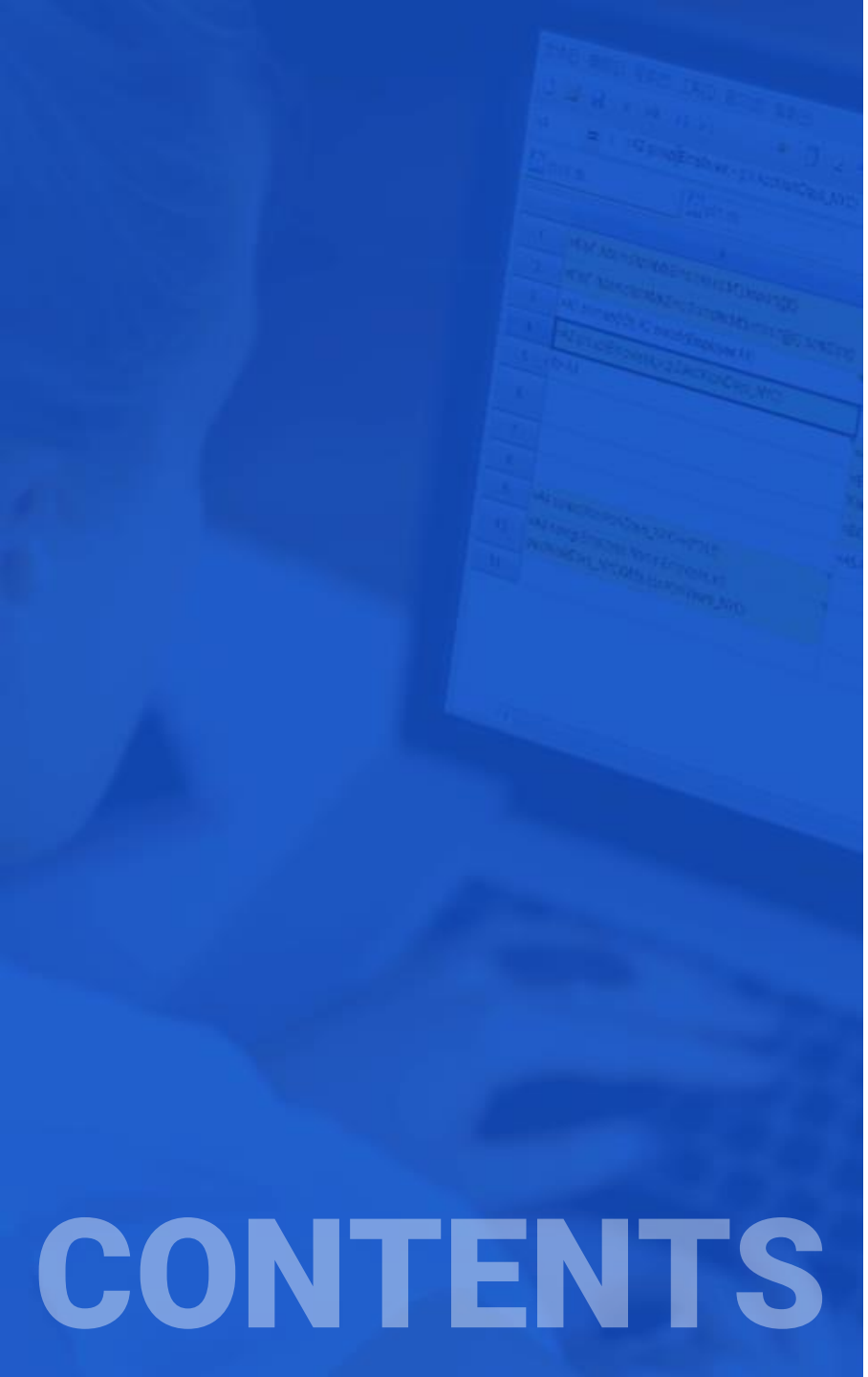
Data computing middleware

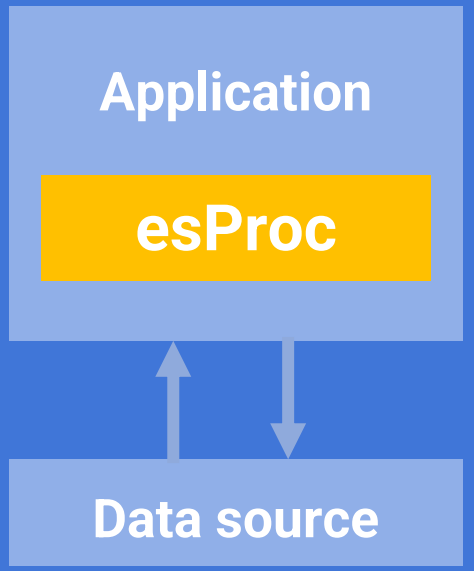


CONTENTS

01 What is esProc?

What is esProc?

- Data computing middleware(DCM), located between data sources and applications, provides general data computing and processing services
- Innovative SPL(Structured Process Language) syntax



- Simple syntax, agile development
- In-built computing ability, independent of database
- Seamlessly embedded into application
- Support diverse data sources

What does esProc solve?

Oriented for data computing scenarios like **business logic/microservice development, report data preparation**

- It is not a rare thing to have a N-layer nested SQL statement or a stored procedure of dozens of KBs, and the programmer himself could not understand it after three months
- Sometimes you have to write very lengthy Java code to carry out a computing goal, which is very exhausting; Each change of code will require the restart of machine, which is unbearable for the users
- Data comes from dozens of sources like DB/NoSQL/Text/Json/Web, and cross-source mixed computation is highly needed
- When the hot data and cold data are separated into different databases due to huge data amount, it is extremely hard to perform the real-time queries on the whole data.
- Too much relied on the stored procedures, the application framework is hard to adjust and expand
- There are too many (intermediate) tables in the database, almost exhausting the storage and computing resources, and you dare not to delete them
- There are constant report demands in an enterprise, and how can the cost of personnel be relieved?

Counterpart technologies of esProc

Database

- Heavy and closed
- Difficult to perform cross-database/no-database computations
- Loading data into database costs time, effort and resources
- Cannot be embedded for use
- SQL/stored procedure development and debugging are difficult

JAVA

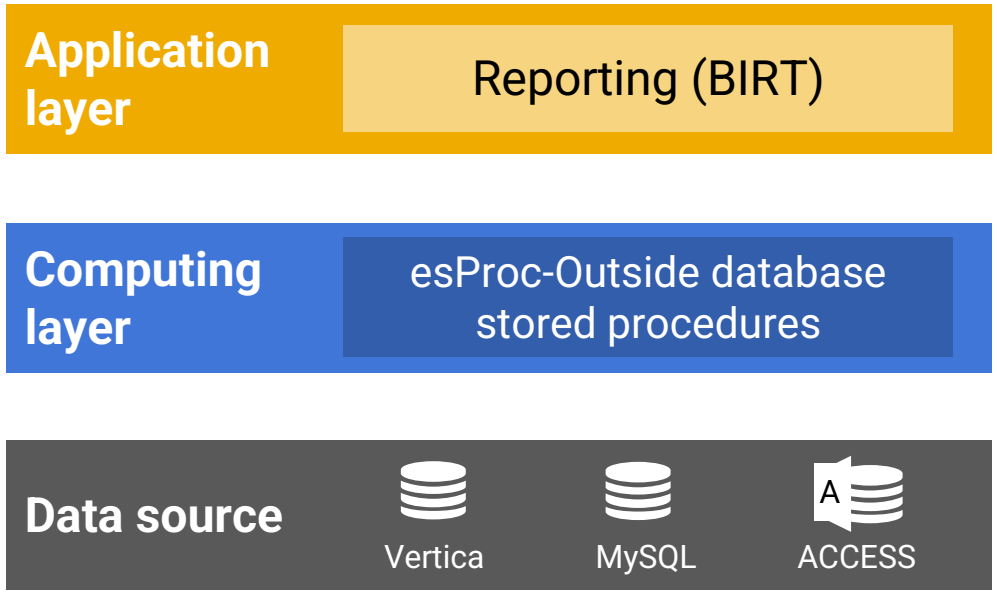
- Lack of necessary structured data computing class library
- It is hard to code for implementing computations
- Codes are very long, difficult to code and maintain
- Do not support hot swap

PYTHON

- Pandas is not a professional structured data computing package, and complex computing is cumbersome to implement
- Python has weak integration ability and is difficult to be seamlessly embedded into applications

An insurance company-Outside database stored procedure

Previously, when data comes from sources like Vertica/MySQL, an application should use a single source as much as possible; when multi-source mixed computation is indeed needed, it can only be done by Java hardcoding, which is very tedious.



User comments

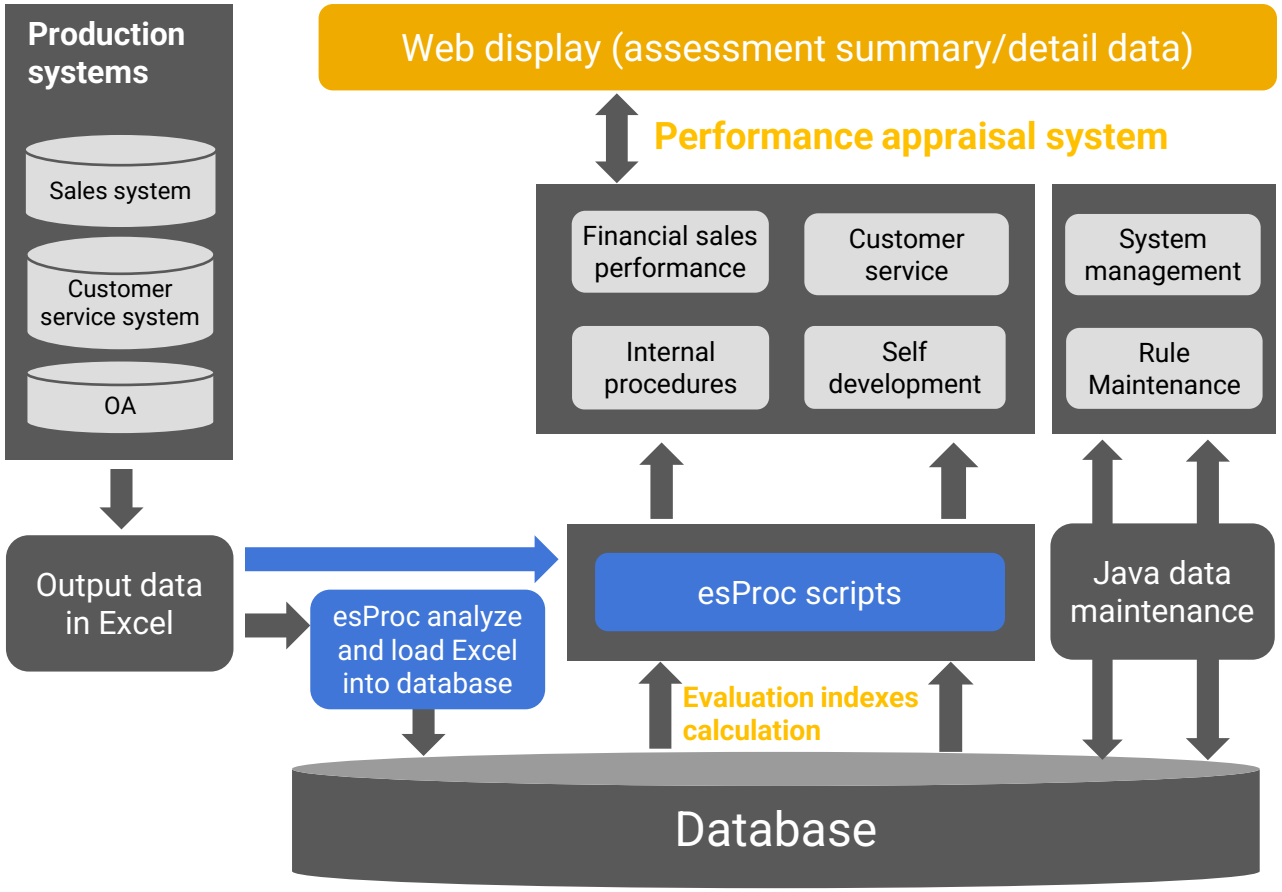
The best use for us is to pass parameters to the Vertica database.

Each cell becomes a data array that are easy to use, compare and manipulate. It is very logical and you have made it user friendly.

esProc helps Vertica to support stored procedures, which makes cross-source computations convenient.

A bank-computing middleware

Previously, Java and stored procedures were used to calculate assessment data, which resulted in long development cycle, low performance and difficult maintenance.



Applying effects

Reduce development costs

A total of 44 types Excel parsing, reduced from 30 manday to 6 manday

Improve development efficiency

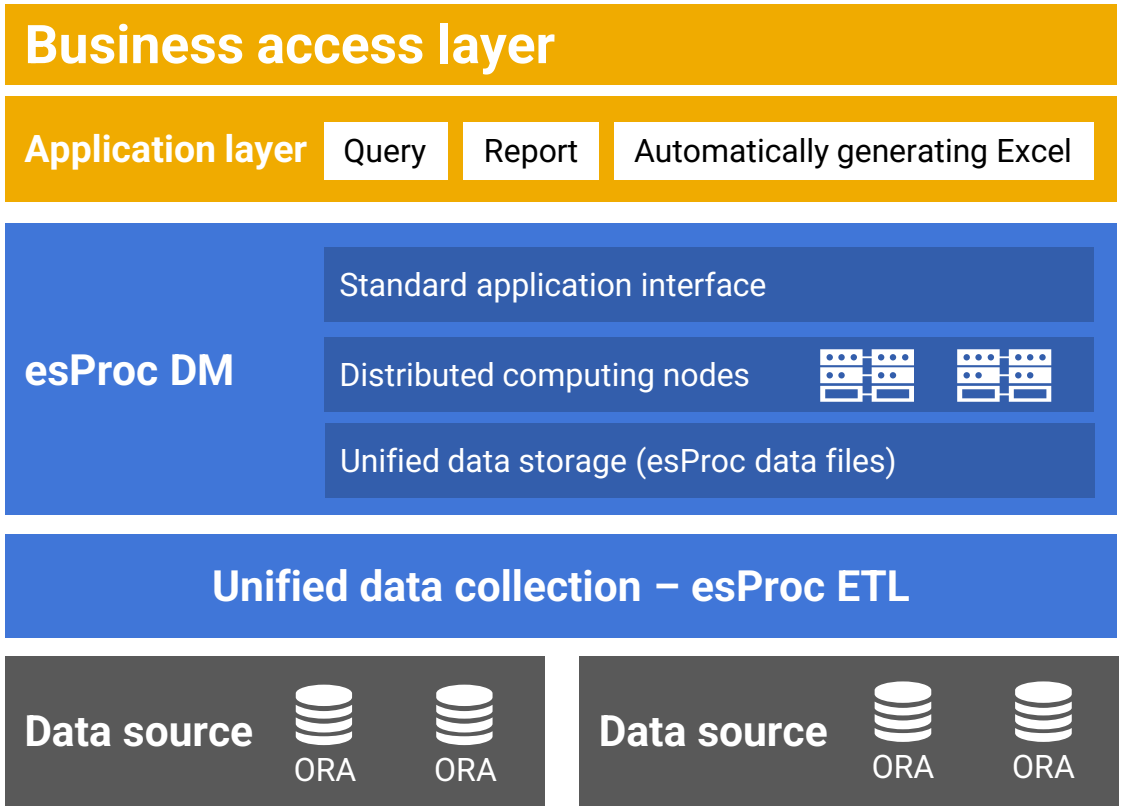
Each Excel parsing codes are reduced from 100 lines to 3 lines

Reduce maintenance costs

In addition to short codes and easy maintenance, the scripts are hot deployed, taking effect immediately upon modification

A large equity exchange-data mart

Before the transformation, the data formats and specifications of the new and old systems were different, which led to the failure of intercommunication between the systems.

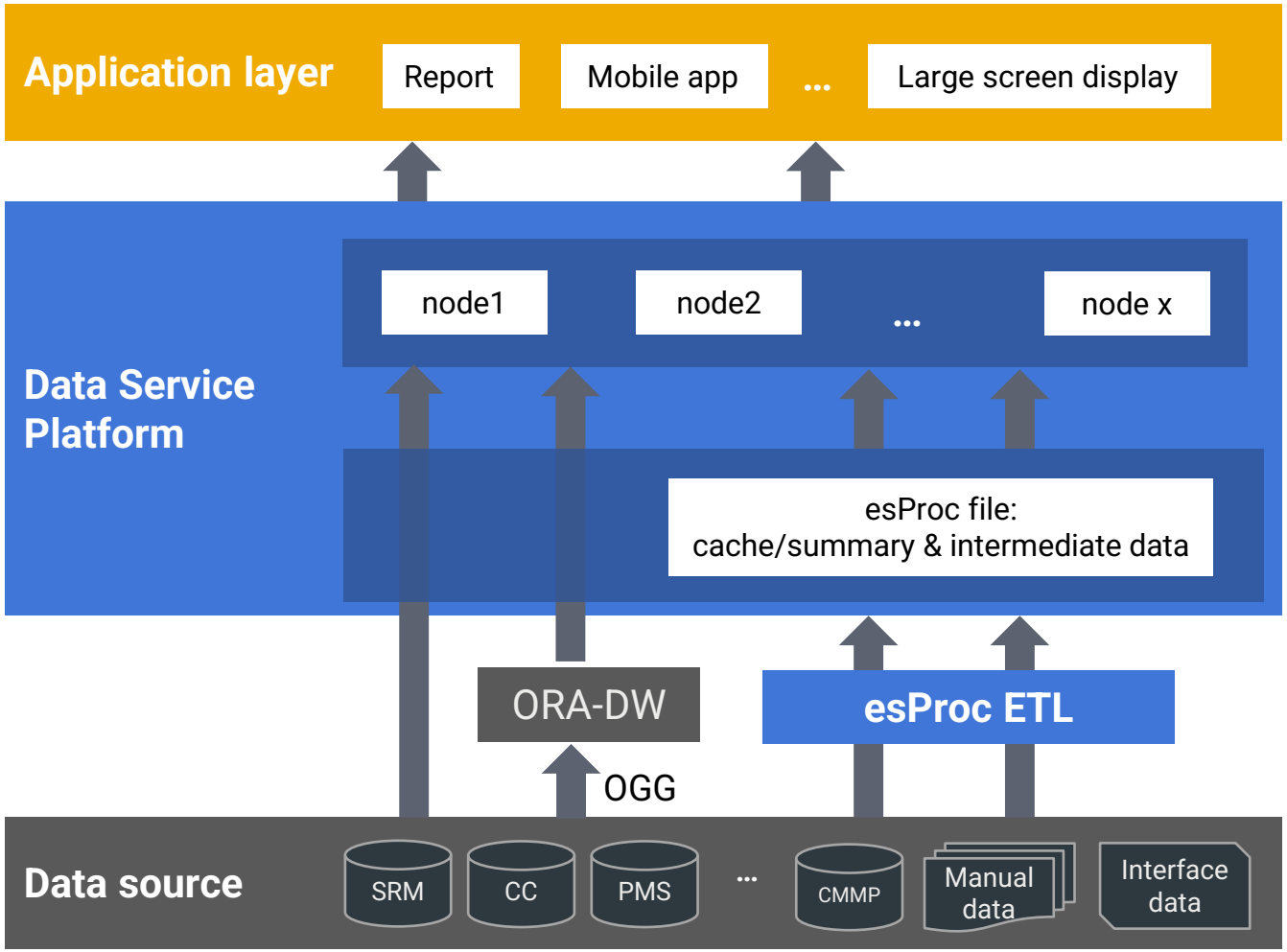


Applying effects

- ✓ **Unified processing of heterogeneous data sources**
  ETL on multi heterogeneous data sources simultaneously (multi-source mixed computation)
- ✓ **File system storage**
  File storage saves database costs and is more flexible
- ✓ **Computational logic reuse**
  Unified computation of one business logic to avoid inconsistency
- ✓ **Easy to scale out**
  Nodes are easy to scale out to improve computing capacity

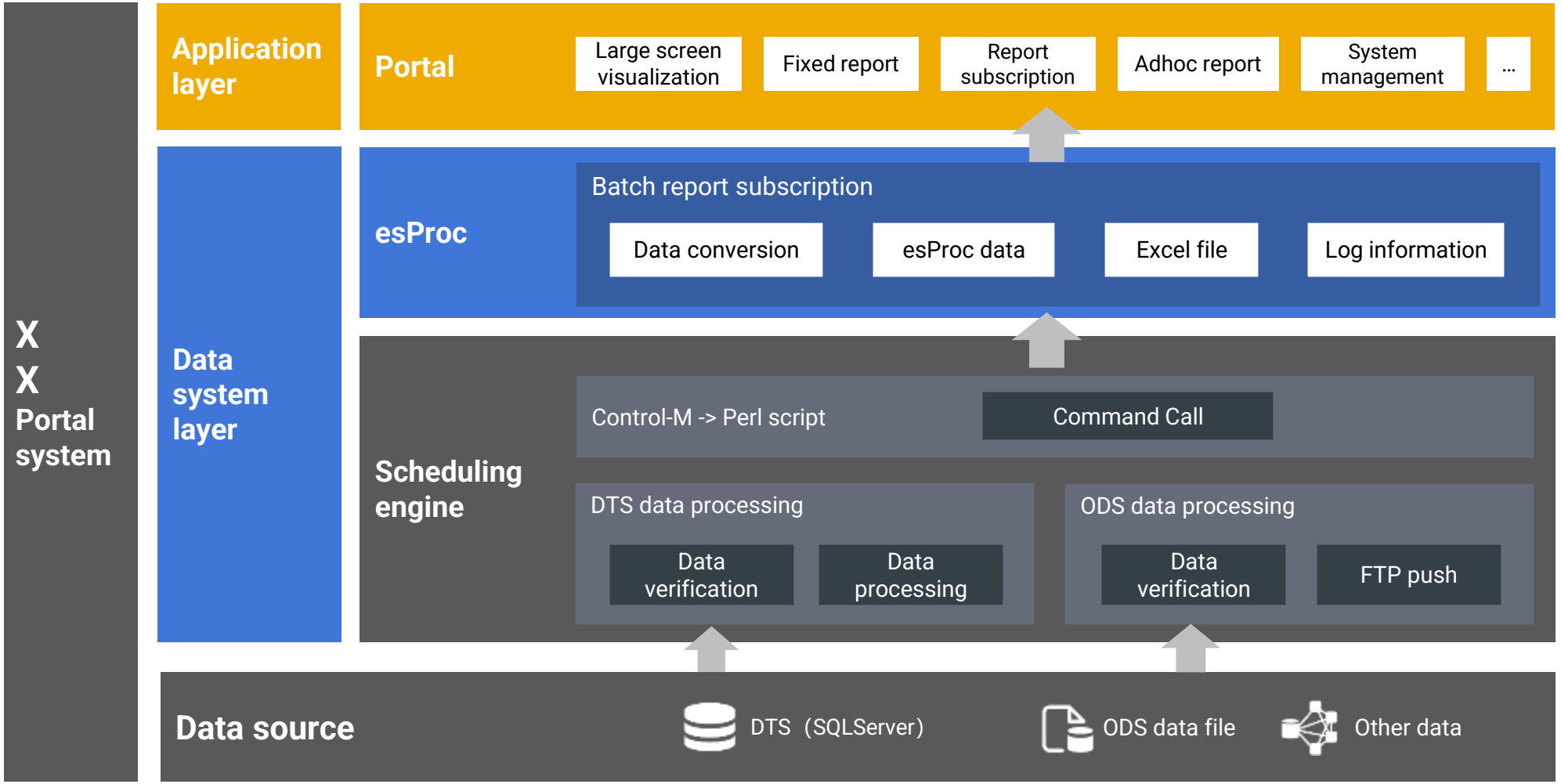
Realize data interchange between systems, unified management and computing reuse.

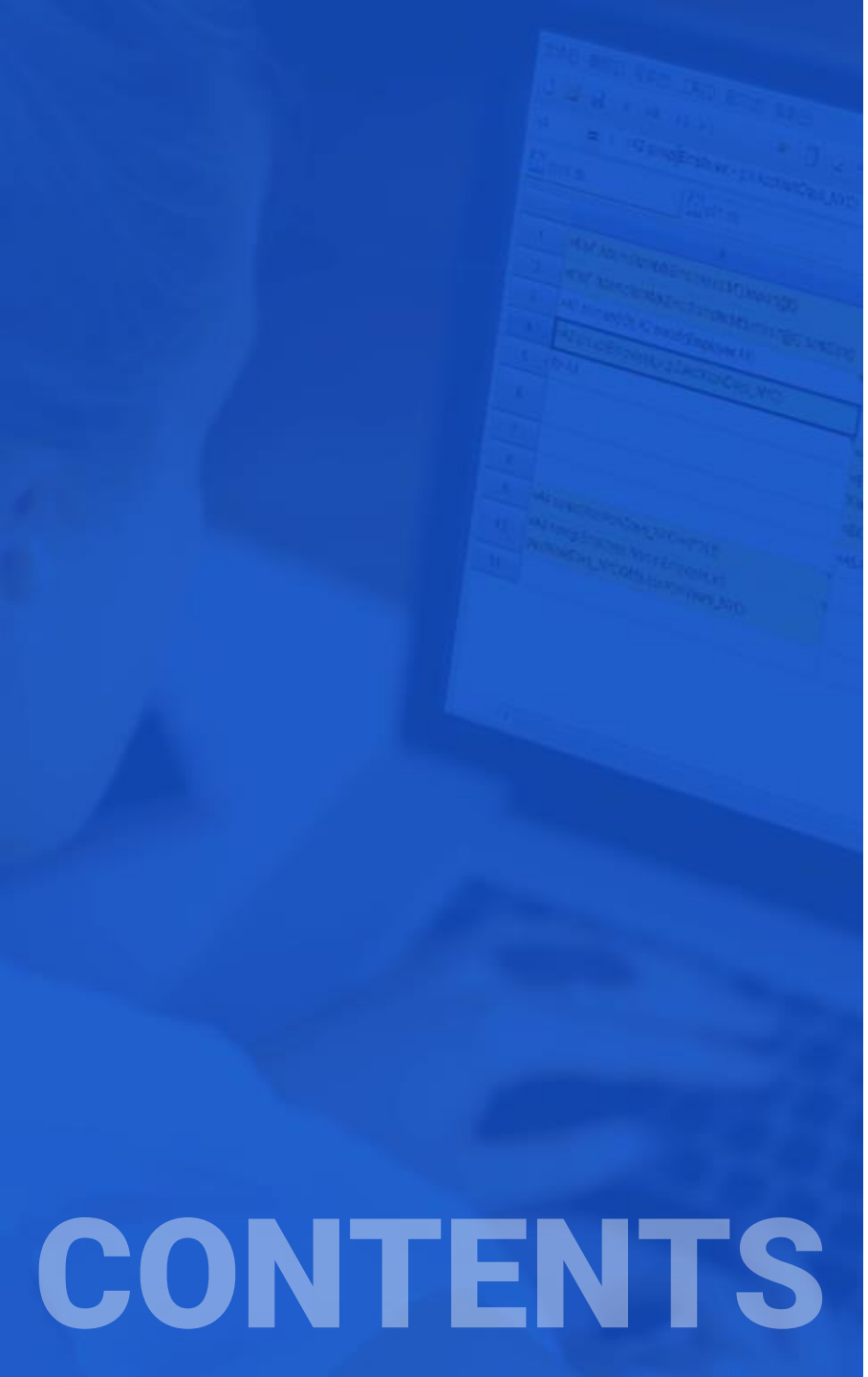
An airline-data service platform



- ✓ **Unified data management**
  Shielding underlying data differences and standardizing data management
- ✓ **Multi-source mixed computation**
  Direct mixed computation of data from multi sources, and data does not need to be in the same database
- ✓ **Historical data archiving**
  Historical data is stored in files, and computations are performed on files.
- ✓ **Improve computing performance**
  Data computing performance improved by 5-20 times
- ✓ **Easy to scale out**
  Nodes are easy to scale out to improve computing capacity

Financial industry-subscription system



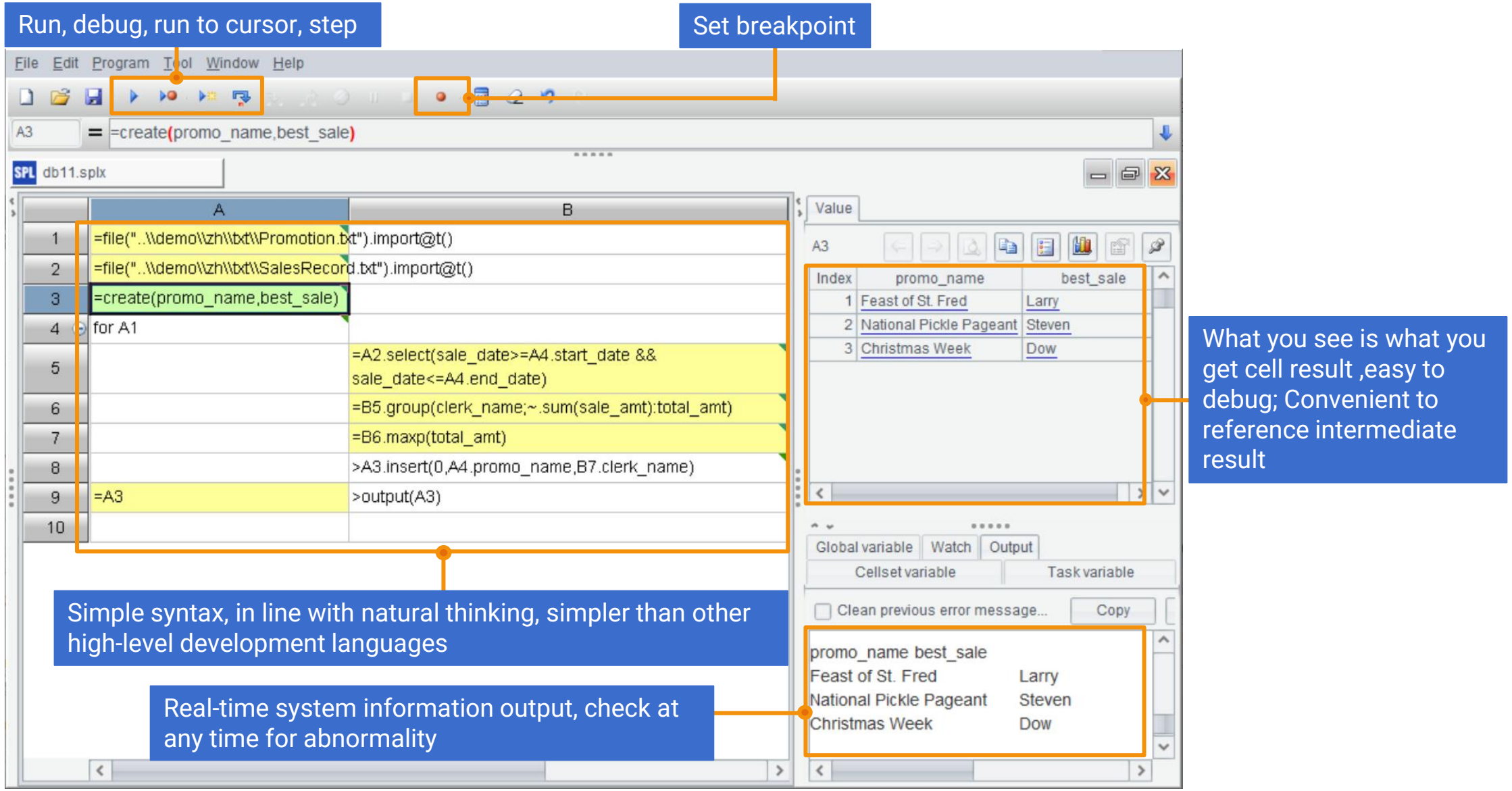


CONTENTS

02

Technical characteristics

Simple and easy-to-use development environment



Agile syntax

Calculation task: what is the max number of days has a stock been rising continuously?

```
select max(continuousDays)-1
from (select count(*) continuousDays
        from (select sum(changeSign) over(order by tradeDate) unRiseDays
                from (select tradeDate,
                        case when closePrice>lag(closePrice) over(order by tradeDate)
                        then 0 else 1 end changeSign
                    from stock) )
        group by unRiseDays)
```

SQL solution

- SQL has to nest in three layers with the help of window function;
- Can you understand it?

	A
1	=stock.sort(tradeDate)
2	=0
3	=A1.max(A2=if(closePrice>closePrice[-1],A2+1,0))

SPL solution

In fact, this calculation is very simple, according to the natural thinking: first, sort by the trading date (line 1), then compare the closing price of the day with that of the previous day, and +1 if it is higher, otherwise, reset, and finally calculate the maximum value (line 3).

Specially designed syntax system

SPL is especially suitable for complex process operations.

	A	B	C	D	E	F
1	=esProc.query("SELECT orderID as contract, orderDate as date, customer, amount, empID as salesman FROM sales where year(orderDate)=? OR year(orderDate)					
2	=esProc.query(select * from employeeInfo")					
3	>A1.run(salesman=A2.select@1(ID:A1.salesman))		/field value is record			
4	>A1.group(salesman)					
5	=create(salesman, thisyearAmount, lastyearAmount, growthRate, custNumber, bigCustNumber,bigCustProportion)					
6	for A4	=A6(1).salesman.name				
7		=A6.select(year(date)==year).sum(amount)				
8		=A6.select(year(date)==year-1).sum(amount)				
9		=B8/B7-1	/growth rate			
10		=A6.group(customer).(~.sum(amount))				
11		=B10.count()	/number of customer			
12		=B10.count(~>=10000)	/number of big customer			
13		=B12/B11				
14		=A5.insert(0,B6,B7,B8,B9,B11,B12,B13)				

Natural & clean step-by-step computation, direct reference of cell name **without specifically defining a variable**

Rich class libraries

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		=B7.count()	
9		=B7.count(~>=10000)	
10			

Grouping & Loop

	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			
9			
10			

Sorting & Filtering

	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	=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			

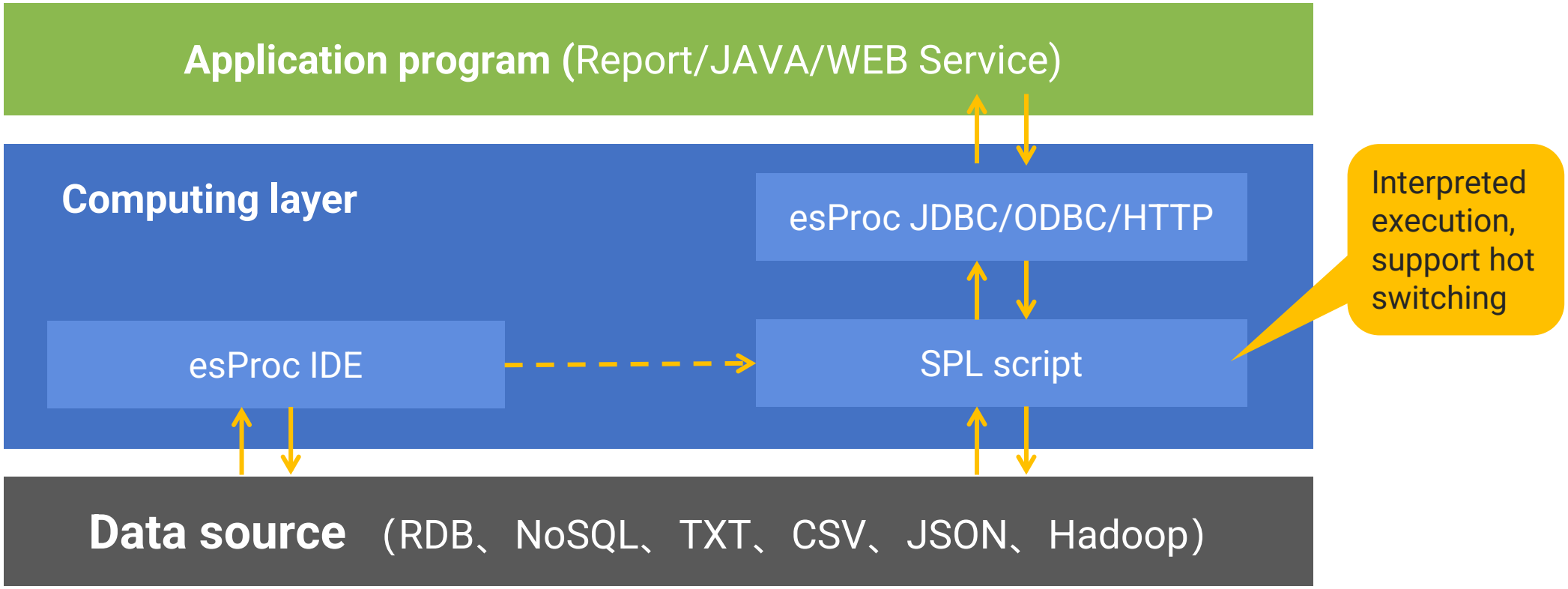
Ordered sets

Diversified data sources

- Multiple data sources are directly used for mixed calculation, and there is no need to unify the data (ETL) before calculation.
- Support SQL query files, NoSQL and other data sources.

Integration

esProc is developed in JAVA, provides standard application interfaces and can be seamlessly integrated into applications.



DCM evaluation criteria

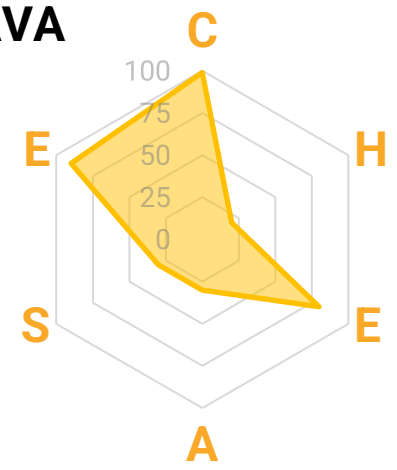
CHEASE

C	H	E	A	S	E
Compatible	Hot-deploy	Efficient	Agile	Scalable	Embeddable

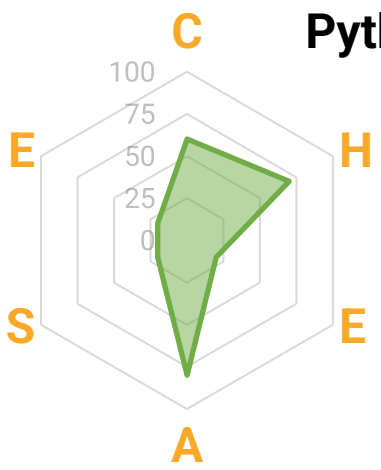
Extended Reading: http://c.raqsoft.com/article/1654672374108

esProc SPL evaluation result

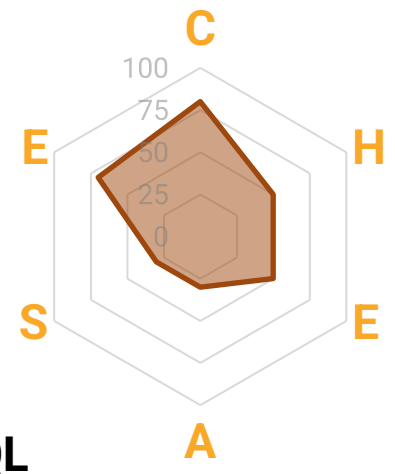
JAVA



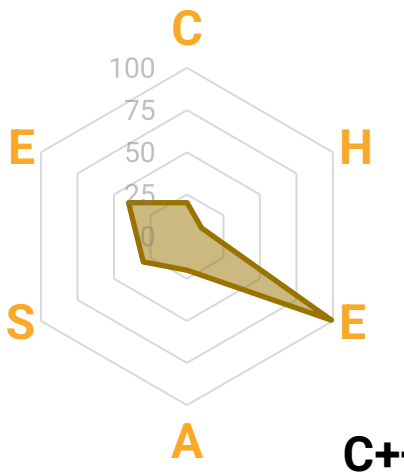
Python



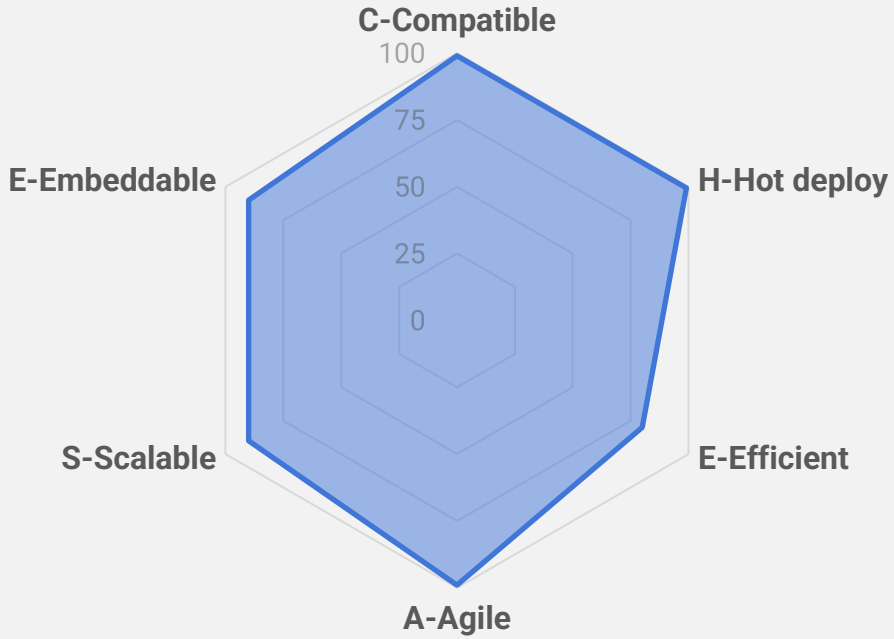
SQL

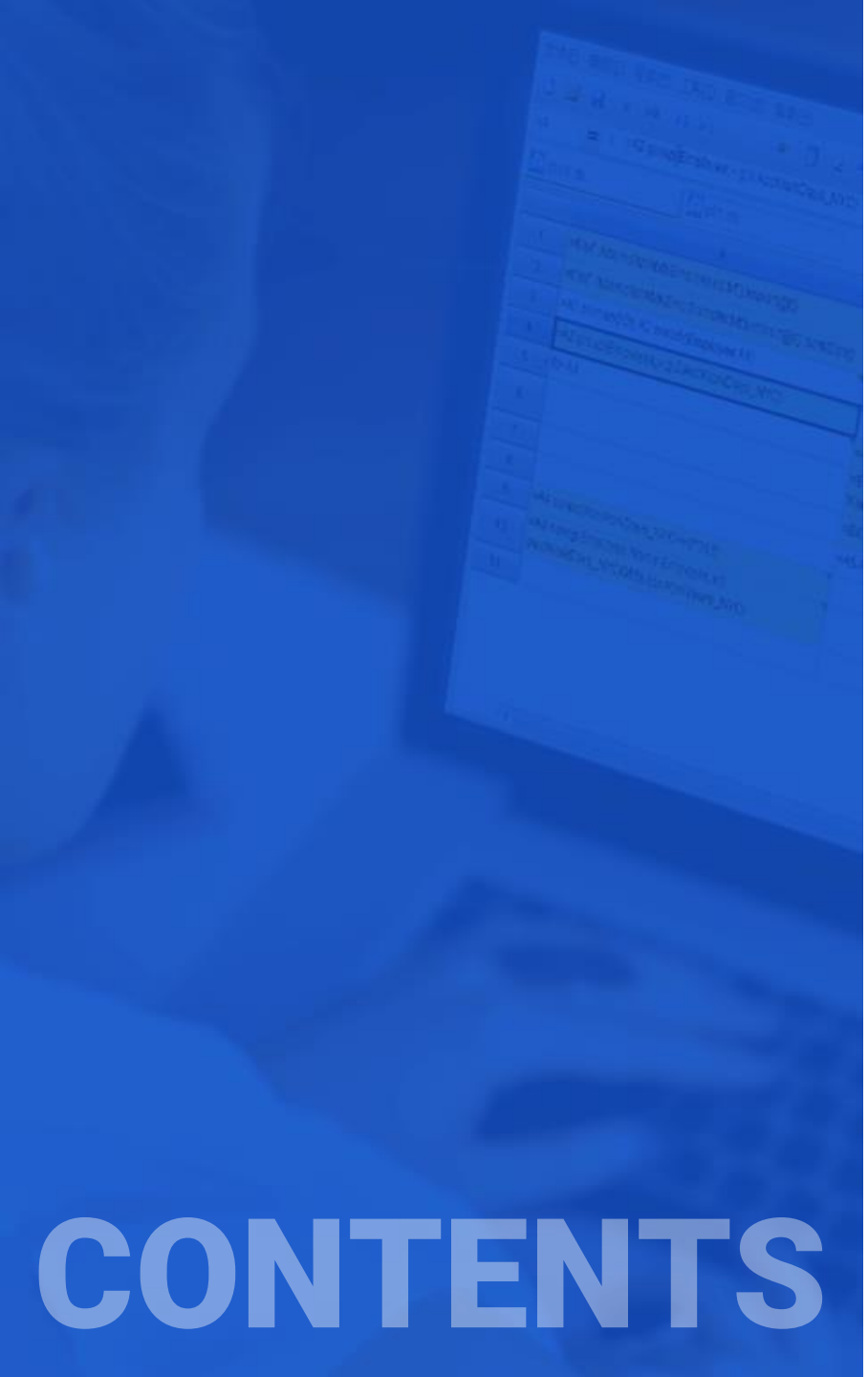


C++



SPL





CONTENTS

03

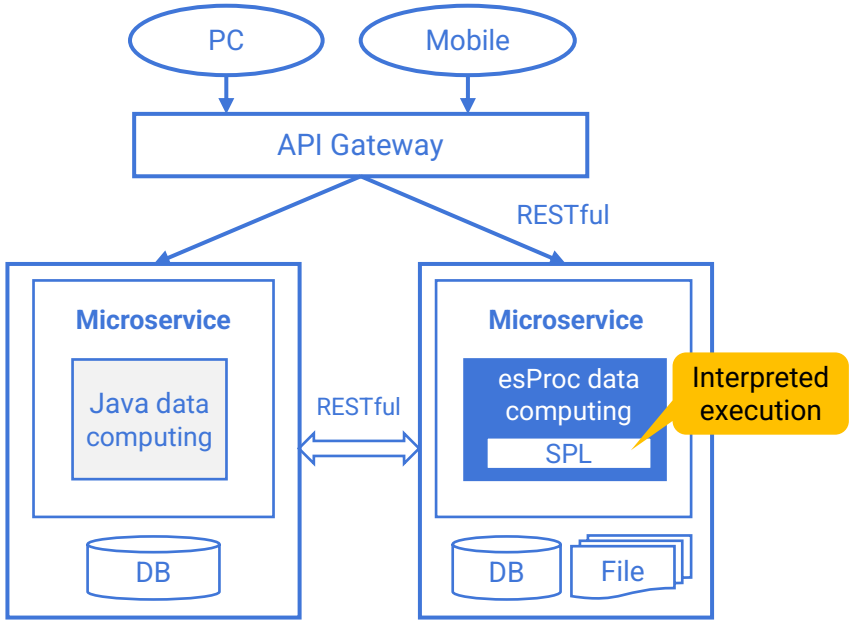
Pain points solution

Replace JAVA/ORM to implement business logic

Current situation

- Mainstream frameworks such as microservices require data processing at the application side.
- The database is difficult to be embedded in front-end applications, thus hardcoding is the only choice.
- JAVA lacks sufficient structured computing class library, which makes it difficult to develop data processing, and hot swap can not be achieved.

Solution



- ✓ SPL replaces JAVA/ORM to implement data computing in (microservice) applications.
- ✓ Rich class library and agile syntax simplify the development.
- ✓ The system is open and can process data of any source in real time.
- ✓ SPL is interpreted executed, naturally supporting hot swap.
- ✓ Efficient algorithms and parallel mechanism ensure computing performance.

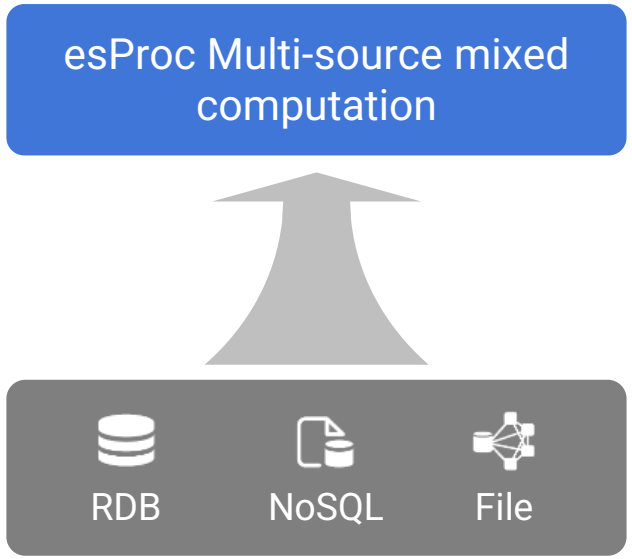
Extended reading：http://c.raqsoft.com/article/1647501354587

Diversified data sources and mixed computing

Current situation

- There are more and more types of enterprise data sources, and cross source hybrid queries are often required.
- The mixed computing ability of databases across data sources is weak (usually only supports same types), and the performance is low.
- Hardcoding at the application side is tedious to implement and difficult on subsequent computations.

Solution



- ✓ SPL supports mixed computing of multiple data sources.
- ✓ Support SPL and SQL syntax
- ✓ Data does not need to be in the same database physically; esProc can handle data in real time.
- ✓ Real-time mixed computation of data from same-type/heterogeneous-type sources
- ✓ Make effective use of the advantages of various data sources
- ✓ Embedded in applications to provide multi-source mixed computing capability for applications

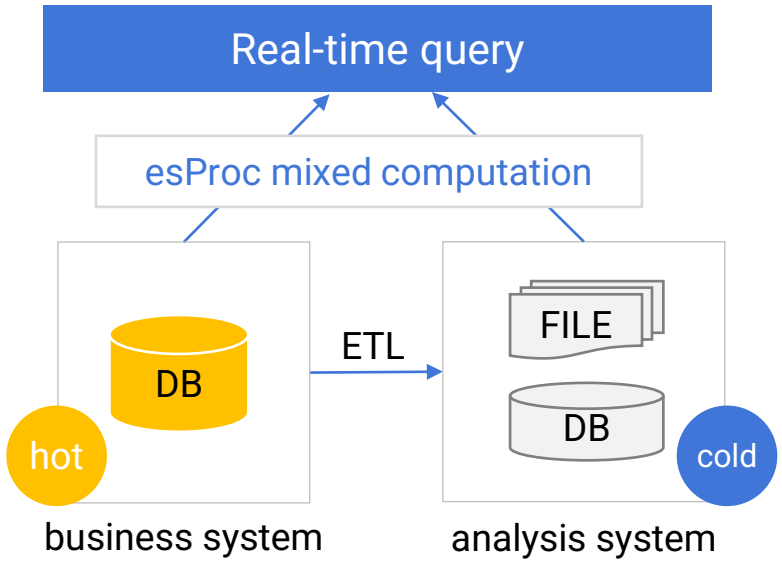
Extended reading: http://c.raqsoft.com/article/1647502473041

Implement real-time query

Current situation

- Using the production database for transactions and queries will affect the business as the amount of data increases.
- After the separation of cold and hot data (data in separated databases), only delayed data can be queried, and it is difficult to query the whole real-time data.
- Real-time query is needed but cannot be met, resulting in poor business experience.

Solution



- ✓ With SPL's multi-source mixed computation capability, data can be retrieved and calculated simultaneously from hot and cold data sources to implement real-time query.
- ✓ Support different types of data sources.
- ✓ Cold data can be stored in file systems, achieving higher computing performance and lower storage costs.
- ✓ Utilize SPL's file computing ability to perform the mixed computation of hot and cold data.

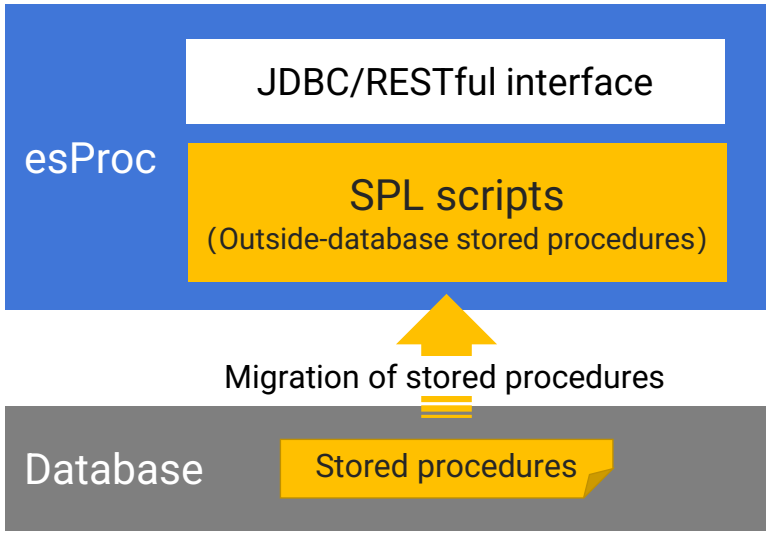
Extended reading: http://c.raqsoft.com/article/1651906058277

Replace stored procedures

Current situation

- Stored procedures are hard to edit and debug, and lack migratability.
- Compiling stored procedures requires high privilege, causing poor security.
- The shared use of a stored procedure by multiple applications will cause tight coupling between applications.

Solution



- ✓ SPL is intuitively suitable for complex multi-step data computation.
- ✓ SPL scripts are naturally migratable.
- ✓ The script only requires the read privilege of the database and will not cause database security problems.
- ✓ Scripts of different applications are stored in different directories, which will not cause coupling between applications.

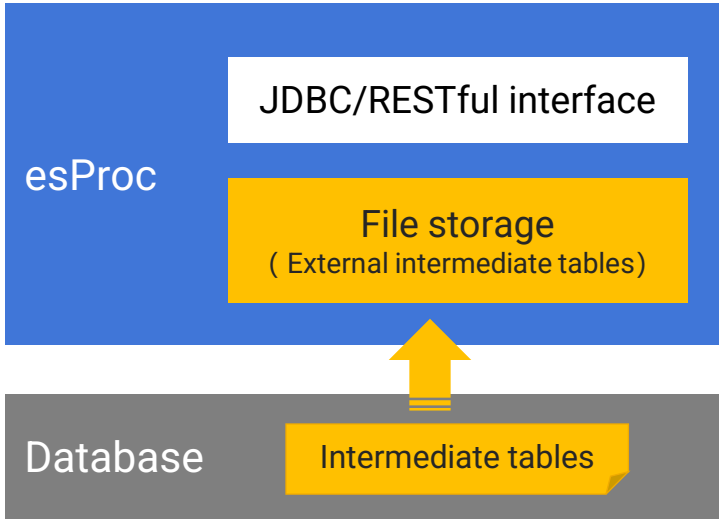
Extended reading: http://c.raqsoft.com/article/1646637335078

Eliminate intermediate tables from databases

Current situation

- For query efficiency or simplified development, a large number of intermediate tables are generated in the database.
- The intermediate tables take up large space, causing the database to be excessively redundant and bloated.
- The use of the same intermediate table by different applications will cause tight coupling, and it is difficult to manage the intermediate tables (hard to delete).

Solution



- ✓ The aim for storing intermediate tables in the database is to employ the database's computational ability for subsequent computations; SPL can implement the subsequent computations after using file storage.
- ✓ External intermediate tables (files) are easier to manage, and using different directories for storage will not cause coupling problems between applications.
- ✓ External intermediate tables can fully reduce the load on the database.

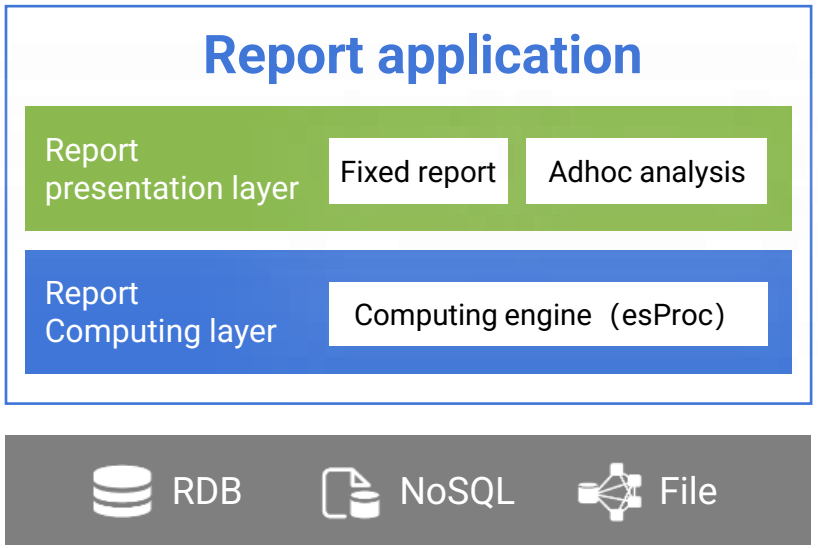
Extended reading: http://c.raqsoft.com/article/1652163420524

Handle endless report development needs

Current situation

- Reporting tools/BI tools can only solve the problems in the report presentation stage and can do nothing about data preparation.
- Data preparation implemented in SQL/stored procedure/JAVA hardcoding is difficult to develop and maintain, and the cost is high.
- The report development needs are objectively endless, and data preparation is the main factor leading to high development costs.

Solution



- ✓ Add a computing layer between report presentation and data source to solve the data preparation problems.
- ✓ SPL simplifies the data preparation of reports, makes up for the lack of computing ability of reporting tools, and comprehensively improves the efficiency of report development.
- ✓ Both report presentation and data preparation can quickly respond to handle endless report development needs at low cost.

Extended reading：http://c.raqsoft.com/article/1642764157224


THANKS