esProc SPL

High performance computing platform

What is esProc SPL?

- Analysis **Database**
- Computing and processing of structured and semi-structured data
- neither **SQL system** nor NoSQL Technology
- **Self created SPL syntax**, more concise, efficient and powerful
- Theoretical innovation, **not originated from open source**

SPL: Structured Process Language

What can esProc SPL do?

Offline batch job

- Nightly batch processing jobs are headache for many corporate IT orgs to fit in certain time window.
- Especially, for mission critical application and/or at quarter or year end.

Online query/report

- Both management demands of real time analytics and decision driven data analysis require fast query response as well as high concurrency
- Being forced to wait for minutes for a query/report, the business personnel becomes angry.

esProc SPL：Improve big data computing performance X times

What are the counterparts of esProc SPL?

MPP/ Distributed data warehouse/Main-Memory Database

- Rely on multi machine cluster or huge memory to improve computing power and consume more hardware resources
- Typical technology：**Hadoop/Spark**

Large database and its special machine

- Decades of rich optimization experience, special high-grade hardware cooperation
- Typical technology：**Oracle ExaData**

esProc SPL: Single machine matches the cluster, order of magnitude faster

Case National Astronomical Observatory – Star aggregation

TASK

- 11 photos, 500000 objects/photo
- Celestial bodies with close astronomical distance (trigonometric function calculation) are regarded as the same
- Complexity: 500000 * 500000 * 10 = 2.5 trillion (times comparison)

- 500000 celestial bodies test
- Python 200 lines, single thread 6.5 days
- SQL 100CPU cluster 3.8 hours

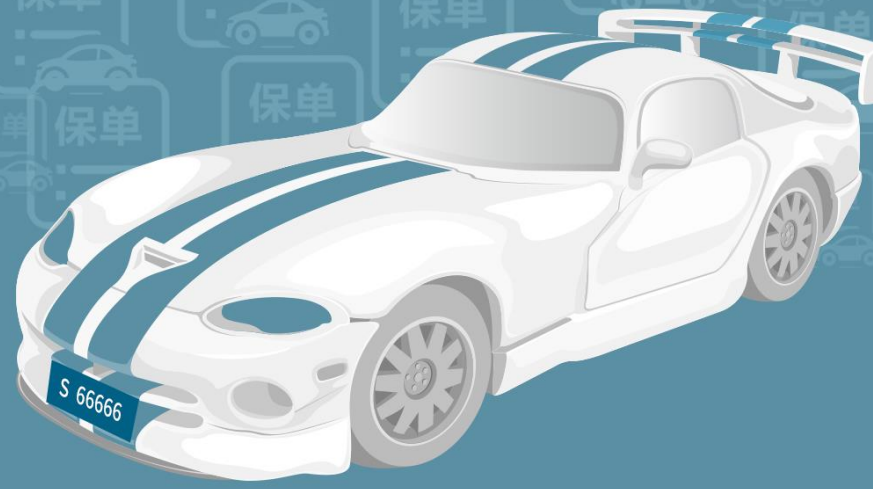
esProc SPL

- 500000 celestial bodies test， 2.5 minutes
- 5 million celestial bodies， 3 hours
- Codes: 50 lines

Increase speed

2000 times

Case Batch running of insurance policies of an auto insurance company



TASK

- Insurance policy table: 35 million lines， details table: 123 million lines
- There are various ways of association, which need to be handled separately

- AS400+Informix
- Determine premium 3600 seconds
- Historical policy matching 6672 seconds

esProc SPL

- Determine premium 68 seconds
- Historical policy matching 1020 seconds

Increase speed

52.9 times

Details： *http://c.raqsoft.com/article/1644827119694*

Case Batch running of loan agreements of a bank

TASK

- SQL: 48 steps，3300 lines
- Historical data: 110 million lines，daily increase: 1.37 million lines
- Complex multi-table join

- AIX+DB2
- Calculation time: 1.5 hours

esProc SPL

- Calculation time: 10 minutes，codes: 500 lines

Increase speed

8.5 times

Details： *http://c.raqsoft.com/article/1644215913288*

Case Mobile banking: multi concurrent account query

TASK

- Huge number of users, and large concurrent accesses
- Institutional information changes frequently and needs to be associated in time

- Commercial data warehouses on Hadoop cannot meet the high concurrency requirements
- Using 6 ElasticSearch cluster can cope with concurrency, but can not associate in real time. The data update time is long, and the service must be stopped during this period.

esProc SPL

- Single machine can cope with the same concurrent volume as ES cluster
- Real time association, zero waiting time for institutional information update

1 server VS 6 servers

Details： *http://c.raqsoft.com/article/1643533607375*

Case Calculation of the number of unique loan clients of a bank

TASK

- Too many labels, and hundreds of labels can be arbitrarily combined to query
- Association, filtering and aggregation calculation of a 20 million row large table and even larger detailed tables
- Each page involves the calculation of nearly 200 indexes, and 10 concurrency will cause the concurrent calculation of more than 2000 indexes

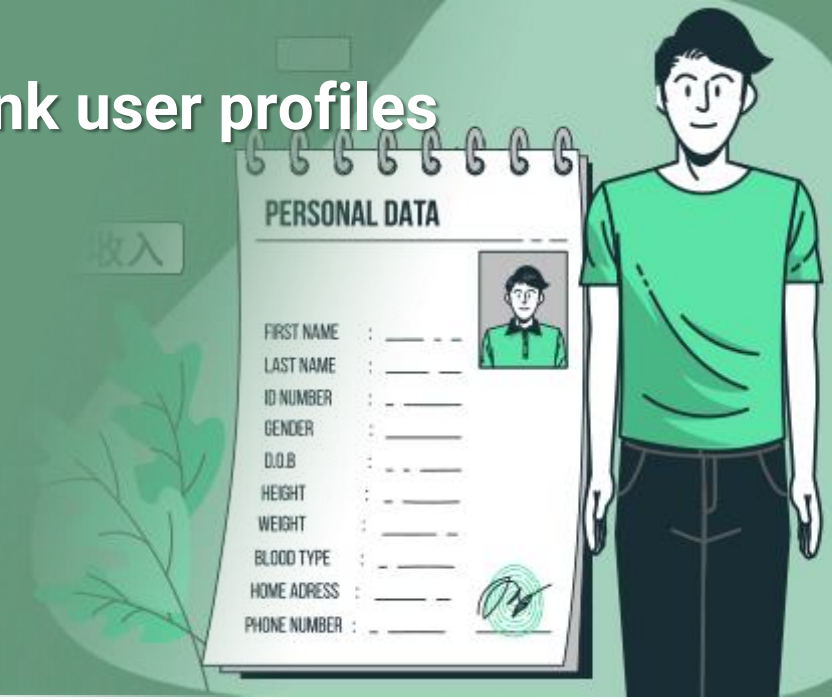
- Oracle
- Unable to calculate in real time; The query requirements have to be agreed in advance, and the calculation is carried out one day earlier.

esProc SPL

- 10 concurrency, 2000 indexes in total, less than 3 seconds
- No need to prepare in advance, instantly select any label combination, and get query results in real time

Turn pre-calculation into real-time calculation

Case Intersection calculation of customer groups in bank user profiles



TASK

- Huge amount of data, hundreds of millions of customers, thousands of customer groups (many to many relationship), dozens of dimensions
- Arbitrarily select several customer groups to calculate the intersection: the intermediate result set is huge and cannot be pre calculated
- More than 10 concurrent requests

- Famous OLAP server on Hadoop,100CPU cluster
- Single task: 2 minutes

esProc SPL

- 12CPU single task: 4 seconds
- 10 concurrent tasks can be completed in 10 seconds

Increase speed

250 times

Details：http://c.raqsoft.com/article/1643333049458

SPL High Performance Computing Concept

What can we do?

Look forward !

Oh, so it is like this.

Yes, that's not magical.

Then find a programmer to do it.

Not so easy !

Isn't that all you can do is stare in despair?

Hey hey, that's how it works most of the time.

So **High performance computing** = **Algorithm design** + **Algorithm Implementation** ⟶ **Becoming the bottleneck of high performance computing**

SQL, NoSQL, NewSQL, Hadoop, all restrict the implementation of algorithms

SQL example: How to get the top 10 of 100 million rows of data?

```
SELECT TOP 10 customer, amount
FROM order
ORDER BY amount DESC
```

It is not difficult to code this query requirement in SQL, but there is an ORDER BY clause here, which means that the data should be sorted completely.

Sorting is a costly operation, and we know that there is a low complexity algorithm that does not need sorting, but it can't be described in SQL. We can only hope for the automatic optimization of the database engine, but it's not reliable.

For example, Oracle can optimize this statement, but some open source databases can not, and Oracle will not optimize it in complex situations (such as getting the top n in a group).

Several scenarios to beat SQL

1. Complex orderly Computing: funnel analysis of user behavior transformation

- Calculate the user churn rate after each event (page browsing, search, shopping cart addition, order placement, payment, etc.)
- Multiple events are effective only when they are completed within a specified time window and occur in a specified order. It is very hard to implement it in SQL, not to mention optimize it.

2. Multi-step big data batch running

- Complex business requirements are difficult to implement directly in SQL, cursor reading is slow and difficult to calculate in parallel, wasting computing resources
- The implementation by a stored procedure requires thousands of lines and tens of steps, with the repeated landing of intermediate results, the batch running cannot be completed within the specified time window

3. Multi index calculation on big data, repeated use and multiple associations

- Complete the calculation of hundreds of indexes at one time, and use the detailed data for many times, and association is also involved. SQL needs to traverse data repeatedly.
- Mixed calculation of large table association, conditional filtering, grouping and aggregation, and deduplication; Real-time calculation with high concurrency.

Relational algebra, which was born 50 years ago, can not meet the needs of contemporary software and hardware environment.

It is difficult to make up for the theoretical limitations by "engineering optimization" (the degree of improvement is limited).

The difficulty of SQL seriously wastes hardware resources.

esProc theoretical breakthrough： SPL Inside

Breakthrough in mathematical theory: Discrete dataset; A new generation of computational syntax：SPL

High performance depends on algebra rather than code

Lower complexity algorithms obtain stronger computing power, revolution from carriage to car

Traditional database SQL

Relational algebra

- Lack of discreteness, incomplete set orientation, unable to effectively solve the big data needs emerged in the past 10 years
- Many high-performance algorithms cannot be implemented with SQL

esProc SPL

Discrete dataset

- Complete set orientation, effective combination of set orientation and discreteness
- Dozens of high-performance algorithms, more than half of which are self invented

Examples of SPL High Performance Algorithms

Understanding aggregation

	A	
1	=file("data.ctx").open().cursor()	
2	=A1.groups(;top(10,amount))	Orders in the Top 10
3	=A1.groups(area;top(10,amount))	Top 10 orders per region

Converting high-complexity sorting to low-complexity aggregation

Multi-purpose Traversal

	A	
1	=file("order.ctx").open().cursor()	Prepare for traversal
2	=channel(A1).groups(product;count(1):N)	Configure reuse
3	=A1.groups(area;sum(amount):amount)	Traverse and get grouped results
4	=A2.result()	Get the results

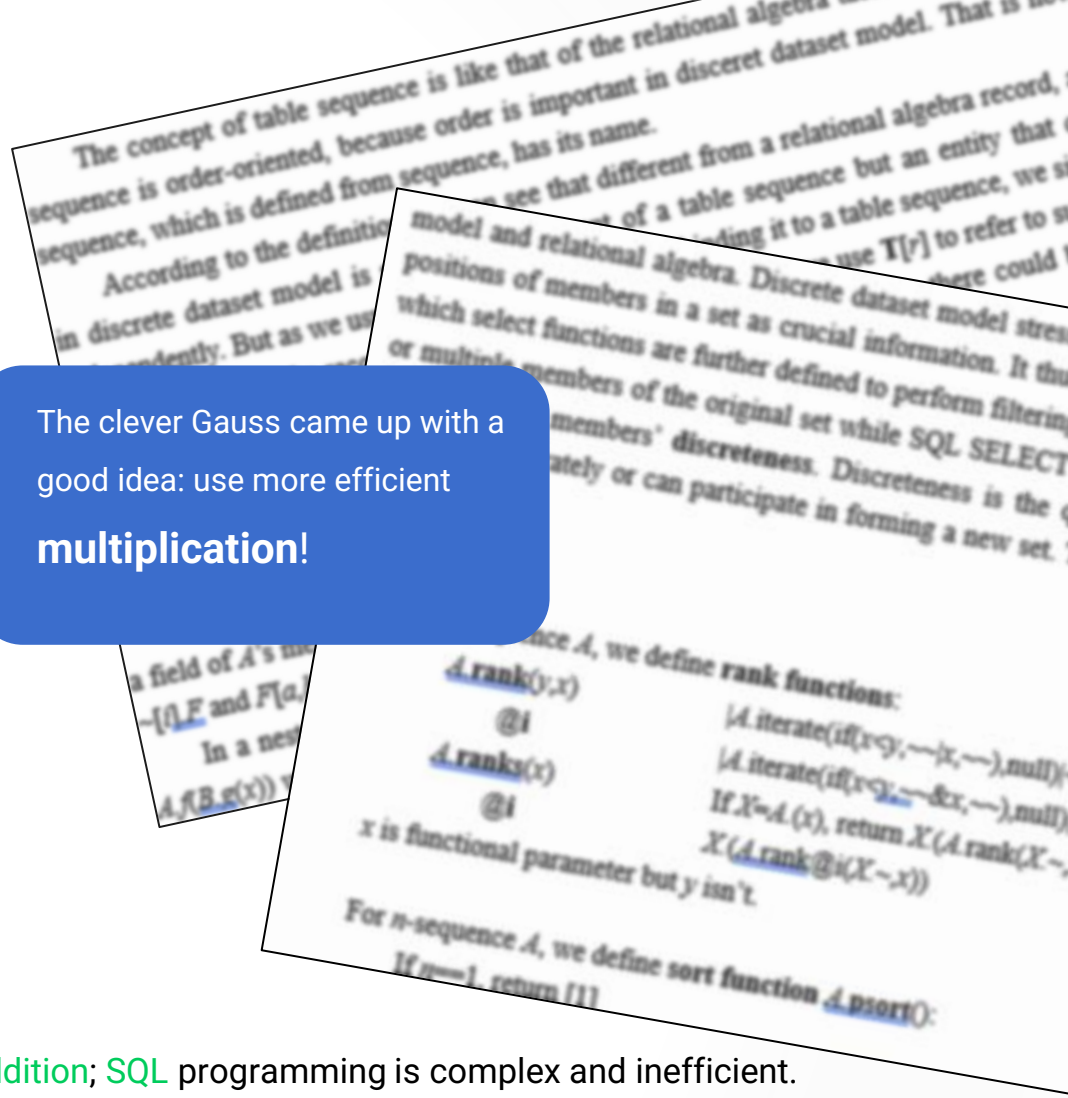
Multiple result sets can be returned by one traversal

Why is esProc SPL faster?

【analogy】Calculate 1+2+3+...+100=?

Ordinary people will do like this	Gauss does like this
1+2=3 3+3=6 6+4=10 10+5=15 15+6=21 21+7=28 ...	1+100=101 2+99=101 ... A total of fifty 101 50*101=5050

The clever Gauss came up with a good idea: use more efficient **multiplication**!



The theoretical basis of SQL relational algebra is like an arithmetic system with only addition; SQL programming is complex and inefficient.

While **esProc**'s own theoretical system **discrete dataset** is equivalent to the invention of **multiplication**! SPL programming is simple and efficient.

SPL also has more multiplications (high-performance computing and storage libraries), and everyone can be Gauss (fast implementation of high-performance algorithms).

Extended reading:SPL: a database language featuring easy writing and fast running

Part of High Performance Computing Mechanism Provided by SPL

Traversal technique	Highly efficient Joins	High performance storage	Cluster computing
Delayed cursor	※ Foreign key as pointer	Orderly Compressed Storage	Preemptive Load Balancing
Aggregate Understanding	※ Numbering of foreign keys	Free column storage	※ Multi-zone composite table
※ Ordered cursor	Order-based merge	※ Hierarchical Numbering positioning	※ Cluster dimension table
※ Multi-purpose traversal	※ Attached table	Index and Caching	※ Memory spare tire fault tolerance
Prefilter traversal	※ Unilateral HASH Join	※ Double increment segmentation	External storage redundancy fault tolerance

※ Many algorithms and storage schemes here are the original inventions of SPL!

Extended reading: Performance Optimization for Big Structured Data

esProc is not only high-performance, but also open

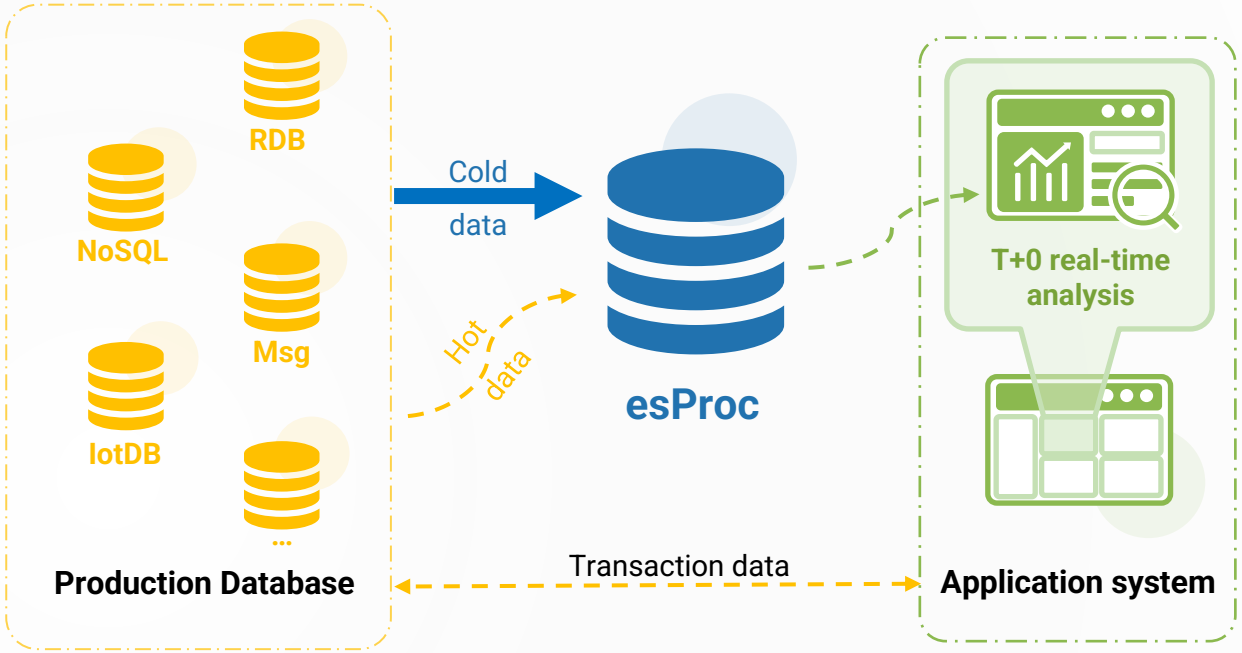
- **Mixed calculation of hot and cold data to implement**

  T+0 real-time analysis

  - Organized historical cold data
  - Real time reading of transaction hot data

- **The production system needs no modification**

  - Make full use of the advantages of the original multiple data sources

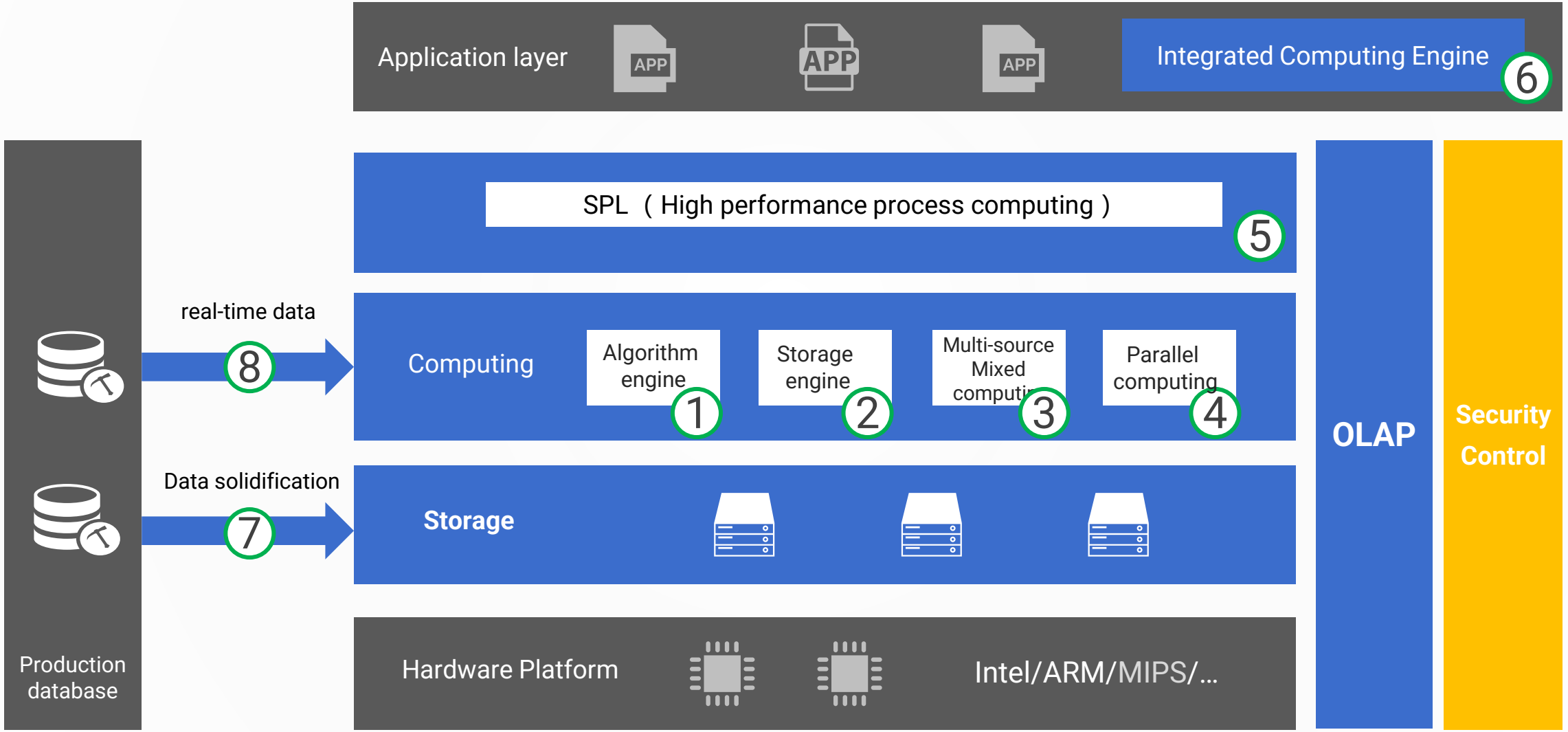


Support low-risk, high-performance and strong real-time HTAP with open multi-source hybrid computing capability

- **HTAP databases are difficult to meet HTAP requirements**
  - Requiring replacement of production system database, with high risk
  - Insufficient SQL computing power, insufficient historical data preparation, low performance
  - Closed computing capacity, complex ETL process required due to external diverse data sources, poor real-time ability

Extended reading： *http://c.raqsoft.com.cn/article/1658456594393*

Technical Architecture



New programming language

What is the max days has a stock been rising?

	A
1	=stock.sort(tradedate).group@i(closing<closing[-1]).max(~.len())

SPL

```
SELECT max(continuousdays)-1 FROM
        (SELECT count(*) continuousdays FROM
                (SELECT SUM(flag) OVER ( ORDER BY tradedate) norisingdays FROM
                ( SELECT tradedate,
                        CASE WHEN  closing>LAG(closing) OVER( ORDER BY tradedate THEN 0 ELSE 1 END flag
                FROM stock ))
        GROUP BY norisingdays)
```

SQL

High Performance Storage

High Performance Data Storage

Private data storage format: bin files, composite tables

File System Storage Form

Support to store data by business classification in tree directory

BIN FILE

Double increment segmentation supports arbitrary number parallelism

Self-owned efficient compression technique (reduced space; less CPU usage; secure)

Generic Storage, allowing set data

COMPOSITE TABLE

Mixed row and column storage

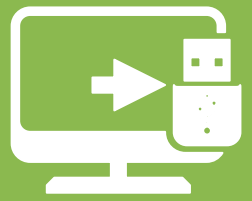
Ordered storage improves compression rate and positioning performance

Efficient intelligent index

Double increment segmentation supports arbitrary number parallelism

Integration of main and sub table to reduce storage and association

Numbering keys to achieve efficient positioning Join

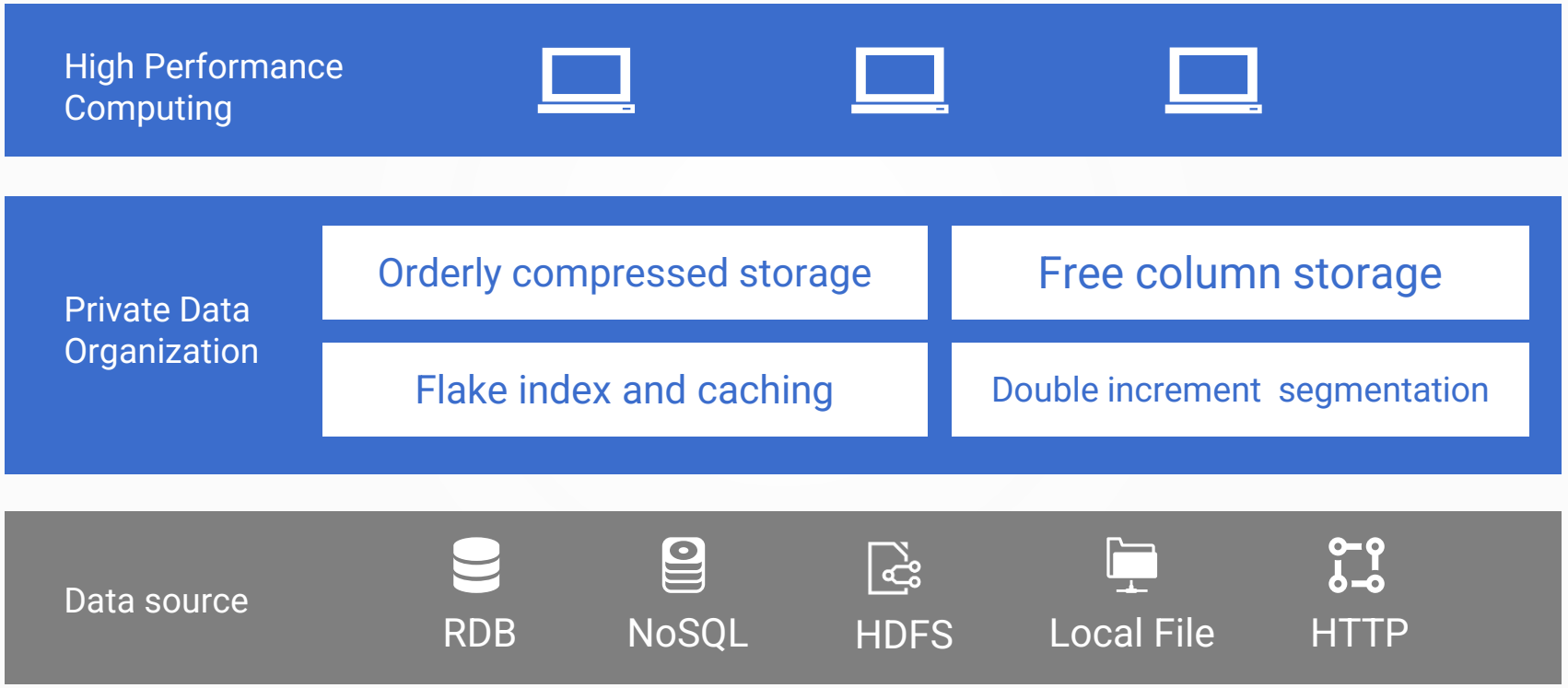


FAQ

Frequently asked questions

Does esProc store data by itself?

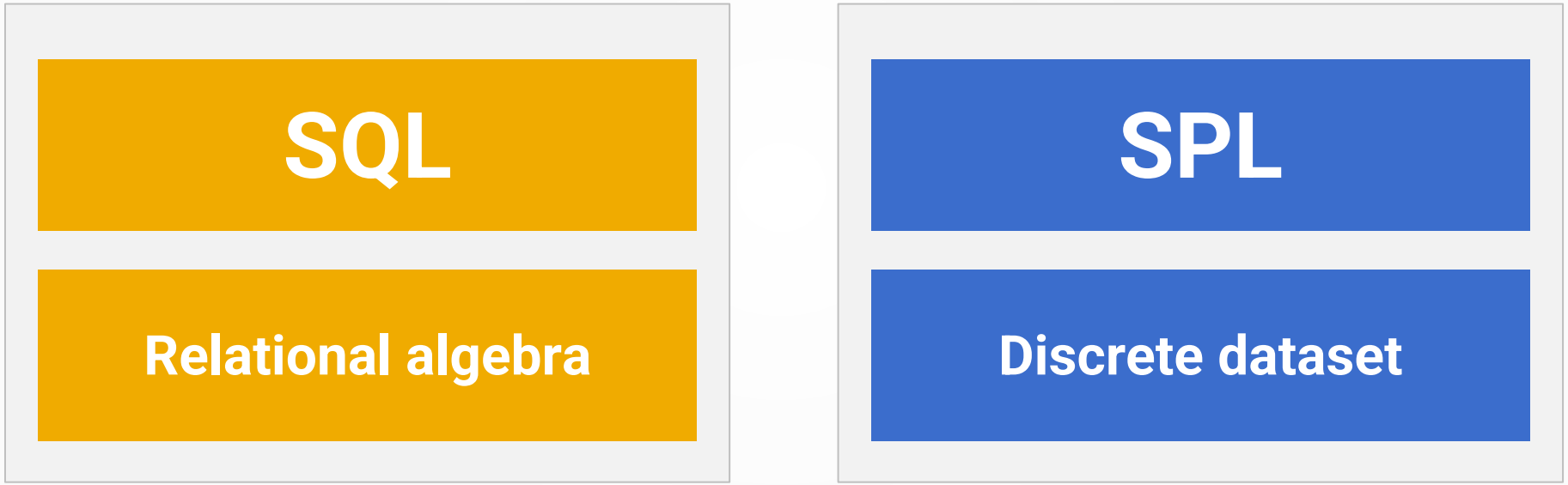
It must! The storage of data-intensive computing is the guarantee of performance. The inefficient storage of traditional RDB and HADOOP can not achieve high performance.



esProc designs special and efficient data organization schemes for memory, external storage and cluster, which are suitable for a variety of computing scenarios.

Is esProc based on open source or database technology?

esProc is based on a brand-new computing model, no open source technology can be cited, and all independent innovation from theory to code.



SPL based on innovation theory can no longer use SQL to achieve high performance, and SQL can not describe most low complexity algorithms.

How difficult is it to learn SPL?

SPL is dedicated to performance optimization and provides a dedicated syntax

Learning SPL is not difficult. It can be mastered in hours and skilled in weeks.

What is difficult is to design optimization algorithms!

To learn SPL is as easy as turning over the palm

Optimizing algorithm is much more difficult to design

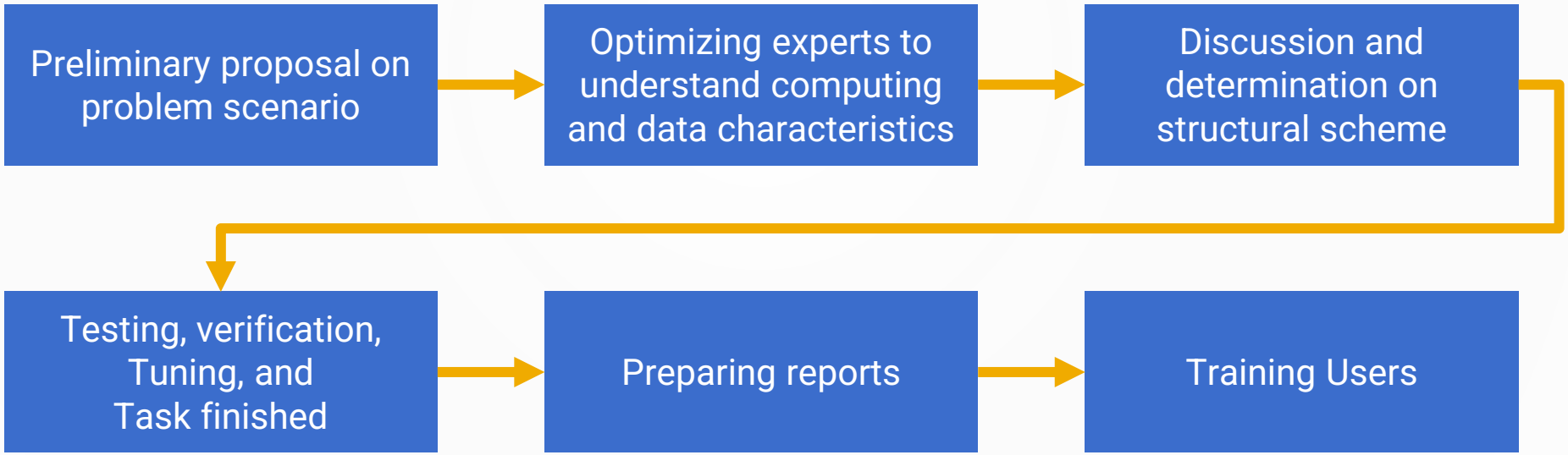
So we designed the following optimization process

Performance optimization process

The first 2-3 scenarios will be implemented by Scudata engineer in collaboration with users.

Most programmers are used to the way of thinking in SQL and are not familiar with high performance algorithms. They need to be trained to understand in one or two scenarios.

Dozens of performance optimization routines will be experienced and learned. Algorithmic design and implementation are not so difficult.



Give a man a fish and you feed him for a day. Teach him how to fish and you feed him for a lifetime!

Extended reading：What should we do when SQL (and stored procedure) runs too slowly?

THANKS