集合C1数据如下：

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
…		

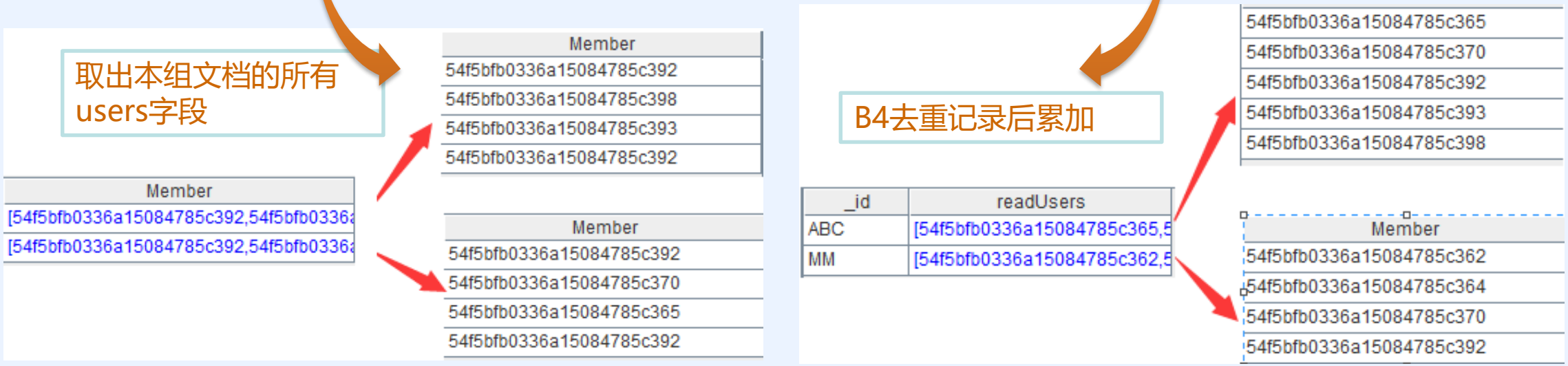
C、多属性子文档

要求按name分组，每组数据是相同的name对应的子文档中的users字段，且数据不能重复。最后的计算结果类似下面这样：

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

C、多属性子文档

	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子文档

```
{
        "_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"
                    ] }
                }
        ]}
```

集合Cbettwen含有多级子文档，其中dataList是List型，含有多个字符串，每个字符串由多个数字组成。需要找出符合如下条件的字符串：第1个数字大于6154并小于等于6155。

D、List子文档

	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)

将dataList中每条记录拆分得到首个字符串的值，筛选其值符合条件的记录。

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

7、复杂分组

A、交叉汇总

交叉汇总是数据统计中一种实用的分类统计。一般情况下，我们通过某一变量分组后作为行，然后用其它变量或变量的组合作为列，形成数据库表进行统计分析。例如下面的表结构：

		成绩				
学校	学科	1	2	3	4	5
A	Sub1	人数	人数	…	…	…
	Sub2	人数	…			
B	Sub1	人数				
	Sub2	人数				

如果将学科、成绩合并，则可进一步演化为：

	学科-成绩					
学校	sub1-1	sub1-2	…	sub1-5	sub2-1	…
A	人数	…				
B	人数					

A、交叉汇总

用MongoDB能够比较清晰、自然地存储类似的数据，但要实现交叉汇总却比较困难。
Student集合记录了学校、学生名称、学科及成绩，数据如下：

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

期望汇总结果：

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、交叉汇总

	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)

每组内部成绩按照[1,2,3,4,5]的序列对齐分组，并求出每个分组序列长度

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

按学科与成绩组合成列存放到位

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、分段分组

按指定的分段进行分组统计。
统计各段内的记录数量。下面按销售量分段，统计各段内的数据量，数据如下：

_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

分段方法：0-3000;3000-5000;5000-7500;7500-10000;10000以上。

计算结果：

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()

根据SALES区间分组统计员工数

C、分类分组

统计分类项下的总数及各子项数。
分类方法：按addr分类统计book总数及其下不同的book数量。

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()

将A4中的Total按addr键值关联后合并到序表

计算结果：

_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、导入导出

将非结构化的MongoDB数据转换成结构化的数据，导出成csv文件或导入数据库中，也可将其它数据源的数据导入MongoDB。

A、导出成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
……		

对car字段进行拆分成行后组成序表，函数conj对序表数据合并，导出生成csv文件

B、导入数据库

将MongoDB数据导入关联数据库中。
将MongoDB中集合course数据导入MySQL数据库表course2中。

集合course:

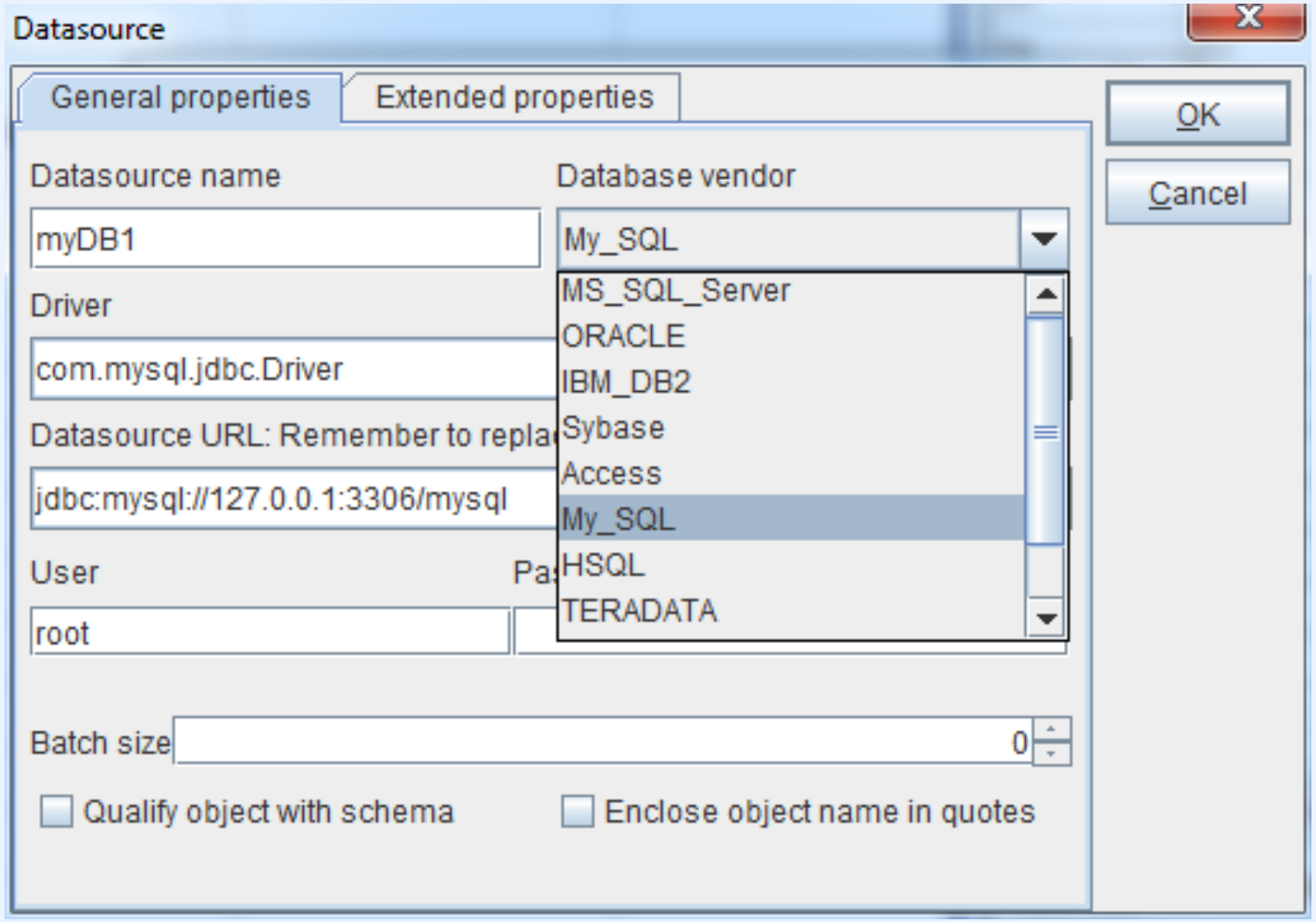
Sno	Cno	Grade
200810121	1	96
200810121	2	93
200810121	3	85
200810122	2	88
200810122	3	90
200810123	4	56
200810123	2	76

	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)

函数update支持当记录若不存在则插入，记录若已存在则更新。

B、导入数据库

其中myDB1是数据源名称，JDBC配置界面如下：



数据源使用的是JDBC连接，它可支持任意数据库。JDBC数据源可以自动连接/关闭，也可以像MongoDB那样手工连接/关闭，这里采用前者。

C、导入MongoDB

将数据从其它数据源导入MongoDB中。
将MySQL数据库表course2的数据导入MongoDB集合course中。

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)

将MySQL表course2导入MongoDB集合course

9、多源混算

集算器SPL语言面向结构化处理的强计算引擎，支持多样性数据源，集成简单，可以协助不同报表工具方便地实现此类需求，下面用MongoDB+MySQL例子说明。

emp是MongoDB的集合，cities是MySQL的table，emp中的字段CityID逻辑上相当于外键，指向cities的CityID字段，cities有CityID、CityName等字段。现在需要按时间段查询出emp中的员工，并将CityID显示为CityName。

集合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

表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、多源混算

	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

查询条件中的**begin**和**end**是来自报表的外部参数，分别表示Birthday的起始时间和终止时间

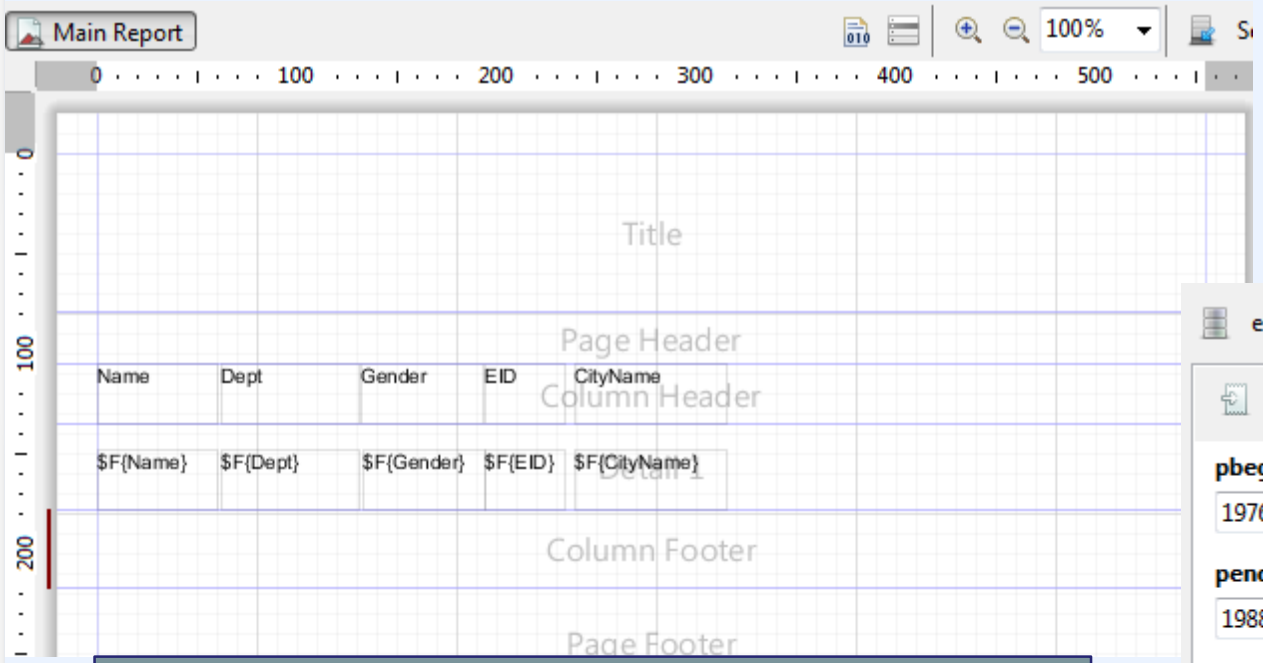
MongoDB和MySQL混合运算

返回结果:

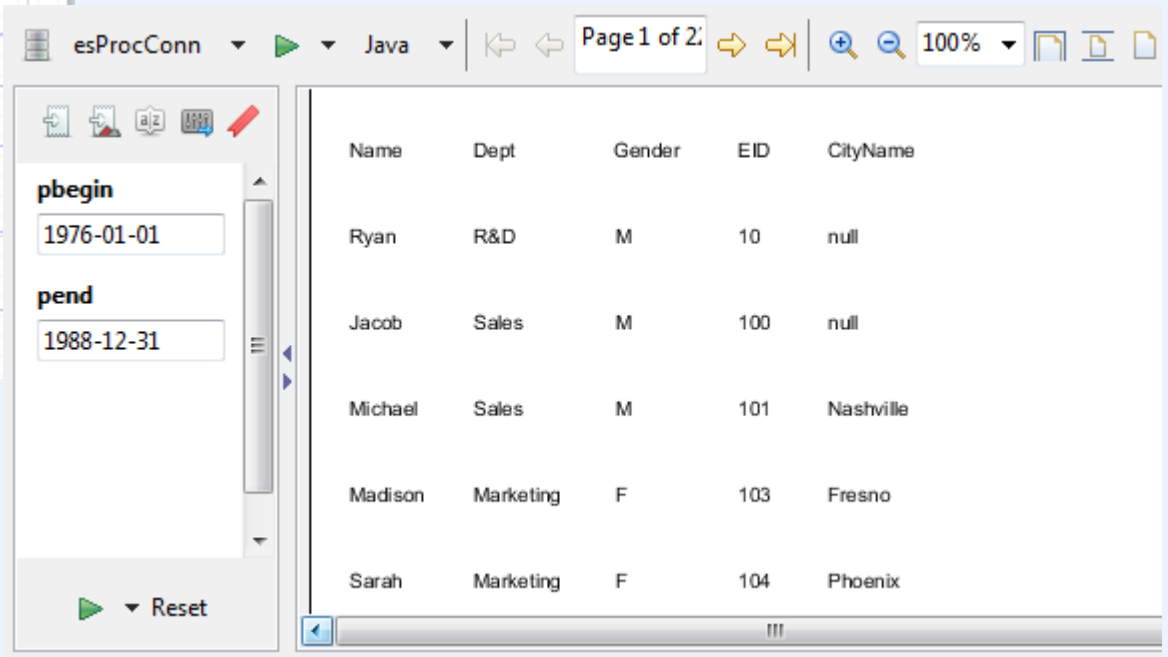
EID	Dept	CityName	Name	Gender
10	R&D		Ryan	M
100	Sales		Jacob	M
101	Sales	Nashville	Michael	M
103	Marketing	Fresno	Madison	F
104	Marketing	Phoenix	Sarah	F
105	Marketing	Houston	Tyler	M
107	Marketing	Los Angeles	Madison	F
108	Marketing	Phoenix	William	M

9、多源混算

集算器提供JDBC接口，报表工具会将集算器识别为普通数据库，集成方案请参考相关文档。
以JasperReport为例设计报表，表样如下：



需要定义两个报表参数Pbegin、Pend，分别对应SPL中的两个参数。预览后可以看到报表结果：



报表调用SPL的方法和调用存储过程一样，此例中可以将本脚本保存为mongodbJoin2.dfx，在JasperReport的SQL设计器中可以用mongodbJoin2 $P{pbegin},$P{pend}来调用。

10、最后总结

从前面的多个不同类型的示例可见，MongoDB存储的数据结构相对关系数据库更复杂灵活。若用MongoDB脚本来实现这些功能，需要熟悉的函数也不少，函数之间如何组合应用也比较难掌握，要熟练应用它并非易事。

由于集算器SPL的功能强和易用性，把MongoDB作为外部数据源进行SQL或增强SQL式处理，能弥补MongoDB这方面的不足，它有效降低了MongoDB学习成本及MongoDB使用维护中的难度与复杂度，让MongoDB的功能得到更充分的展现。

同时，MongoDB与集算器SPL 相互融合，优势互补，在各行各业、大数据领域将拥有更广阔的发展空间。