



**esProc**

Innovative big data computing engine

# Reporting Process Optimization Engine

By Raqsoft

# CONTENTS

1

**Solution**

2

Competitive advantages

3

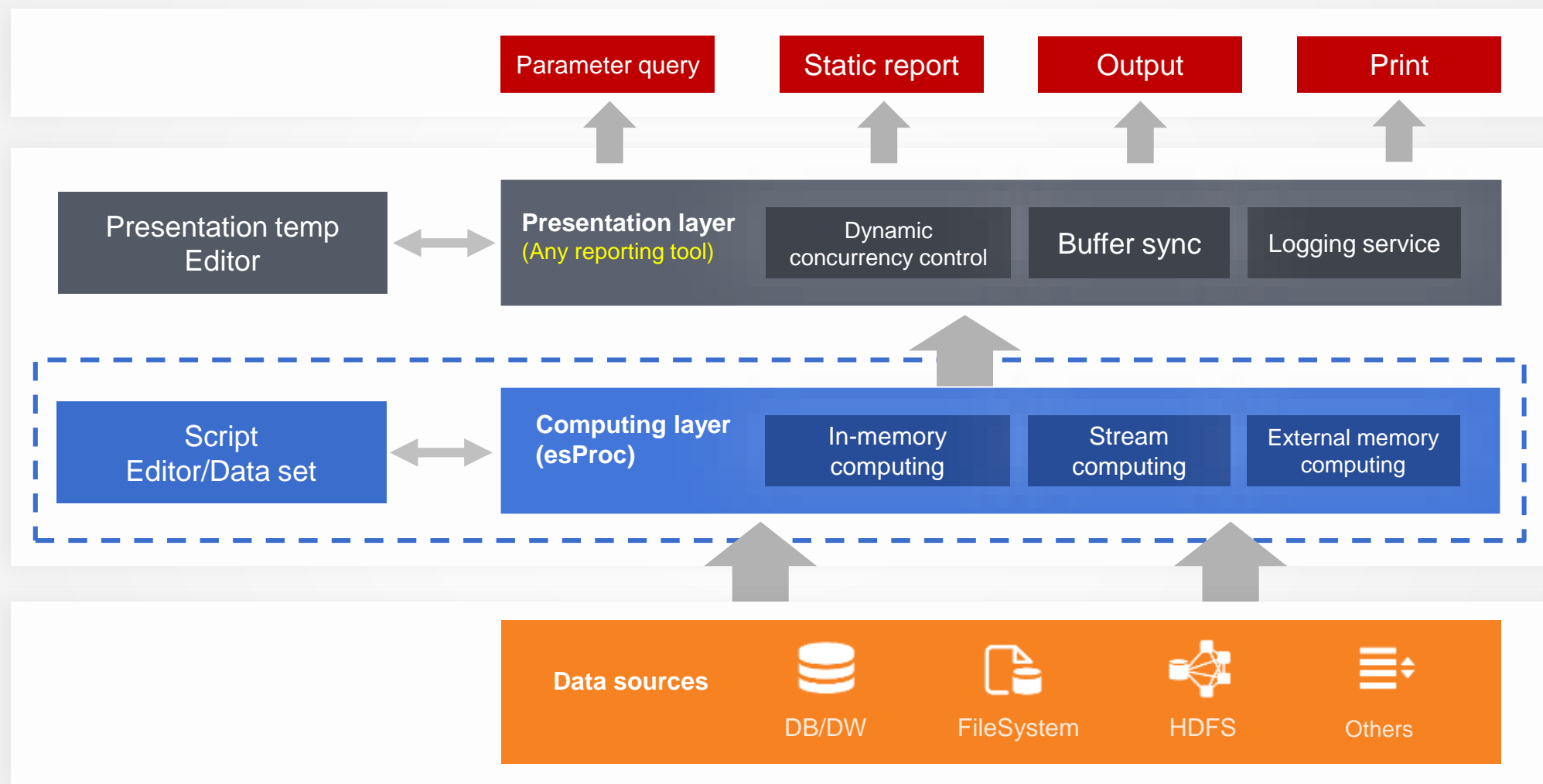
Tech features

A person wearing a blue suit jacket and a white shirt is standing next to a laptop. The person's hands are in their pockets. The laptop is open and the screen is visible. The background is blurred.

## esProc A computing problem buster!

1. An **efficient** Dev Tool specializing in **structured data** analysis & processing and intended for programmers & data analysts;
2. Dynamic Java-based **interpreted** language that adopts **innovative** computing model & **original** design to enable **streamlined** dev process & **high** performance;
3. With a **rich** class library & a **lightweight** architecture, esProc is **flexible** & **cost-effective** in handling real-world problems.

# esProc – Data computing layer



# Data computing layer – A stand-alone tool & module



## Computing tool

1. A computing tool helps to simplify the whole report development process;
2. Environment config (like data sources) becomes unnecessary; complex computing logics are achieved with simple code;
3. A script for handling unstructured data sources, like Excel & text formats, is simple.

## Independent module

1. A data computing layer is completely independent of an app, in operation & maintenance;
2. A modification in reporting module won't affect the computing module.

# CONTENTS

1

Solution

2

**Competitive advantages**

3

Tech features

# Competitive advantages



Streamlined dev  
process



High performance



Optimized  
structure



Efficient big data  
reporting



# Process-mode computation

The descriptive computing mode reporting tools use hampers  
process-mode computations

To handle process-mode computations common to complicated  
report dev, reporting tools use:

Hidden cell;

Java or stored procedure





# Process-mode computations – *Code examples*

Find big customers whose sales amount accounts for the 1st half of the total.

	A	B	C
1	Customer	Amount	=ds.sum(Amount)/2
2	=ds.select(Customer)	=ds.Amount	=C2[-1]+B2
3	NumOfVIP	=count(B2{C2[-1]<C1})	
4	AvgSales	=avg(B2{C2[-1]<C1})	

Hidden cell: Column C is set as hidden, with display condition C2[-1]<C1 in the 2nd row. Using a simple condition C2<=C1 will result in error with the first row of the 2<sup>nd</sup> half. Besides, the conditional expression will be computed repeatedly, and some of the reporting tools' special functionalities, like cell set filtering, are needed.

## Round-off error control

If there is disagreement between the detailed data and the totals after round-off, we need to find the appropriate round-off values for the detailed values according to the round-off value of the totals.

# esProc solution



## Stage 1: Data preparation

	A	B
1	=db.query("select Customer,Amount from CustomerSales order by Amount desc")	
2	=A1.sum(Amount)/2	=0
3	=A1.pselect((B1+=Amount)>=A2)	return A1.to(A3)

## Step 2: Data reporting

	A	B
1	Customer	Amount
2	=ds.select(Customer)	=ds.Amount
3	NumOfVIP	=ds.count()
4	AvgSales	=ds.avg(Amount)

The stepwise computation is clearer, and can cooperate with any reporting tool

# esProc vs JAVA



esProc is **deeply set-oriented** syntax, and thus can produce **more concise** code.

Java, however, doesn't have direct support for structured data processing.

## Faster & Shorter

- Java-based esProc offers high-level class library & methods

## Easy to understand & debug

- The ratio of pseudo code to real code is about 1:1.5; most of the time, a data preparation algorithm can be displayed **within the screen**
- Code can be displayed as much as possible in one page, which is easy to understand and debug



# esProc vs SQL/Stored procedure

Find big customers whose sales amount accounts for the 1st half of the total.

1	SELECT CUSTOMER, AMOUNT, SUM_AMOUNT
2	FROM (SELECT CUSTOMER, AMOUNT,
3	SUM(AMOUNT) OVER(ORDER BY AMOUNT DESC) SUM_AMOUNT
4	FROM (SELECT CUSTOMER, SUM(AMOUNT) AMOUNT
5	FROM ORDERS GROUP BY CUSTOMER))
6	WHERE 2 * SUM_AMOUNT < (SELECT SUM(AMOUNT) TOTAL FROM ORDERS)

Stepwise computation makes debugging and development convenient;

Discreteness support enables deep set orientation & order-based computations

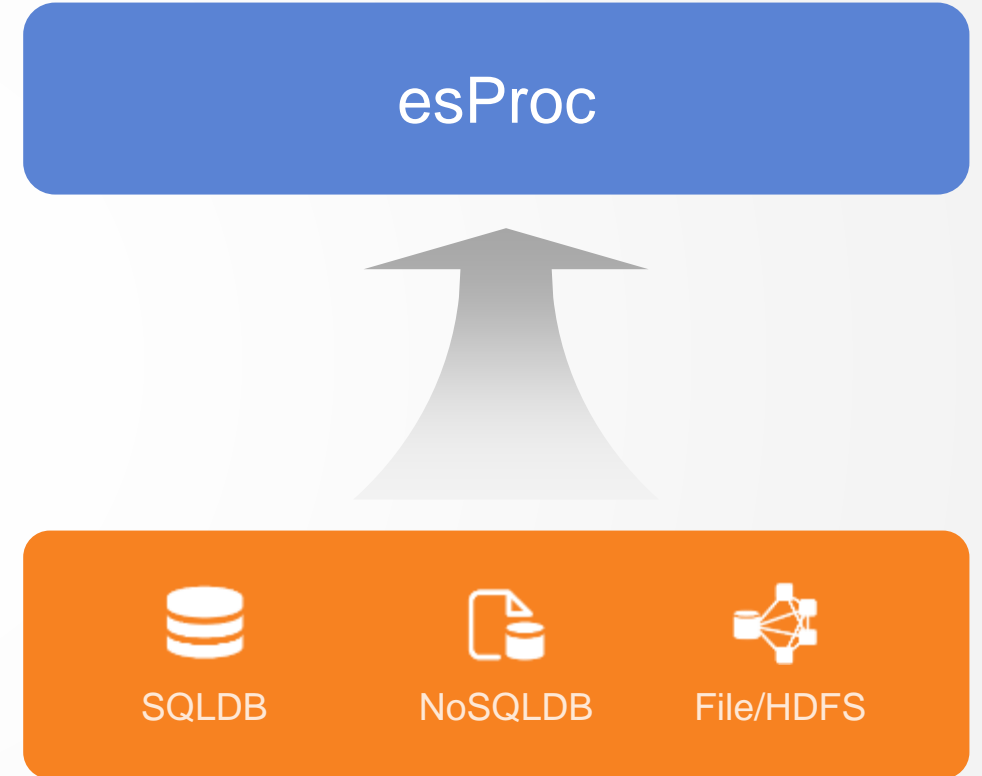
# Support of heterogeneous data sources



Reporting tools fall down on processing heterogeneous data sources in terms of computing ability & design capacity;

With esProc computing layer:

Data loading is unnecessary;  
Multilevel data structure is supported; which facilitate dev process;





# Dynamic data source/set

## Dynamic data source

Parameter-controlled data source connection  `${pds}.query("select * from T where F=?",pF)`

## Dynamic data set

Programming-logic-assisted dynamic SQL

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

## Result set capacity control

	A	B	
1	<code>=db.cursor("select * from T")</code>	<code>=A1.fetch(1000)</code>	
2	<code>if B1.fetch@0(1)</code>	<code>&gt;B1.insert(0,"loop")</code>	/Insert a mark if all data isn't fetched
3	<code>&gt;A1.close()</code>	<code>return B1</code>	

# Special report layouts



## Layouts unsupported by reporting tools:

- ▶ Horizontal column group

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

	A	B	C
1	=db.query("select a,b,c from T ")		
2	=A1.step(3,1)	=A1.step(3,2)[[null]]	=A1.step(3,3)[[null]]
3	=A2.derive(B2(#).a:a2,B2(#).b:b2,B2(#).c:c2,C2(#).a:a3,C2(#).b:b3,C2(#).c:c3)		

- ▶ Append blank rows

	A	
1	=db.query("select * from T")	
2	=pn-A1.len()%pn	/Calculate the number of to-be-appended blank rows
3	=A1 if(A2!=pn,A2*[null])	/The result set with to-be-appended blank rows

# Competitive advantages



Streamlined dev  
process



High performance



Optimized  
structure



Efficient big data  
reporting





# Reporting performance problems

## Caused by inefficient data preparation

A reporting tool is only competent to handle a small amount of data;

Reporting stage optimization cannot solve slow data preparation

Efficient data preparation can be achieved, if

We: Use a better computing method;

Can reduce the use of hidden cells

*eProc offers:* Fast data read/write;

Manageable buffer capacity;

Shared memory resources

# esProc computing layer – No hidden cells



Hidden cell

Store intermediate results

Hidden cell  
byproduct

Cells with appearance property use more memory resources, which reduces reporting performance

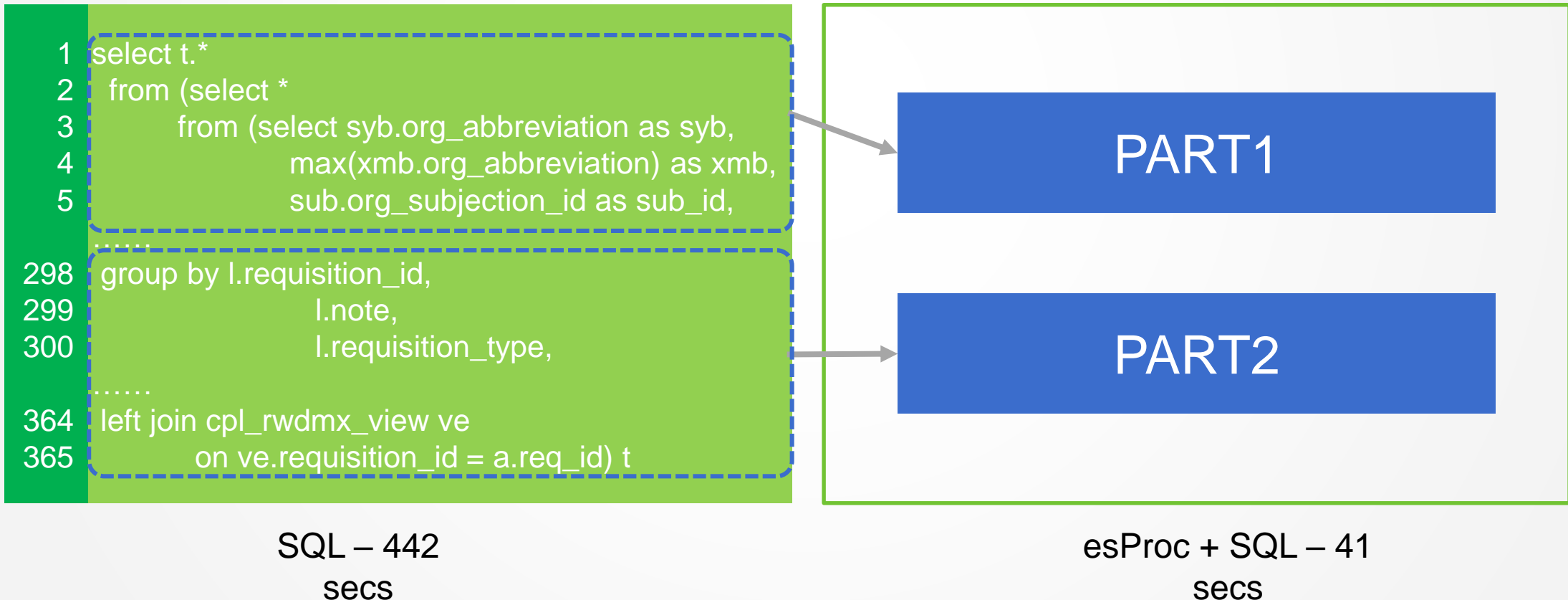
Solutions

- ✓ A separate computing layer makes intermediate result reuse convenient;
- ✓ No hidden cells and appearance property leads to efficient memory use



# Flexible SQL execution path

- Database transparency is user-friendly but execution-path-optimization-unfriendly;
- esProc **supports flexible execution paths**, and thus can execute certain computations outside the database and increase the overall performance





# Parallel retrieval

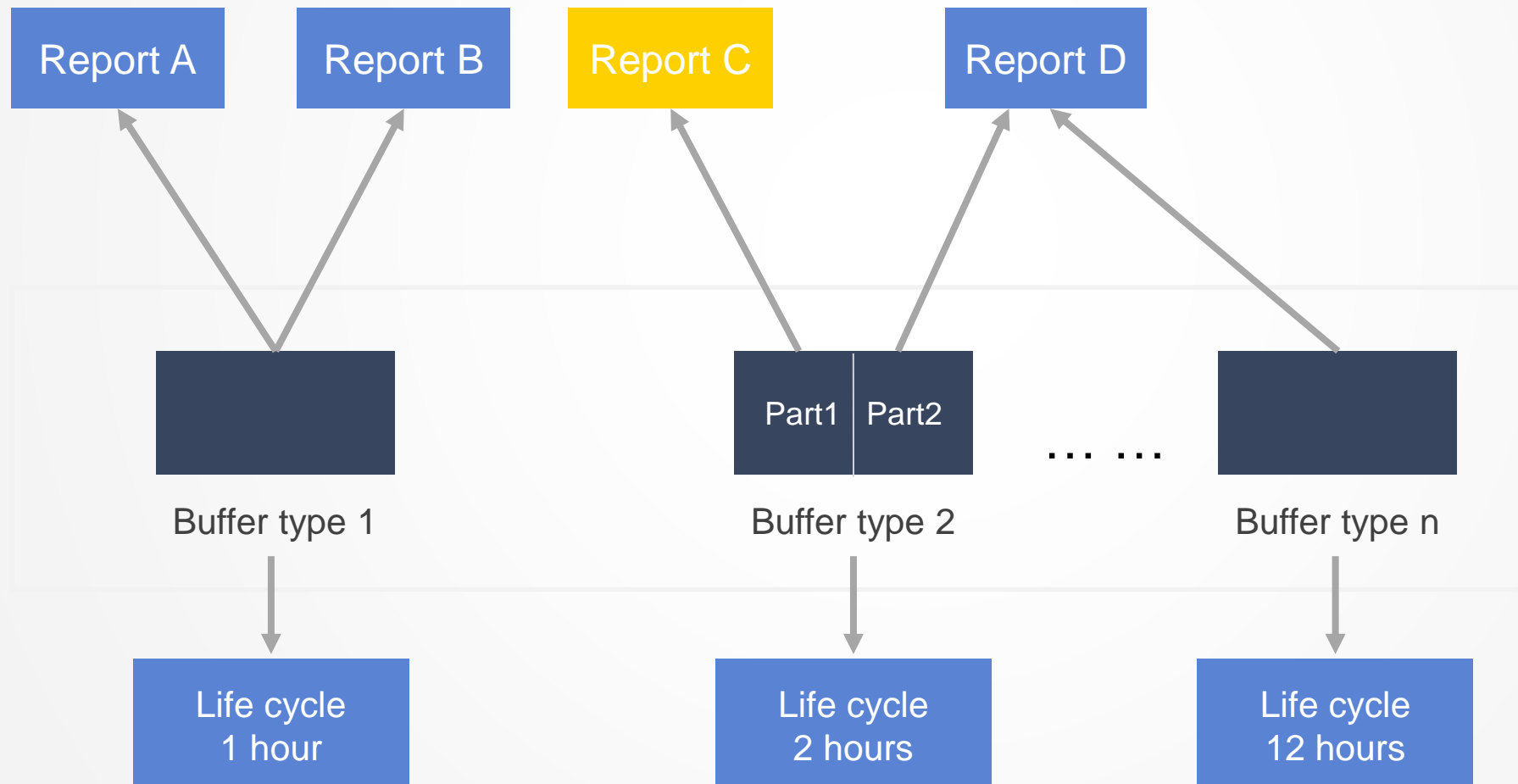
Data retrieval is critical to high reporting performance, but JDBC is so inefficient; esProc retrieves data by segments **in parallel** by creating multiple database connections, which increases performance **multiple fold**

	A	B	C
1	fork 4	=connect(db)	/4 threads, which connects to database respectively
2		=B1.query@x("select * from T where part=?",A1)	/Retrieve 4 parts one by one
3	=A1.conj()		/Union the returned result sets

# Flexible buffer



esProc supports **partial buffering**, **buffer reuse among reports**, and **different buffer life cycles**

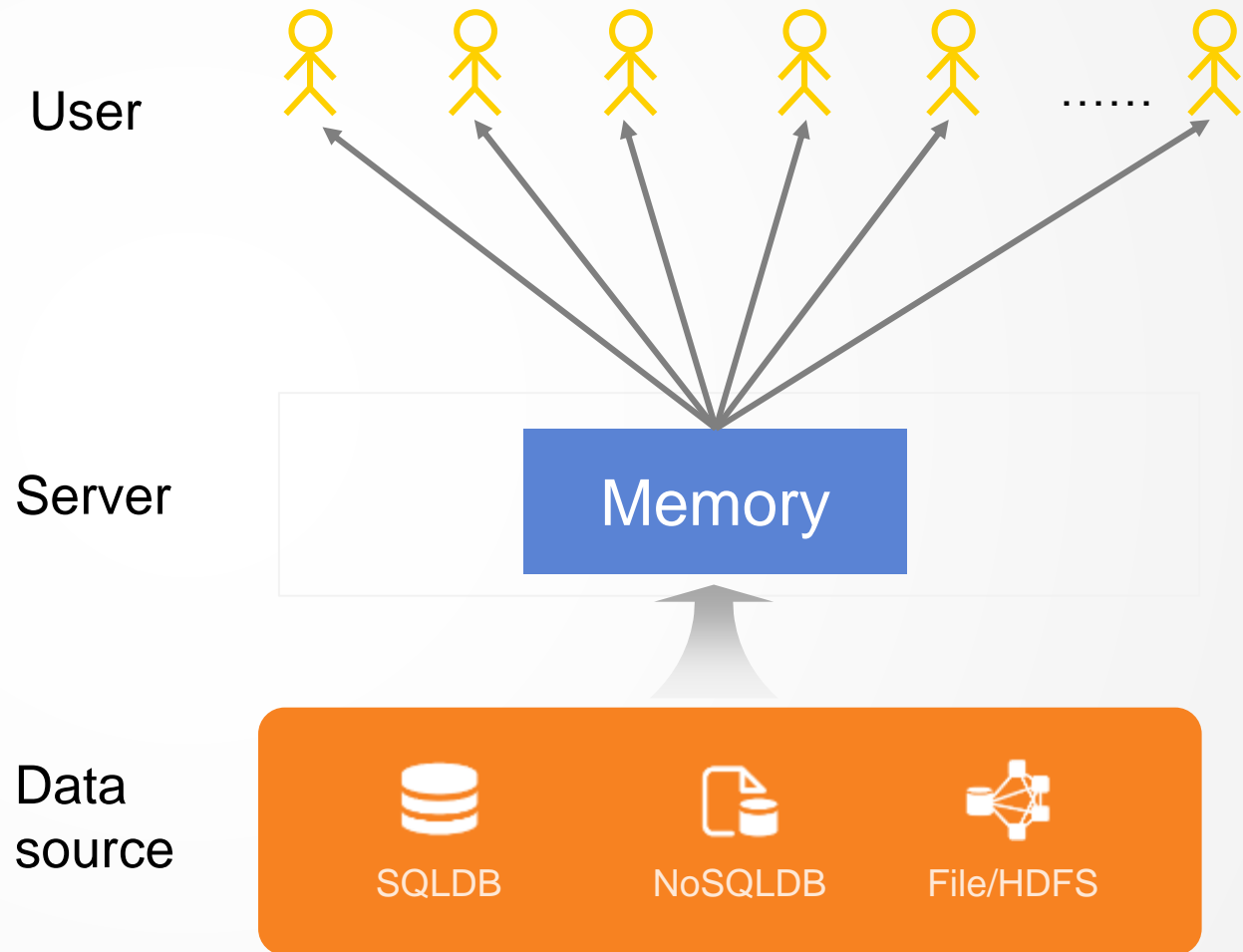


# Shared memory resources



In a **high concurrency**

environment, esProc can use shared memory mechanism to achieve higher performance and easier parallel processing



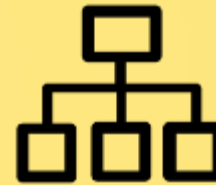
# Competitive advantages



Streamlined dev  
process



High performance



Optimized  
structure



Efficient big data  
reporting

# Reduced coupling with interpreted execution



Report's data preparation in Java and esProc:

## JAVA

### Low modularity

Java code has to be **compiled and packaged with the main app**, causing tight coupling

### “Cold” switching

A modification of report's data preparation algorithm in Java leads to an overall **recompilation & repackaging**

## esProc

### High modularity

An esProc **script file** is stored and maintained along with the report template, which creates a separate reporting module

### Hot switching

esProc **interpreted execution** enables hot switching



# External-database algorithms reduce stored procedures



## Tight coupling between report components and database is caused by stored-procedure-based data preparation algorithms

Separate storage makes it hard to match a stored procedure to its report;

Modifying stored procedures needs database privilege, posing potential security risks;


A stored procedure could be used by multiple apps, causing tight coupling between apps

Report's data preparation in esProc will **greatly reduce the use of stored procedure**; an external algorithm stored and managed with report template is a part of an app, which **looses coupling between the report and other parts of the app or other apps**

# External intermediate data helps to trim the database



Problems of intermediate tables resulted from accumulated data or complex computations:



**Chaotic database management**

Accumulated app-generated, linearly-stored intermediate tables mess up the database



**Wasted database resource**

Update of useless intermediate tables with ETL is a waste of resources

With esProc, it's **convenient-to-manage** to put intermediate data in a file system outside the database; there will be high IO performance & computing ability, and **as few intermediate tables as possible** and a slim database



# Direct processing of various data sources & cross-database computing ability

Advantages of handling various data sources directly:

1. The database becomes slim without data loading and the resulting intermediate tables;

2. Real-time data retrieval reduces the risk of inconsistency;

3. Make best use of the strengths of each type of data source

# Competitive advantages



Streamlined dev  
process



High performance

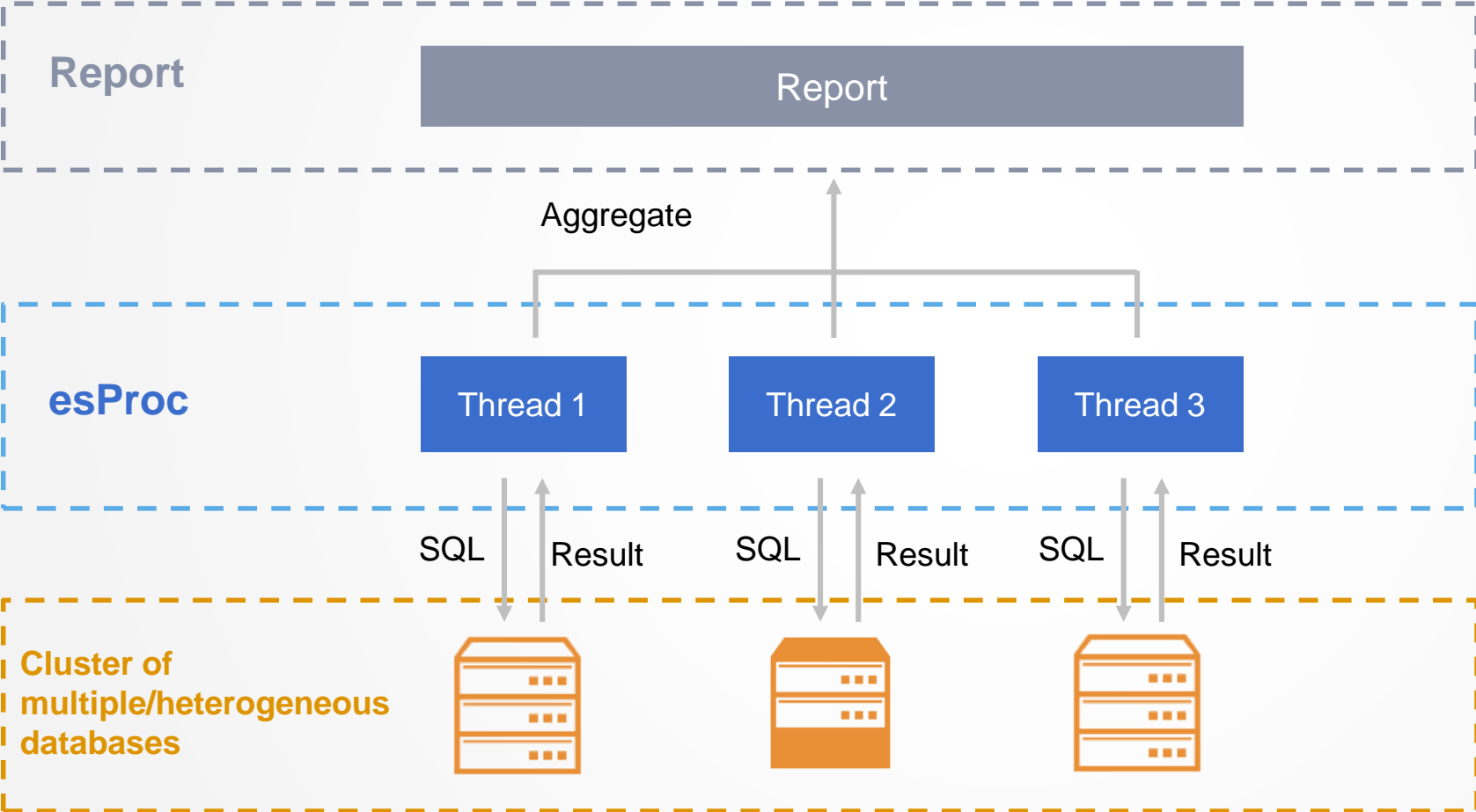


Optimized  
structure



Efficient big  
data reporting

# Concatenate result sets of handling different databases



esProc concatenates result sets returned from handling a cluster of same-structure or different-structure databases in parallel, and passes the aggregate to the report

# Create T+0 reports via hybrid computing



Storage type & common problems of T+0 reports:

1

Store historical & current data in one database

Huge amounts of historical data causes high storage cost and low performance

2

Store historical & current data in different database

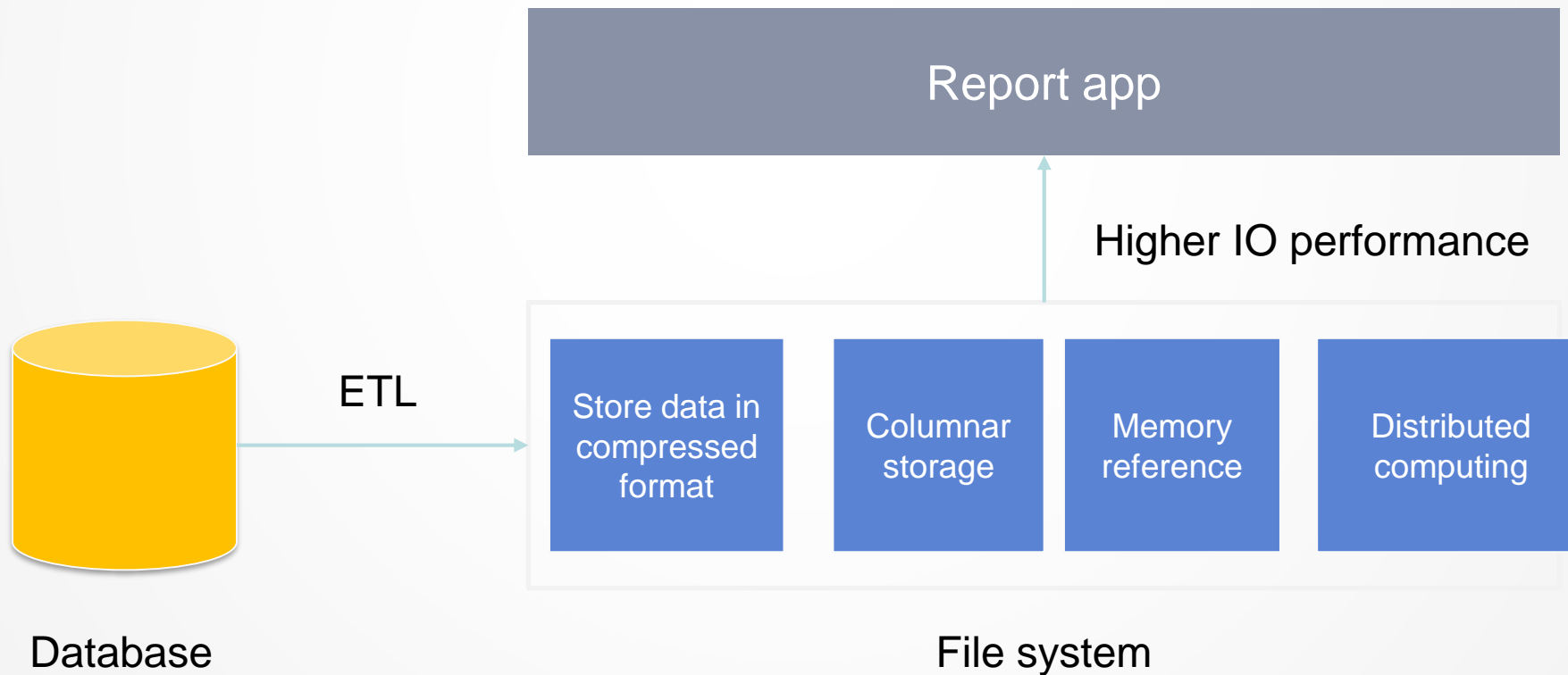
Cross-database computing is required; it's complex & low-performance, and it's hard to implement over different types of databases

- esProc can:
- Perform queries over multiple **different-structure databases**;
- Store historical data in the **file system** with better IO performance and handle it with cluster computing to get higher performance with lower cost

# External data & parallel processing



A file system is better than a database in terms of IO performance, and supports storing data in a compressed format, columnar storage, memory reference, and distributed computing; that makes external data processing is higher performing and helps to **reduce database workload**



# Hadoop



## Hadoop: unsatisfactory computing mechanism

Complex dev process;

Result of computation needs to be loaded into RDB

## Prepare Hadoop data in esProc for reporting

Easy dev process;

Support cluster computing and have a better performance;

No need of data loading to RDB



# Summary



To handle the reporting dynamicity and achieve expected goals, esProc adopts certain ways:



## High efficiency

Rich syntax & class library



## Loose coupling

Store algorithm along with its report template



## Modularity

Separate report data preparation stage



## Hot switching

No need to restart app for interpreted execution



## Easy dev process

No need to configure environment & reference app-level



## Low cost

Suitable for non-professional programmers

# CONTENTS

1

Solution

2

Competitive advantages

3

**Tech features**

# esProc dev environment



Execute/Debug/Step

Set breakpoint

The screenshot shows the esProc development environment. At the top, there is a menu bar (File, Edit, Program, Tools, Window, Help) and a toolbar with icons for file operations and execution. Below the toolbar, a script editor displays a series of commands in a spreadsheet-like format with columns A, B, C, and D. A console window on the left shows system information and error messages. On the right, a data table is displayed with columns for Index, Datetime, Commodity, and Volume. The interface is annotated with orange boxes and lines pointing to various features.

Index	Datetime	Commodity	Volume
1	2009-06-01 08	20077	28
2	2009-06-01 08	20056	47
3	2009-06-01 08	20094	34
4	2009-06-01 08	20020	19
5	2009-06-01 08	20013	42
6	2009-06-01 08	20077	19
7	2009-06-01 08	20069	19
8	2009-06-01 09	20011	22
9	2009-06-01 09	20007	22
10	2009-06-01 09	20005	39
11	2009-06-01 09	20085	31
12	2009-06-01 09	20054	8
13	2009-06-01 09	20011	49

Real-time system info output

WYSIWYG-style interface that enables easy debugging and convenient intermediate result reference

Simple syntax, natural & intuitive computing logic

Ready-to-use  
Easy-to-debug



# Procedure-oriented computing

## Reliable loop branch control

	A	B	C	D
1	=demo.query("SELECT ORDERID AS CONTRACT,CLIENT,SELLERID AS SALE,AMOUNT,ORDERDATE AS DATE FROM SALES")			
2	=demo.query("SELECT * FROM EMPLOYEE")	=Year=2012		
3	>A1.run(SALE=A2.select@1 (EID:A1.SALE))	/Field Value is Record		
4	=A1.group(SALE)			
5	=create(Sale,ThisYear,LastYear,GrowthRate,NumOfClients,NumOfBigClients,RatioOfBigClients)			
6	for A4	=A6(1).SALE.NAME		
7		=A6.select(year(DATE)==Year).sum(AMOUNT)	/Sales Amount This year	
8		=A6.select(year(DATE)==Year-1).sum(AMOUNT)	/Sales Amount Last Year	
9		=B8/B7-1		
10		=A6.group(CLIENT).(sum(AMOUNT))		
11		=B10/count(=10000)	Number Of Clients	
12		=B10/count(=10000)	Number Of Big Clients	
13		=B12/B11		
14		>A5.insert(0,B6,B7,B8,B9,B11,B12,B13)		
15	result A5			

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

# Agile syntax



Count the longest consecutively rising trading days for a stock

```
1 select max(ConsecutiveDays)
2 from (select count(*) ConsecutiveDays
3       from (select sum(ChangeMark) over(order by TradingDate) Non-risingDays
4             from (select TradingDate,
5                   case when ClosingPrice>lag(ClosingPrice) over(order by
6                     TradingDate)
7                       then 0 else 1 end ChangeMark
8                   from StockPrice) )
9       group by Non-risingDays)
```

SQL

```
      A
1 =StockPrice.sort(TradingDate)
2 =0
3 =A1.max(A2=if(ClosingPrice>ClosingPrice[-1],A2+1,0))
```

esProc

Syntax suitable for describing a natural way of thinking;  
Data model enabling efficient algorithms



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



# Rich class library

## Intended for structured data processing

	A	B	C
1	=esProc.query("SELECT OrderID AS Contract,OrderDate AS		/Retrieve Orders table
2	=A1.group(Seller)		
3	=create(Seller,Sales(This year),Sales (Last year),CustomerNumber,BigCustomerNumber)		
4	for A2	=A4(1).Seller	
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)	
10		>A3.insert(0,B4,B5,B6,B8,B9)	

### Grouping & Loop

	A	B	C
1	=esProc.query("select * from Employees")		
2	=A1.select(Gender=="Male")		
3	=A1.select(BirthDate>=date("1970-01-01"))		
4	=A2*A3		/Intersection; find the male employees who were born after 1970
5	=A2&A3		/Union; find the male employees or the employees who were born after 1970
6	=A2\A3		/Difference; find the employees who were born before 1970
7	=A4.sum(Wages)		
8	=A5.avg(Age)		
9	=A6.sort(BirthDate)		
10			/Set is a widely-used, basic data type
11			

### Set operations

	A	B	C
1	=file("Transaction.txt").import@t()		
2	=A1.sort(CustomerID,TransactionDate)		
3	=A2.select(CarType=="Jetta"    CarType=="Magotan").dup@t()		
4	=A3.derive(interval(TransactionDate[-1],TransactionDate):Interval)		
5	=A4.select(CarType=="Jetta" && CarType=="Magotan" && CustomerID==CustomerID[-1])		
6	=A5.avg(Interval)		
7			
8			
9			
10			

### Sorting & Filtering

	A	B	C
1	=esProc.query("select * from Employees")		
2	=A1.sort(EntryDate)		
3	=A2.pmin(BirthDate)		/Sequence number of the record of the oldest employee
4	=A2(to(A3-1))		/Access a record with the sequence number
5	=esProc.query("select * from StockPrice table where StockCode='000062'")		
6	=A5.sort(TransactionDate)		
7	=A6.pmax(ClosingPrice)		/Sequence number of the record with the highest closing price
8	=A6.calc(A7,ClosingPrice,ClosingPrice[-1]-1)		
9			
10			/Access a record with the sequence number
11			

### Ordered sets



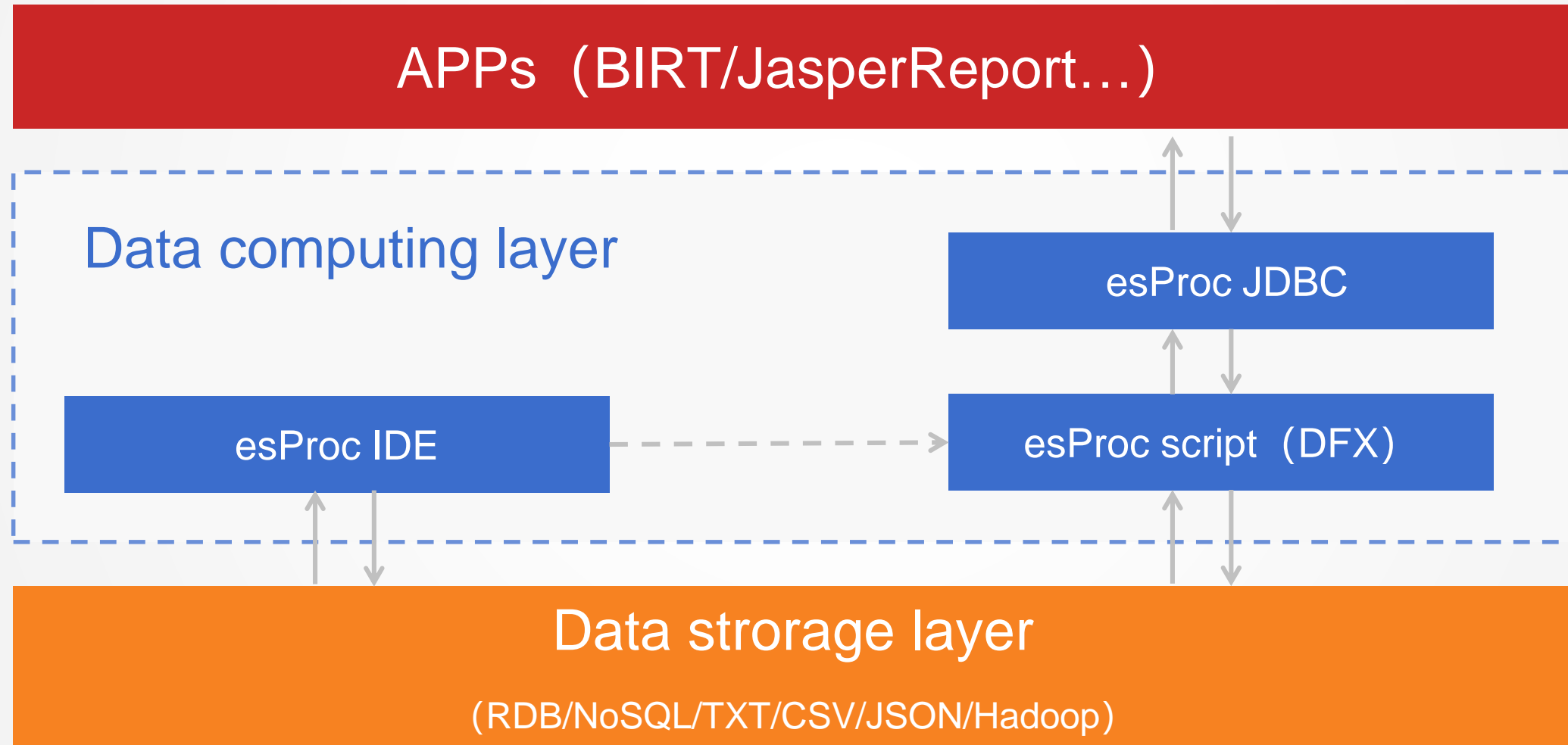
# Various data source interface

- ▶ High-efficiency binary compressed format & columnar storage
- ▶ RDB: Oracle,DB2,MS SQL,MySQL,PG,.....
- ▶ TXT/CSV, JSON/XML, EXCEL
- ▶ Hadoop: HDFS, HIVE, HBASE
- ▶ MongoDB, REDIS, ...
- ▶ HTTP、ALI-OTS
- ▶ ... ..
- ▶ Ready-to-use, built-in interface



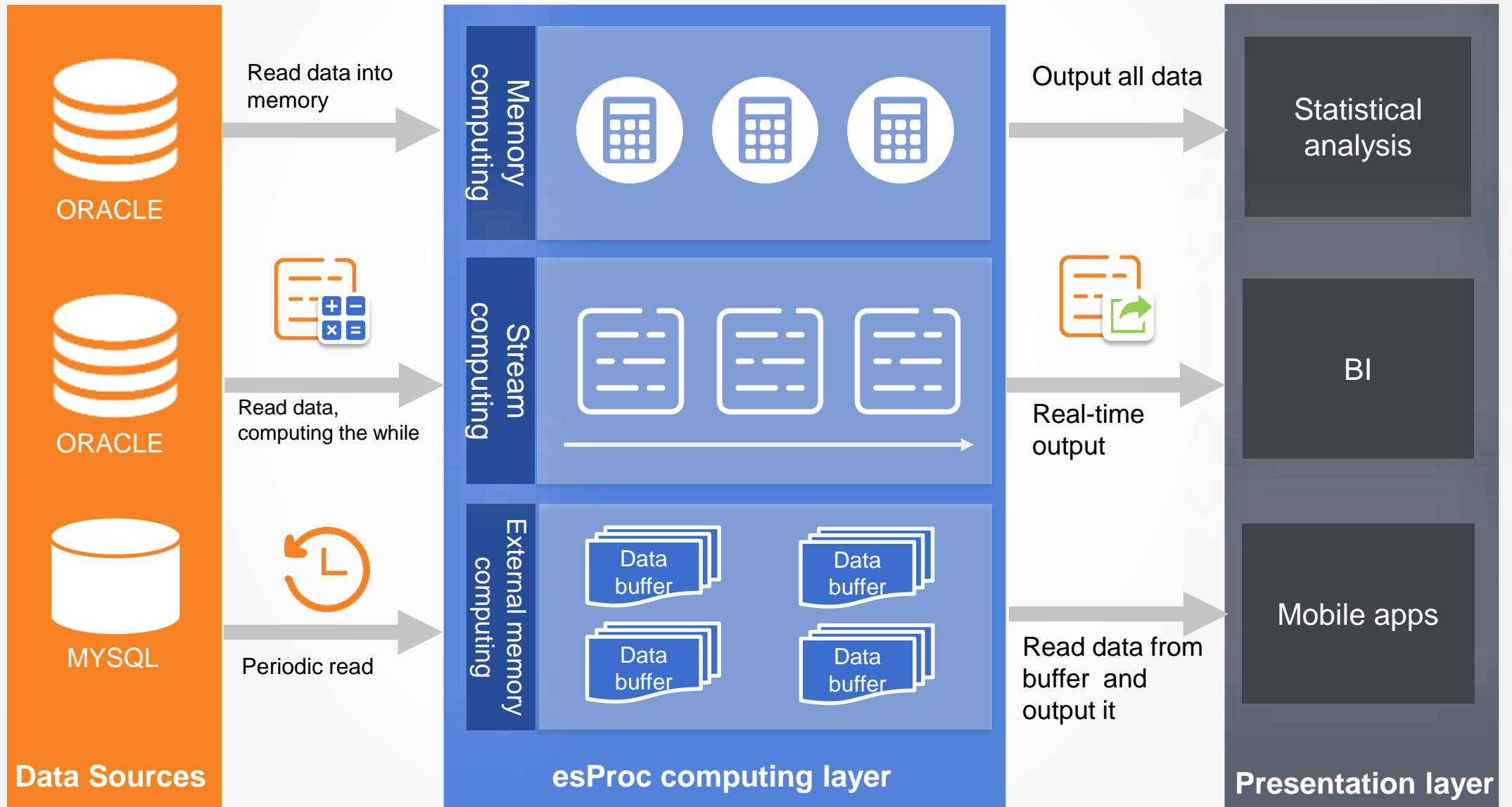
# Integration & management system

Seamless integration & easy-to-manage code





# Data stream models



# Innovation makes progress!

