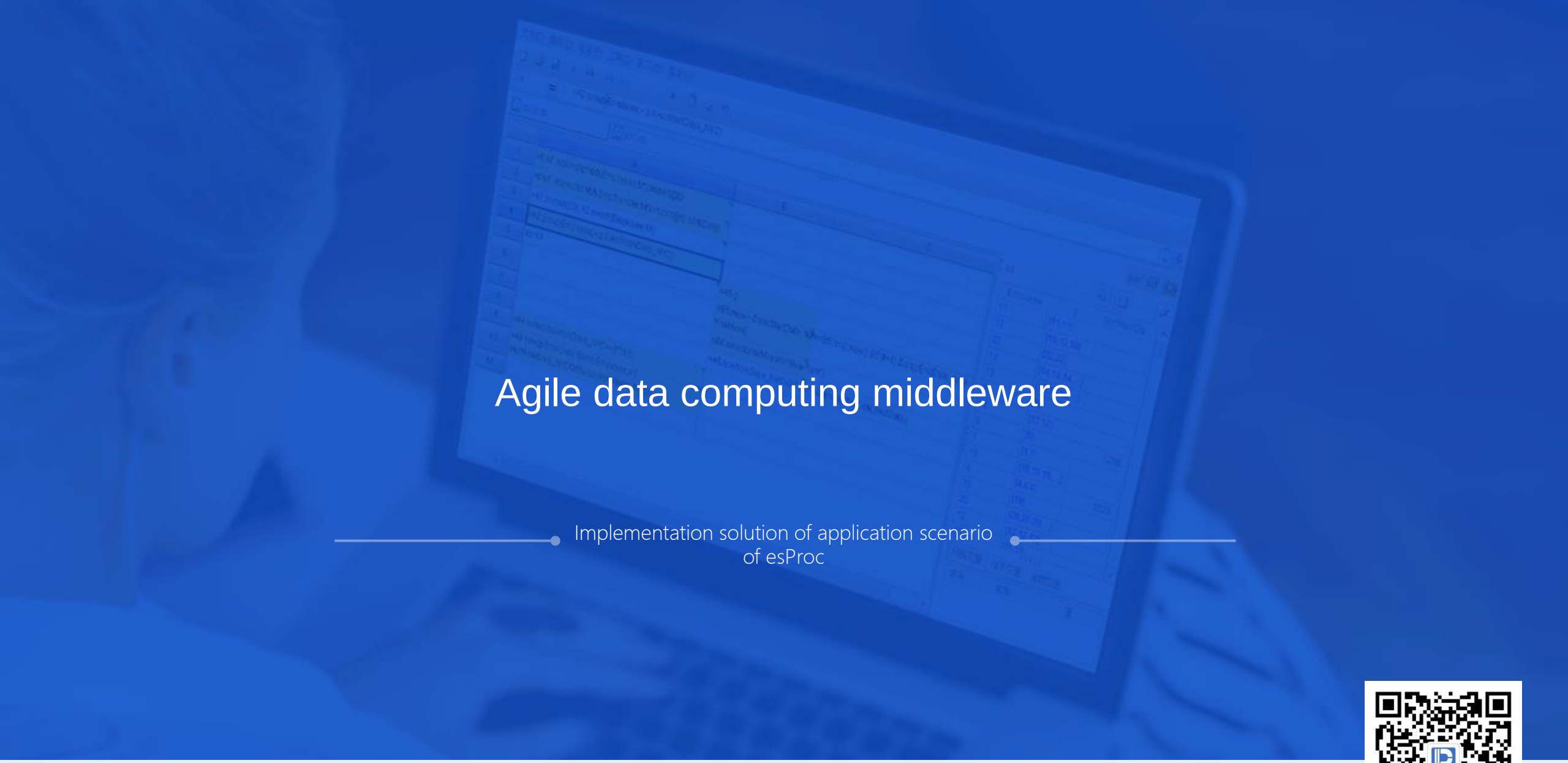


Agile data computing middleware

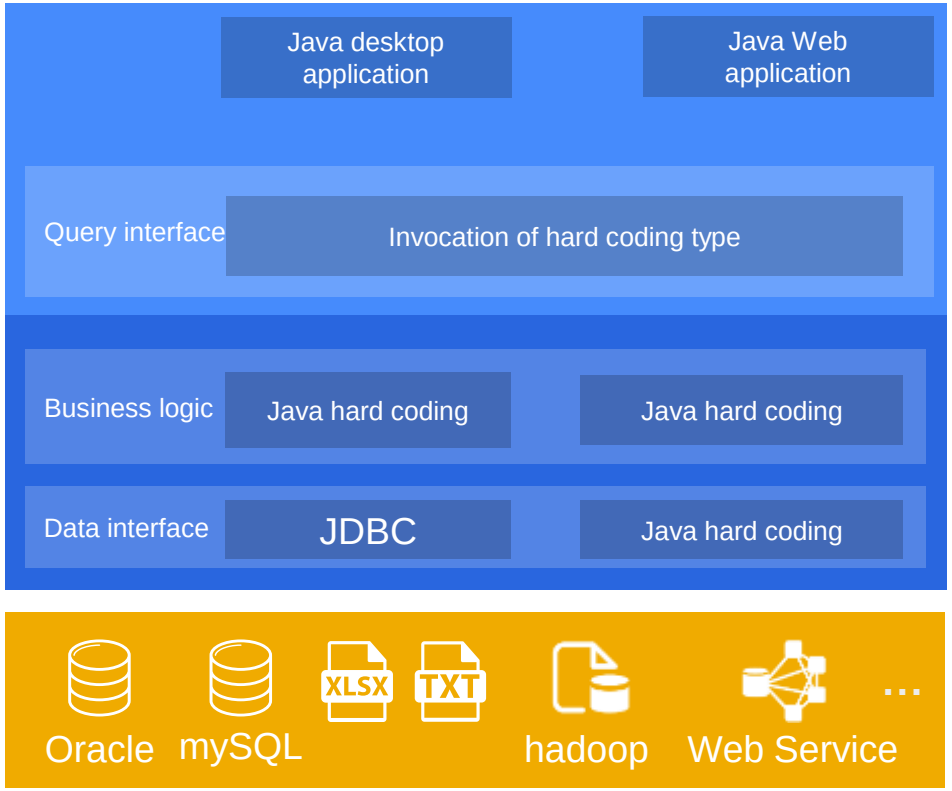
Implementation solution of application scenario of esProc

www.raqsoft.com

Architecture contrast

Computing middleware: a programmable general software between application and data, which can perform computing independently. It is often used to solve problems such as loose coupling, high performance, special source computing, multi-source hybrid computing, complex logic, etc.

Traditional hard coding solution

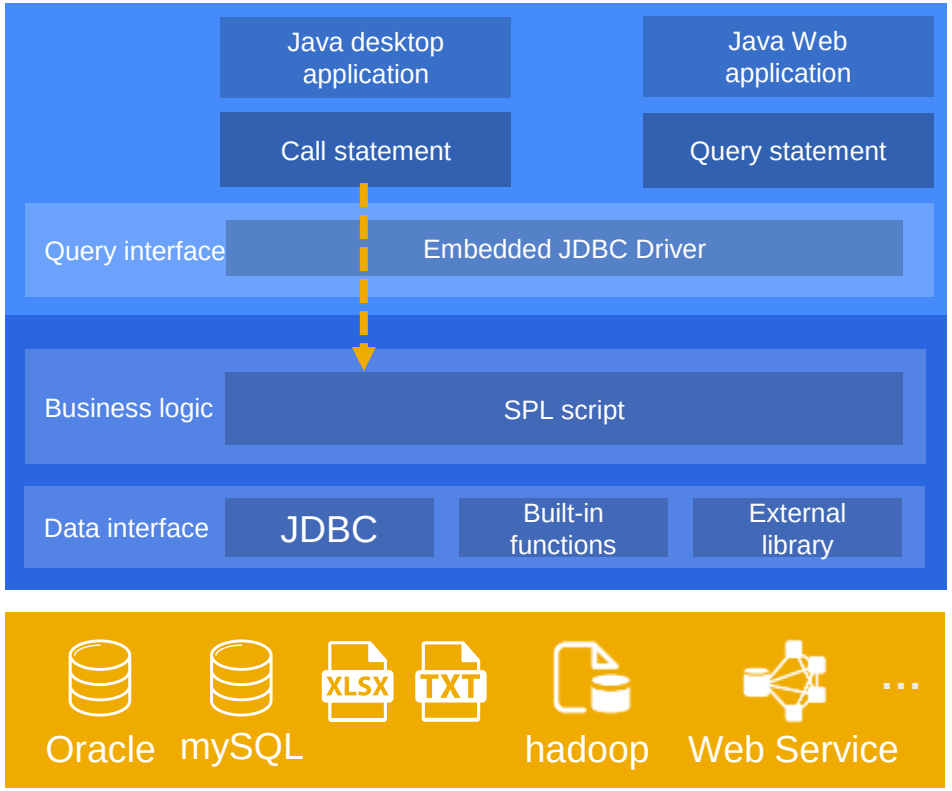


Front end application

Computing middleware

Source data

esProc solution



Note： This article focuses on the mainstream embedded and Java application architecture, and esProc also supports the independent and non Java application architecture.

Characteristic contrast

Similarity

- Consistent logical architecture
- Consistent implementation process

Main difference

- Front end application query interface
  esProc solution：JDBC driver
  Hard coding solution: Hard coding of basic class library

- Business logic implementation
  esProc solution： Structured computing function
  Hard coding solution ： Hard coding of basic class library

- Non database access interface
  esProc solution： Access library function
  Hard coding solution ： Hard coding of basic class library

Advantages

esProc underlying capability	Middleware feature	Advantage
Independent calculation script	Separation of algorithm and main application Hot switching possible	Reduce the coupling between algorithm and Application
Complete computing class library	Replace stored procedure Independent of specific databases	Reduce the coupling between Database and Application Reduce database pressure
Direct hybrid computing of external source and database	Reduce intermediate table	Reduce the difficulty of development and maintenance Reduce database pressure
Data in the database can be moved out as files	Reduce intermediate table	Reduce database pressure
Support discrete dataset, ordered set, process calculation, high performance algorithm	Can simplify complex calculation	Improve development efficiency

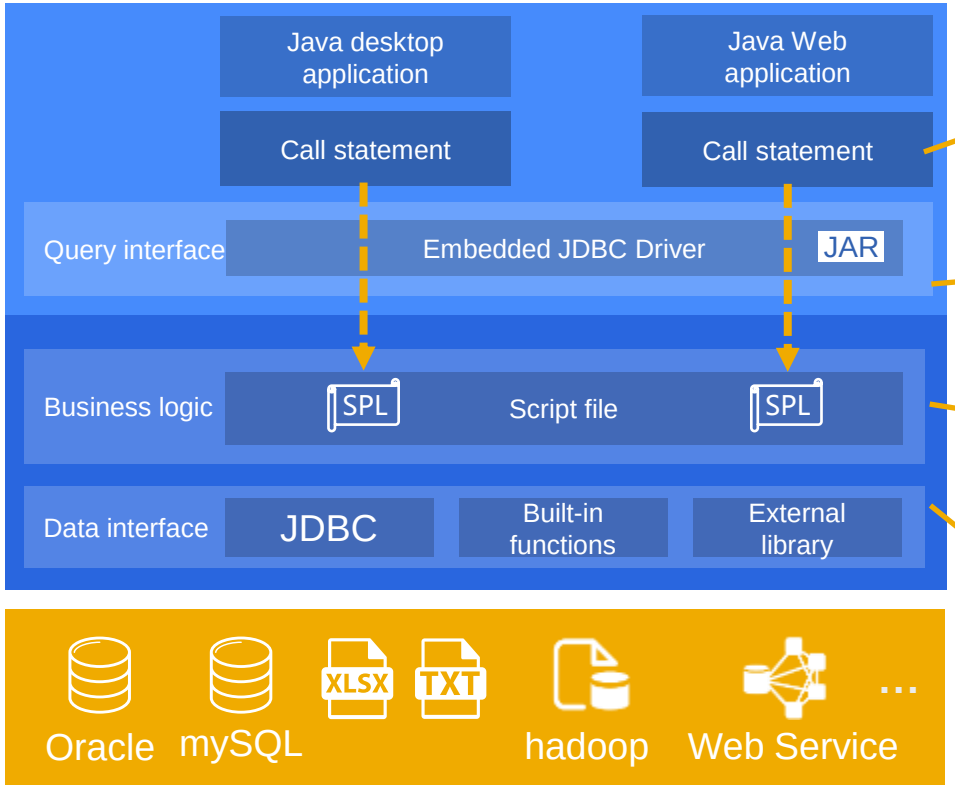
CONTENTS

01

Java application integration

Integration steps

Java application integrates esProc and calls the SPL script file.



4. Execute the call statement in the Java code, call the SPL file through the JDBC of esProc, and the esProc JDBC interprets and executes the SPL file, and returns the calculation result.

1. Deploy the esProc driver jar packages in Java application.

3. Write specific business algorithm in SPL file and store it in file system.

2. Configure the running environment in raqsoftconfig.xml, such as the jdbc driver and URL of the data source (not esProc itself).

Note：For esProc deployment and JDBC configuration, please refer to http://doc.raqsoft.com/esproc/tutorial/jdbcbushu.html

For JAVA calls esProc，please refer to http://doc.raqsoft.com/esproc/tutorial/bjavady.html

Deploy esProc JDBC

The esProc JDBC provides an open and friendly interface for programmers to call esProc Middleware in JAVA applications.

Deploy method: Copy driver to Java application's classpath.

esproc_bin_xxxx.jar	esProc computing engine and JDBC driver package
icu4j_60.3.jar	Deal with internationalization
jdom-1.1.3.jar	Parse configuration file
raqsoftConfig.xml	Running environment configuration file

Note: For deploy details, please refer to http://doc.raqsoft.com/esproc/tutorial/jdbcbushu.html

Configure running environment

The running environment of esProc is stored in raqsoftconfig.xml, which is mainly the data source information of business database or data warehouse, and also includes authorization, script file directory, character set, external library, etc.

Configuration information of an Oracle data source:

```
<DB name= “orcl">
        <property name="url" value="jdbc:oracle:thin:@192.168.1.8:1521:runqian"/>
        <property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
        <property name="type" value="1"/>
        <property name="user" value="ydxx"/>
        <property name="password" value="password"/>
        <property name="batchSize" value="0"/>
……
</DB>
```

Note：The JDBC package of the data source is also deployed in the classpath.

For detail configuration of raqsoftConfig.xml，please refer to http://doc.raqsoft.com/esproc/tutorial/jdbcbushu.html

Similarity

- The deployment method is similar
- Follow the same set of interface specifications

Main difference

- Calling location

esProc JDBC：in java application
Data source JDBC：in esProc script file

- Calling method

esProc JDBC：java code
Data source JDBC ： SPL function

- Effect

esProc JDBC：Access esProc script file to get the calculation results of the business algorithm.
Data source JDBC: Access database / warehouse to obtain raw data.

Script file

The SPL script file is the business middleware algorithm, which is written by the programmer. Generally, it needs to fetch data from the data source, then realize the business algorithm, and finally return the calculation result to the main program by the JDBC.

Example：orcl is database data source, and java application needs to fetch data from sales table. Some of the data are as follows:

ordered	client	sellerid	amount	orderdate
1	UJRNP	17	392	2012/11/2 15:28
2	SJCH	6	4802	2012/11/9 15:28
3	UJRNP	16	13500	2012/11/5 15:28
4	PWQ	9	26100	2012/11/8 15:28
5	PWQ	11	4410	2012/11/12 15:28

The performance of the original single thread fetching is poor. The following is to use esProc as the middleware, use the 8-thread parallel query, and merge the query results. The script file conj.dfx is as follows:

	A	B	C
1	=8.(range(1,100000000L,~:4))	/	
2	fork A1	=connect("dbsource")	/Multi-thread parallel
3		=B2.query@x("select * from sales where orderid>=? and orderid<?",A2(1),A2(2))	/Query in thread
4	=A2.conj()		/Merge and output

Note： The starting and ending range of OrderID is 1-100000000L. A1 code divides it equally into eight time intervals, one for each thread. For example, thread 2 is 12500000-25000000.

Script file

Java application calls script file through esProc JDBC.

```
……
Class.forName("com.esproc.jdbc.InternalDriver");
con=DriverManager.getConnection("jdbc:esproc:local://");
PreparedStatement pstmt = con.prepareStatement(  “call conj()");
ResultSet rs=pstmt. executeQuery()
……
```

The script file can take parameters, such as filtering data by order amount range (minamount, maxamount). The script file is as follows:

	A	B	C
1	=8.(range(1,100000000L,~:4))	/	
2	fork A1	=connect("dbsource")	/Multi-thread parallel
3		=B2.query@x("select * from sales where orderid>=? and orderid<?“ and amount>=? and amount<?, A2(1),A2(2),minAmount,maxAmount)	/Query in thread
4	=A2.conj()		/Merge and output

At this time, java code should be written according to parameter query:

```
…
PreparedStatement pstmt = con.prepareStatement(  “call conj(?,?)”  );
//Can also be written as call conj(4000,8000)
pstmt.setObject(1, 4000);pstmt.setObject(2, 8000);
ResultSet rs=pstmt. executeQuery()
……
```

//For more contents on Java calls esProc, please refer to http://doc.raqsoft.com/esproc/tutorial/bjavady.html

Script file

The data source can also be text, Excel, etc.

The source data is the tab separated text file sales.txt. The first few lines are as follows:

ordered	client	sellerid	amount	orderdate
1	UJRNP	17	392.0	2012-11-02 15:28:05
2	SJCH	6	4802.0	2012-11-09 15:28:05
3	UJRNP	16	13500.0	2012-11-05 15:28:05

Query by parameter, group by client and sum the amount. The script file run.dfx is as follows:

	A	B
1	=file("d:/sales.txt").import@t()	/Open txt file
2	=A1.select(amount>=minAmount && amount<maxAmount)	/Parameter query
3	=A1.groups(client;sum(amount):sAmount)	/Group and aggregate

The above scripts can also be combined into one sentence:

	A
1	=file(“d:/sales.txt”).import@t().select(amount>=minAmount &&amount<maxAmount).groups(client;sum(amount):sAmount)

The calculation result is as follows:

client	sAmount
AVU	482640
AYWYN	455320
BTMMU	496226

Script file

As a comparison, we use Java hard coding (traditional computing middleware) to implement the grouping aggregation algorithm.

```
Comparator<salesRecord> comparator = new Comparator<salesRecord>() {
                public int compare(salesRecord s1, salesRecord s2) {
                        if (!s1.client.equals(s2.client)) {
                                return s1.client.compareTo(s2.client);
                        } else {
                                return s1.ID.compareTo(s2.ID);
                        }
                }
};
Collections.sort(sales, comparator);
ArrayList<resultRecord> result=new ArrayList<resultRecord>();
salesRecord standard=sales.get(0);
float sAmount=standard.value;
for(int i = 1;i < sales.size(); i ++){
                salesRecord rd=sales.get(i);
                if(rd.client.equals(standard.client)){
                        sAmount=sAmount+rd.value;
                }else{
                        result.add(new resultRecord(standard.client,sAmount));
                        standard=rd;
                        sAmount=standard.value;
                }
}
result.add(new resultRecord(standard.client,sAmount));
return result;
```

Script file

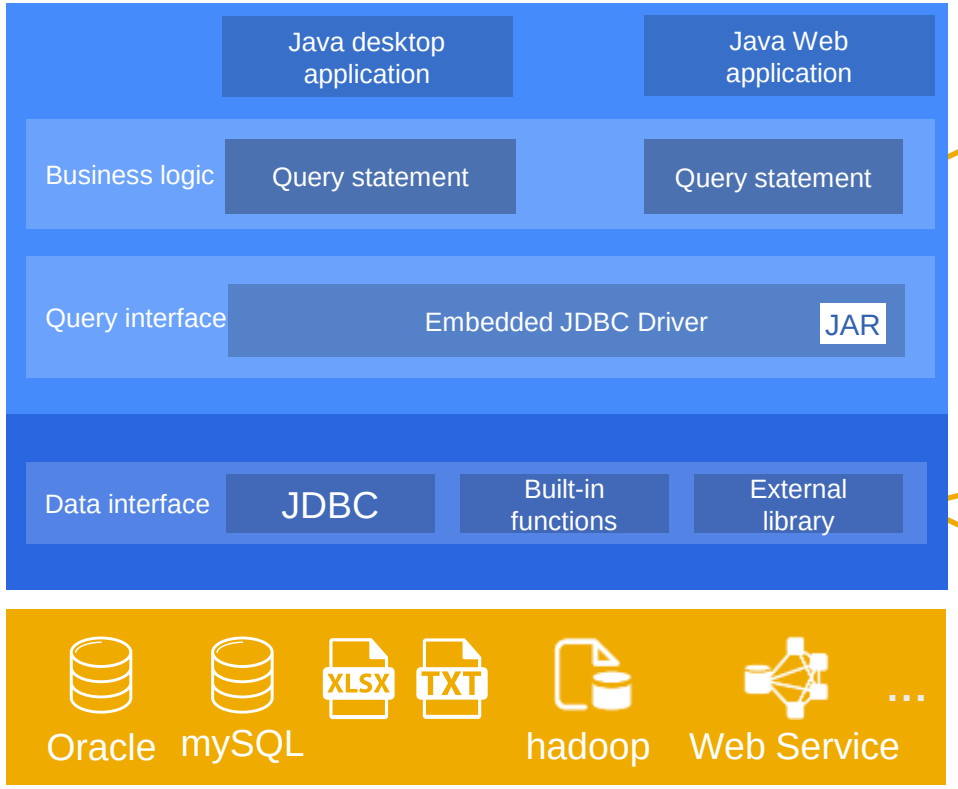
SPL script files are stored in the operating system directory.

Generally, it is recommended to divide directories by functional modules, or by application type, time and version.

```
-Main Directory
    -Customer management
        -run.dfx()
    -Attendance performance
    -Enterprise resource management
        -Financial management
            -Fixed assets statistics.dfx
            -Cash flow query.dfx
            -Bad debt early warning analysis.dfx
        -Inventory management
            ...
```

Query statement

If the script file is simple, the SPL query statement can be used instead of the SPL call statement + script file. The former is similar to Java calling SQL statements, and the latter is similar to calling stored procedures.



3. Directly implement business logic through query statements:

1. Deploy the esProc driver jar packages in Java application.

2. Configure the running environment in raqsoftconfig.xml.

Query statement

SPL call statement + script file

```
…
PreparedStatement pstmt =
con.prepareStatement(  “call run(?,?)”  );
pstmt.setObject(1, 4000);
pstmt.setObject(2, 8000);
ResultSet rs=pstmt. executeQuery()
…
```

SPL query statement

```
…
PreparedStatement pstmt =
con.prepareStatement(“=file(\“d:/sales.txt\”).import@t(
).select(amount>=?
&&amount<?).groups(client;sum(amount):sAmount)");
pstmt.setObject(1, 4000);
pstmt.setObject(2, 8000); );
ResultSet rs=pstmt. executeQuery()
…
```

run.dfx

	A
1	=file(“d:/sales.txt”).import@t().select(amount>=minAmount &&amount<maxAmount).groups(client;sum(amount):sAmount)

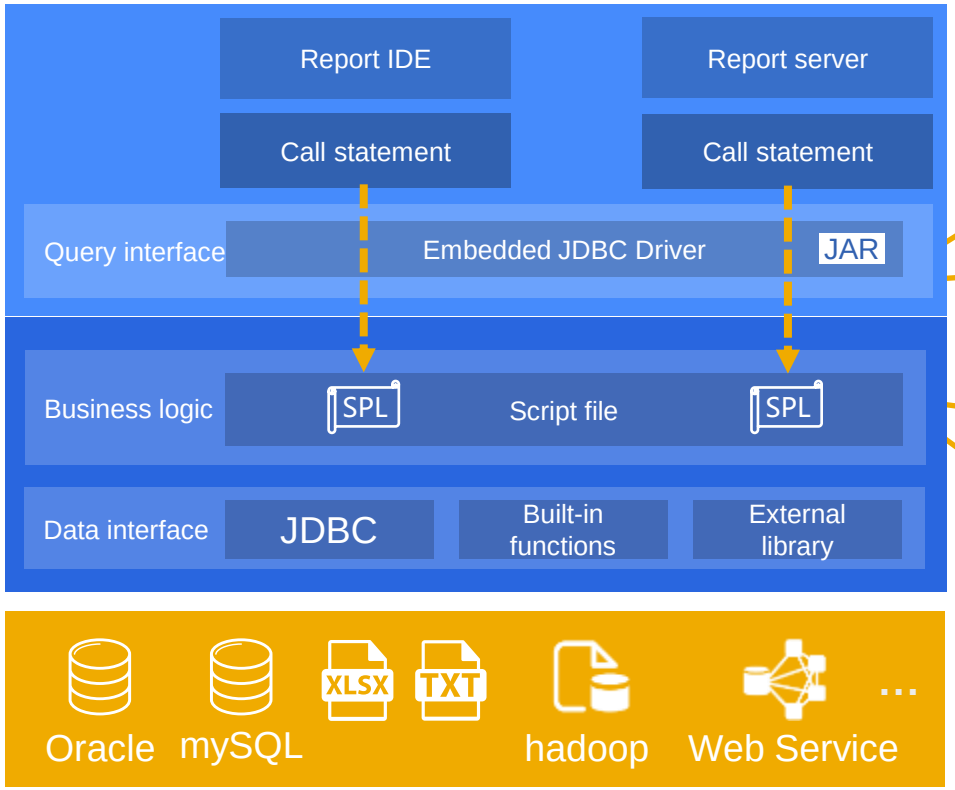
CONTENTS

02

Report integration

Integration steps

Java report tool is a kind of Java application, in which report IDE is generally desktop application and report service is generally web application. Both of them can integrate esProc to realize the computing middleware.



4. Create a new stored procedure dataset in the report, call the esProc script file, or create a new SQL dataset, and write the SPL query statement directly.

1. Deploy the esProc driver jar packages in IDE or web service.

3. Write specific business algorithm in SPL file and store it in file system.

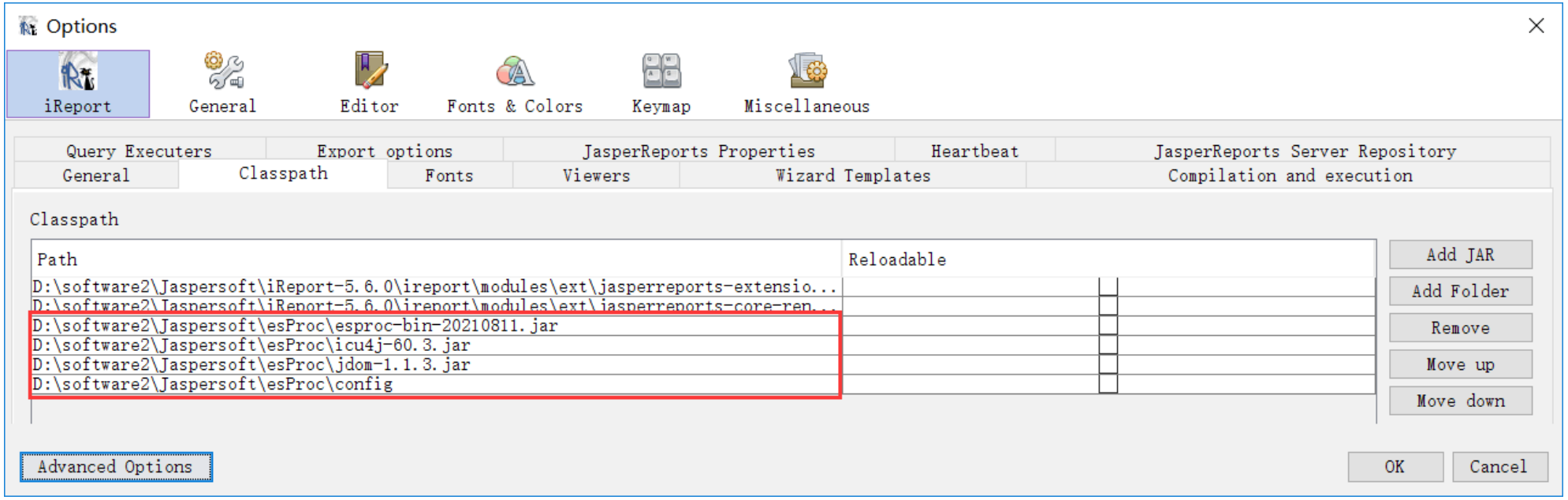
2. Configure the running environment in raqsoftconfig.xml, such as the jdbc driver and URL of the data source (not esProc itself).

Deploy esProc JDBC

The report tool deploys the esProc JDBC in the same way as the common Java application does.

For example, to deploy in the IDE of open source reporting tool BIRT, you only need to copy the esProc driver to:
[Installation directory]\plugins\org.eclipse.birt.report.data.oda.jdbc_4.6.0.v20160607212
The way to configure raqsoftconfig.xml remains the same, and we won't go into details here.

Some reporting tools provide a visual interface, which makes it easier to specify the driver jar package, such as JasperReport.

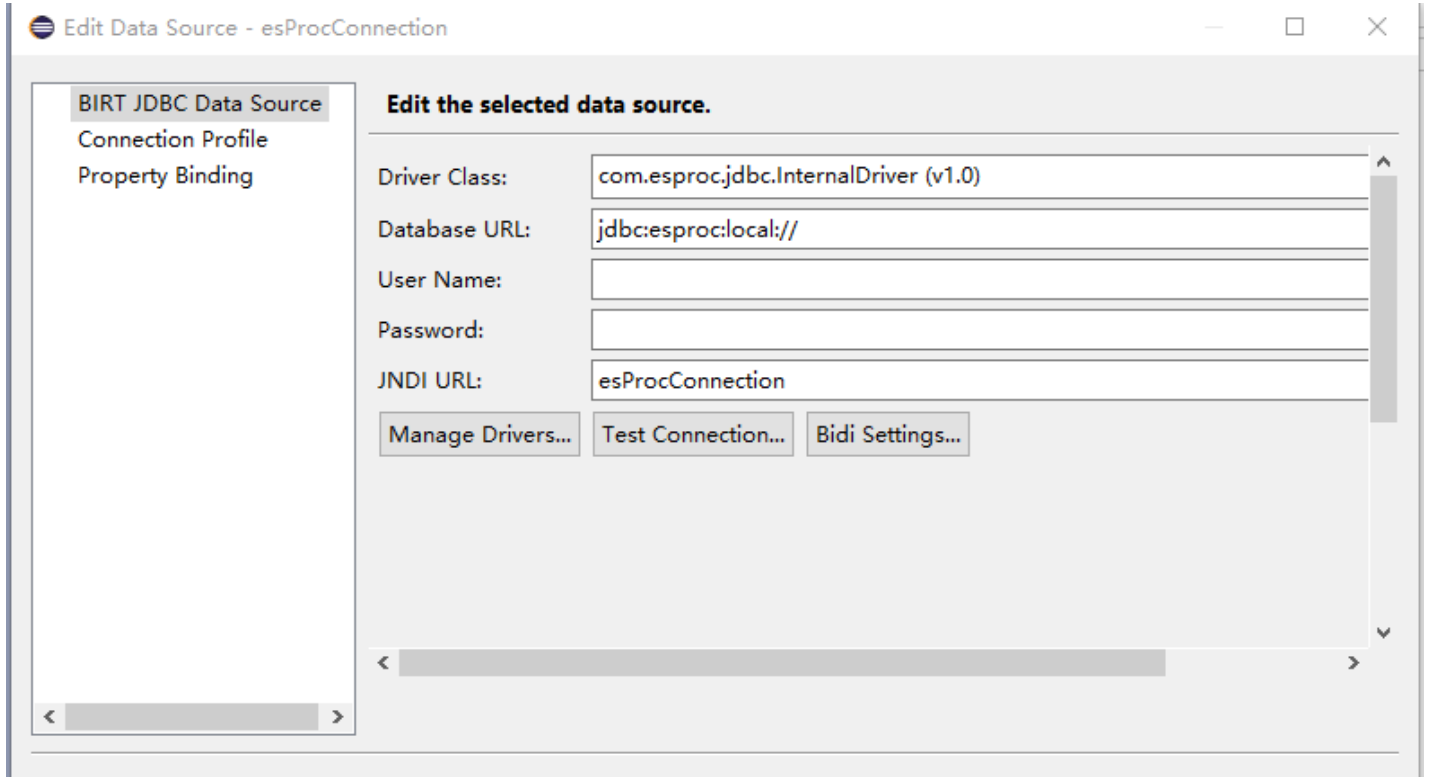


Note: In the above figure, raqsoftconfig.xml is placed in the config directory, and it can also be copied to any jar package.

Report calls esProc script

After writing the script file, you should first establish the data source in the report IDE and point to esProc. Take the BIRT report as an example.

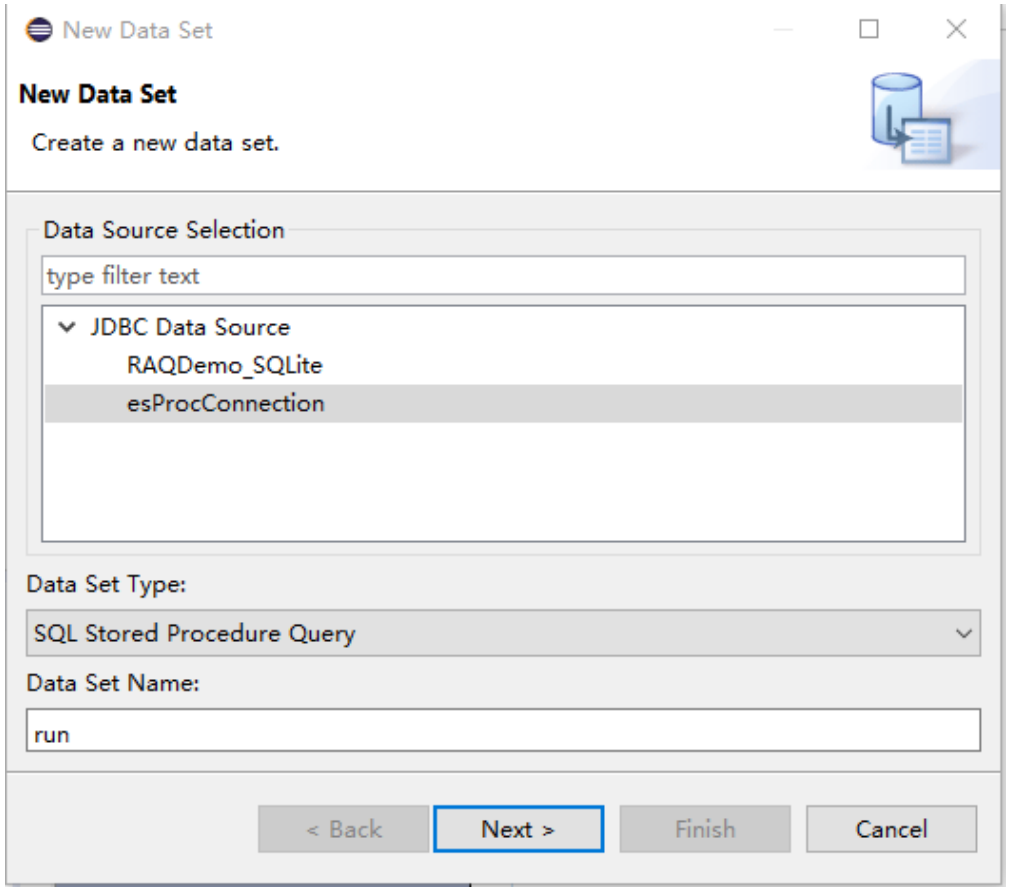
Driver Class selection：com.esproc.jdbc.IntervalDriver (v1.0)
Database URL：jdbc:esproc:local://
JNDI data source name： Free to fill in

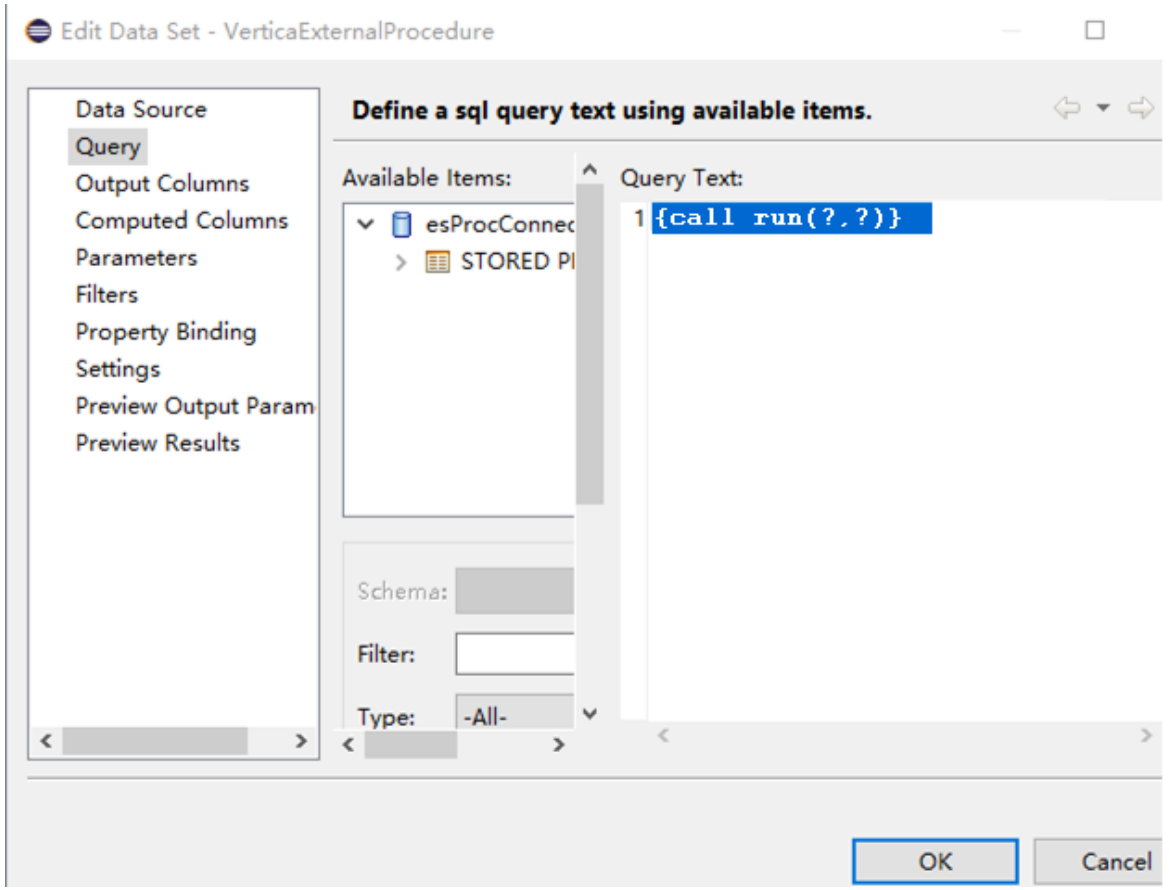


Report calls esProc script

Create a new stored procedure dataset, similar to database stored procedures.

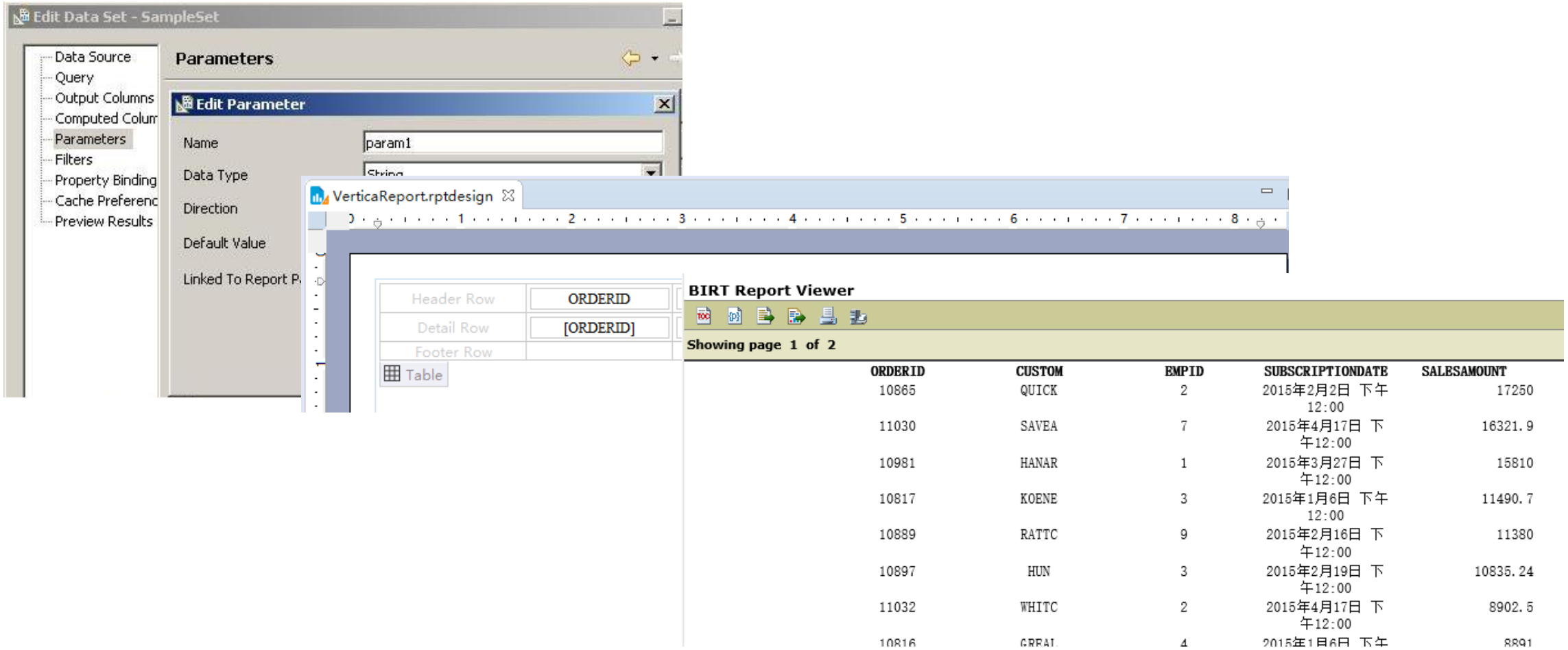
Choose esProc as data source, and call esProc script file.





Report calls esProc script

Set parameters, design reports, and preview reports. The usage is the same as ordinary data sets.



Algorithm example

Horizontal columns: use the column number pColNum as the parameter to place the employee table in horizontal columns in the report.

When pColNum = 2, the report should present:

EID	NAME	DEPT	EID2	NAME2	DEPT2
1	Rebecca	R&D	2	Ashley	Finance
3	Rachel	Sales	4	Emily	HR
5	Ashley	R&D	6	Matthew	Sales
7	Alexis	Sales	8	Megan	Marketing
9	Victoria	HR	10	Ryan	R&D
11	Jacob	Sales	12	Jessica	Sales

When pColNum = 3, the report should present:

EID	NAME	DEPT	EID2	NAME2	DEPT2	EID3	NAME3	DEPT3
1	Rebecca	R&D	2	Ashley	Finance	3	Rachel	Sales
4	Emily	HR	5	Ashley	R&D	6	Matthew	Sales
7	Alexis	Sales	8	Megan	Marketing	9	Victoria	HR
10	Ryan	R&D	11	Jacob	Sales	12	Jessica	Sales
13	Daniel	Finance	14	Alyssa	Sales	15	Alexis	Sales
16	Christopher	Production	17	Hannah	Marketing	18	Jonathan	Administration

Using script file to implement the algorithm

	A	B	C
1	=demo.query("select EId,Name,Dept from employee ")		
2	=A1.step(pColNum,1)		/Take the first column，as intermediate result
3	for to(2:pColNum)	=A1.step(pColNum,A3)	/Take the Nth column
4		=A2=A2.join(#,B3:#,EID:${"EID"/A3}, NAME:${"DEPT"/A3}, DEPT:${"DEPT"/A3})	/Add column N to the right of the intermediate result as the new intermediate result
5	return A2		/Return the final result

Note： Macro in the dynamic syntax of esProc is used in the algorithm，please refer to http://doc.raqsoft.com/esproc/tutorial/huoyongzifuchuan.html#_141
The algorithm uses the method of join by sequence number， please refer to http://doc.raqsoft.com/esproc/func/join.html

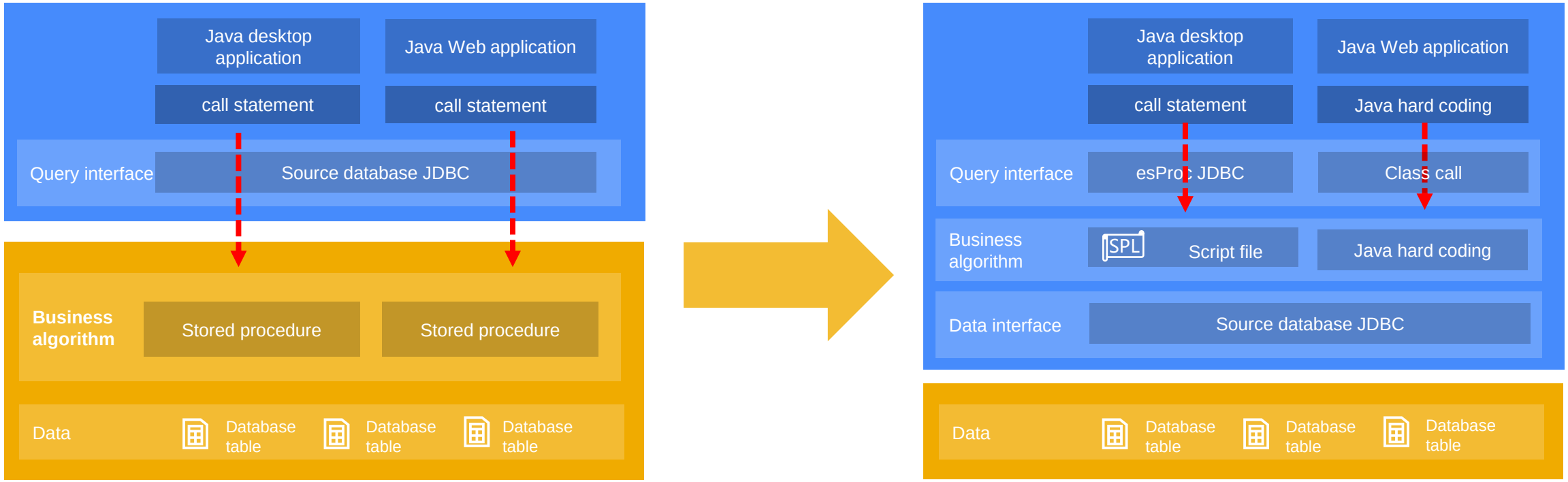
CONTENTS

03

Stored procedure outside database

Implementation ideas

Instead of the stored procedure of the database, esProc script is used to decouple the business algorithm and the application program.



Algorithm example

Switch data source with the same structure

History and current are two databases of the same type with the same structure but different data. When querying and calculating, the middleware needs to select which database to use through parameters. The following is implemented with esProc switch.dfx script.

	A	B
1	=${pSource}.query(“select * from sales”)	

pSource is a macro parameter representing the data source name. If the pSource value in the Java program is "history", the query is executed on history database, that is:

```
PreparedStatement pstmt = con.prepareStatement(“call switch(?)");
pstmt.setObject(1, “history”);
pstmt.execute()
```

The calculation result is shown on the right:

orderid	client	sellerid	amount	orderdate
1	UJRNP	17	392	2012/11/2 15:28
2	SJCH	6	4802	2012/11/9 15:28
3	UJRNP	16	13500	2012/11/5 15:28
4	PWQ	9	26100	2012/11/8 15:28
5	PWQ	11	4410	2012/11/12 15:28

If the pSource value is ”current”， the calculation result is different.

orderid	client	sellerid	amount	orderdate
982	SJCH	12	10900	2019/7/12 15:28
983	GLH	13	8330	2019/7/14 15:28
984	SJCH	19	5684	2019/7/20 15:28
985	YZ	14	27100	2019/7/13 15:28
986	HANAR	10	11100	2019/7/15 15:28

Algorithm example

Switch data source with different structure

Application systems may be migrated between multiple databases, such as Mysql to Oracle, with different database structure. The standard SQL independent of the database is used in development, and the SQL statement does not need to be modified during migration. Only the database type is passed into the script file as a parameter, which can be translated into a specific database SQL. The following is the script file run.dfx.

	A	B
1	select left(client,2),year(orderDate) y,sum(amount) sAmount from sales group by left(client,2),year(orderDate)	/SPL standard SQL
2	=A1.sqltranslate(sqlType)	/Translate standard SQL into SQL of specified database according to parameters
3	=connect@l("dbsource").query@x(A2)	/Execute translated SQL

If the sqltype value in the Java program is "MySQL", the SQL in A2 will be translated as:

select left(client,2),year(orderDate) y,sum(amount) sAmount from sales group by left(client,2),year(orderDate)

If sqltype is “Oracle”, the translation result of A2 is:

select left(client,2),EXTRACT(YEAR FROM orderDate) y,sum(amount) sAmount from sales group by left(client,2),EXTRACT(YEAR FROM orderDate)

Algorithm example

Dynamic alignment

For many algorithms databases are difficult to implement, or some databases are difficult to implement. For example, the external parameter pclient is a dynamic list of major customers. Please count the order amount in the order of the list. If pclient = ["HL", "MIP", “SJCH"], the calculation result should be as follows:

client	samount
HL	305320
MIP	397000
SJCH	500298

Use esProc script file to implement:

	A
1	=connect@l("dbsource").query@x("select client,sum(amount) sAmount from sales where client in(?) group by client ",pclient)
2	=A2.align(A1,client)

CONTENTS

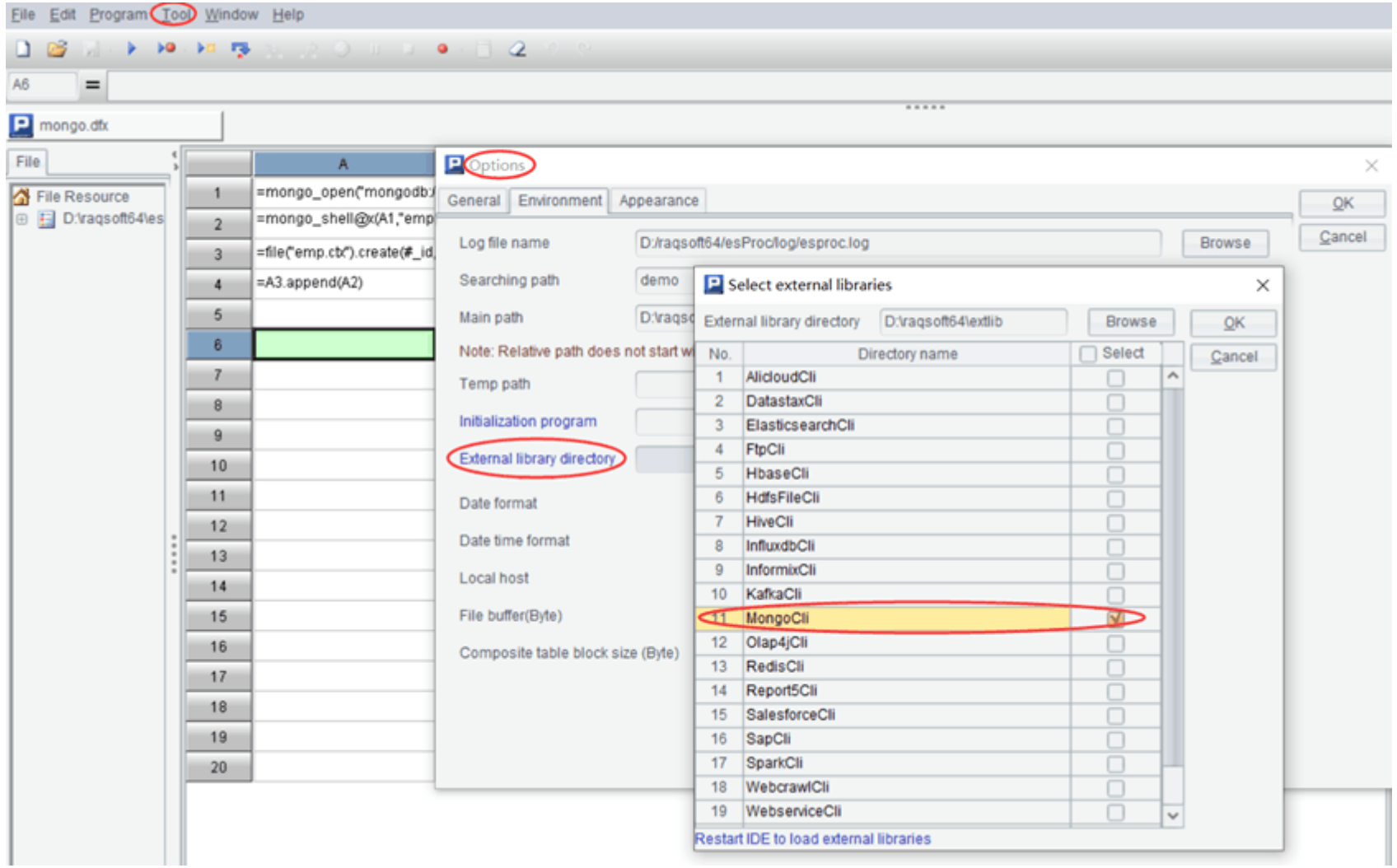
04

Diversified data sources

Special data sources

Scenario: for special data sources such as WebService, mongodb, hive, etc., esProc provides an external library interface. With the jar package provided by the data source, it can realize a convenient and fast special source computing middleware.

Take mongodb for example, first configure the external library



Special data sources

Put mongodb's jar package in the mongodb directory of the external library, as follows:

In the script file, use the shell command to query the EMP collection, complete the query and group calculation.

	A	B
1	=mongo_open("mongodb://192.168.1.7:27017/mydb")	/Connect
2	=mongo_shell@x(A1,"emp.find()")	/Query
3	=A2.groups(department;count(empid):total)	/Group and aggregate

Extended reading： For more external library usage, please refer to http://doc.raqsoft.com/esproc/func/wbk.html

SQL syntax

For text and excel, esProc provided built-in functions to access, such as the previous example: text grouping summary.

	A	B
1	=file("d:/sales.txt").import@t()	/Open txt file
2	=A1.groups(client;sum(amount):sAmount)	/Group and aggregate

In addition to the built-in functions, esProc also provides SQL syntax, which can access the text in a more convenient way. The above script can be written as a SQL in Java:

```
……
Class.forName("com.esproc.jdbc.InternalDriver");
con=DriverManager.getConnection("jdbc:esproc:local://");
ResultSet rs = con.executeQuery(“select client, sum(amount) sAmount from d:/sales.txt")
……
```

SQL examples

Filter

select ID,NAME,GENDER,AGE from students.txt where GENDER='F' and AGE>24

Sort

select ID,NAME,GENDER,AGE from students.xlsx order by AGE

Set

select ID,NAME,GENDER,AGE from class1.txt union select ID,NAME,GENDER,AGE from class2.xls

Merge of data from multi databases

Background: the same logical table is scattered in multiple physical databases, and such data is merged and calculated.

MySQL stores the order data of 2015, Oracle stores the data of 2013-2014. Please merge the data of the two databases and return.

	A	B
1	=connect("org.hsqldb.jdbcDriver","jdbc:hsqldb:hsql://127.0.0.1/demo?user=sa")	=connect("com.mysql.jdbc.Driver","jdbc:mysql://127.0.0.1:3306/demo?user=root&password=password")
2	=A1.query@x("select * from sales")	=B1.query@x("select * from sales")
3	=A2 | B2	

Please merge the data of two databases and sort by order amount.

	A	B
1	=connect("org.hsqldb.jdbcDriver","jdbc:hsqldb:hsql://127.0.0.1/demo?user=sa")	=connect("com.mysql.jdbc.Driver","jdbc:mysql://127.0.0.1:3306/demo?user=root&password=password")
2	=A1.query@x("select * from sales order by amount")	=B1.query@x("select * from sales order by amount")
3	=[A2,B2].merge(AMOUNT)	

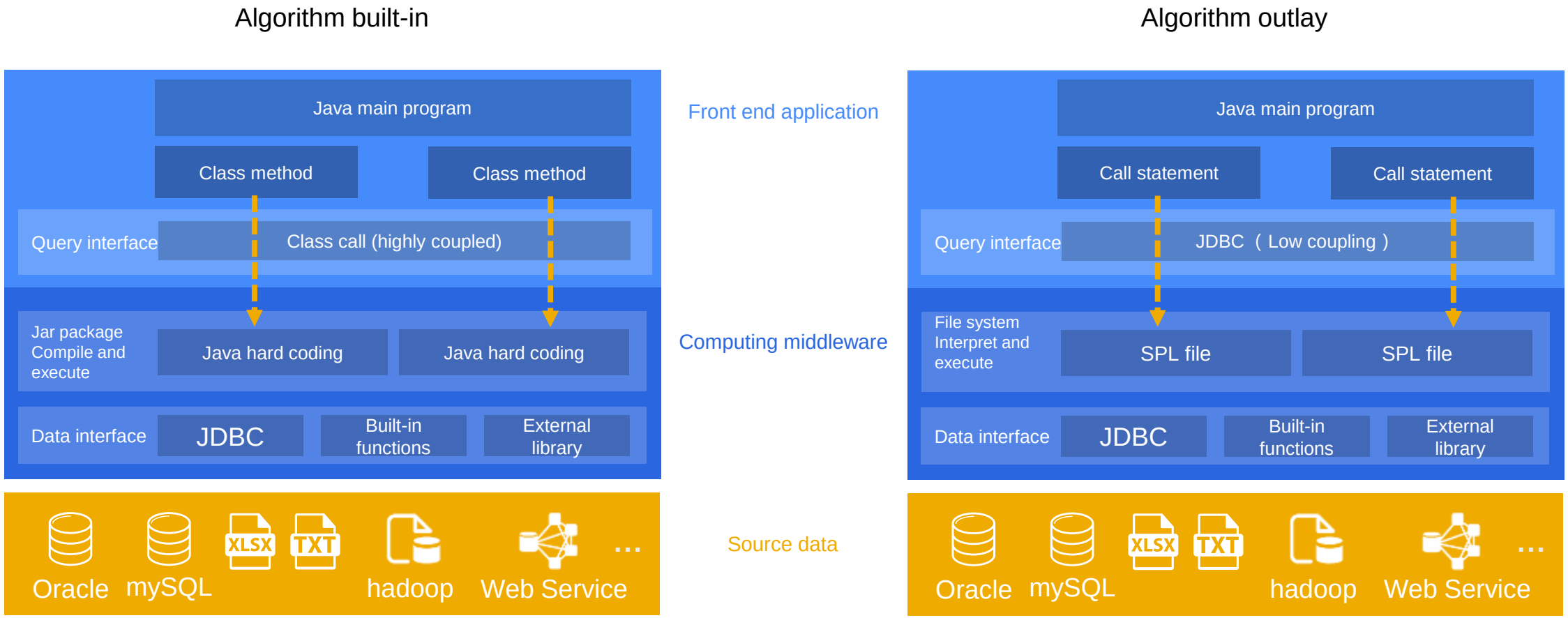
CONTENTS

05

Java algorithm outlay

Algorithm outlay

The algorithm is placed outside in the file system, and the call statement (string) is used to reduce the coupling between the main program and the algorithm.



Algorithm outlay

Hot switch: after the business algorithm is modified, it can be replaced directly without compilation or downtime.

The business algorithm queryorder.dfx originally needs to query the text file.

	A	B
1	=file(“sales.txt”).import@t()	/Open the text file
2	=A1.select(orderID==pClient)	/Return query result. pClient is parameter，representing customer number

Change the algorithm to query data from excel and directly cover the original algorithm.

	A	B
1	=file(“sales.xlsx”).xlsimport@t()	/Open Excel file
2	=A1.select(orderID==pClient)	/Return query result.

Json calculation

In order to reduce system coupling completely, some Java applications need to get data from microservices, which requires that Java programs have the ability to calculate JSON.

Access the microservice, read the multi-layer JSON, calculate the order amount of each employee, as a new field amount of employee record, the result is output as a two-dimensional table. The source data JSON is as follows:


```
[{
"EID":1,"NAME":"Rebecca","SURNAME":"Moore","GENDER":"F","STATE":"California","BIRTHDAY":"1974-11-20","HIREDATE":"2005-03-
11","DEPT":"R&D","SALARY":7000,
"orders":[{"ORDERID":14,"CLIENT":"JAYB","SELLERID":1,"AMOUNT":7644.0,"ORDERDATE":"2012-11-16 15:28:05"},
        {"ORDERID":77,"CLIENT":"HANAR","SELLERID":1,"AMOUNT":13200.0,"ORDERDATE":"2013-01-17 15:28:05"},
                        {"ORDERID":78,"CLIENT":"YZ","SELLERID":1,"AMOUNT":11600.0,"ORDERDATE":"2013-01-20 15:28:05"}
                       ]
},{
"EID":2,"NAME":"Ashley","SURNAME":"Wilson","GENDER":"F","STATE":"New York","BIRTHDAY":"1980-07-19","HIREDATE":"2008-03-
16","DEPT":"Finance","SALARY":11000,
"orders":[{"ORDERID":7,"CLIENT":"EGU","SELLERID":2,"AMOUNT":17800.0,"ORDERDATE":"2012-11-06 15:28:05"},
                        {"ORDERID":19,"CLIENT":"JOPO","SELLERID":2,"AMOUNT":3430.0,"ORDERDATE":"2012-11-18 15:28:05"},
                        {"ORDERID":46,"CLIENT":"UJRNP","SELLERID":2,"AMOUNT":1274.0,"ORDERDATE":"2012-12-20 15:28:05"}
                       ]
},{
"EID":3,"NAME":"Rachel","SURNAME":"Johnson","GENDER":"F","STATE":"New Mexico","BIRTHDAY":"1970-12-17","HIREDATE":"2010-12-
01","DEPT":"Sales","SALARY":9000,
"orders":[{"ORDERID":17,"CLIENT":"PJIPE","SELLERID":3,"AMOUNT":7154.0,"ORDERDATE":"2012-11-19 15:28:05"},
 ...
```

Json calculation

Using esProc external algorithm, and the script is as follows:

	A	B
1	=httpfile("10.0.0.4:8080/recordQuery?comp=cidc").read()	/read microservice
2	=json(A1)	/Convert JSON to sequence table
3	=A8.new(EID,NAME,SURNAME,GENDER,DEPT,SALARY,orders.sum(AMOUNT):sAmount)	/Calculate employee's order amount

Calculation result of A2

	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY	orders
1	[illegible]	oore	F	California	1974-11-20	2005-03-11	R&D	7000	[[14,JAYB,1..
2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000	[[7,EGU,2, ...
3	Rachel	Johnson	F	New Mexico	1970-12-17	2010-12-01	Sales	9000	[[17,PJIPE,...
4	Emily	Smith	F	Texas	1985-03-07	2006-08-15	HR	7000	[[16,AYWY...
5	Ashley	Smith	F	Texas	1975-05-13	2004-07-30	R&D	16000	[[21,DILRT,..

ORDERID	CLIENT	SELLERID	AMOUNT	ORDERDATE
14	JAYB	1	7644.0	2012-11-16 15:28:05
77	HANAR	1	13200.0	2013-01-17 15:28:05
78	YZ	1	11600.0	2013-01-20 15:28:05

ORDERID	CLIENT	SELLERID	AMOUNT	ORDERDATE
21	DILRT	5	16900.0	2012-11-29 15:28:05
34	HANAR	5	784.0	2012-12-12 15:28:05
43	HANAR	5	196.0	2012-12-15 15:28:05

Calculation result of A3

EID	NAME	SURNAME	GENDER	DEPT	SALARY	sAmount
1	Rebecca	Moore	F	R&D	7000	32444.0
2	Ashley	Wilson	F	Finance	11000	22504.0
3	Rachel	Johnson	F	Sales	9000	55154.0
4	Emily	Smith	F	HR	7000	33364.0
5	Ashley	Smith	F	R&D	16000	17880.0
6	Matthew	Johnson	M	Sales	11000	19680.0

Json calculation

esProc can also parse JSON parameters

Example: query the order table according to the start date, end date and customer list. Parameters are passed in the form of JSON string (named pjson), as follows:

```
{beginDate:2012-01-01, endDate:2012-12-10,
 clientList:[{client:UJRNP},{client:UJRNP},{client:PWQ}]
}
```

Use script file to implement query.

	A	B	C
1	=json(pJson)		/Convert JSON parameter to sequence table
2	=beginDate=A1.beginDate	=endDate=A1.endDate	/Resolve start and end date parameters, and automatically convert to date type
3	=clientList=A1.clientList.(client)		/Resolve clientList
4	=demo.query("select * from sales where orderdate>=? and orderdate<? and client in(?)",beginDate,endDate,clientList)		/Execute SQL with parameters

The return result of the above algorithm is a two-dimensional table. If the Java main program needs the algorithm to return JSON format, write the following in cell A5:

5	=json(A4)		

CONTENTS

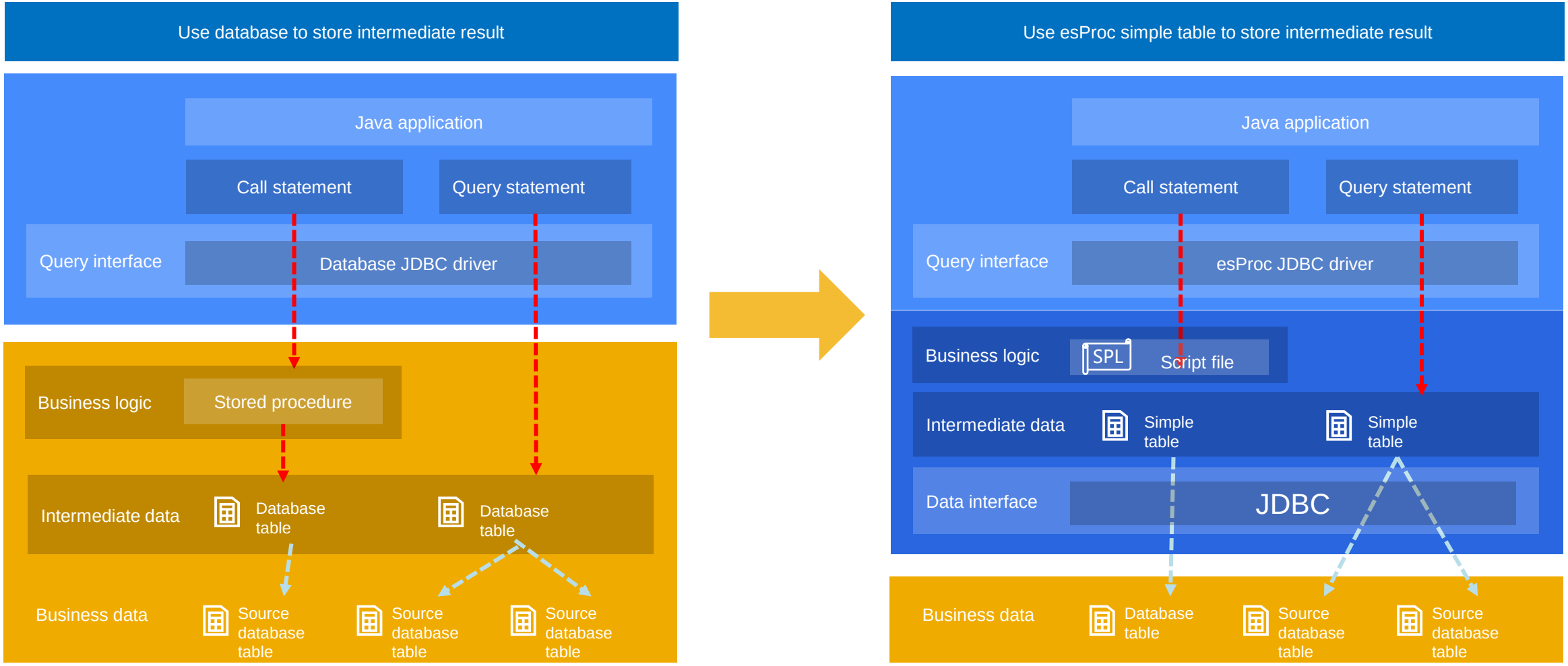
06

Application data cache

Application Architecture

When the source database is under great pressure, the intermediate data should be calculated in advance so that the application server can access it quickly.

The intermediate data is often cached in the source database before, and can be cached in the application server after using the middleware of esProc.



Simple table and database table

Simple table: the intermediate data is stored in the application server without consuming database resources.

Database table: the intermediate data is stored in the source database, which still consumes database resources.

	esProc simple table	Database table
Calculation ability	Sufficient structured algorithms	Sufficient structured algorithms
Addition	High performance addition	Append relatively slowly
Modify	Not good at	Be good at
Occupancy space	Small	Large
Relationship between tables	No physical foreign key and constraint relationship	Can have physical foreign key and constraint relationship
The order of access	Natural order	Order by is required due to uncertain order
Segment to fetch data	Arbitrary segmentation, simple way	Use where segmentation in a complex way
Segment by group	Support	Not supported
Computational performance	High	Low

Select cached data

Data suitable for caching: intermediate tables with complex algorithm and large calculation amount such as summary statistics.

Data that does not need to be cached: business tables that are not computationally stressed.

Source database	esProc simple table
Accounting balance table (Intermediate table with complex algorithms) Budget risk assessment （Intermediate table with complex algorithms） General ledger summary （Summary intermediate table） Customer churn Statistics (Summary intermediate table） Monthly sales （Summary intermediate table） Financial general ledger (business table) Budget table（business table） Order table（ business table） Customer table（business table） Sales table（business table）	-ERP -Financial management -Balance sheet.btx -Budget risk assessment.btx -General ledger summary.btx -Sales management -Customer churn statistics.btx -Monthly sales.btx -CRM…

Note 1: if necessary, frequently accessed business tables can also be cached.
Note 2: the simple table is stored in the operating system directory, which can be divided by functional modules or by application type, time and version.

Generating simple tables

The most direct and simple way: export the intermediate table in the source database as a simple table.

In the source data, the aggregation intermediate table salesAgg is generated according to the business table sales. Some of the data are as follows:

year	month	sellerid	samount	cquantity
2012	11	17	35792	3
2012	11	6	4802	1
2012	11	16	39334	3
2012	11	9	26100	1
2012	11	11	25610	2

Export salesAgg as a simple table:

	A	B
1	=connect@l("db").cursor("select * from salesAgg")	/SQL fetch data
2	=file("salesAgg.btx").export@b(A1)	/Create simple table

After reading the simple table, you can see that its data is the same as the intermediate table.

	A	B
1	=file("salesAgg.btx").import@b()	/Read simple table

year	month	sellerid	samount	cquantity
2012	11	17	35792	3
2012	11	6	4802	1
2012	11	16	39334	3
2012	11	9	26100	1
2012	11	11	25610	2

Generating simple tables

More common way: use SPL script to transform (replace) the generation process of the original intermediate table, and generate the simple table directly from the business table.

Generate a simple table directly with business table sales:

	A	B
1	=connect@l("db").cursor("select year(orderdate) as year,month(orderdate) month as month,sellerid,sum(amount)sAmount,count(1) cQuantity from sales group by year(orderdate),month(orderdate) ,sellerid")	/SQL fetch data
2	=file("salesAgg.btx").export@b(A1)	/Create simple table

Incremental addition

Scenario: it is applicable to appending data to historical simple table regularly.

Note: the source data must have a time stamp for increment.

Take yesterday's incremental data from the business table sales in the early morning of each day and add it to the simple table salesagg.btx.

	A	B
1	=connect@l("db").cursor@x("select year(orderdate) as year,month(orderdate) month as month,sellerid,sum(amount)sAmount,count(1) cQuantity from sales where orderdate=? group by year(orderdate),month(orderdate), sellerid ", pDate)	Take incremental data by date
2	=file("salesAgg.btx.btx").export@ab(A1)	Append to existing simple table

Note 1: The simple table cannot be modified. If the data changes, the simple table should be regenerated. esProc group table is used for data warehouse, and this file format supports modification.

Note 2: The way to take incremental is flexible. You can get data by time interval or by date. In the above example, pdata is a parameter, which is used to calculate yesterday's date from the outside and pass it in, so that it is convenient to control which date of data to append. If you want to extract yesterday's data, you can also use the SPL expression date(elapse(now(),-1)) instead of pdate.

Extended reading： Oracle has a unique incremental addition method of OGG, which is supported by esProc. For details, please refer to OGG Incrementally collected data importing into database

Using simple table

Simple table supports esProc SQL syntax, and examples of usage are as follows:

Query

```
select * from salesAgg.btx where samount>=10000 && samount<20000
```

Aggregate

```
select year, month, sum(samuont) as total from salesAgg.btx group by year,month
```

Join

```
select s.year, s.month, e.name from salesAgg.btx s, employee.btx e where s.sellerid=e.eid
```

Using simple table

When the business logic is complex, the SPL script should be used for calculation.

Example: Stockagg.btx, a simple table generated by the inventory table, records the entered and issued quantity of each product every day. Part of the data is as follows:

IDATE	INAME	ENTER	ISSUE
2014-04-01	Item1	19	0
2014-04-02	Item1	3	10
2014-04-03	Item1	3	0
2014-04-04	Item1	0	5
2014-04-07	Item1	4	0

Query the simple table by time period, and calculate the following inventory status: open, enter, total, issued, and close of each product every day.

	A	B
1	=fle("stockAgg.btx").import@b().select(IDATE>=start &&IDATE<=end)	
2	=A1.group(INAME)	=periods(start,end,1)
3	for A2	=A3.align(B2,IDATE)
4		>c=0
5		=B3.new(A3.INAME,B2(#):IDATE, c:OPENING, ?ENTER,(b=c+ENTER):TOTAL,ISSUE,(c=b-ISSUE):CLOSE)
6		=@|B5
7	return B6	

	A	B	C	D	E	F	G
1	INAME	IDATE	OPENING	ENTER	TOTAL	ISSUE	CLOSE
2	Item1	2014-04-01	0	19	19	0	19
3	Item1	2014-04-02	19	3	22	10	12
4	Item1	2014-04-03	12	3	15	0	15
5	Item1	2014-04-04	15	0	15	5	10
6	Item1	2014-04-05	10		10		10
7	Item1	2014-04-06	10		10		10
8	Item1	2014-04-07	10	4	14	0	14
9	Item1	2014-04-08	14		14		14
10	Item1	2014-04-09	14		14		14

Optimized storage

When a simple table is generated, it can be stored optimally according to the data characteristics, so as to achieve better computing performance.

Orderly storage: salesagg.btx often performs orderly calculation (such as grouping and aggregating by year and month), so the simple table should be generated in the order of grouping field.

	A	B
1	=connect@l("db").cursor("select year(orderdate) as year,month(orderdate) month as month,sellerid,sum(amount)sAmount,count(1) cQuantity from sales group by year(orderdate),month(orderdate) ,sellerid order by year,month")	/SQL sorting
2	=file("salesAgg.btx").export@b(A1)	/Orderly storage

Segmented storage: salesagg.btx often performs segmented calculation (such as parallel query), so segmented storage should be performed.

	A	B
1	=connect@l("db").cursor("select year(orderdate) as year,month(orderdate) month as month,sellerid,sum(amount)sAmount,count(1) cQuantity from sales group by year(orderdate),month(orderdate) ,sellerid")	/SQL sorting
2	=file("salesAgg.btx").export@z(A1)	/Segmentation

Ordered segmentation: when a field is known to be used for segmentation calculation, it can be sorted and stored in segments according to the field.

	A	B
1	=connect@l("db").cursor("select year(orderdate) as year,month(orderdate) month as month,sellerid,sum(amount)sAmount,count(1) cQuantity from sales group by year(orderdate),month(orderdate) ,sellerid order by sellerid")	/SQL sorting
2	=file("salesAgg.btx").export@z(A1;sellerid)	/Ordered segmentation

SQL+ statement

If you know the data characteristics of a simple table, you can use SQL + syntax for high-performance optimization queries.

Parallel computing

```
select  /*+parallel (4) */ * from sales.txt where orderid=100
```

Large table cursor query

```
select * from /*+external*/ emp.btx where orderid=100
```

Ordered foreign table join

```
select o.Orderid ,o.amount,s.name from sales.btx
o /*+foreign*/ join seller.btx s on s.id = o.sellerid
```

Note：Using SQL+statement，the url should be written as jdbc:esproc:local://sqlfirst=plus。

Extended reading：SQL+ http://doc.raqsoft.com/esproc/func/sqljia.html

Life cycle

There are three life cycles of cache: timed cache, temporary cache and controllable cache.

Timed cache: generates a cache in advance for multiple business algorithms.

The database table sales has frequent access and many algorithms will use it. Now it is cached as a simple table. The following script file can be executed in the early morning of each Monday.

	A	B
1	=connect@l("db").query@x("select orderid,client,sellerid,amount,orderdate from sales order by eid ")	Query data
2	=file("sales.btx").export@z(A1)	Create simple table

Business algorithm A: Group and aggregate by client, sellerid.

	A	B
1	=file("sales.btx").cursor@b(A1)	Open simple table
2	=A2.groups(client,sellerid;sum(amount):sAmount,count(1):cOrderid)	Group and aggregate

Business algorithm B: Query records by ordered.

	A	B
1	=file("salse.btx")	Open simple table
2	=A2.iselect@b(10,orderid; orderid,client,amount)	Orderly query

Life cycle

Temporary cache: generates a cache temporarily before calculation, which is generally used repeatedly in the local area by the current algorithm to reduce frequent query actions of the database.

The front-end application needs to take large table data for page by page display. In this case, it can temporarily generate a simple table, which can greatly improve the performance by using the I / O and ordered fetching of the simple table. In order to achieve this goal, the front end must pass in the unique identification of the current algorithm, so that the same simple table can be used each time when fetching data, and the start and end positions of records (which can be converted to page numbers) must be passed in, so that different pages can be fetched each time.

	A	B	C
1	=file(uuid)		Take the unique ID of the current algorithm as the simple table name
2	if !A1.exists()	=connect@l("dbsource").cursor@x("select * from sales order by orderid")	If the simple table does not exist, generate
3		=A1.export@z(B2)	
4	=A1.iselect@b(begin:end,orderid; orderid,client,sellerid,orderdate,amount)		If the simple table exists, then fetch data by page

Note: The temporary cache can be stored in the temporary directory specified by esProc, and the files in the directory will be cleaned automatically at regular intervals.

Life cycle

Controllable cache: the business algorithm controls the exact life cycle of the cache, which is used by the current algorithm or other algorithms.

When performing group aggregation calculation on the sales table, first check whether the timeliness of the cache is within one hour. If the timeliness is met, the cache is directly used for calculation. If the timeliness has passed, the latest cache is temporarily generated. This cache can be used by this algorithm and other algorithms.

	A	B	C
1	=file(“sales”)		
2	if interval@s(A1.date(),now())>=3600 || !A1.exists()	=connect@l("dbsource").cursor@x("select * from sales order by orderid")	If the simple table times out or does not exist, generate
3		=A1.export@z(B2)	
4	=A1.groups(client,sellerid;sum(amount):sAmount)		If the simple table exists, group and aggregate

Other items for attention

In the pursuit of higher concurrency, more computation, more data (including data compression), high-performance index query, data modification and other characteristics, you can use esProc group table format.

Regularly execute the SPL script (including the EPT script of the visualizer) to generate the cache. You can use the scheduling tool of the operating system, such as scheduled task, crontab, or the third-party visualizer, such as opencron.

CONTENTS

07

Multi source hybrid computing method

Basic method

Multi-source hybrid computing is one of the main uses of computing middleware. With highly encapsulated interfaces, esProc can simplify the hybrid computing between any data sources.

The employee table is located in Oracle and the performance table is located in MySQL. Please join the two tables across databases and calculate the actual salary.

employee(oracle)

key	empID
	empName
	baseSalary
	post
	dept

performance(mysql)

key	empID
	bonus

Output result

key	empID
	empName
	realSalary
	post
	dept

	A	B
1	=connect(“orcl”).cursor@x(“select * from employee”)	/Connect employee table
2	=connect(“mysql”).query@x(“select * from performance”)	/Connect performance table
3	=A1.switch(empID,A2:empID)	/Join across databases
4	=A3.new(empID.empID:empID,empName,baseSalay+empID.bonus:realSalary,post,dept)	/Calculate real salary

Basic method

esProc uses a unified data model to access data sources, and can use the same method to calculate different data sources.

The employee table is located in Oracle, and the performance table is located in the text file. Please join the two tables across sources and calculate the actual salary.

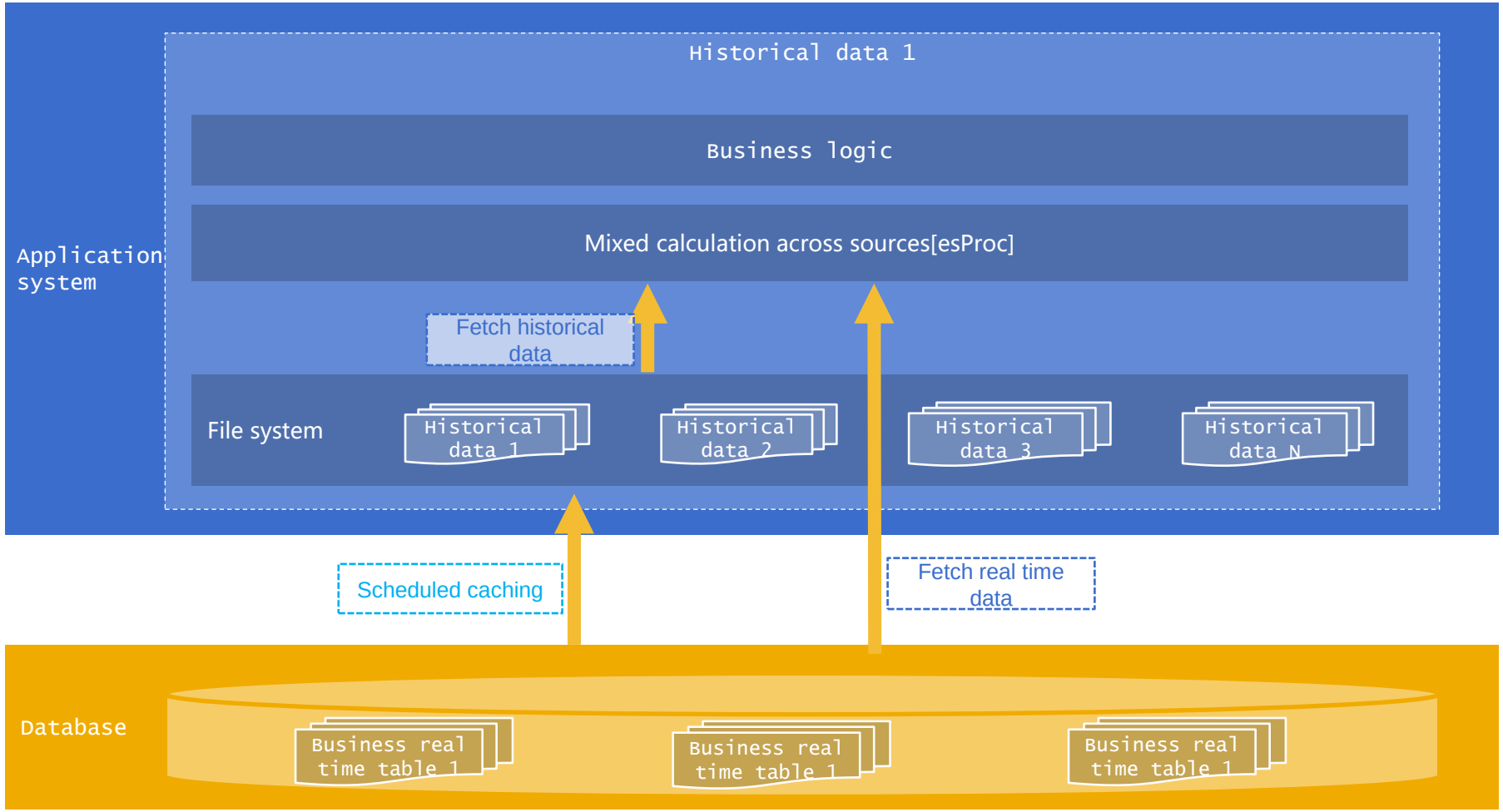
	A	B
1	=connect("orcl").cursor@x("select * from employee")	/Employee
2	=file("performance.txt").import@t()	/Performance
3	=A1.switch(empID,A2:empID)	/Join across sources
4	=A3.new(empID.empID:empID,empName,baseSalay+empID.bonus:realSalary,post)	/Calculate actual salary

The employee information is in the simple table, and the performance information is in the text file. Please join the two tables across sources and calculate the actual salary cost of each department.

	A	B
1	=file("employee.btx").cursor@b()	/Employee
2	=file("performance.txt").import@t()	/Performance
3	=A1.switch(empID,A2:empID)	/Join across sources
4	=A3.groups(dept; sum(baseSalay+empID.bonus):realSalary)	/Actual salary of each department

T+0 mixed calculation

Historical data t is stored in simple tables (group tables if amount is large) and real-time data 0 is stored in production database. By real-time mixed calculation of the two, high-performance full amount of data query can be realized and precious production database resources can be saved at the same time.



T+0 mixed calculation

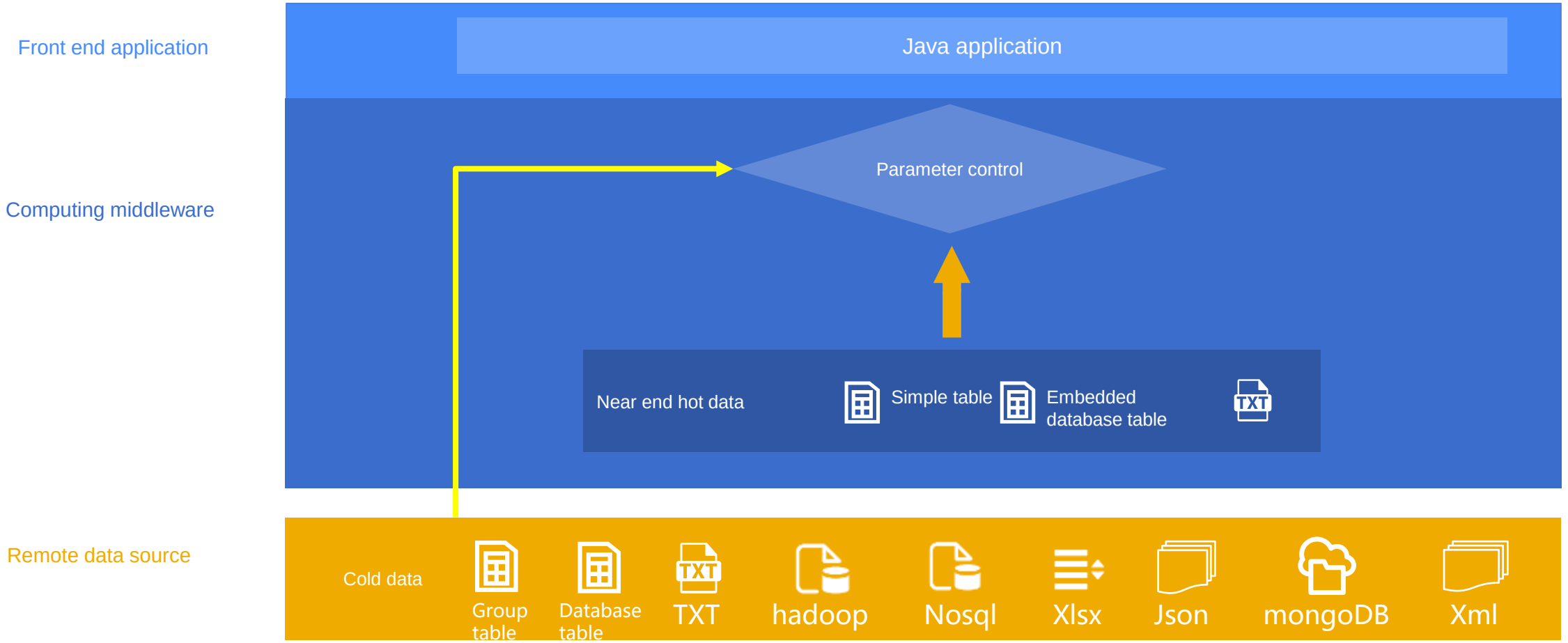
The order history is stored in the simple table, and the real-time order data is stored in the production database. Please count the number of orders per customer so far.

	A	B
1	=connect(“db”).cursor@x(“select customer,count(1) total from sales where orderDate=?” ,now())	/Query database order of the day
2	=connect().cursor@x(“select customer,count(1) total from sales.btx”)	/Query historical orders
3	=[A1,A2].conjx().groups(customer,sum(total):total)	/Group aggregation again

Extended reading： For more comprehensive and detailed information about T + 0 real-time mixed calculation, please refer to 《Statistical Query after Database Split》

Cold and hot routing

The frequently accessed hot data can be cached at the near end of the application server, and the occasionally accessed cold data can be stored at the far end. During the calculation, the cold data or hot data can be returned by judging the parameter interval.



Cold and hot routing

The hot data of the current year (2016) is stored in the application server in the form of simple tables, and the cold data of previous years (before 2016) is stored in the RDB data warehouse. Now, getdate.dfx is used to implement hot and cold routing and group aggregation algorithm. When the upper Java program calls, the parameters pbegindate and penddate are passed in, that is, the starting and ending time range of the required data.

	A	B
1	=hotcoldLine=date("2016-01-01")	/Cold and hot dividing point
2	=file(“saleshot.btx”).iselect@b(max(pBeginDate,hotcoldLine):max(pEndDate,hotcoldLine), orderdate; orderid,client,sellerid,amount,orderdate)	/Fetch hot data
3	=connect@l("demo").cursor@x("select * from sales where orderdate>=? and orderdate<?",min(pBeginDate,hotcoldLine),min(pEndDate,hotcoldLine))	/Fetch cold data
4	=A2|A3	/Combine cold and hot data, one of which may be empty
5	=A4.groups(year(orderdate),month(orderdate);sum(amount):sAmount)	/Calculate

Note： Cold data should be indexed according to the OrderDate field, and the group table can be sorted and segmented according to the OrderDate, so as to return quickly when it is empty. The routing process can be separated into public script for specific script to call to reduce coupling.

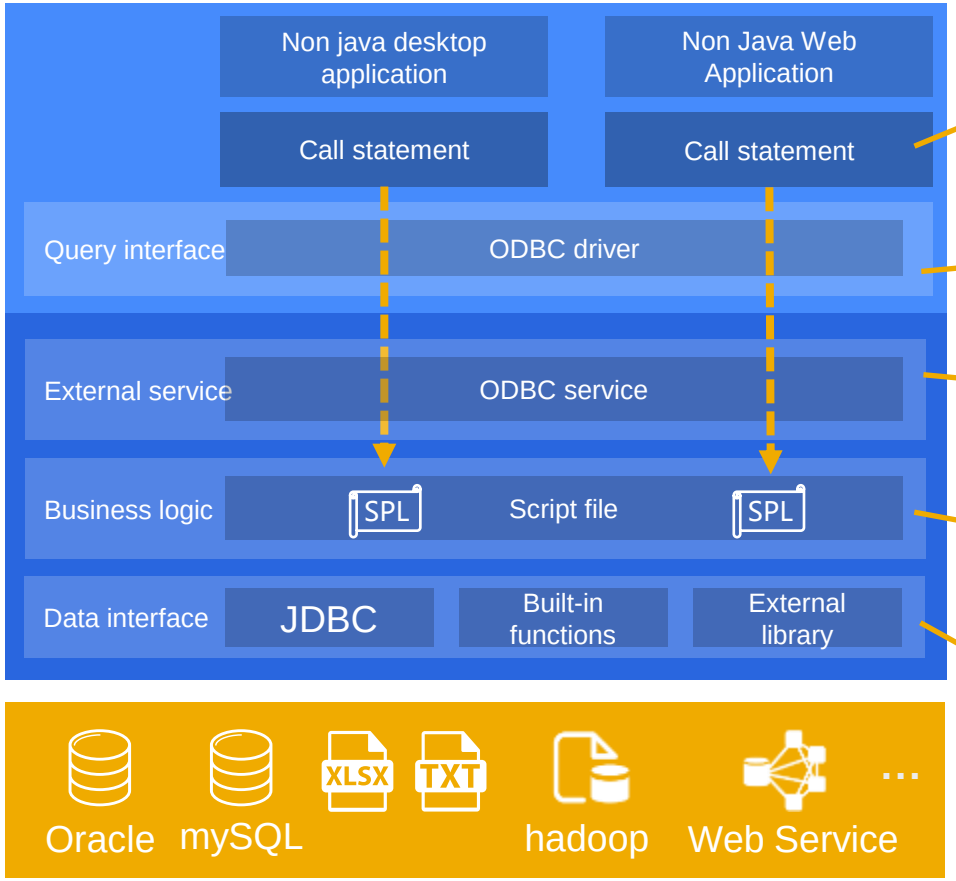
CONTENTS

08

ODBC and HTTP integration

ODBC integration

In addition to the JDBC interface, esProc also provides ODBC services to support non Java front-end applications, such as VB, C#, asp.net, crystal reports.



5. Execute the call statement in non java code, call the SPL file through esProc ODBC, which interprets and executes the SPL file and returns the calculation result.

3.Deploy esProc ODBC driver

2. Start esProc ODBC service

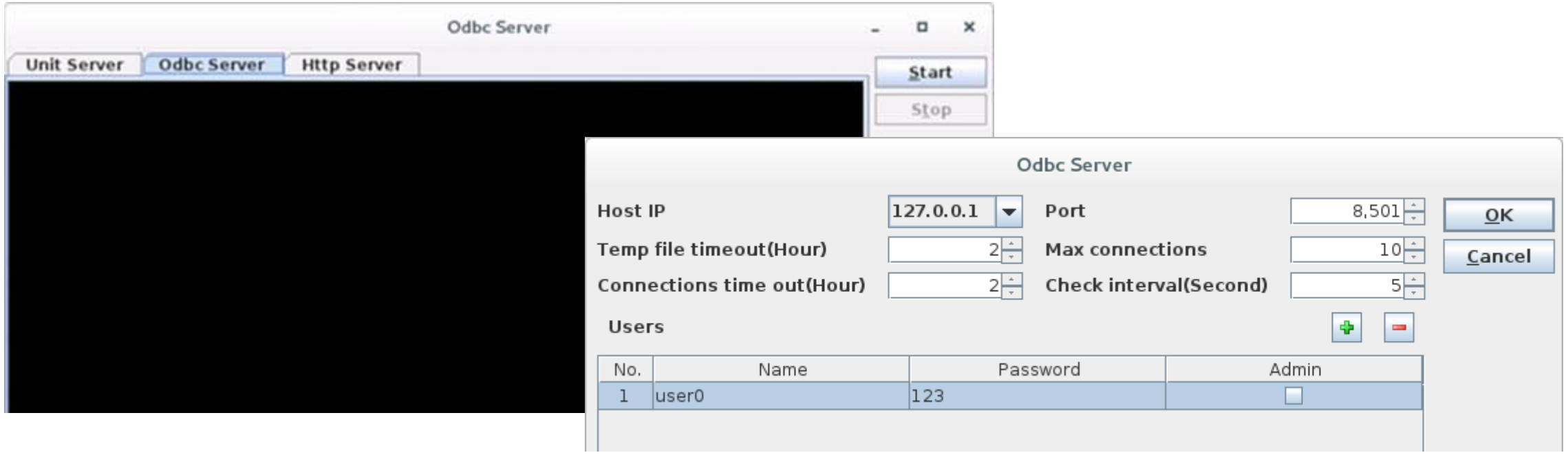
4. Write specific business algorithm in SPL file，no change.

1. Configure the running environment in raqsoftconfig.xml, no change.

ODBC integration

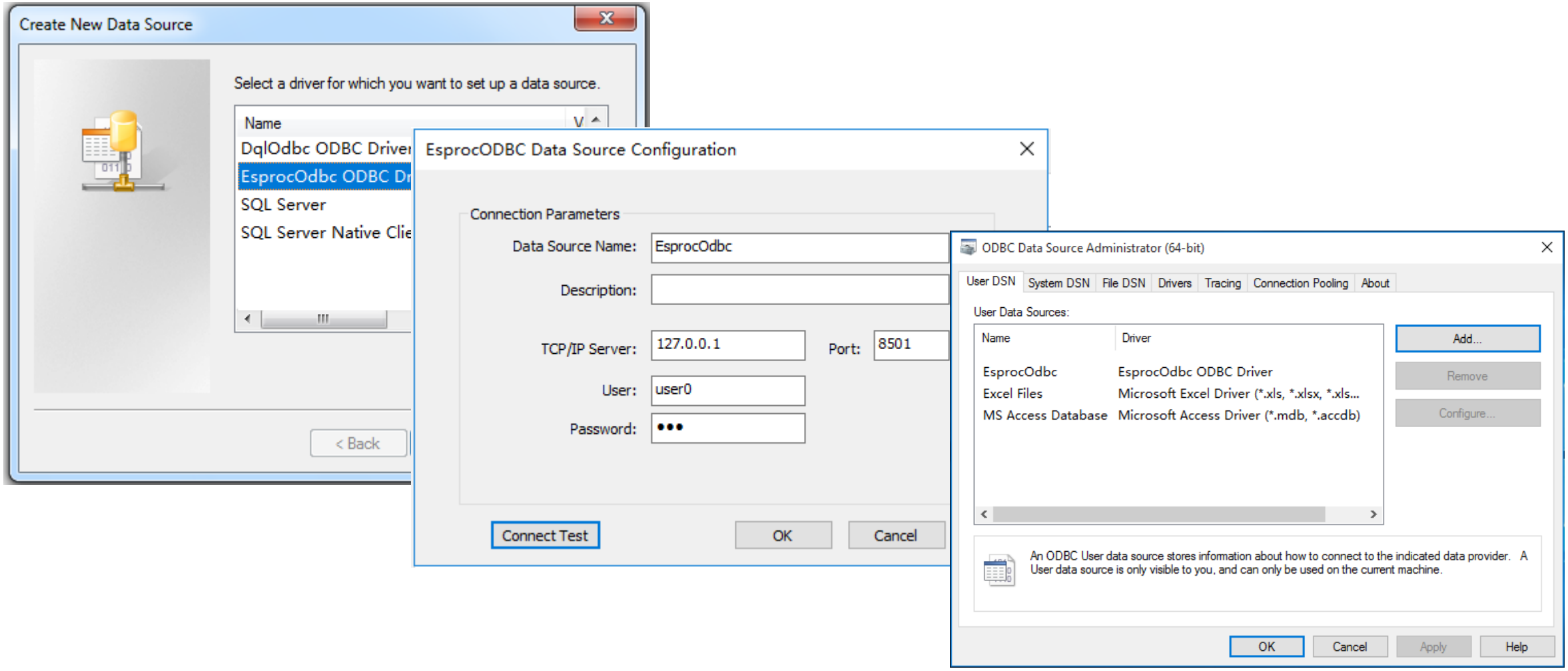
1. Configure the running environment in raqsoftconfig.xml, and there is no change in this step.
2. Start the esProc ODBC service. This step is the main change.

esProc provides a graphical interface to start ODBC service，please refer to http://doc.raqsoft.com.cn/esproc/tutorial/jsqzqdodbcfw.html



ODBC integration

3. Deploy the ODBC driver of esProc. First, execute esprocodbcinst.exe with administrator permission to automatically deploy ODBC driver, and then configure ODBC connection word in ODBC data source manager of windows.



Note：For details of ODBC deploy, please refer to http://doc.raqsoft.com/esproc/tutorial/odbcbushu.html

ODBC integration

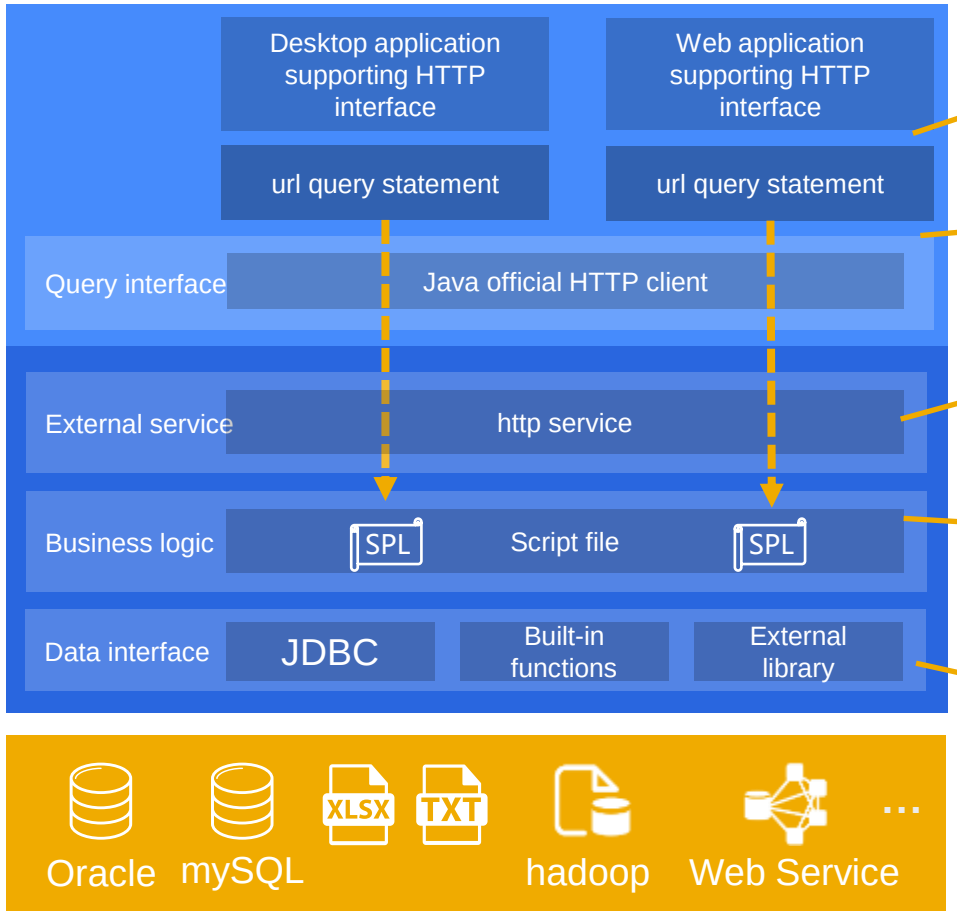
4. Write the specific business algorithm in the SPL file. There is no change in this step.
5. Execute the call statement in the non java code. This step only needs to write the code according to the standard ODBC specification.

For example: in asp.net code, access the conj.dfx script file through ODBC.

```
……
   OdbcConnection odbcConn = new OdbcConnection(“DSN=testOdbc;”);
   odbcConn.Open();
   OdbcCommand odbcCmd = new OdbcCommand(“call conj(?,?)", odbcConn);
   odbcCmd.Parameters.Add("minamount", OdbcType.Int).Value = 4000;
   odbcCmd.Parameters.Add("maxamount", OdbcType.Int).Value = 8000;
  ……
```

HTTP integration

In order to provide loosely coupled data services (such as data in the middle), esProc provides HTTP services. It should be noted that in this case, it is usually required to deploy esProc independently.



5. Use any language to access the SPL file through HTTP protocol. The HTTP service of esProc interprets and executes the SPL file, and returns the calculation result.

3. Deploy the official HTTP client jar package of Java.

2. Start esProc http service.

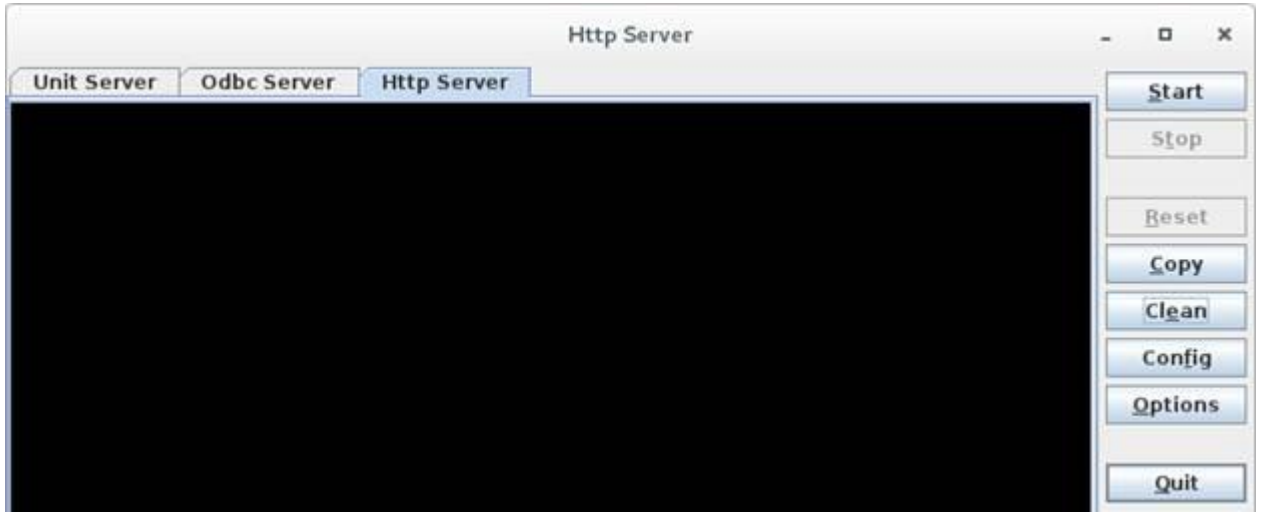
4. Write specific business algorithm in SPL file and store it in file system.

1. Configure the running environment in raqsoftconfig.xml, no change.

HTTP integration

1. Configure the running environment in raqsoftconfig.xml, and there is no change in this step.
2. Start the HTTP service of esProc. This step is the main change.

esProc provides a graphical interface to start HTTP service，please refer to http://doc.raqsoft.com.cn/esproc/tutorial/httpfuwu.html



HTTP integration

3. Deploy the official HTTP client jar package of Java. This step has nothing to do with the specific HTTP service. Please refer to the official JAVA standard.
4. Write the specific business algorithm in the SPL file. The algorithm itself does not change in this step. Only the format of the returned data needs to be processed.

The original algorithm (non HTTP service) queries the sales table. Begindate and enddate are external parameters. The script file is as follows:

	A	B	C
1	=connect@l(“demo”).query@x("select * from sales where orderdate>=? and orderdate<?",beginDate,endDate)		/Query database

If you want esProc to provide a restful JSON service, just use the JSON function to return the result.

	A	B	C
1	=connect@l(“demo”).query@x("select * from sales where orderdate>=? and orderdate<?",beginDate,endDate)		/Query database
2	=json(A1)		

To access the HTTP service of esProc with a browser, you can see the following data structure:

Note: if you want to return XML format, you should use XML function. If the returned result is not processed, the HTTP client receives a string in 2D table format, separated by tab by default.

HTTP integration

For Java middleware, it's difficult to return multi-layer JSON, while it's relatively simple to use esProc.

Associate the salesperson with his / her order to form multi-layer JSON and return.

	A	B
1	=connect@l("dbsource")	
2	=A1.query("select * from employee")	=A1.query@x("select * from sales where orderdate>=? and orderdate<?",beginDate,endDate)
3	=A2.run(eid=B2.select(sellerid==eid))	/Join
4	=json(A3)	

View return results in browser.


```
1 [{"eid":[{"orderid":14,"client":"JAYB","sellerid":1,"amount":7644.0,"orderdate":"2012-11-16 15:28:05"},{"orderid":77,"client":"HANAR","sellerid":1,"amount":13200.0,"orderdate":"2013-01-17 15:28:05"},
{"orderid":78,"client":"YZ","sellerid":1,"amount":11600.0,"orderdate":"2013-01-20 15:28:05"},{"orderid":93,"client":"AVU","sellerid":1,"amount":21800.0,"orderdate":"2013-02-05 15:28:05"},
{"orderid":104,"client":"HL","sellerid":1,"amount":26400.0,"orderdate":"2013-02-18 15:28:05"},{"orderid":109,"client":"PWQ","sellerid":1,"amount":17500.0,"orderdate":"2013-02-21 15:28:05"},
{"orderid":120,"client":"FHYBR","sellerid":1,"amount":16000.0,"orderdate":"2013-03-03 15:28:05"},{"orderid":127,"client":"HP","sellerid":1,"amount":13600.0,"orderdate":"2013-03-15 15:28:05"},
{"orderid":189,"client":"DNEDL","sellerid":1,"amount":26100.0,"orderdate":"2013-05-13 15:28:05"},{"orderid":200,"client":"EGU","sellerid":1,"amount":14000.0,"orderdate":"2013-05-20 15:28:05"},
{"orderid":201,"client":"DNEDL","sellerid":1,"amount":7350.0,"orderdate":"2013-05-25 15:28:05"},{"orderid":221,"client":"EGU","sellerid":1,"amount":7742.0,"orderdate":"2013-06-14 15:28:05"},
{"orderid":237,"client":"PWQ","sellerid":1,"amount":23400.0,"orderdate":"2013-06-24 15:28:05"},{"orderid":267,"client":"QUICK","sellerid":1,"amount":17400.0,"orderdate":"2013-07-28 15:28:05"},
{"orderid":278,"client":"FHYBR","sellerid":1,"amount":23900.0,"orderdate":"2013-08-12 15:28:05"},{"orderid":279,"client":"MIP","sellerid":1,"amount":5880.0,"orderdate":"2013-08-12 15:28:05"},
{"orderid":288,"client":"UJRNP","sellerid":1,"amount":4998.0,"orderdate":"2013-08-22 15:28:05"},{"orderid":351,"client":"SAVEA","sellerid":1,"amount":6958.0,"orderdate":"2013-10-19 15:28:05"},
{"orderid":380,"client":"JAYB","sellerid":1,"amount":13000.0,"orderdate":"2013-11-17 15:28:05"},{"orderid":384,"client":"AYWYN","sellerid":1,"amount":22200.0,"orderdate":"2013-11-20 15:28:05"},
{"orderid":401,"client":"DILRT","sellerid":1,"amount":20200.0,"orderdate":"2013-12-08 15:28:05"},{"orderid":461,"client":"EGU","sellerid":1,"amount":5880.0,"orderdate":"2014-02-04 15:28:05"},
{"orderid":465,"client":"AYWYN","sellerid":1,"amount":22400.0,"orderdate":"2014-02-11 15:28:05"},{"orderid":468,"client":"UJRNP","sellerid":1,"amount":15500.0,"orderdate":"2014-02-18 15:28:05"},
```

HTTP integration

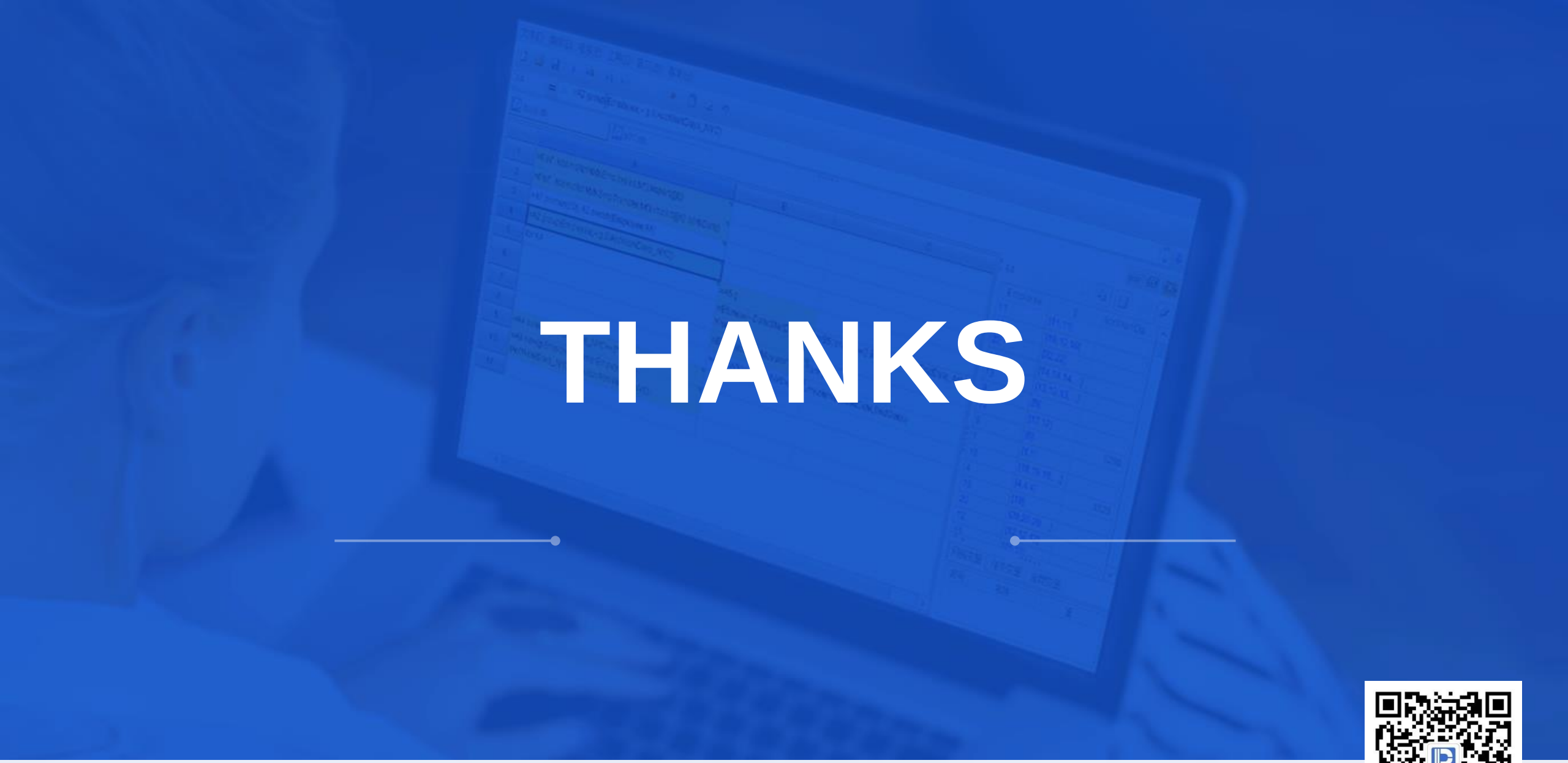
5. Access the SPL file through HTTP protocol, which has nothing to do with the specific HTTP service. Note that the esProc HTTP service supports two URL styles.

Default style，http:///*IP:port* /dfx1.dfx(arg1,arg2,...). Part of the code of Java calling esProc through HTTP protocol is as follows:

```
.....
   URL url = new URL("http://192.168.1.107:8503/getsales.dfx(2010-03-01,2019-04-01)");
   HttpURLConnection httpUrlConn = (HttpURLConnection) url.openConnection();
  .....
```

SAP style，http:///*IP:port* /sapPath/dfx/arg1/arg2/.... The calling code is as follows:

```
.....
   URL url = new URL(“http://192.168.1.107:8503/getsales/2010-03-01/2019-04-01)");
   HttpURLConnection httpUrlConn = (HttpURLConnection) url.openConnection();
  .....
```



THANKS

www.raqsoft.com