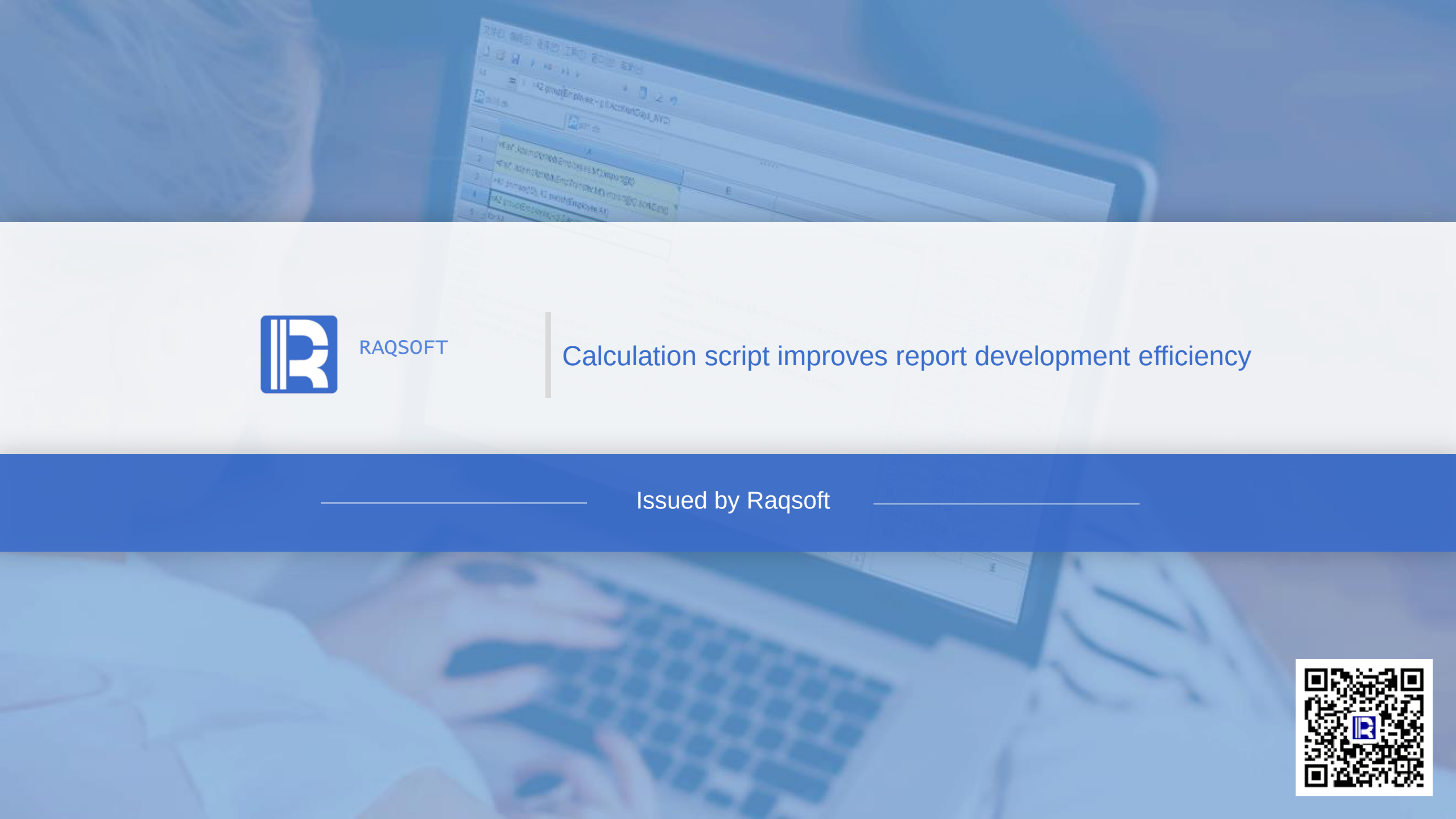


RAQSOFT

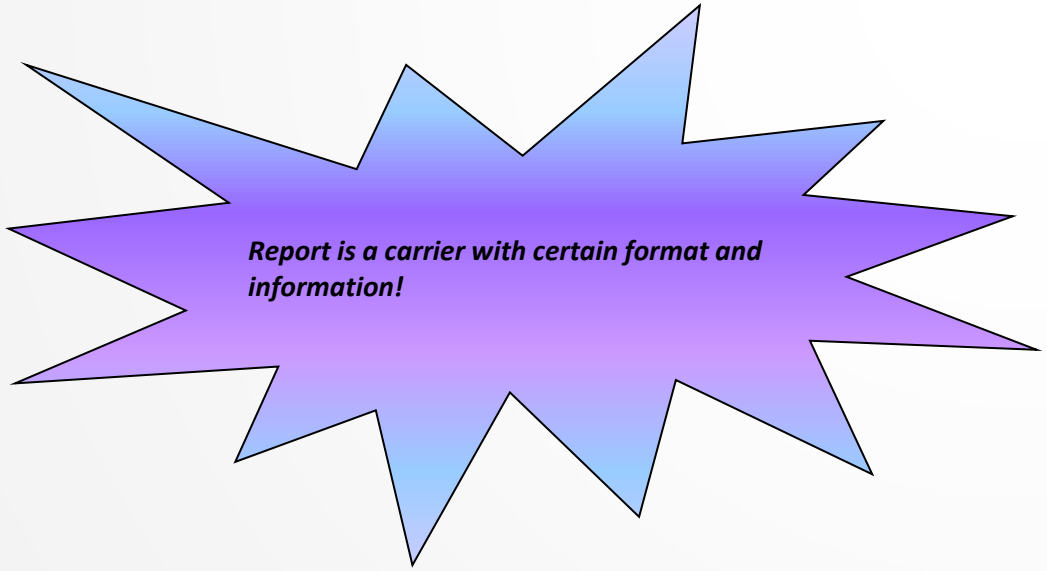
# Calculation script improves report development efficiency

Issued by Raqsoft

# Contents

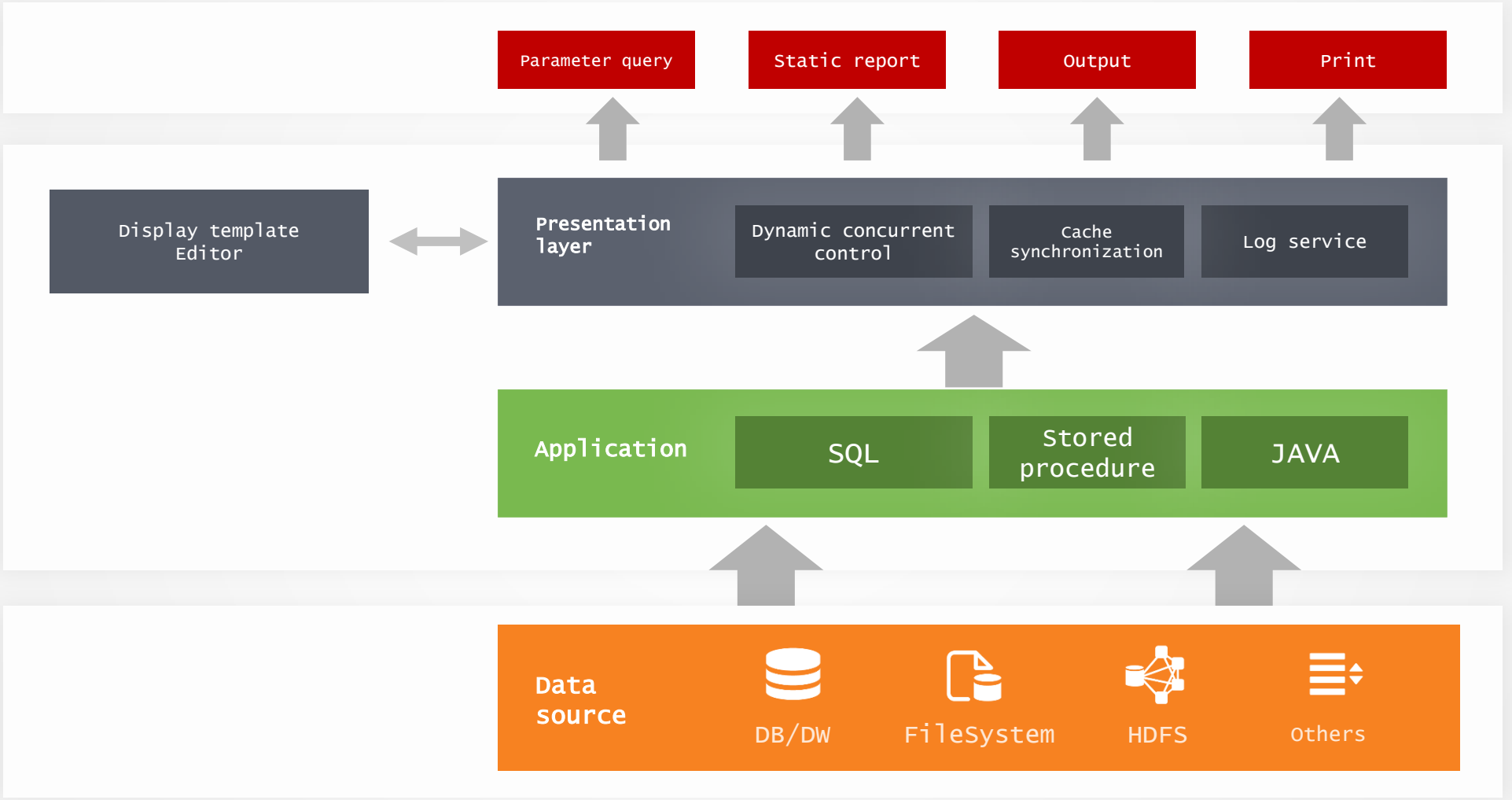
# Report

| Class | Name | Gender | Math | Sport | Art |
|---|---|---|---|---|---|
| Class one | James | M | 8 | 3 | 6 |
| Class one | Jack | M | 2 | 5 | 9 |
| Class one | Alice | F | 7 | 5 | 6 |
| Class one | Bob | M | 4 | 8 | 5 |

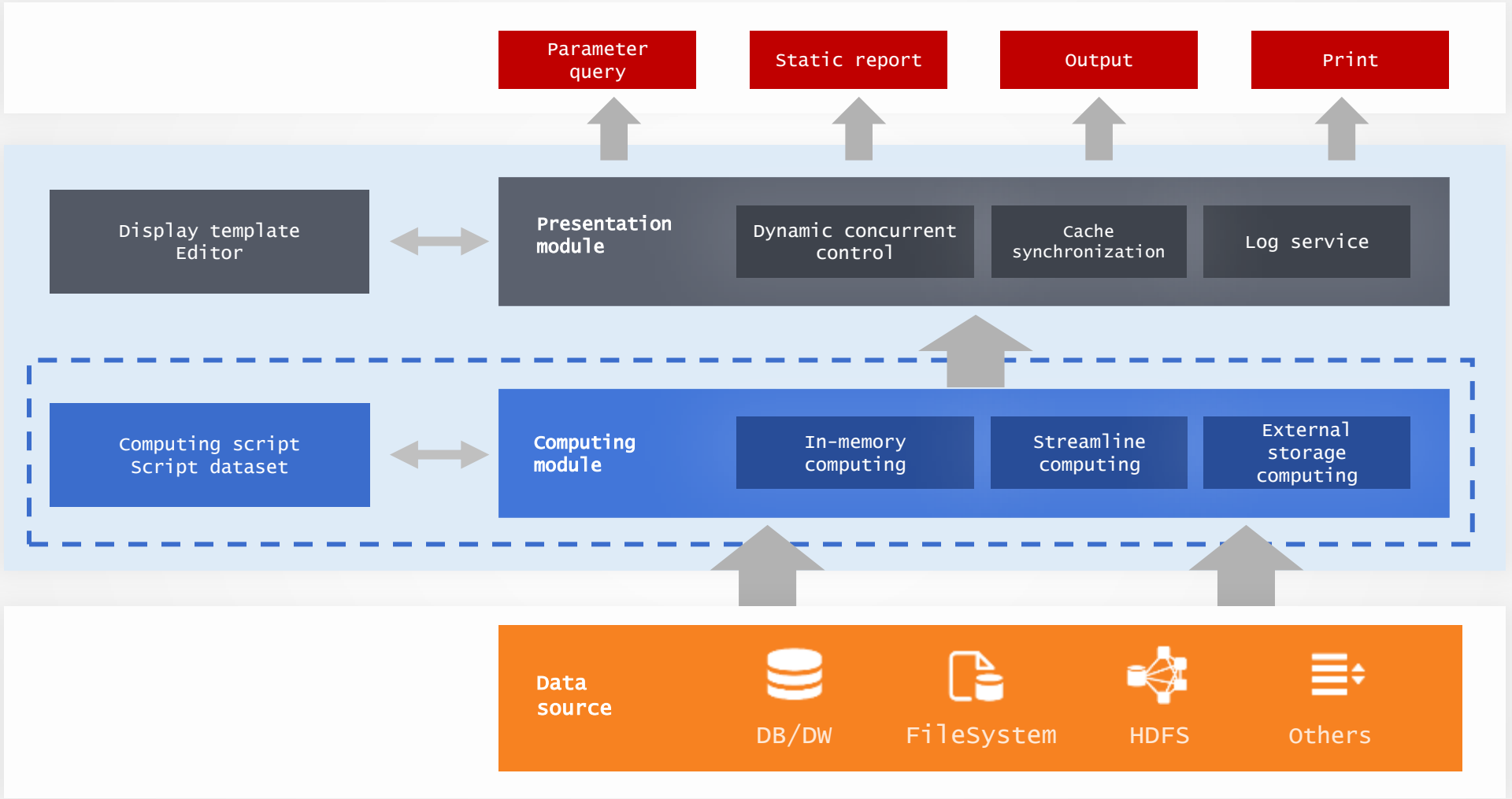




| | Math | Sport | Art |
|---|---|---|---|
| Class one | 7 | 6 | 8 |
| Class two | 6 | 8 | 7 |
| Class three | 7 | 7 | 9 |
| Class four | 8 | 6 | 8 |

# Traditional mode



# Introducing Data Computing Layer




Report template (calculation and presentation) is placed outside the application system, which can be developed by an independent team, and the report implementation process is completely tool enabled.

# Using script to implement difficult computing

## Cross-row calculation

Example: Present the sales amount of each month in a certain year in order of ranking, and calculate the difference between ranks, as well as the growth rate of the current month compared with the previous month.

| | A |
|---|---|
| 1 | =myDB1.query("select month(OrderDate) month,sum(Amount) amount from salesAll where year(OrderDate)=? group by month order by month",theYear) |
| 2 | =A1.derive((amount-amount[-1])/amount[-1]:LRR) |
| 3 | =A2.sort(-amount) |
| 4 | =A3.derive(amount-amount[1]:DIFF) |
| 5 | result A4 |

| month | amount | LRR | DIFF |
|---|---|---|---|
| 12 | 328556.0 | 0.543313448 | 47782.0 |
| 4 | 280774.0 | 0.173804347 | 7134.0 |
| 10 | 273640.0 | 0.124623433 | 9263.0 |
| 7 | 264377.0 | 0.206159980 | 20663.0 |
| 1 | 243714.0 | | 397.0 |
| 9 | 243317.0 | 0.020227008 | 4117.0 |
| 3 | 239200.0 | 0.102197483 | 707.0 |
| 8 | 238493.0 | -0.097905642 | 19304.0 |
| 6 | 219189.0 | 0.020399707 | 2168.0 |
| 2 | 217021.0 | -0.109525919 | 2214.0 |
| 5 | 214807.0 | -0.234946968 | 1917.0 |
| 11 | 212890.0 | -0.222007016 | 212890.0 |

## Reuse of result set

Example: Sales Ranking, map distribution pie chart report

| | A |
|---|---|
| 1 | =connect("db2") |
| 2 | =A1.query("select * from (select e.eid eno,max(e.name) name,max(e.dept) state,sum(amount) amount from sales s,employee e where e.eid=s.sellerid group by e.eid order by amount desc) fetch first 10 rows only" ) |
| 3 | =A1.close() |
| 4 | =A2.groups(state;count(eno):TOP10).sort(TOP10) |
| 5 | result A2,A4 |

# Special format

Some special formats can be implemented by data sources.

| 员工号 | 姓名 | 部门 | 员工号 | 姓名 | 部门 | 员工号 | 姓名 | 部门 |
|---|---|---|---|---|---|---|---|---|
| 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 |

Example: Horizontal columns

| | A | B | C |
|---|---|---|---|
| 1 | =DB.query("select EId,Name,Dept from emp where EId>=? and EId<=? order by EId ",begin,end) | | |
| 2 | =A1.step(3,1) | =A1.step(3,2)\|[null] | =A1.step(3,3)\|[null] |
| 3 | =A2.derive(B2(#).EId:EId2,B2(#).Name:Name2,B2(#).Dept:Dept2,C2(#).EId:EId3,C2(#).Name:Name3,C2(#).Dept:Dept3) | | |

Example: Supplement empty rows

| | A | B |
|---|---|---|
| 1 | =DB.query("select * from employee where gender ='F'") | |
| 2 | =length-A1.len()%length | /Calculate the number of rows to be supplemented |
| 3 | =A1\|if(A2!=length,A2*[null]) | /Result set after the number of empty rows is supplemented |

| Index | Member |
|---|---|
| 259 | [493,Morgan,Davis, ...] |
| 260 | [494,Emily,Johnson, ...] |
| 261 | [496,Rachel,Johnson, ...] |
| 262 | [499,Nicole,Smith, ...] |
| 263 | (null) |
| 264 | (null) |
| 265 | (null) |
| 266 | (null) |
| 267 | (null) |
| 268 | (null) |
| 269 | (null) |
| 270 | (null) |

Result set when the length of rows per page is 15

# Contents

# SQL usage for data outside of database

Using scripts can query external data like database tables with SQL.

Example: Group and filter

| | A |
|---|---|
| 1 | $(esProcOdbc) select Date, count(ID) Count, sum(Quantity) Sum from Order_Electronics.txt group by Date having sum(Quantity)>110000 order by Sum |

| Index | Date | Count | Sum |
|---|---|---|---|
| 1 | 2013-12-25 | 139 | 110002 |
| 2 | 2013-07-22 | 136 | 110149 |
| 3 | 2013-04-19 | 138 | 110417 |
| 4 | 2013-07-11 | 136 | 110655 |
| 5 | 2013-02-10 | 139 | 110717 |
| 6 | 2013-04-10 | 137 | 110811 |
| 7 | 2013-08-29 | 138 | 110856 |
| 8 | 2013-07-26 | 139 | 111163 |
| 9 | 2013-01-05 | 139 | 111454 |
| 10 | 2013-04-29 | 136 | 111526 |

After grouping, use the having clause to select the data whose sum (quantity) is greater than 110000 from the results.

Example: Join

| | A |
|---|---|
| 1 | $(esProcOdbc) select C.NAME,C.POPULATION, S.StateId, S.Abbr from states.txt S join cities.txt C on S.StateId=C.STATEID |

| Index | C.NAME | C.POPULATION | S.StateId | S.Abbr |
|---|---|---|---|---|
| 1 | Argentina | 435245.0 | 5 | CA |
| 2 | Auckland | 402777.0 | 5 | CA |
| 3 | Anaheim | 334425.0 | 5 | CA |
| 4 | Bakersfield | 308392.0 | 5 | CA |
| 5 | Chula Vista | 212756.0 | 5 | CA |
| 6 | Chicago | 2886251.0 | 13 | IL |
| 7 | Buffalo | 276059.0 | 32 | NY |
| 8 | Austin | 671873.0 | 43 | TX |
| 9 | Arlington | 349944.0 | 43 | TX |
| 10 | Corpus Christi | 285267.0 | 43 | TX |

Use join statement to associate two external tables and list the city information that can find the corresponding state abbreviation.

# Multilayer data

Simple analysis and calculation of string with JSON format.

Example:  Using JSON data as a report data source.

```
1  {
2      "order": [{
3              "client": "Google",
4              "date": "2015-6-23",
5              "item": [{
6                      "product": "HP NoteBook",
7                      "number": 8,
8                      "price": 3200
9                  },
10                 {
11                     "product": "Dell Server",
12                     "number": 4,
13                     "price": 22100
14                 }
15             ]
16         },
```

Multilayer JSON

| client | product | quantity | price |
|---|---|---|---|
| Apple | | | |
| | Lenove PC | 3 | 2700 |
| | SONY TV | 1 | 14900 |
| | Dell Server | 1 | 3400 |
| Audi | | | |
| | Lenove PC | 2 | 5100 |
| | Dell Server | 1 | 7800 |
| | Bandy fridge | 1 | 2500 |
| Google | | | |
| | HP NoteBook | 8 | 3200 |
| | Dell Server | 4 | 22100 |
| Tesla | | | |
| | iPad | 1 | 7800 |
| | Midea Air Filter | 1 | 9800 |
| | Bandy fridge | 1 | 4400 |

# Multilayer data

Simple analysis and calculation of string with JSON format.

Example: Using JSON data as a report data source.

| | A |
|---|---|
| 1 | =file("order.json":"utf-8").read().import@j().(order) |
| 2 | =A1.news(item;client,product,quantity,price) |
| 3 | return A2 |

A.news: Calculate the field value of the sequence table and merge to generate a new sequence table.

| client | date | item |
|---|---|---|
| Google | 2015-6-23 | [[HP NoteBook,8,3200],[Dell Server,4,22100]] |
| Apple | 2017-7-12 | [[Lenove PC,3,2700],[SONY TV,1,14900],[De... |
| Tesla | 2016-8-3 | [[iPad,1,7800],[Midea Air Filter,1,9800],[Band... |
| Audi | 2013-9-20 | [[Lenove PC,2,5100],[Dell Server,1,7800],[Ba... |

| client | product | quantity | price |
|---|---|---|---|
| Google | HP NoteBook | 8 | 3200 |
| Google | Dell Server | 4 | 22100 |
| Apple | Lenove PC | 3 | 2700 |
| Apple | SONY TV | 1 | 14900 |
| Apple | Dell Server | 1 | 3400 |
| Tesla | iPad | 1 | 7800 |
| Tesla | Midea Air Filter | 1 | 9800 |
| Tesla | Bandy fridge | 1 | 4400 |
| Audi | Lenove PC | 2 | 5100 |
| Audi | Dell Server | 1 | 7800 |
| Audi | Bandy fridge | 1 | 2500 |

A2 is the company's purchase detail list

# Cross database SQL translation

Translate standard SQL into SQL of various databases.

Example: Translate "the week of the year" in standard SQL into SQL for various databases.

| Standard function | Meaning | oracle | sql server | db2 | mysql | teradata | hsql | PostgresSQL |
|---|---|---|---|---|---|---|---|---|
| WEEKOFYEAR (d) | No. of weeks in the year | TO_NUMBER(TO_CHAR(d,'WW')) | DATEPART (WW,d) | WEEK(d) | WEEK(d) | TD_WEEK_OF_YEAR (d) | WEEK(d) | EXTRACT(WEEK FROM d) |

| | A |
|---|---|
| 1 | SELECT ID,WEEKOFYEAR(DATES),CUSTOMER,AREA FROM CLUE |
| 2 | =A1.sqltranslate("ORACLE") |
| 3 | =A1.sqltranslate("SQLSVR") |
| 4 | =A1.sqltranslate("MYSQL") |

| Value |
|---|
| SELECT ID,TO_NUMBER(TO_CHAR(DATES,'WW')),CUSTOMER,AREA FROM CLUE |

A2 is the SQL syntax corresponding to Oracle

| Value |
|---|
| SELECT ID,DATEPART(WW,DATES),CUSTOMER,AREA FROM CLUE |

A3 is the SQL syntax corresponding to SQL Server

| Value |
|---|
| SELECT ID,WEEK(DATES),CUSTOMER,AREA FROM CLUE |

A4 is the SQL syntax corresponding to MySQL

All kinds of databases have their own string functions, numerical functions, time functions, conversion functions, arbitrary functions.
It will bring great convenience for development and maintenance to unify these different standards.
Currently supports but not limited to: Oracle、SQL Server、DB2、MySQL、HSQL、Teradata、PostgreSQL

# Dynamic data source

Dynamic data source can be realized by script computing layer.

One solution is to pass in jdbc url/username/password with parameters, but it will affect security (password is passed in parameters).

Example: Query details with amount greater than a certain threshold in different data sources.

| | A |
|---|---|
| 1 | =${pSource}.query("select * from sOrder where Amount>? ",pAmount) |

Where pSource, pAmount are report parameters，pSource represents the data source name，pAmount represents the order amount.

# Cross database computing

Single database with too large amount of data will affect the performance, so it will usually divide the databases and share the computing pressure.

Simple query example: The order table orders is stored in two Oracle databases, and the data source names are orcla and orclb respectively. Filter out orders with an amount greater than or equal to 10000, and sort by order of order amount.

| | A | B | C |
|---|---|---|---|
| 1 | =[connect("orclA"),connect("orclB")] | | /Connect multiple data sources |
| 2 | SELECT * FROM orders WHERE amount >= 10000 ORDER BY amount | | /Sort |
| 3 | fork A1 | =A3.query(A2) | /Parallel computing |
| 4 | =A3.merge(AMOUNT) | | /Merge result |
| 5 | =A3.(~.close()) | | /Close |

Group aggregation example: Group the order table by year and month, and aggregate the amount field of each group of data.

| | A | B | C |
|---|---|---|---|
| 1 | =[connect("orclA"),connect("orclB")] | | /Connect multiple data sources |
| 2 | SELECT EXTRACT(YEAR FROM orderTime) AS y, EXTRACT(MONTH FROM orderTime) AS m, SUM(amount) AS amount FROM orders GROUP BY EXTRACT(YEAR FROM orderTime), EXTRACT(MONTH FROM orderTime) | | /Group aggregation |
| 3 | fork A1 | =A3.query(A2) | /Parallel computing |
| 4 | =A3.conj() | | /Merge result |
| 5 | =A4.groups(Y,M;sum(AMOUNT):AMOUNT) | | /Group aggregation again |
| 6 | =A3.(~.close()) | | /Close |

# Cross database computing

Single database with too large amount of data will affect the performance, so it will usually divide the databases and share the computing pressure.

Example of join: Sales table is the dimension table of orders table. The two tables take salesid as the associated field and group by dept field of sales table to calculate the sales of each department. Assume that the sales table has been fully stored in each database.

| | A | B | C |
|---|---|---|---|
| 1 | =[connect("orclA"),connect("orclB")] | | /Connect multiple data sources |
| 2 | SELECT sales.dept, SUM(orders.amount) AS amount FROM orders, sales WHERE orders.salesID = sales.salesID GROUP BY sales.dept | | /Group aggregation |
| 3 | fork A1 | =A3.query(A2) | /Parallel computing |
| 4 | =A3.conj() | | /Merge result |
| 5 | =A4.groups(DEPT;sum(AMOUNT):AMOUNT) | | /Group aggregation again |
| 6 | =A3.(~.close()) | | /Close |

Example of heterogeneous databases: Orders table is stored in Oracle and MySQL, and the data source name is ora and my. Query the records whose amount field is greater than or equal to 10000, and truncate and round the amount field.

| | A | B | C |
|---|---|---|---|
| 1 | =[[connect("ora"),"ORACLE"],[connect("my")] ,"MYSQL"] | | /Connect data source, mark database type |
| 2 | SELECT ORDERID, ORDERTIME, truncate(AMOUNT, 0) , CLIENTID, SALESID FROM orders WHERE amount >= 10000 | | /Standard SQL |
| 3 | fork A1 | =A2.sqltranslate(A3(2)) | /translate into local SQL |
| 4 | =A3.conj() | =A3.query(B3) | /Query |
| 5 | =A3.conj() | | /Merge result |
| 6 | =A3.(~.close()) | | /Close |

# NoSQL/Hadoop

For some databases that do not support ODBC and JDBC connection, you can use the external library function.

Take MongoDB for example

1. Deploy external database:
Download the following two files from the Internet (please download the corresponding jar files according to the actual version) and put them into the external Library folder of the product; the path of Mongo external library file is: installation directory\esproc\extlib\mongodbcli; and the Raqsoft core jar is mongocli.jar.

bson-3.6.3.jar
mongo-java-driver-3.6.3.jar

2、Access to MongoDB:
The external library functions that can be used are mongo_open()、mongo_shell()、mongo_close().

| | A | B |
|---|---|---|
| 1 | =mongo_open("mongodb://localhost:27017/mydb") | /Connect mydb of mongo server |
| 2 | =mongo_shell(A1,"emp.find()").fetch() | Query the records in emp set of mydb |
| 3 | =mongo_close(A1) | /Close |

The external database can also respectively access Alibaba cloud, elasticsearch, hive, spark, HBase, redis, Cassandra, Informix database, read report files, connect HDFS file system, multidimensional database, WebService, FTP, sap, Kafka system.

# Multi data source mix

It is difficult to develop the multi-source mixed computation directly, so the computing module can be used.

Example: The client column in the sales table is the customer name, and the amount column is the order amount. There is also a list of potential customer names in the potential.json file. It is required to summarize the order amount in groups in the sales table according to the order in the potential customer file.

| | A |
|---|---|
| 1 | =sales=demo.query("select * from sales") |
| 2 | =potential=file("potential.json").read().import@j().(potential) |
| 3 | =sales.align@a(potential,CLIENT) |
| 4 | =A3.new(potential(#):CLIENT,~.sum(AMOUNT):AMOUNT) |

| ORDERID | 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 |
| 4 | PWQ | 9 | 26100.0 | 2012-11-08 15:28:05 |

Partial data of sales table in A1

| Index | Member |
|---|---|
| 1 | GLH |
| 2 | EGU |
| 3 | YZ |
| 4 | MIP |
| 5 | VILJX |
| 6 | DNEDL |
| 7 | PJIPE |
| 8 | HANAR |
| 9 | BTMMU |
| 10 | JOPO |

A2 is the list of potential customers

| Index | Member |
|---|---|
| 1 | [[39,GLH,14, ...],[49,GLH,2, ...],[68,GLH,13, ...], ...] |
| 2 | [[7,EGU,2, ...],[20,EGU,8, ...],[90,EGU,4, ...], ...] |
| 3 | [[30,YZ,19, ...],[65,YZ,8, ...],[78,YZ,1, ...], ...] |
| 4 | [[15,MIP,16, ...],[69,MIP,16, ...],[74,MIP,11, ...], ...] |
| 5 | [[8,VILJX,7, ...],[35,VILJX,6, ...],[41,VILJX,8, ...], ...] |
| 6 | [[18,DNEDL,16, ...],[59,DNEDL,3, ...],[101,DNEDL,5, ...], ...] |
| 7 | [[17,PJIPE,3, ...],[26,PJIPE,20, ...],[111,PJIPE,11, ...], ...] |
| 8 | [[6,HANAR,18, ...],[34,HANAR,5, ...],[43,HANAR,5, ...], ...] |
| 9 | [[23,BTMMU,17, ...],[40,BTMMU,17, ...],[92,BTMMU,2, ...], ...] |
| 10 | [[19,JOPO,2, ...],[27,JOPO,20, ...],[80,JOPO,9, ...], ...] |

Results after grouping potential customers in A3

| ORDERID | CLIENT | SELLERID | AMOUNT | ORDERDATE |
|---|---|---|---|---|
| 39 | GLH | 14 | 22200.0 | 2012-12-15 15:28:05 |
| 49 | GLH | 2 | 8526.0 | 2012-12-24 15:28:05 |
| 68 | GLH | 13 | 24700.0 | 2013-01-07 15:28:05 |
| 75 | GLH | 6 | 20140.0 | 2013-01-13 15:28:05 |

| ORDERID | CLIENT | SELLERID | AMOUNT | ORDERDATE |
|---|---|---|---|---|
| 7 | EGU | 2 | 17800.0 | 2012-11-06 15:28:05 |
| 20 | EGU | 8 | 11700.0 | 2012-11-21 15:28:05 |
| 90 | EGU | 4 | 26700.0 | 2013-01-30 15:28:05 |
| 110 | EGU | 10 | 11000.0 | 2013-02-24 15:28:05 |

... ...

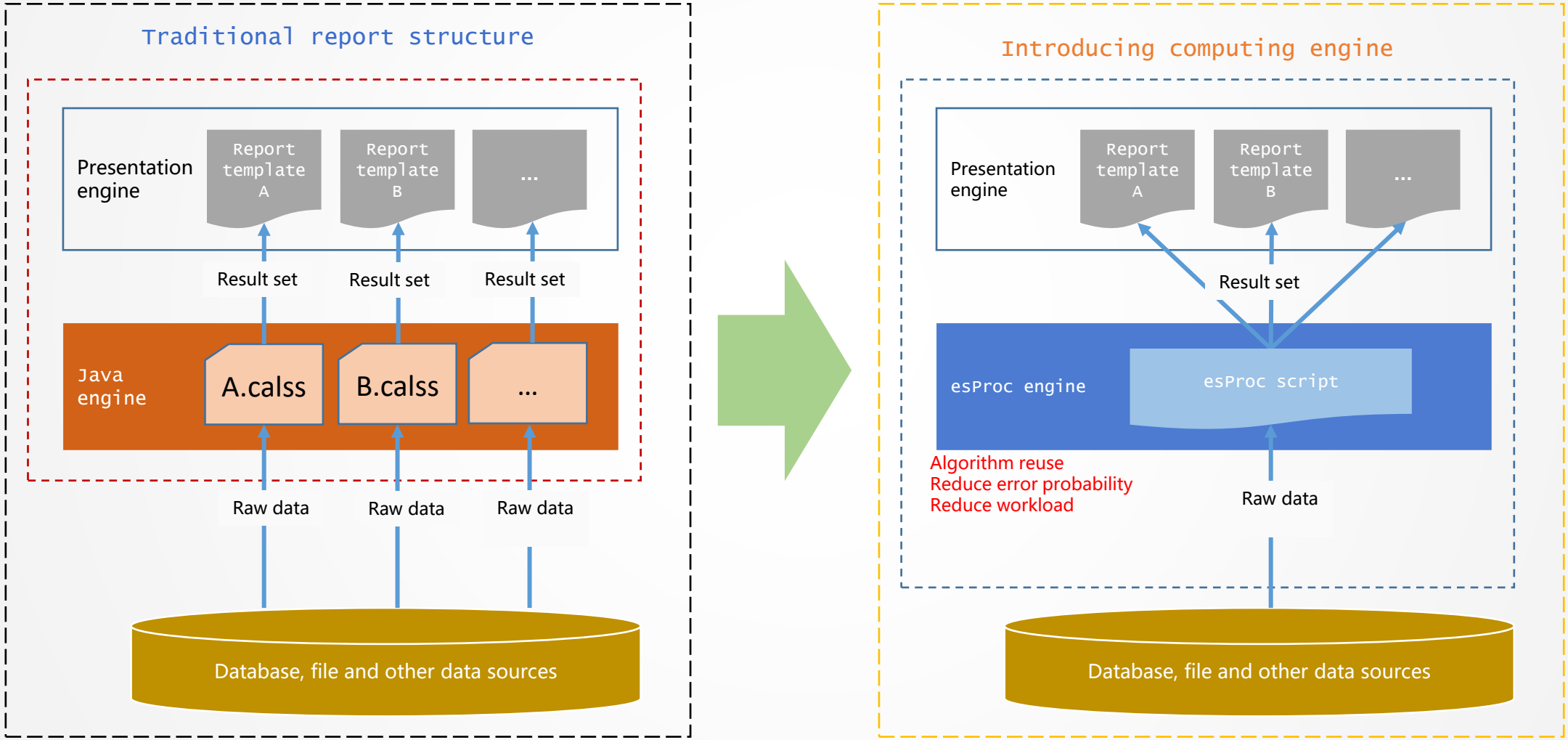
Records in the first two groups in A3

| CLIENT | AMOUNT |
|---|---|
| GLH | 461052.0 |
| EGU | 598344.0 |
| YZ | 550882.0 |
| MIP | 397000.0 |
| VILJX | 585320.0 |
| DNEDL | 601164.0 |
| PJIPE | 443102.0 |
| HANAR | 669736.0 |
| BTMMU | 496226.0 |
| JOPO | 632798.0 |

A4 is the total amount of potential customers

# Contents

# Algorithm reuse - structure chart



# Algorithm reuse

General algorithm reuse reduces error probability and workload.

Example, "employee performance salary details" and "department performance salary summary" both need to calculate employee performance salary. Although the algorithm is complex, it is basically the same.

| | A | B |
|---|---|---|
| 1 | =demo.query("select * from employee") | |
| 2 | if (trim(where) !=null) | =A1.select@o(${where}) |
| ... | //Omit N lines complex code here | |
| 13 | result A12 | |

| EMPID | SCORE | DEPT | EMPNAME | BASESALARY | STATE | PRETAX |
|---|---|---|---|---|---|---|
| 3 | 1.10 | Sales | Rachel | 8000 | New Mexico | 8052.80... |
| 6 | 1.18 | Sales | Matthew | 4000 | California | 4037.76... |
| 1 | 0.75 | R&D | Rebecca | 6000 | California | 6000 |
| 2 | 0.90 | Finance | Ashley | 10000 | New York | 10000 |
| 4 | 0.80 | HR | Emily | 800 | Texas | 800 |
| 5 | 1.40 | R&D | Ashley | 3000 | Texas | 3000 |

When the query criteria where is empty, the returned result set is suitable for "department performance salary summary"

| EMPID | SCORE | DEPT | EMPNAME | BASESALARY | STATE | PRETAX |
|---|---|---|---|---|---|---|
| 6 | 1.18 | Sales | Matthew | 4000 | California | 4037.76... |
| 1 | 0.75 | R&D | Rebecca | 6000 | California | 6000 |
| 4 | 0.80 | HR | Emily | 800 | Texas | 800 |
| 5 | 1.40 | R&D | Ashley | 3000 | Texas | 3000 |

When the query criteria where is state = = "California" | state = = "Texas", the returned result set is suitable for "employee performance salary details"

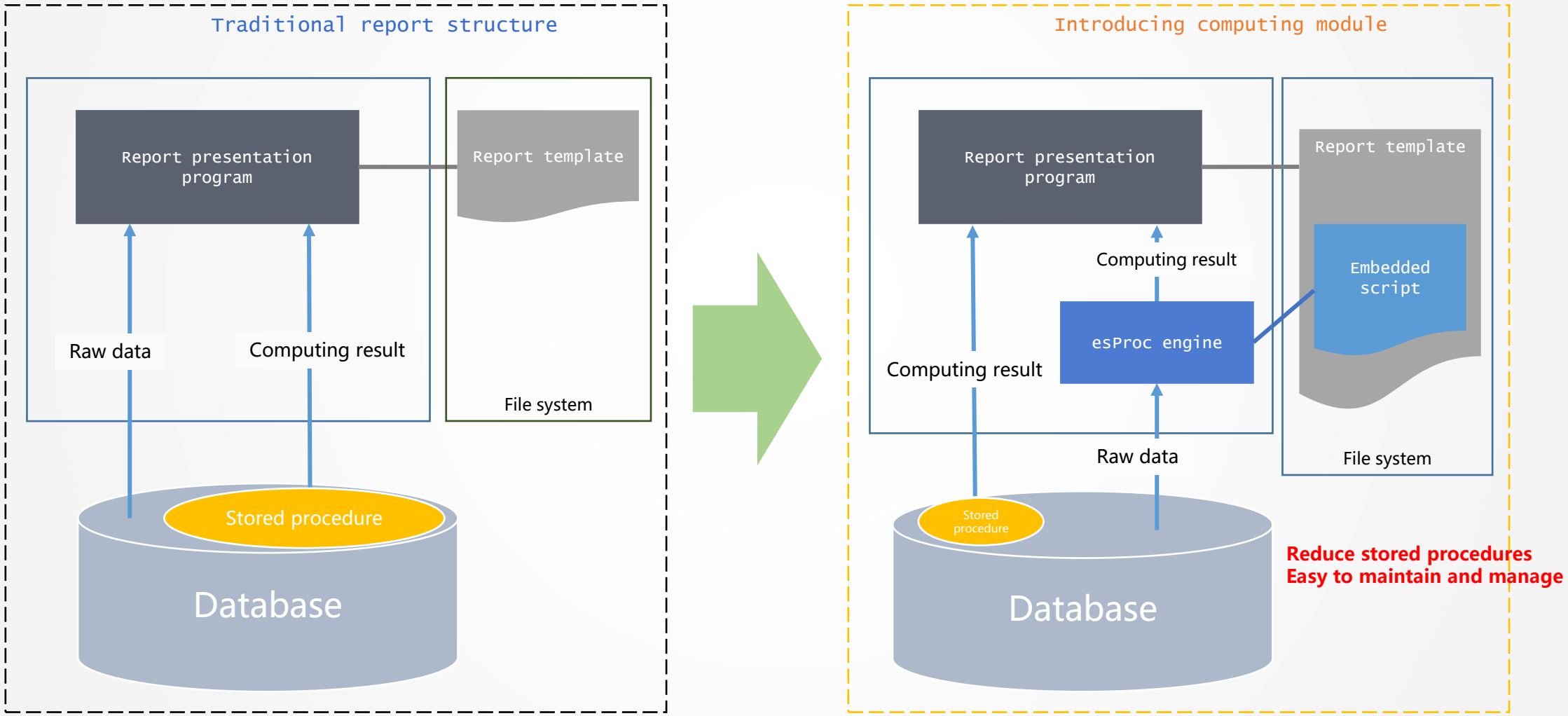
| EMPNAME | BASESALARY | PRETAX | STATE |
|---|---|---|---|
| Matthew | $4000.00 | $4037.76 | California |
| Rebecca | $6000.00 | $6000.00 | California |
| Emily | $800.00 | $800.00 | Texas |
| Ashley | $3000.00 | $3000.00 | Texas |

employee performance salary details

| department | count | total:basesalary | total:pertax |
|---|---|---|---|
| Finance | 1 | $10,000.00 | $10,000.00 |
| HR | 1 | $800.00 | $800.00 |
| R&D | 2 | $9,000.00 | $9,000.00 |
| Sales | 2 | $12,000.00 | $12,090.56 |

department performance salary summary

# Stored procedure outside the database - structure chart



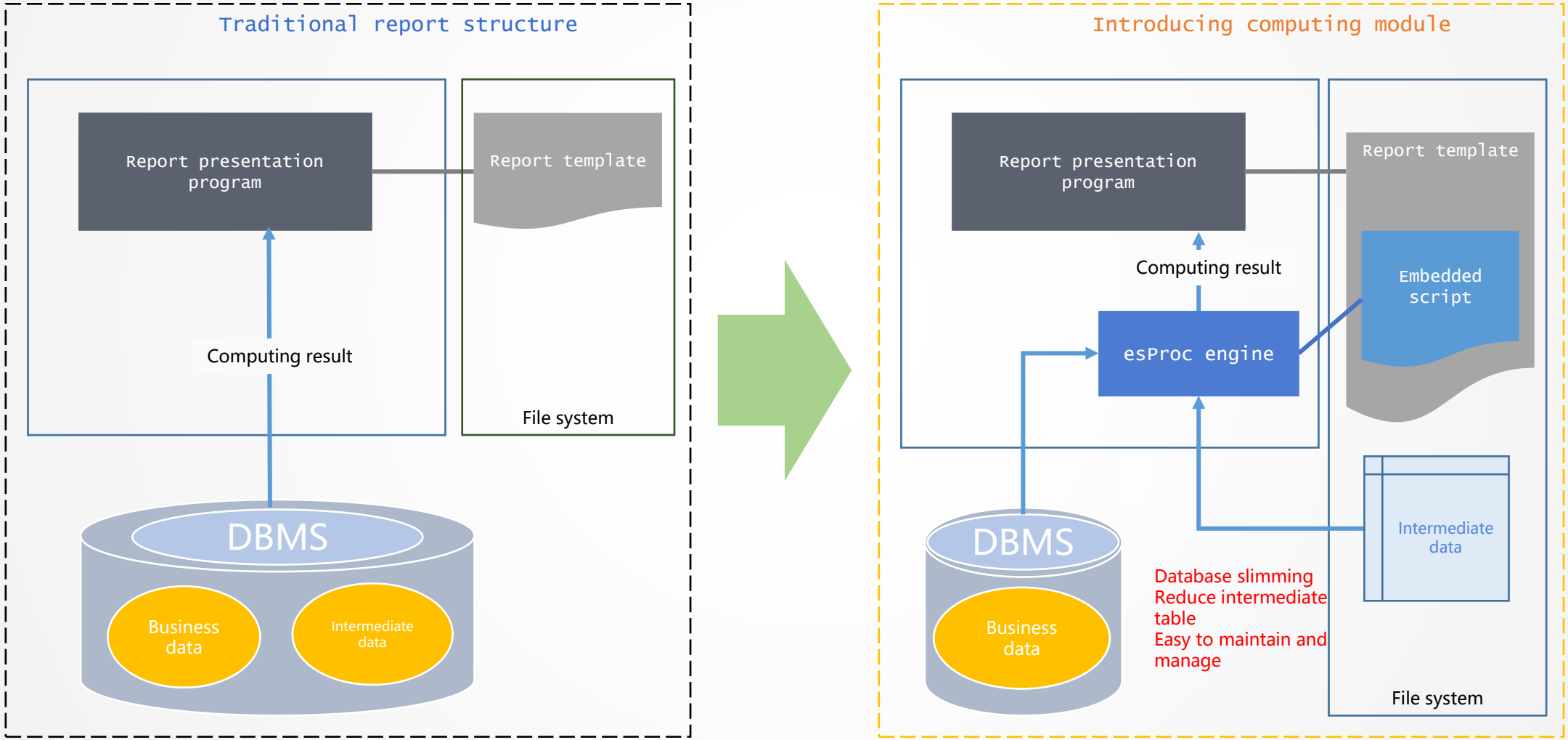
# Stored procedure outside the database

When it is inconvenient to modify the stored procedure in the database, the calculation outside the database can be realized with the help of the computing module.

Example: The calculation goal of a development team is to specify a certain time period in a year, and calculate the top 20 superior products sold in each state in that time period. The monthly sales amount of these superior products and the growth rate of sales amount of each month compared with that of last month are calculated.

| | A | B | C |
|---|---|---|---|
| 1 | =ora.query("select State,Product,Amount,Time from sales where to_char(Time,'YYYMMDD')>=? and to_char(Time,'YYYMMDD')<=? ";start,end) | | /Get data from sales table according to time parameter |
| 2 | =A1.group(State) | =A2.(~.rank(Amount).pselect@a(~<=20)) | /A2: Group the data in A1 by state. B2: In each group (state) of A2, find the record number of the top 20 products sold. |
| 3 | =A2.(~(B2(#)).(Product)) | | /Take the product corresponding to each state from A2 with serial number |
| 4 | =A3.isect() | | /Find the intersection of each group of products in A3. The result here is the superior product. |
| 5 | =A1.select(A4.pos(Product)>0) | | /Filter out the sales records of the superior products |
| 6 | =A5.groups(Product,month(Time):Month;sum(Amount):MonthAmount) | | /According to A5, summarize the monthly sales amount of each product |
| 7 | =A6.derive(Rate) | | /Add a field rate to A6 for future storage of growth rate |
| 8 | =A7.group(Product) | | /Regroup A7 by product |
| 9 | =A8.(~.run(Rate=MonthAmount[-1]/MonthAmount-1)) | | /Calculate the monthly growth rate of each product over the previous month |
| 10 | =A9.union() | | /Merge the grouped data in A9. That's the ultimate goal |
| 11 | return A10 | | /Return result A10 |

# Application intermediate table - structure chart



# Application intermediate table

The increase of database intermediate table will cause the decrease of database performance, increase the cost of management and maintenance.

Example: a company's customer sales compared with last year's annual sales (large amount of annual data).

1. Create an intermediate table: read the sales data of last year from the database and export it as an intermediate file.

| | A | B |
|---|---|---|
| 1 | =db.query("select client,count(orderid) c,sum(amount) s from sales where year(orderdate)=? group by client",year(now())-1) | /Read 2009 sales data from database |
| 2 | =file(string(year(now())-1)+"sales.b").export@b(A1) | /Last year's data was exported as a set file |

2. Calculate the intermediate file (last year's data) and the database data(this year's data) together to get the sales comparison results.

| | A | B |
|---|---|---|
| 1 | =db.query("select client,count(orderid) c,sum(amount) s from sales where year(orderdate)=? group by client",year(now())) | /Calculate and retrieve the number of orders and sales amount of the year to query from the database. |
| 2 | =file(string(year(now())-1)+"sales.b").import@b() | /Extract data from data files of the previous year |
| 3 | =A2.align(A1:CLIENT,CLIENT) | /Align the data in A2 with the client field in A1, and fill in blank lines if there is one in A1 but not in A2 |
| 4 | =A1.new(CLIENT,C:COUNT,S:TOTAl,A3(#).C:lastCOUNT,A3(#).S:lastTOTAL,S/A3(#).S:PROPORTION) | /Use A1 to generate a new sequence table. The corresponding row data of A3 is added, for example,A3(#).C:lastCOUNT means the c field of corresponding row data of A3, where # is the current row number of A1 |
| 5 | return A4 | /Return result set |

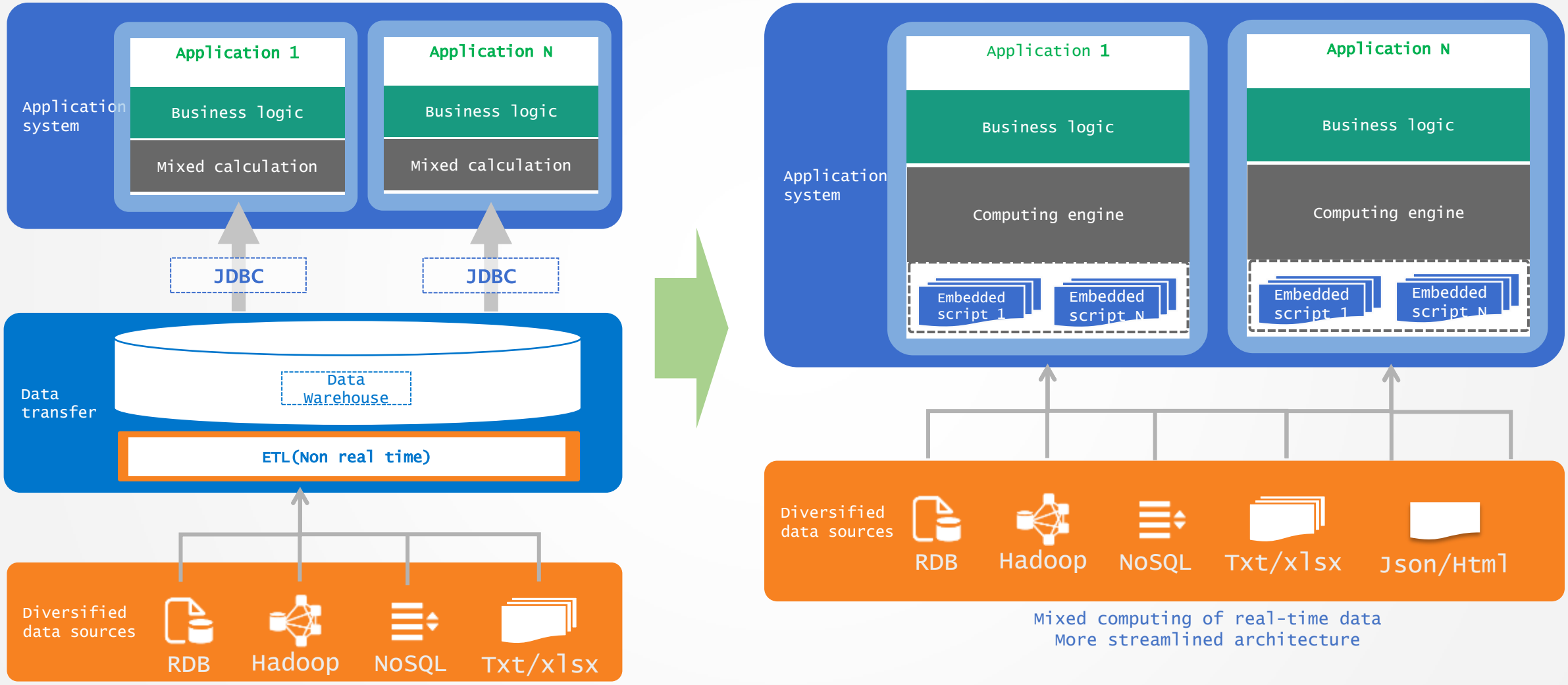
| client | quantity | amount | L_quantity | L_amount | L/N_amount |
|---|---|---|---|---|---|
| JOPO | 12 | 107010.0 | 19 | 255012.0 | 41.96% |
| AYWYN | 8 | 136578.0 | 12 | 152600.0 | 89.50% |
| SAVEA | 13 | 137916.0 | 13 | 158874.0 | 86.81% |
| PJIPE | 8 | 106436.0 | 17 | 225042.0 | 47.30% |
| HANAR | 18 | 275464.0 | 19 | 242076.0 | 113.79% |
| ERNSH | 15 | 158446.0 | 16 | 141660.0 | 111.85% |
| DNEDL | 13 | 209652.0 | 17 | 232994.0 | 89.98% |
| BTNMU | 15 | 217186.0 | 10 | 168790.0 | 128.67% |
| JAYB | 13 | 201706.0 | 14 | 188348.0 | 107.09% |
| HP | 15 | 217986.0 | 23 | 244816.0 | 89.04% |

The customers, orders and sales in this report are calculated directly from the database, that is, the data of this year.

The number of orders and sales amount of the whole year last year are read from the xxxsales.B file in the file system.

"Sales / last year's sales" is calculated by the data of this year and last year.

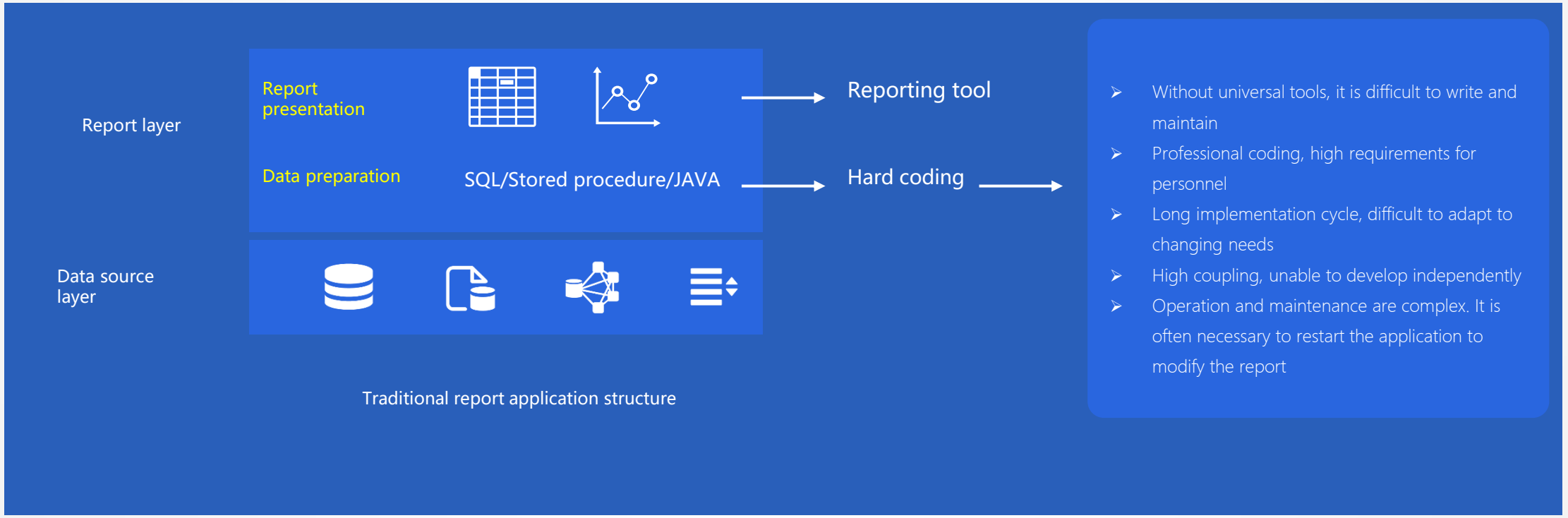
# Real time multi-source Report




Mixed computing of real-time data
More streamlined architecture

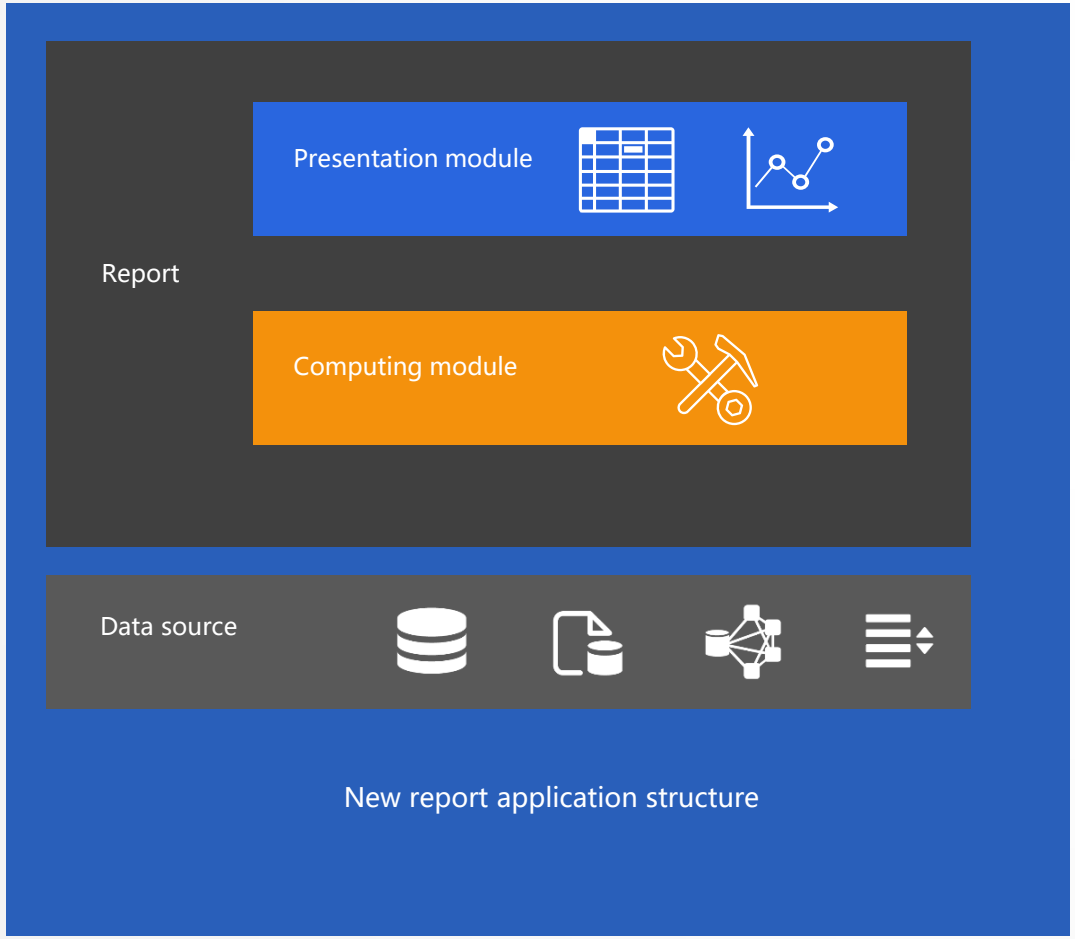
# Endless reports

It's not enough to only solve the report presentation phase. Most of the workload of report development is in the data preparation phase.



# Solution

Add data preparation layer (computing module) in reporting tool to make report development tool enabled completely and realize high efficiency!




### Make report development tool enabled

Both report presentation and data preparation can be developed using report tools.

### Make report module independent

Report presentation and data preparation are independent from application and database, maintained separately and decoupled from application.

### Make developers extensive

General technicians can do it, with low cost and quick response.

# Staffing and knowledge transmission

To deal with the endless problems of reports is a comprehensive business involving technology, management and other aspects.

Solve presentation problems
Reduce workload

With the help of reporting tool

Improve communication mechanism

Improve the development efficiency of data preparation

With the help of computing module

Establish effective communication mechanism
Reduce misunderstanding in communication

Independent report module

Build report team

Adjust application structure
Decouple report module from business system

Reduce developer costs
Improve the response speed

# Contents

# Parallel fetch data

The performance of database JDBC is poor, which can be solved by multithreading in computing module.

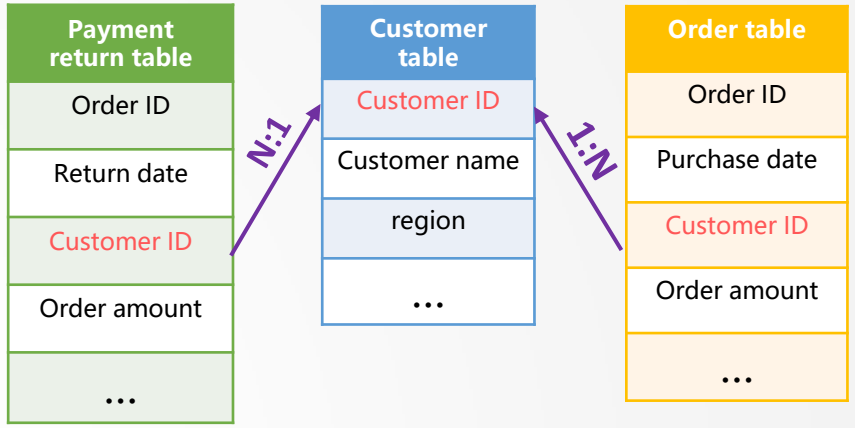
Example: According to the data field part in table t, the data is divided into four parts, and each thread reads a part of the data.

| | A | B | C |
|---|---|---|---|
| 1 | fork 4 | =connect(db) | /4 threads, set up connection respectively |
| 2 | | =B1.query@x("select * from T where part=? ",A1) | /Select each segment separately |
| 3 | =A1.conj() | | /Merge results |

# Multi-table Join

The efficiency of multi table join is low in the presentation stage, and it is implemented in the script stage instead.

Example: There are three datasets in the report, which are from the return table, customer table and order table. The customer ID of the return table points to the customer ID of the customer table primary key, and the customer ID of the order table also points to the customer ID of the customer table primary key.




| | A |
|---|---|
| 1 | =connect("demo") |
| 2 | =A1.query("SELECT Customer ID, SUM(Order amount) AS Order amount FROM Order WHERE Purchase date >=? AND Purchase date <? GROUP BY Customer ID",begin,end) |
| 3 | =A1.query( "SELECT Customer ID, SUM(Return amount) AS Return amount FROM return WHERE return date>= ? AND return date<= ? GROUP BY Customer ID",begin,end) |
| 4 | =A1.query@x("select Customer ID, Company name from Customer") |
| 5 | =join@1(A4:customer,Customer ID;A2:Order,Customer ID;A3:Return ID,customer) |
| 6 | =A5.new(Customer.Company name:Customer name,Order.Order amoumt:Order amount,Return.Return amount:Return amount) |

| client | orders | Fund_received |
|---|---|---|
| [RANCH,Apple] | [RANCH,2844.10000... | [RANCH,1149.4] |
| [LILAS,Google] | [LILAS,16005.199969... | [LILAS,11831.0] |
| [LINOD,Dell] | [LINOD,16476.56498... | [LINOD,7803.950000... |
| [RATTC,Microsoft] | [RATTC,50134.67998... | [RATTC,30520.30000... |

| CLIENTID | COMPAN |
|---|---|
| RANCH | Apple |

| CLIENTID | AMOUNT |
|---|---|
| RANCH | 2844.10... |

| CLIENTID | FUND_RECEIVED |
|---|---|
| RANCH | 1149.4 |

A5 left join order and payment return data according to the customer

| company | amount | Fund_received |
|---|---|---|
| Apple | 2844.1 | 1149.4 |
| Google | 16005.2 | 11831.0 |
| Dell | 16476.56 | 7803.95 |
| Microsoft | 50134.68 | 30520.3 |

A6 get join result and return report dataset.

# Disassembling SQL calculation

The execution path is difficult to control, which hinders the report optimization. The calculation script can disassemble SQL to complete the report optimization.

Example: A detail table has a large amount of data, which involves many database tables, and the join between database tables is frequent. The execution path of SQL is not controllable.

```
select syb.org_abbreviation as syb,
        max(xmb.org_abbreviation) as xmb,
        …
        -- Omit multiple join, judgment and summary statements
        left join losrrr losr on losr.requisition_id =
            l.requisition_id
        where l.table_type = '1'
             and l.requisition_state =
'0100500005000000006'
             and nvl(l.bsflag, 0) != 1
        group by l.requisition_id,
                 l.note,
                 -- Omit multiple grouping fields
                 losr.standby_param3,
                 losr.standby_param6
```

Complex dataset SQL (nearly 400 lines)

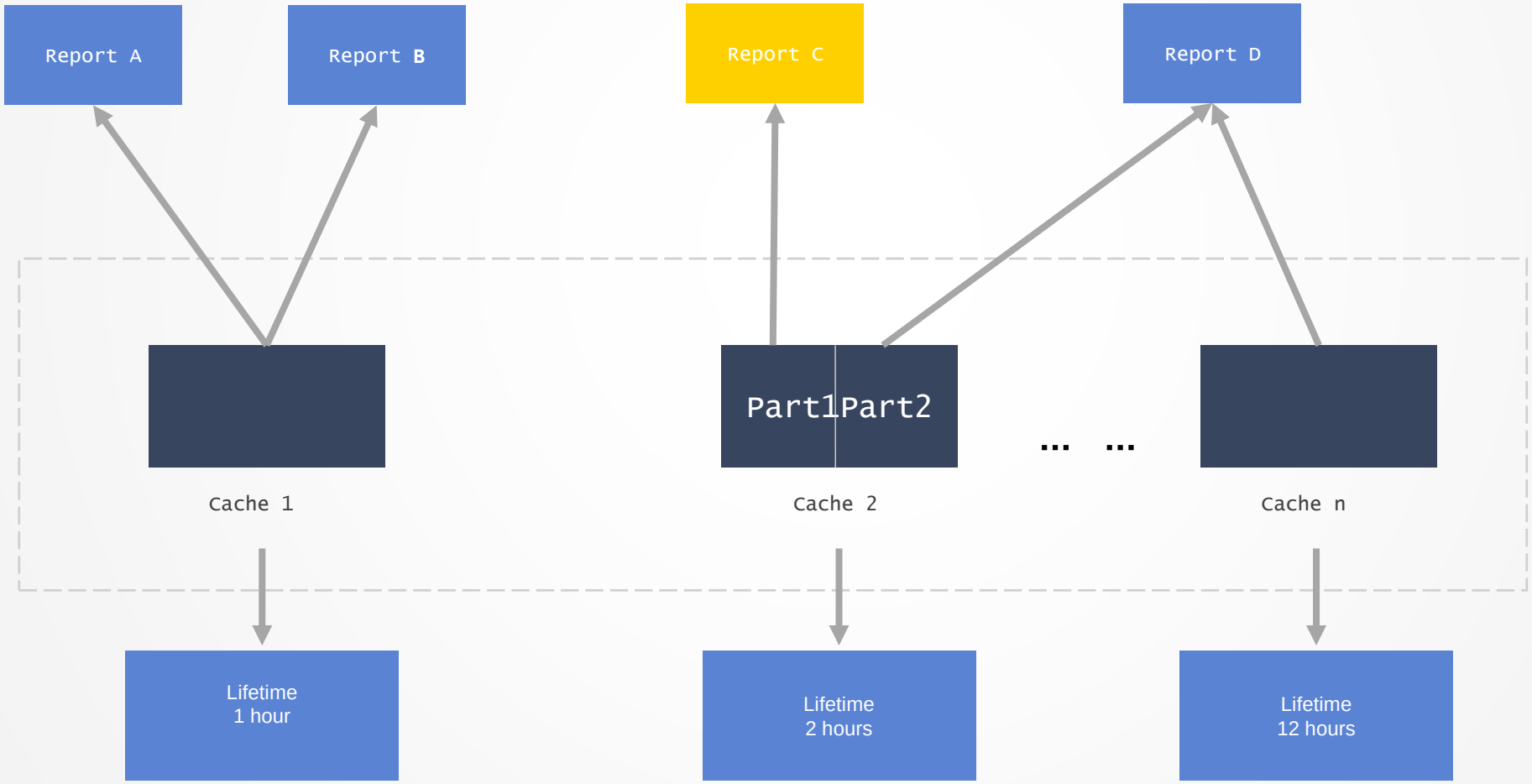
| | A | B |
|---|---|---|
| 1 | =connect("database") | |
| 2 | =A1.query("select syb.org_abbreviation as syb …") | /First half of left join SQL |
| 3 | =A1.query("select l.requisition_id, l.table_type …") | /View part of left join |
| 4 | =A2.switch(REQ_ID,A3:REQUISITION_ID) | |
| 5 | =A4.new(#:SN,SYB, XMB, SUB_ID, ORG_ABB, …) | |
| 6 | //Omit n rows and N columns here, calculate the summary value | |
| 7 | >A5.insert(0,“Summary","", "", "", …) | |
| 8 | return A5 | |

Split the original report dataset SQL. The two sub query SQL are written to the embedded script for execution, and the join is completed in the embedded script.

| | Dataset calculation | Report calculation presentation | Total time |
|---|---|---|---|
| Before optimization | 317 seconds | 85 seconds | 402 seconds |
| After optimization | 52 seconds | 5 seconds | 57 seconds |

# Controllable cache - structure chart

esProc can achieve partial caching of reports, cache reuse among multiple reports, and different lifetimes of different caches.



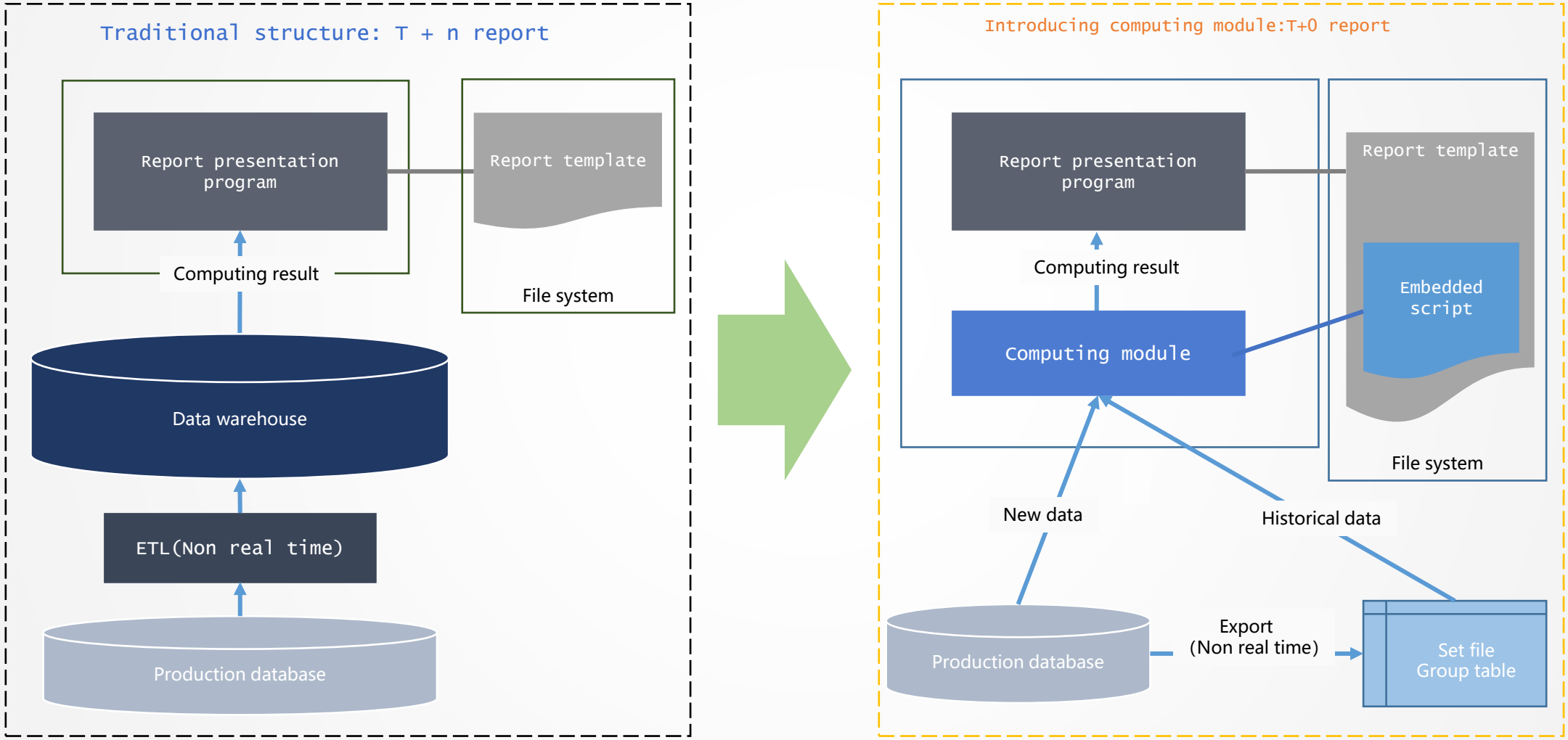
# Controllable cache

esProc can achieve partial caching of reports, cache reuse among multiple reports, and different lifetimes of different caches.

Example: If the query parameters are the same within one hour after the cache file is generated, the data in the cache file will be read directly, otherwise, the data will be obtained from the database and the cache file will be generated.

| | A | B | C |
|---|---|---|---|
| 1 | E:\\work\\esProc\\ | /Cache directory | |
| 2 | ="sales_"+string(d_date) | /Cache file name, report name + parameter (dfxName_ParamName) | |
| 3 | =file(A1+A2) | | |
| 4 | if A3.exists() && interval@s(A3.date(),now())<3600 | return A3.import@b() | end |
| 5 | =connect("demo") | | |
| 6 | =A5.cursor("SELECT customer ID,sum(item price*quantity) Order amount FROM order,order detail WHERE Order.Order ID=Order detail.Order ID and year(Purchase date)>? group by Customer ID order by Order amount desc",d_date) | | |
| 7 | =A6.fetch(5) | >A5.close() | |
| 8 | =A3.export@b(A7) | /Generate cache file | |
| 9 | return A7 | /Return result set for report | |

# T + 0 query and Statistics - structure chart



# T + 0 query and Statistics

Implement low cost T + 0 real-time report using mixed data source capability.

Example: Qquery "sales statistics of each state", and it is required to query real-time (T + 0) data.

| | A |
|---|---|
| 1 | =db2.cursor("select s.*,e.name sellername,e.state sellerstate from sales s,employee e where s.sellid=e.eid and days(current date)=days(orderdate)") |
| 2 | =file("sales.btx").cursor@b() |
| 3 | =[A1,A2].conjx() |
| 4 | =A3.groups(SELLERSTATE;count(ORDERID):count,sum(AMOUNT):TOTAL) |
| 5 | =A4.sort(TOTAL:-1) |
| 6 | return A5 |

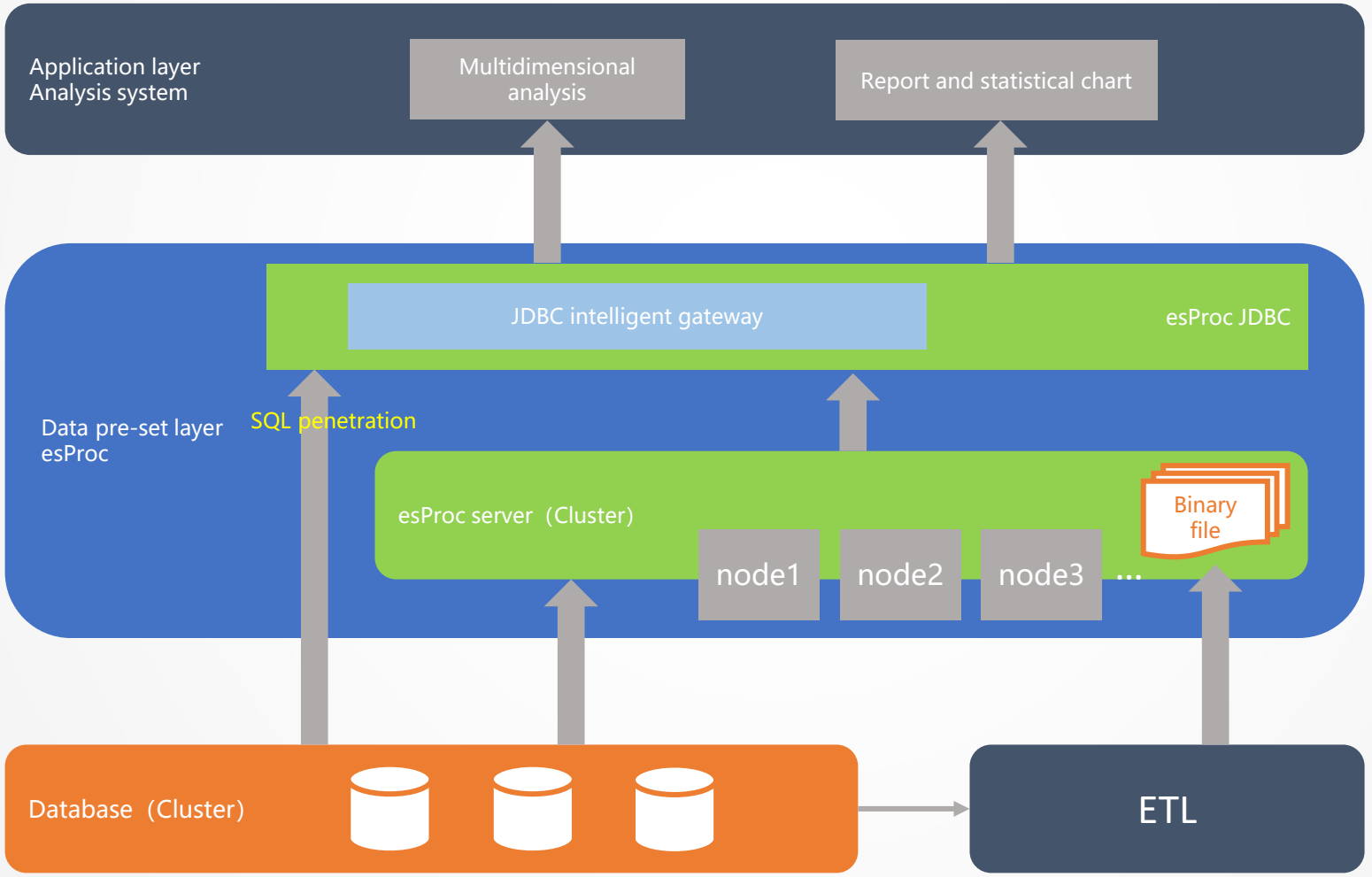
| state | quantity | amount |
|---|---|---|
| California | 34 | $80,492.50 |
| Texas | 35 | $72,957.90 |
| Florida | 35 | $72,089.00 |
| New York | 22 | $34,680.00 |
| Pennsylvania | 16 | $16,346.10 |
| New Mexico | 5 | $7,624.60 |

The report presentation effect is as shown in the figure.

A1: Create a database cursor to read sales data and sales personnel data with simple SQL. From the days(current date)=days(order date) in the where condition, we can see that the sales data only reads the new data of the day.
A2: Create a cursor for the pre exported data file sales.btx. File cursor can read data from large data files in batches to avoid memory overflow. The @b option is to read the file in binary encoding.
A3: Merge database cursor (new data) and file cursor (historical data) vertically.
A4: Use the groups function to complete the grouping aggregation of the merged cursors.
A5: Sort in descending order of total sales.
A6: Return result set.

# Pre-set data and routing -- structure chart

Data pre-set layer reconstructs BI system into three-tier structure.



# Pre-set data and routing

After receiving the SQL, the esProc JDBC intelligent gateway will transfer it to the gateway.dfx program for processing. Gateway.dfx determines whether it is a query within three years. If so, change the table name to the file name, and check the local file orders.btx to return the result. If not, convert SQL to Oracle format and submit to database for processing.

| | A | B | C |
|---|---|---|---|
| 1 | =filename="orders.btx" | | |
| 2 | =sql.sqlparse@w().split(" ") | =A2.select@1(like(~, "ORDERDATE=date('????-??-?? ') ")) | |
| 3 | =mid(right(B2,14),3,10) | =year(now())-year(date(A3)) | |
| 4 | if B3<=3 | =connect() | =sql=replace(sql, "from ORDERS","from  "+filename) |
| 5 | | =B4.cursor@x(sql) | return B5 |
| 6 | else | =connect("orcl") | =sql=sql.sqltranslate("ORACLE") |
| 7 | | =B6.cursor@x(sql) | return B7 |

# Innovation makes progress!