



esProc

Innovated big data
computing engine

Middleware for report source data computing

Issued by Raqsoft

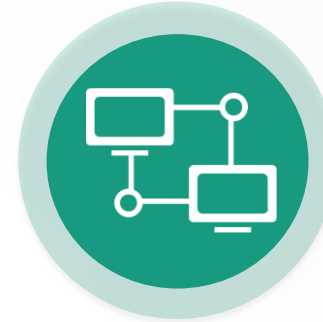


Current Situation Analysis



Report business is instable

In the process of business development, the new reports are always generated and the old ones need to be changed, resulting in endless development of the reports.



Difficulty in data preparation and development

The SQL for preparing data for reports is several K bytes long, which is difficult to write and more difficult to maintain.



Query is slow for Large Report

For large amount of data, report presentation is too slow, sometimes overflow happens. Users often complain.



High coupling with application

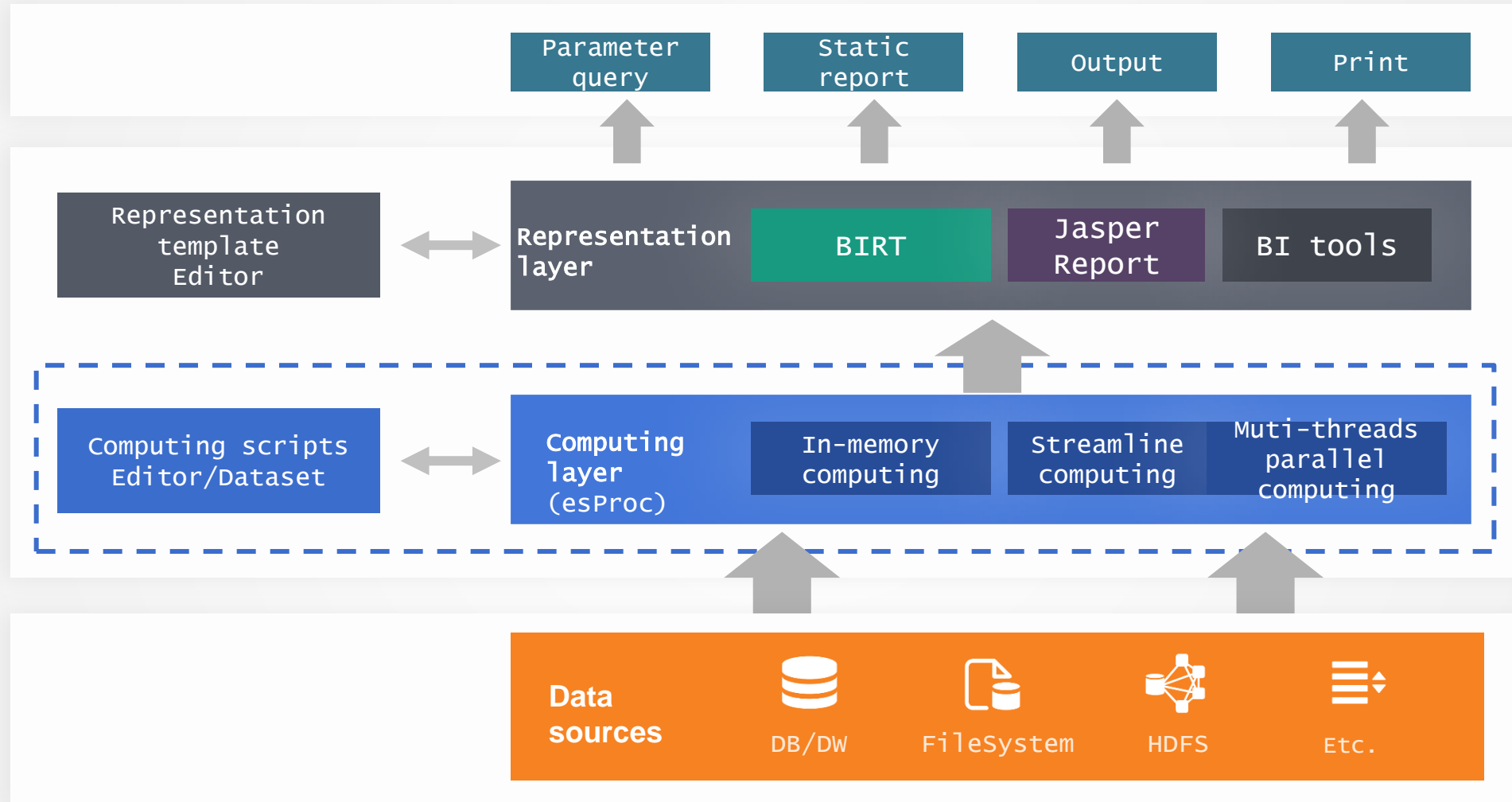
Applications are highly coupled with reports, and revising reports often requires restarting applications.

Difficulties in Report Development have transferred

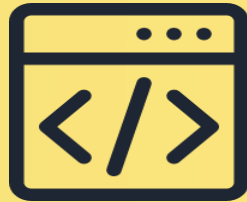


- ✓ Mature reporting tools have been able to solve the presentation problem.
- ✓ More difficulties in report development are on data sources.
- ✓ Most performance problems are also caused by or need to be addressed by data sources.
- ✓ A good data source processing mechanism can also optimize the application structure.

Introducing Data Computing Layer - esProc



Core advantages



Reducing
Development
Difficulty



Improving
operational
performance



Optimizing
application
structure



esProc uses set-oriented syntax, and the code is much shorter than JAVA that does not directly support structured data computing.

Programming faster and shorter

- esProc provides higher level class libraries and methods based on Java

Easy to understand and troubleshoot

- The proportion of pseudo-real code is about 1:1.5, and most report data preparation algorithms can be displayed on a single screen.
- You can see more code in a page to understand the meaning of the code more completely, and the troubleshooting is easier.



Advantages over SQL/stored procedure



Count the longest consecutively rising trading days for a stock

```

1 select max(continuousDays)-1
2 from (select count(*) continuousDays
3       from (select sum(changeSign) over(order by tradeDate) unRiseDays
4             from (select tradeDate,
5                   case when closePrice>lag(closePrice) over(order by tradeDate)
6                       then 0 else 1 end changeSign
7                   from stock) )
8       group by unRiseDays)

```

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

SPL

Syntax suitable for describing a natural way of thinking!

SQL



Can you do it in a more natural way of thinking?



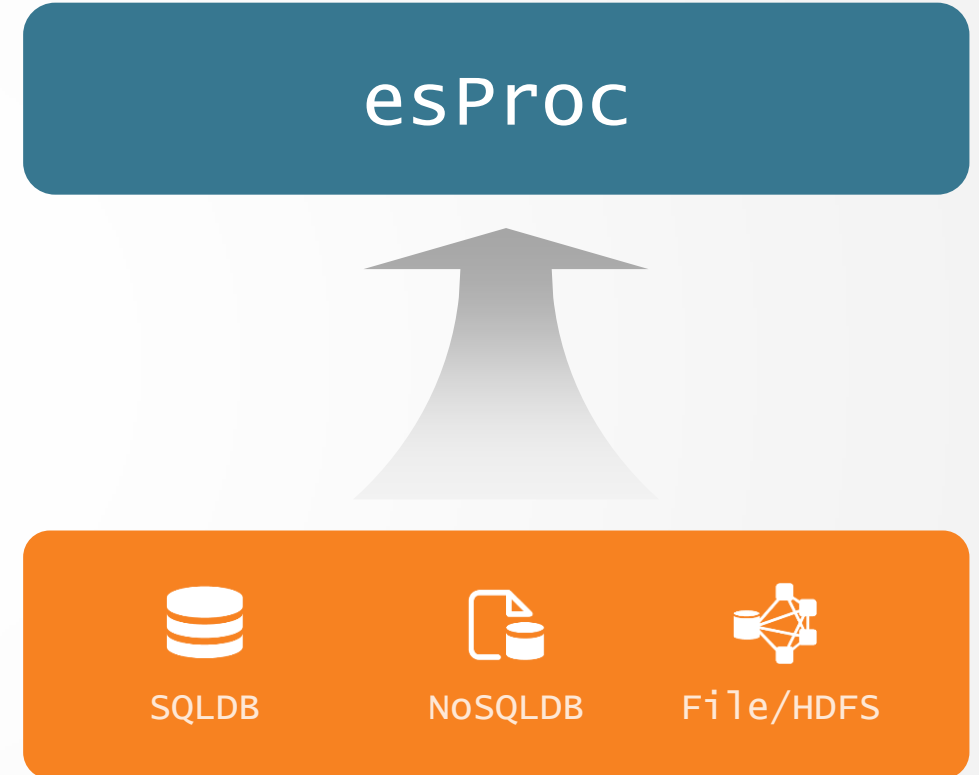
Support multiple data sources

The computing power and capacity of reporting tools are not competent for diverse data sources.

Computing layer processes diverse data sources

No need to import data into database, reduce working steps.

Support multi-tier data format, reduce development workload.





Dynamic Data Source

Decide the database to connect according to the parameters `${pds}.query("select * from T where F=?",pF)`

Dynamic Data Set

Dynamic SQL needs program logic to splice

	A	
1	<code>=sums.array().("sum("+~+") as "+~).string()</code>	<code>/sum(a) as a, sum(b) as b</code>
2	<code>=db.query("select G,"+A1+" from T group by G")</code>	

Capacity control for result set

	A	B	
1	<code>=db.cursor("select * from T")</code>	<code>=A1.fetch(1000)</code>	
2	<code>if B1.fetch@0(1)</code>	<code>>B1.insert(0,"Continue")</code>	<code>/Insert markers when uncompleted</code>
3	<code>>A1.close()</code>	<code>return B1</code>	

Core advantages



Reducing
Development
Difficulty



Improving
operational
performance



Optimizing
application
structure

Data retrieval by using multithreaded parallel processing



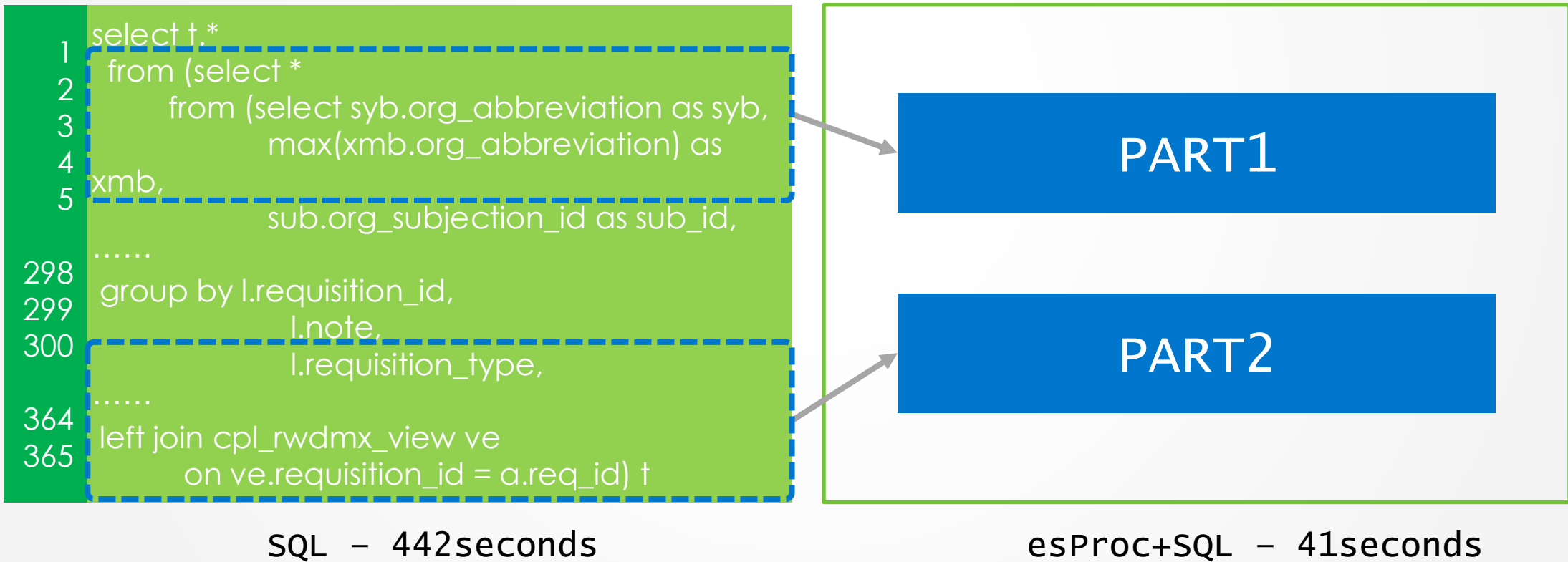
Databases generally have a bad JDBC performance, but the reporting performance relies heavily on the data retrieval performance. esProc can create multiple database connections to retrieve data in segments by using multithreaded parallel processing, and the performance can be increased by several times.

	A	B	C
1	fork 4	=connect(db)	/There are four threads, and each connection is established separately.
2		=B1.query@x("select * from T where part=?",A1)	/Retrieve each segment separately
3	=A1.conj()		/Merge the results



Control SQL execution path

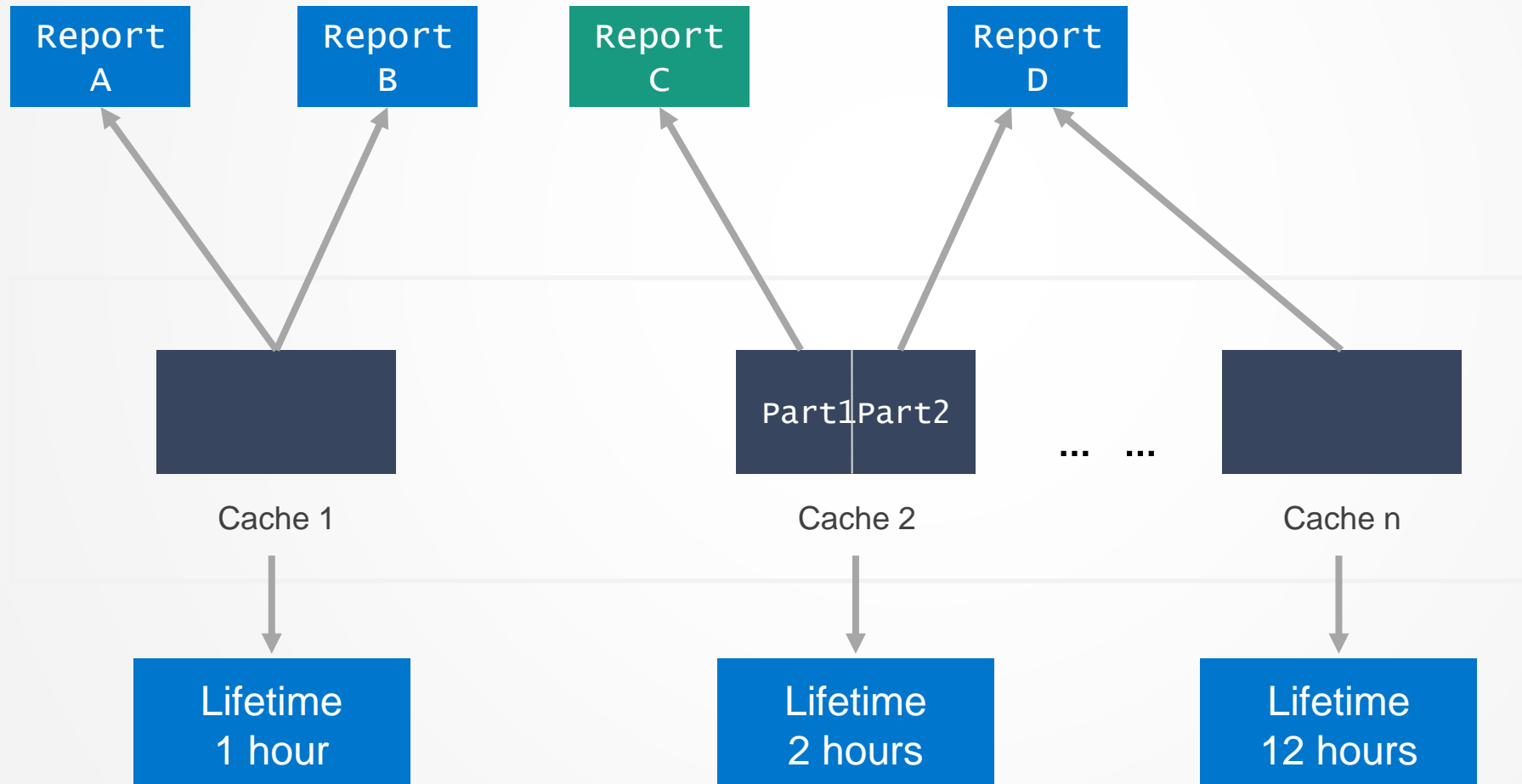
- While the transparency of database brings convenience to users, it is very difficult to optimize the execution path of SQL.
- esProc can freely control the execution path, and part of the operation can be moved out of the database to implement, so as to facilitate the performance optimization.



Controllable caching



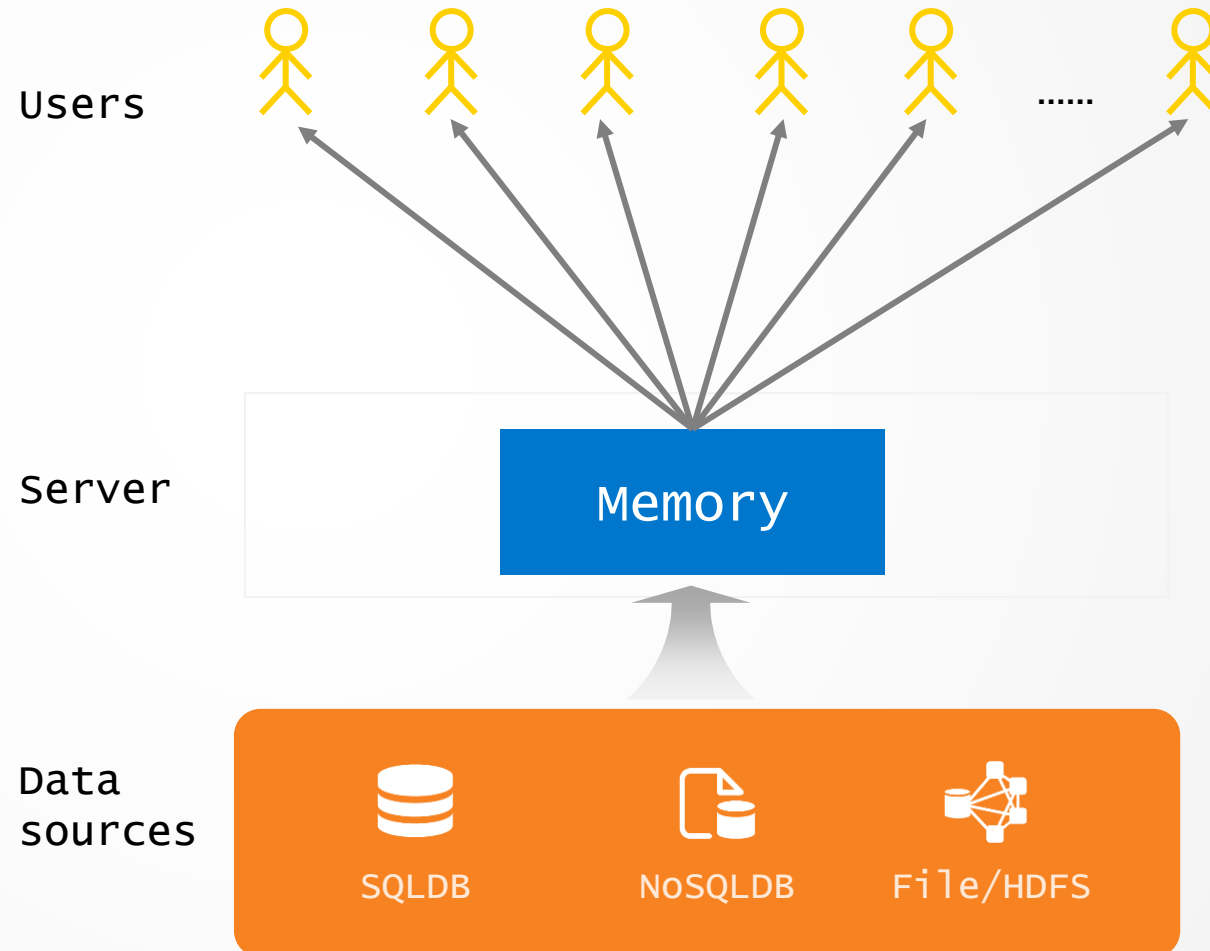
esProc can achieve partial caching of reports, cache reuse among multiple reports, and different lifetimes of different caches.



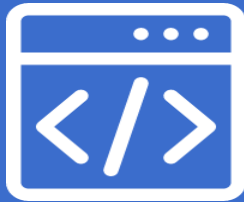


In-Memory Sharing

For highly concurrent reports, memory sharing mechanism can be used, which has higher performance and is more convenient for parallel computing.



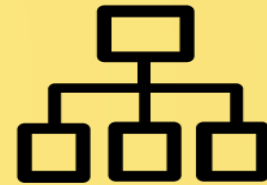
Core advantages



Reducing
Development
Difficulty



Improving
operational
performance



Optimizing
application
structure

Interpreted execution that creates decoupled architecture



To prepare data for report by using JAVA and esProc has the following differences :

JAVA

Difficulties in modularization

Java programs must be compiled and packaged with the main application, which is highly coupled.

Hot switching is almost impossible

The modified report data preparation algorithm written in Java will lead to the recompilation and deployment of the whole application, and it is difficult to achieve hot switching.

Few class libraries

There are few class libraries in structured and semi-structured data computing in JAVA.

esProc

Simple modularization

The esProc script file can be managed and maintained together with the report template, so that the report function can be modularized.

Hot switching is easy

esProc is the interpretive execution language , and it is easy to do hot switching.

Rich class libraries

There are rich syntax and class libraries, making the computing of structured data more efficient.

File-based algorithm storage: Lessening dependence on stored procedures



Using stored procedure to implement data preparation algorithm will cause the coupling between report and database.

Stored procedure and report are stored in different locations, which makes it very difficult to be consistent.

Stored procedure modification needs to allocate corresponding database privileges, and there are security risks.


Stored procedures are easily used by other applications, resulting in coupling between multiple applications.

- esProc helps cut down the procedures in the database greatly. An algorithm will be stored and managed along with the report template in file system and become a part of the reporting module. This will reduce its coupling with the other parts of the application while won't add more coupling with other applications.

The storage of data in the file system reduces intermediate tables




Because of the amount of data or computational complexity, it is often necessary to create intermediate tables in the database. Intermediate tables can cause the following problems:



Disorder of database management

The cumulative storage of a large number of intermediate tables in linear databases causes confusion in database management.

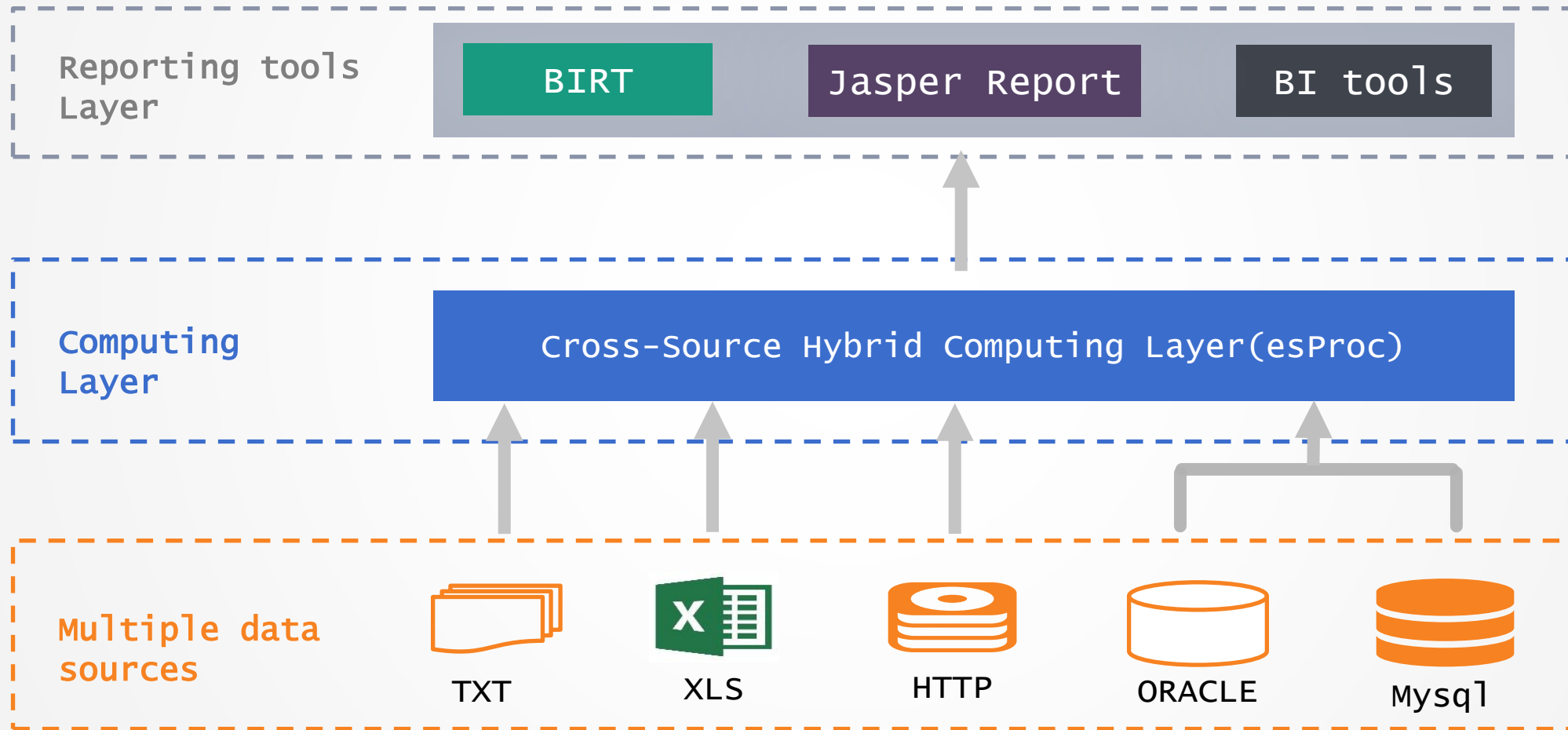


Waste of database resources

Unused intermediate tables still require corresponding ETL processes to update data and waste database resources.

The data of intermediate tables are stored in the file system, which is easy to manage, and the computing power can be obtained through esProc. It can obtain more efficient IO performance and computing power, reduce the database intermediate tables and sort out the database structure.

Direct use of multiple data sources and cross-database computing





Using esProc to implement data computing layer for report — Summary



High performance

Support higher computing performance by using discrete dataset model



Openness

Rich and diverse data source interfaces, direct computing



Low coupling

Report development is independent and decouples with application



Easy integration

Easy to be integrated by providing standard jar packages



Hot switching

No need to restart application by interpretive execution



Easy development

No need for environment configuration and application layer code reference

* Extended reading : <http://c.ragsoft.com.cn/article/1565142431668>



esProc as BIRT data source: handling csv files

BIRT
representation

Showing page 1 of 1

dataset_date	measurement	managedelement	content	last_minute_cpu	last_5_minute_cpu	last_30_minute_cpu	last_hour_cpu
2015-05-26 21:45:00.0	cpqHoCpuUtilTable	nltcom04	20:7:5:5	20	7	5	5
2015-05-26 21:45:00.0	cpqHoCpuUtilTable	nltcom04	17:8:5:5	17	8	5	5
2015-05-26 21:45:00.0	cpqHoCpuUtilTable	nltcom01	20:4:7:7	20	4	7	7
2015-05-26 21:45:00.0	cpqHoCpuUtilTable	nltcom03	8:12:5:5	8	12	5	5
2015-05-26 21:45:00.0	cpqHoCpuUtilTable	nltcom02	11:5:5:6	11	5	5	6

esProc
calculation

	A
1	=file("D:\DummyData.csv").import@t(dataset_date,measurement,managedelement,content:string;","")
2	=file("D:\Mapping.csv").import@t(",")
3	=A2(1).content.split(":")
4	=A3.to(2).("cols("+string(#+1)+")"+"~).string()
5	=A1.derive((cols=string(content).split(":"))(1):\${A3(1)},\${A4})

Data
sources

```

1 dataset_date,measurement,managedelement,content
2 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom04,20:7:5:5
3 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom04,17:8:5:5
4 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom01,20:4:7:7
5 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom03,8:12:5:5
6 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom02,11:5:5:6

```

DummyData.csv

```

1 managedelement,content
2 nltcom,last_minute_cpu_utilization_kpi:last_5_minute_cpu_u
tilization_kpi:last_30_minute_cpu_utilization_kpi:last_hou
r_cpu_utilization_kpi

```

Mapping.csv

esProc as JasperReport data source: Cross-database associative filtering



Jasper
representation

OID	EID	NAME	DEPT	AMOUNT
10262	8	Megan	Marketing	583.199999392
10265	2	Ashley	Finance	1170.0
10268	8	Megan	Marketing	1098.0
10276	8	Megan	Marketing	400.0

esProc calculation
(Cross-database
associative filtering)

	A
1	=mysql.query("select OID, EID, AMOUNT from orders")
2	=pg.query("select EID,NAME,DEPT from employee")
3	=A1.switch(EID,A2:EID)
4	=A3.select(["Marketing","Finance"].pos(EID.DEPT))
5	=A4.new(OID,EID:EID,EID.NAME:NAME,EID.DEPT:DEPT,AMOUNT)

Data
sources

OID	EID	AMOUNT
10248	5	428.0
10249	6	1842.0
10250	4	1523.4999898076...
10251	3	624.94999976083...

MySQL table:orders(partial field)

EID	NAME	DEPT
1	Rebecca	R&D
2	Ashley	Finance
3	Rachel	Sales
4	Emily	HR

PG table:employee(partial field)

External JAVA programs call esProc scripts through JDBC



JDBC class stored procedure calls SPL script file

```
...
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.prepareStatement("call
CountName()");
// Execute stored procedures
st.execute();
//Get result set
ResultSet rs = st.getResultSet();
...
```

JDBC query files directly using SQL

```
...
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.createStatement();
//Query files using SQL,get result set
ResultSet rs = st.executeQuery("$select
name,count(*) from /home/user/duty.txt group by
name");
...
```

Innovation makes progress!

