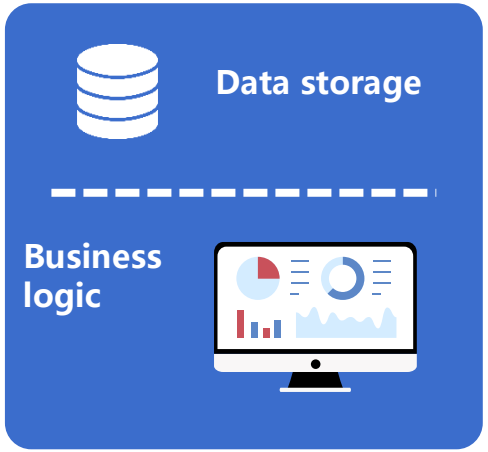
Java Computing Layer

esProc · Issued by Raqsoft

www.raqsoft.com

Data computing in Java programs

For various reasons, we often need to compute data in Java programs.



According to system design requirements, business logic is separated from data storage.

Data is scattered among different data sources.

Computing logic is complex and it is difficult to implement in SQL.

The Pain Points of Implementing Data Computing in Java

Large amount of code

Difficult to implement complex algorithms

High coupling

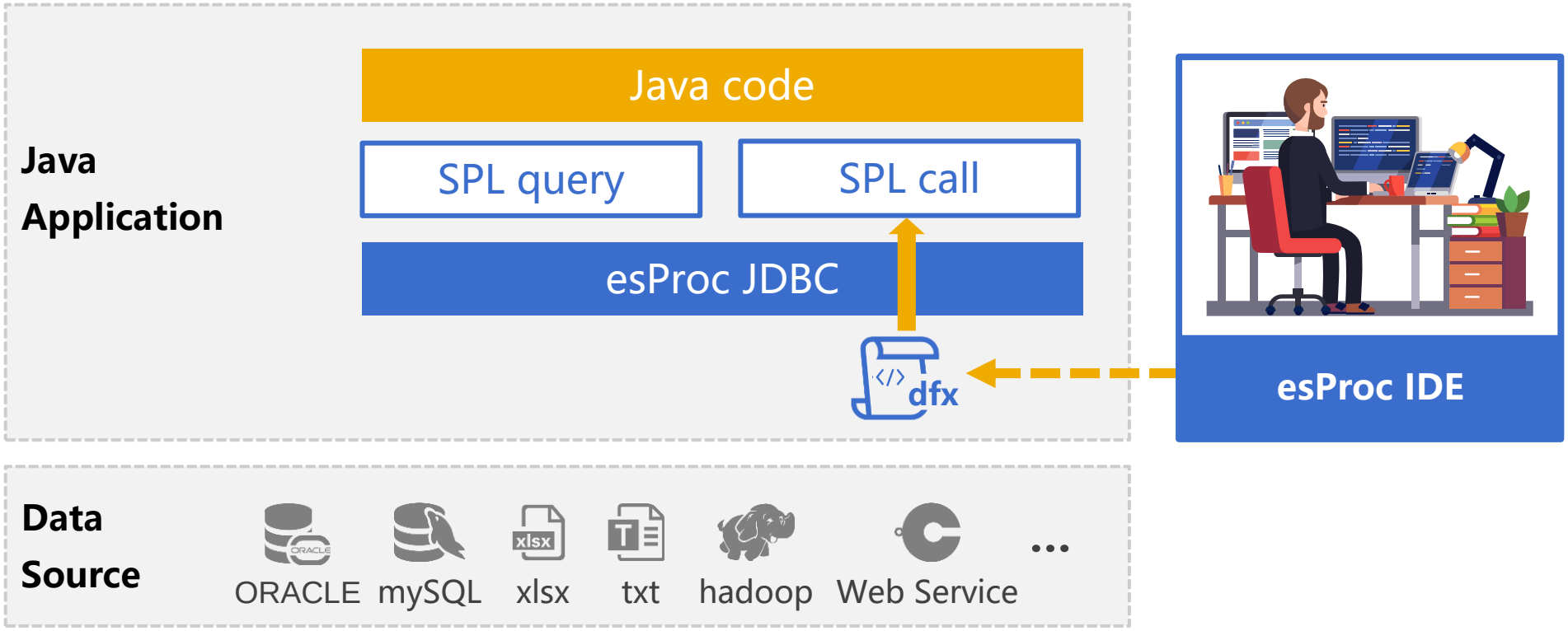
Difficult to maintain

A JAVA example to implement grouping

```java
public static void groupCount(){
    groupBy mainJava = new groupBy();
    List<MainBean> list = mainJava.initList();
    List<MainBean> mainList = new ArrayList<MainBean>();
    Map<String, MainBean> map = new HashMap<String, MainBean>();
    MainBean totalBean = null;
    String employeeId = null;
    for(MainBean mainBean : list) {
        employeeId = mainBean.getEmployeeId();
        if (!map.containsKey(employeeId)) {
            totalBean = new MainBean();
            totalBean.setEmployeeId(employeeId);
            totalBean.setBuyNum(mainBean.getBuyNum());
            totalBean.setGoodsPrice(mainBean.getGoodsPrice());
            totalBean.setTotalBuyNum(mainBean.getTotalBuyNum());
            totalBean.setFreight(mainBean.getFreight());
            totalBean.setTotalGoodsPrice(mainBean.getGoodsPrice() * mainBean.getBuyNum());
            totalBean.setTotalPrice(mainBean.getGoodsPrice() * mainBean.getBuyNum() + mainBean.getFreight());
            mainList.add(totalBean);
            map.put(employeeId, totalBean);
        } else {
            totalBean = map.get(employeeId);
            totalBean.setTotalBuyNum(totalBean.getTotalBuyNum() + mainBean.getTotalBuyNum());
            totalBean.setFreight(totalBean.getFreight() + mainBean.getFreight());
            totalBean.setTotalGoodsPrice(totalBean.getTotalGoodsPrice() + mainBean.getGoodsPrice() * mainBean.getBuyNum());
            totalBean.setTotalPrice(totalBean.getTotalPrice() +
                    mainBean.getGoodsPrice() * mainBean.getBuyNum() + mainBean.getFreight());
        }
    }
```

Such a long code only realizes the function of single field grouping and single field aggregation

Solution：Java computing layer - esProc



Note: SPL is the scripting language used by esProc. DFX is the script file, which is equivalent to an external stored procedure. Java uses the JDBC interface to import SPL statements to realize structured calculation or script execution.

Solution：Java computing layer - esProc

Professional Data Computing Language SPL

- Combination of discreetness and set orientation
- More than 300 Structured Computing Functions
- Characteristic cell-style coding makes layer clear
- Memory, External Storage Cursor, Cluster Operation
- Loop and Branch Adapts Procedural Computation
- Step-by-step calculation and perfect debugging function

Abundant data interfaces

- RDB：Oracle,DB2,MS SQL, MySQL, PG,....
- MongoDB，REDIS，...
- Hadoop：HDFS，HIVE，HBASE
- TXT/CSV，JSON/XML，EXCEL
- HTTP, ALI-OTS
- File SQL interface

Professional Jar packages

- Standard JAR package, easy to integrate
- Multithread Parallel Computing
- Perfect exception handling

Examples of implementing grouping with SPL

Such simple operation can also be written directly in SQL :

SELECT name,count(name) FROM user/test/duty.xlsx GROUP BY name

Conventional grouping

Summarize the number of days on duty for each person

	A
1	=file("/Users/test/duty.xlsx").importxls@tx()
2	=A1.groups(name;count(name):count)

Top N of each group

Obtain overtime records for each month, each person, and the first three days

	A
1	=file("/Users/test/duty.xlsx").importxls@tx()
2	=A1.groups(month(workday):mon,name;~.top(3):top3)

Alignment grouping

List in order the number of countries using Chinese, English and French as official languages

	A
1	=connect("mysql")
2	=A1.query@x("select * from world.countrylanguage where isofficial= 'T' ")
3	[Chinese,English,French]
4	=A2.align@a(A3,Language)
5	=A4.new(A3(#):name,~.len():cnt)

Rich class library

Designed specifically for structured data tables

	A	B	C
1	=esProc.query("SELECT orderID as contract,		/retrieve sales records
2	=A1.group(salesman)		
3	=create(salesman,thisyearAmount, lastyearAmount, custNumber, bigCustNumber)		
4	for A2	=A4(1).salesman	
5		=A4.select(year(date)==year).sum(amount)	
6		=A4.select(year(date)==year-1).sum(amount)	
7		=A4.group(customer).(~.sum(amount))	
8			
9		=B7.count(~>=10000)	

Grouping & Loop

	A	B	C
1	=esProc.query("select * from employee")		
2	=A1.select(sex=="male")		
3	=A1.select(birthday>=date("1970-01-01"))		
4	=A2^A3	/intersect, find out male employee born after 1970	
5	=A2&A3	/union, find out male employee or employee born after 1	
6	=A2\A3	/subtract, find out male employee born before 1970	
7	=A4.sum(salary)		
8	=A5.avg(age)		
9	=A5.sort(birthday)		
10	/set is extensively used as a data type		
11			

Set operations

	A	B	C
1	=file("traderecord.txt").import@t()		
2	=A1.sort(customerID, tradeDate)		
3	=A2.select(autoType=="Jetta" || autoType="Passat").dup@t()		
4	=A3.derive(interval(tradeDate[-1], tradeDate):space)		
5	=A4.select(autoType[-1]=="Jetta" &&atuoType=="Passat" &&customerID=customerI		
6	=A5.avg(space)		
7			
8			

Sorting & Filtering

	A	B	C
1	=esProc.query("select * from employee")		
2	=A1.sort(entryDate)		
3	=A2.pmin(birthday)	/select recordNo of employee born at earl	
4	=A2(to(A3-1))	/directly access employee record via reco	
5	=esProc.query("select * from stock where	stockCode="000062")	
6	=A5.sort(tradeDate)		
7	=A6.pmax(closePrice)	/recordNo of highest exchange closing qu	
8	=A6.calc(A7,closePrice/closePrice[-1]-1)		
9			
10			

Ordered sets

Reduce Java coupling

JAVA hard coding

Compile and execute

Hard coding algorithm must be compiled and packaged together with the main application

Difficult to modularize

Hard coding algorithm and main application are class dependent, hard to isolate and highly coupled

Hard to hot switch

After the hard coding algorithm is modified, the whole application needs to be recompiled, deployed and restarted, so it is difficult to achieve hot switching

esProc

Interpreted execution

esProc algorithm does not need to be compiled in advance

Easy to modularize

The script file of esProc is independent of the main application and can be placed externally in Java and maintained separately

Easy to hot switch

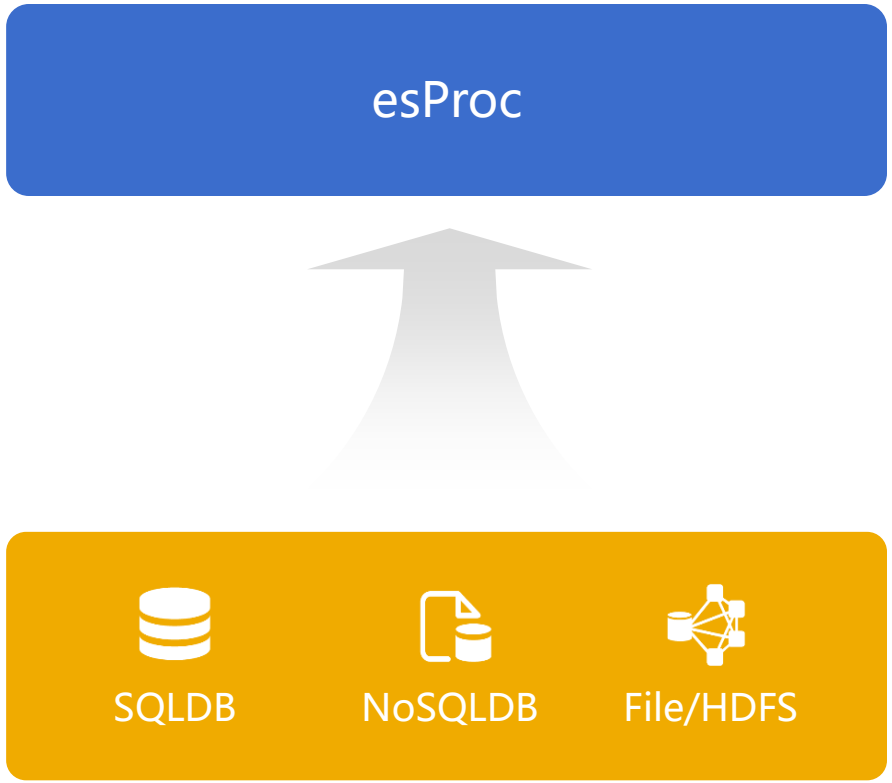
No need to compile and no need to restart, hot switching is easily achieved

Diversity data source support

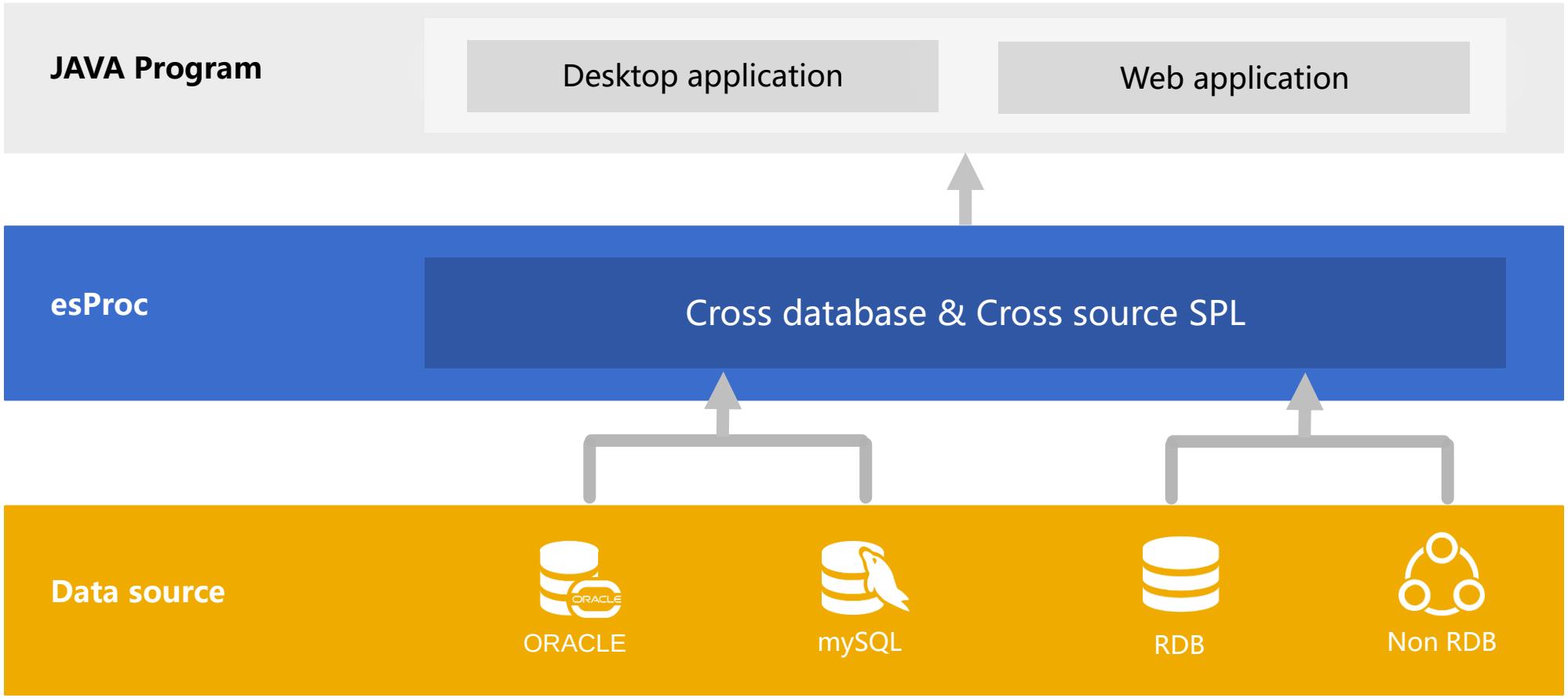
Java computing ability is not competent for diverse data sources.

The built-in interface of esProc can easily access:

- Commercial RDBMS：Oracle, MS SQL Server, DB2, Informix
- Open source RDBMS：MySQL, PostgreSQL
- Open source NOSQL：MongoDB, Redis, Cassandra, ElasticSearch
- Hadoop：HDFS, HIVE, HBase
- Application software：SAP ECC, BW
- File：Excel, Json, XML, TXT
- Others：Http Restful, Web Services, OLAP4j, ...



Cross database computing support



Standard invocation interface

Java code

```
...
Connection con = null;
Class.forName("com.esproc.jdbc.InternalDriver");
con= DriverManager.getConnection("jdbc:esproc:local://");
// Calling stored procedures , CountName is the file name of dfx
st =(com. esproc.jdbc.InternalCStatement)con.prepareCall("call CountName()");
// Execute stored procedures
st.execute();
// Get  result set
ResultSet rs = st.getResultSet();
...
```

Hot deployment of external algorithm reduces coupling effectively

Resource link

- SPL codes of common calculation http://doc.raqsoft.com/
- Query Sharded Databases http://c.raqsoft.com/article/1569381805397
- SPL assists MongoDB calculation http://c.raqsoft.com/article/1573541376418
- Structured Text Computing http://c.raqsoft.com/article/1571711703952
- JSON data calculation and importing into database http://c.raqsoft.com/article/1576466956518
- XML data parsing and calculation http://c.raqsoft.com/article/1577423877262
- How to call an SPL script in Java http://c.raqsoft.com/article/1544066331124

- Installation and free authorization http://c.raqsoft.com/article/1573787506233