High Performance Online Computing Scheme

esProc SPL Base application scenarios

Contact author to learn more

Online Computing Application Scenarios

Multidimensional analysis OLAP

Self-service analysis

Dashboard management

Report/Query

Mobile terminal

Online Computing Faces Difficulties

Front-end applications - high performance requirements!

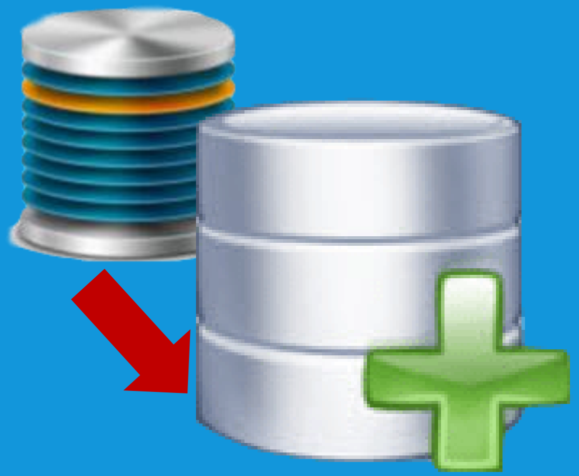
Hundreds of users access and expect second-level response

Expect to query the full amount of data

Database, Data Warehouse - Performance is uncontrollable!

Unstable performance due to excessive application load and great influence from other applications

Existing solution problems: expansion or replacement of databases and data warehouses



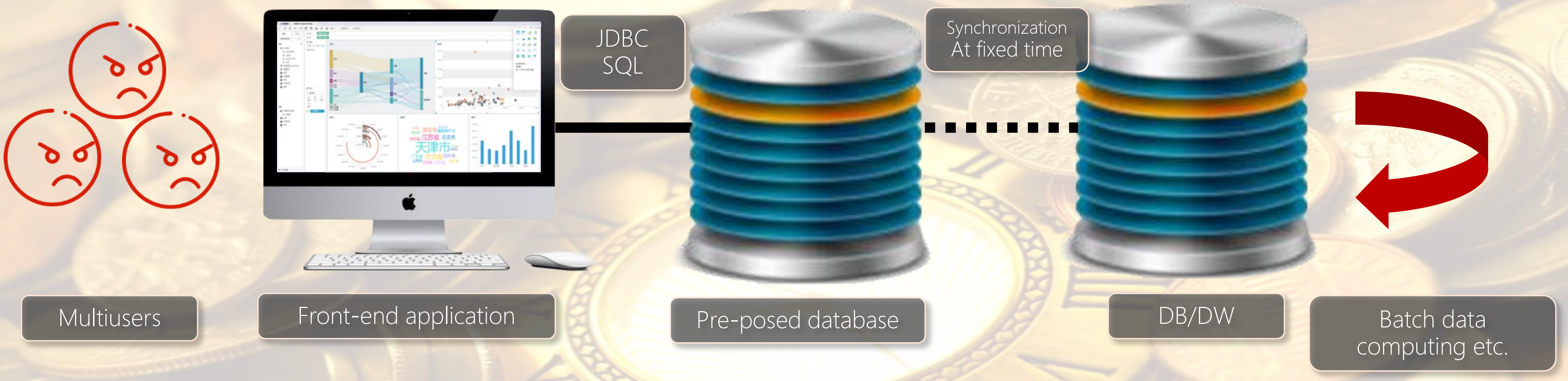
High Cost for Expansion

Expansion cost of database and warehouse is high
Data Warehouse Nodes number is limited
Continuous increase of nodes can not effectively increase speed

Replacement is not feasible

Change of databases and warehouses, involving multiple departments
Multiple other applications, cost is too high
Once changed, can't guarantee quicker

Existing Solution Problem: Adding pre-posed database



Repetitive Construction

The front end expects to query the full amount of data. Clustering is also necessary for the size of pre-posed database is the same as data warehouses

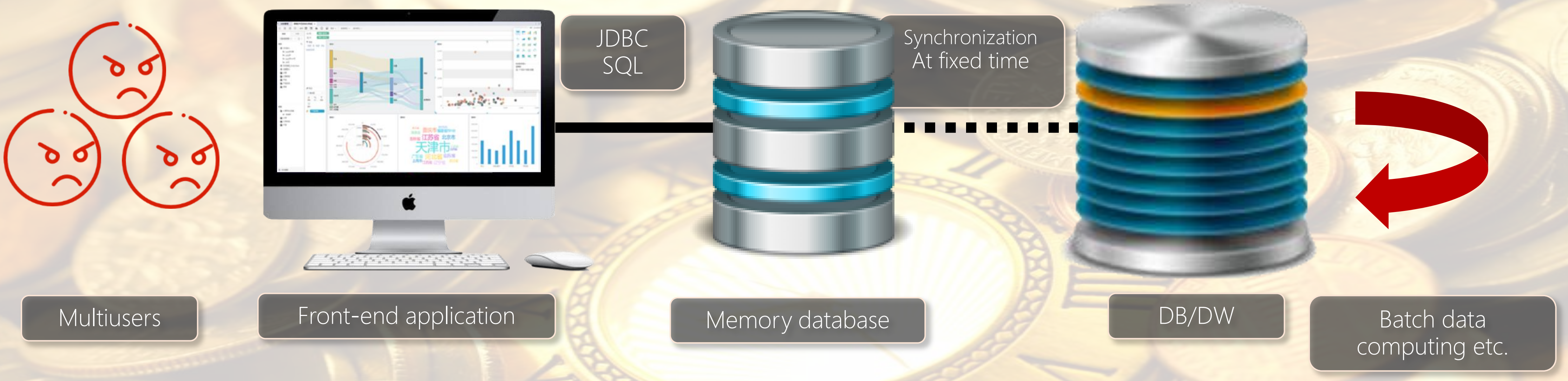
Not fast enough

The row-storage database can’ t achieve second-level response for tens of millions data

No routing capability

Can not achieve high-frequency hot data pre-positioning, a large number of cold data post-positioning routing function

Existing Solution Problem: Adding Memory Database



The price is too high

The purchase price for memory database is millions, or even tens of millions

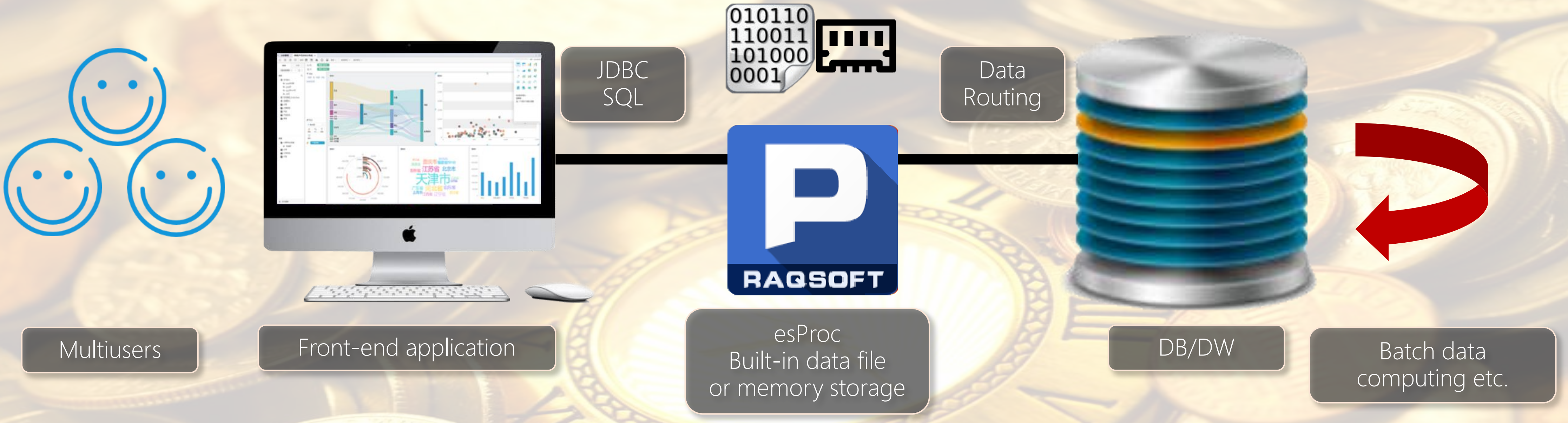
Service cost is super high

Have to pay high service fee annually.
It needs the maintenance of the original factory, tens of thousands of yuan at a time.

Capacity is limited

Capacity is determined by stand-alone memory size, unable to expand horizontally

Solution: Using esProc to implement High Performance Online Computing Backstage



Supporting large concurrency

esProc can cluster in parallel, implementing multi-user large concurrent access

Fast by column-storage

Built-in column-storage data file, implementing second-level response for tens of millions of data

Full memory mode

Supports full memory mode running.
Supports cluster and parallel, High Availability

Successful case: Massive accounts concurrent real-time query project

Large concurrency requirements

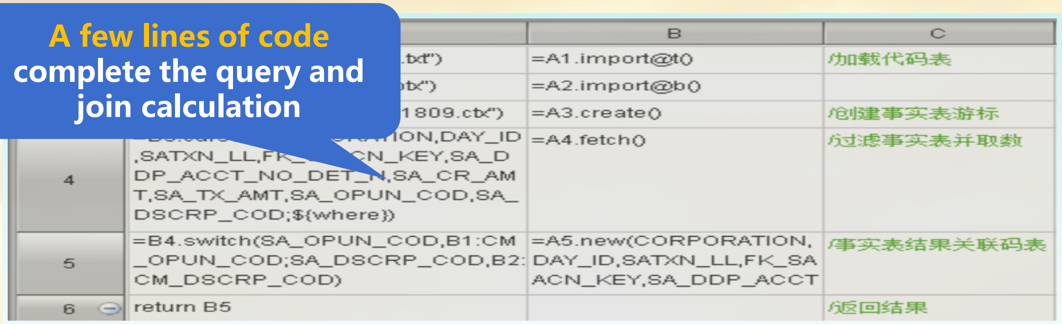
Bank account details inquiry, single server required to support 60 concurrency.

Data scale

From September 2015 to September 2018, more than 300 million detailed data, and multiple dimension tables need to be joined.

esProc response time **0.5** second

A few lines of code complete the query and join calculation



Query time does not exceed 5 seconds!
Comparable to professional column-storage data warehouse

Each server supports 50 concurrencies
Actual support for hundreds of user access without pressure

More than 30 million pieces of data
Search results by filtering according to conditions

Fully compatible with self-service analysis tools
Only need to change JDBC configuration to esProc

Successful Case: Bank Multi-user Self-service Analysis Project with Large Data Volume

Successful case: Bank receipt inquiry

Optimizing requirement

Query the receipt data of a branch in a month, the response time is 18 minutes, and often resulting in memory overflow.

Data scale

Tens of millions pieces of data

esProc increases speed by **108** times

Scene	Before optimization	After optimization	Increase speed
Bank receipt inquiry	18 minutes	10 seconds	108 times

esProc provides SPL language and many high performance computing support means

Traversal technique
Solution for Joins

Storage formats
Using index

Segmentation and Parallel
Cluster solutions

esProc Optimization for Complex SQL

```
Select F1, F2, F3,
  (select FF1 from TABLE1 WHERE...) AS F4,
  (select min(FF1) from TABLE2 GROUP BY...) AS F5,
From
  (select FFF1 from TABLE3 WHERE...) T1
Left join
  (select FFFF1 from TABLE4 WHERE...) T2
On T1.FFF2=T2.FFFF2
WHERE
T1.FFF2 in (select min(FFFF1) from TABLE5 GROUP BY...)
```

- Converting Subqueries to JOIN Computing
- Stripping Multi-Layer Nested SQL to Single Layer Computing
- 。。。

On-line Computing Front-end
SQL multilayer nesting with sub-queries

OLAP engine can only optimize simple SQL

esProc optimizes this type of SQL in depth

esProc provides innovative pre-aggregation capabilities

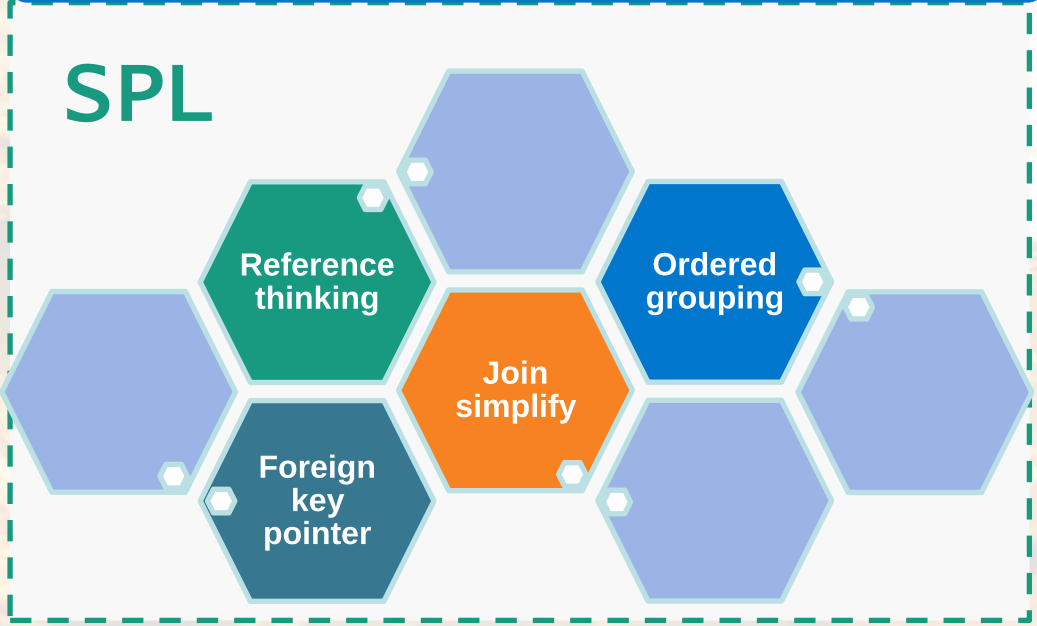
Partial pre-aggregation to effectively balance the contradiction between space and time

Give a feasible method of pre-aggregation for time-period statistics

esProc in-memory computing

Why can SPL achieve high performance in-memory computing?

Discrete dataset model

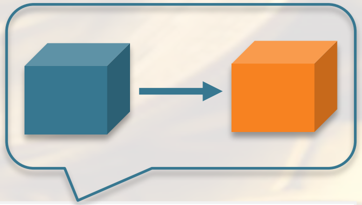


Reasons for SPL High Performance

- ✓ Efficient reuse and reference mechanism reduces data replication; SPL can do more computing tasks with the same memory!
- ✓ Make full use of memory characteristics, foreign keys can be pre-associated with pointers, JOIN can be completed in a constant time!
- ✓ More high-performance optimization algorithms: ordered grouping, efficient Joins, etc.

esProc in-memory computing：SQL vs. SPL

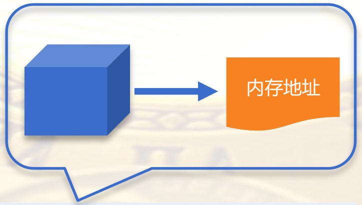
Select the 1998 sales records from the contract table.



1	SELECT * FROM Contract table WHERE YEAR(Sales date)=1998
2	

SQL

Query filtering results are always newly generated, which will copy the data again, not only consume time, but also occupy more memory!

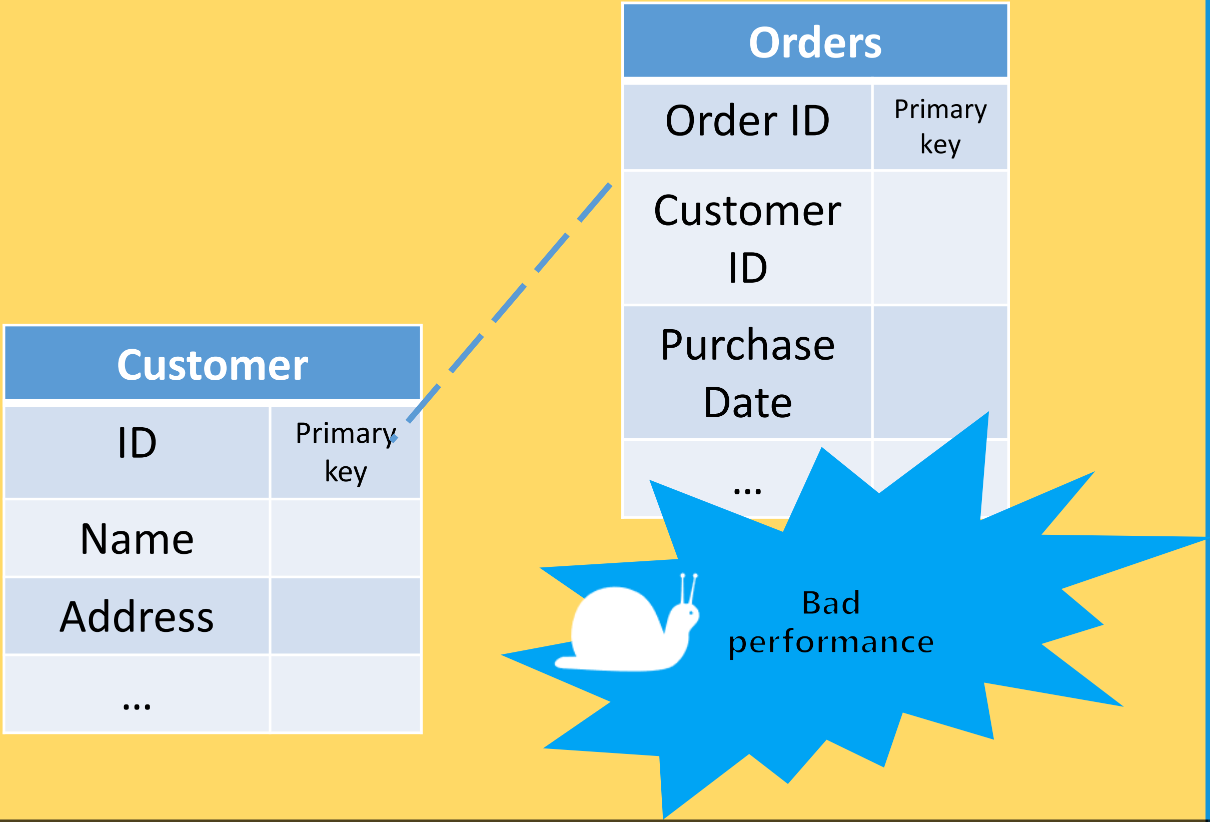


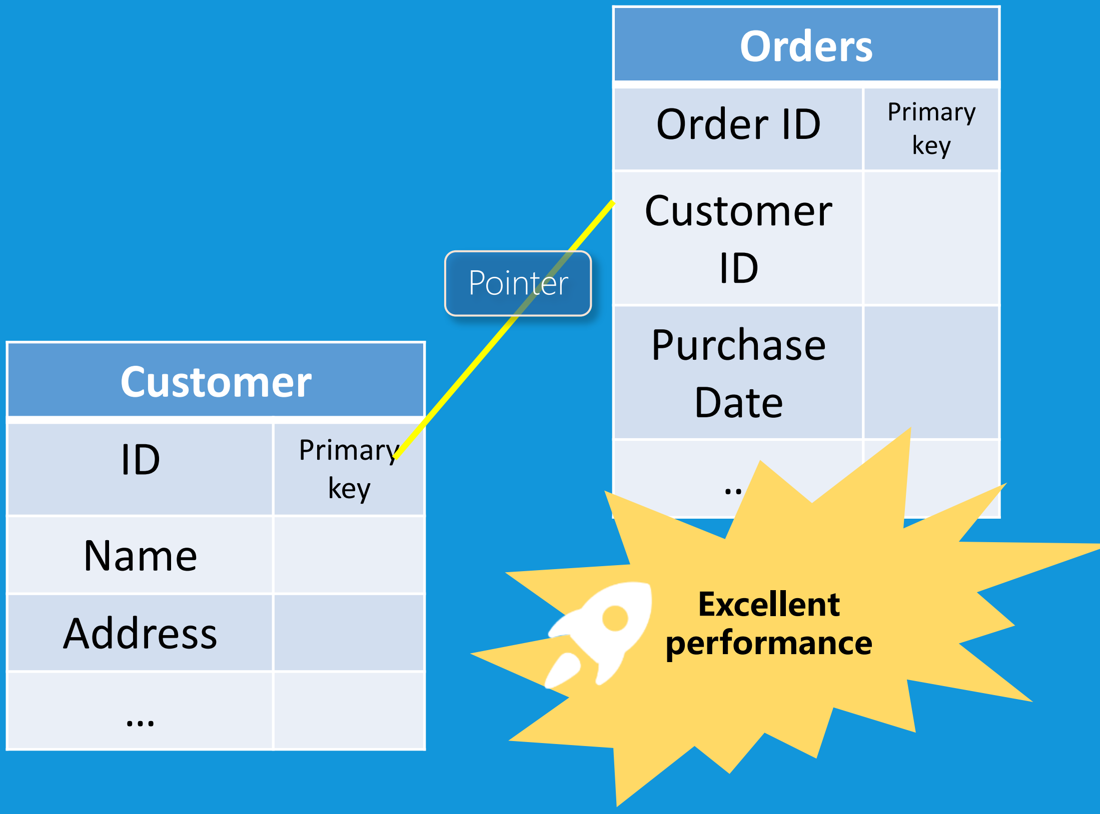
	A
1	=Contract table.select(year(Sales date)==1998)
2	

SPL

The calculation result of A. select () is still a new set of original members. The data is not copied, only the memory record address is copied, which is faster and occupies less memory.

esProc in-memory computing：Pre-association makes JOIN faster





Database JOIN：

Temporarily compute Join at query time

esProc JOIN

Pre-compute Joins
Stored in memory in various ways
There is no need to compute Joins when querying

esProc Hybrid Computing: Implement T+0 Real-time Computing

Current Common Ways and Problems of T+0 Online Computing

1 Historical and current data are stored in the same database

Large amounts of historical data can lead to high database costs (storage costs and performance costs)

2 Historical and current data are stored in separate databases

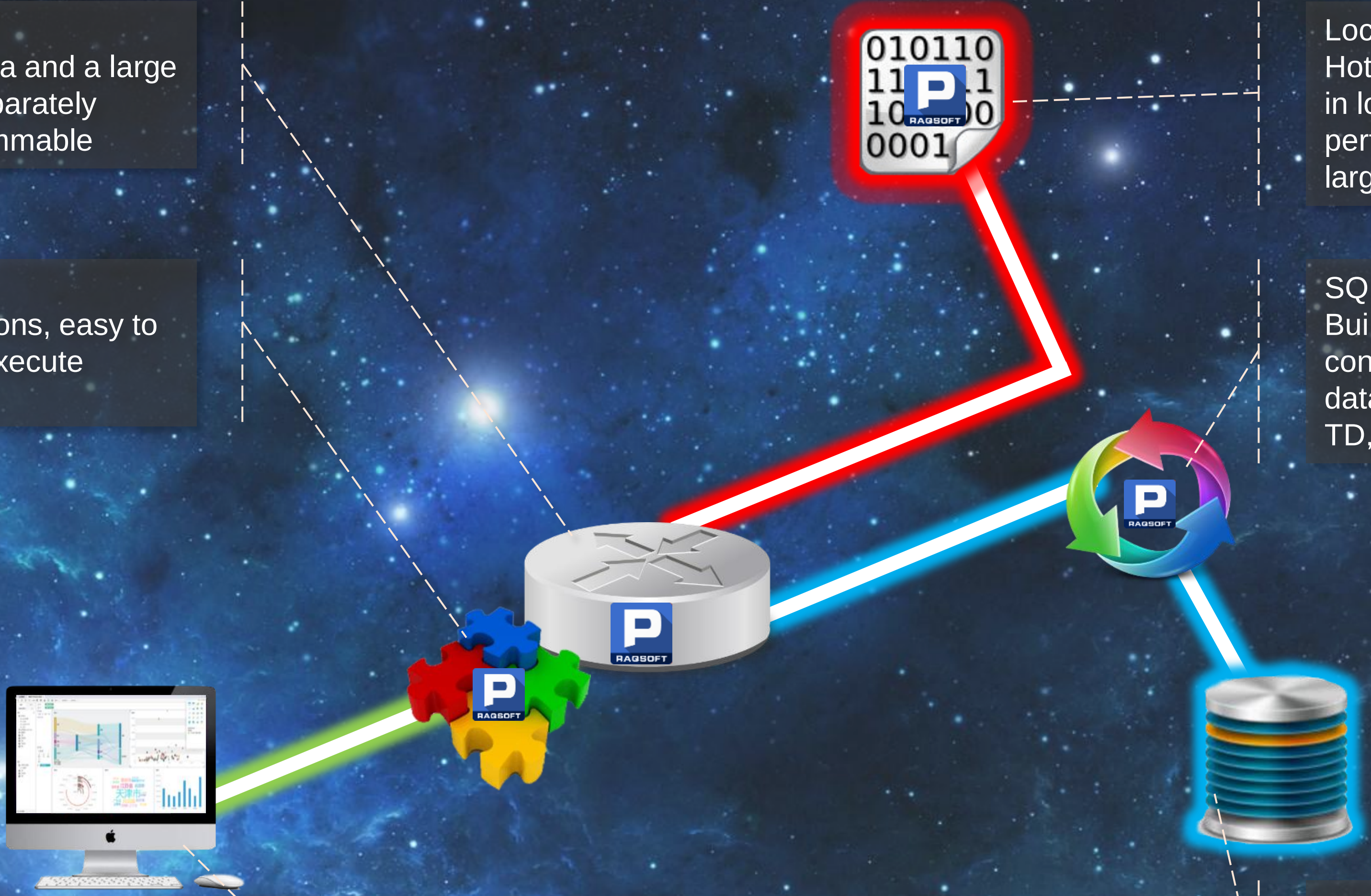
The database is required to have the ability of cross-database calculation, but the implementation complexity is high and the performance is low; when the database type is different, it is difficult to achieve.

- esProc can implement report T+0 query based on multiple heterogeneous databases;
- It can also store historical data in a file system with better IO performance and use cluster computing to achieve higher performance and lower cost.

esProc Programmable Routing

Data Routing
Frequently accessed hot data and a large number of cold data are separately routed, routing rules programmable

SQL parsing
Powerful SQL parsing functions, easy to split SQL clauses, used to execute routing rules



Local High Performance Computing
Hot data completely in memory, warm data in local file storage. esProc high performance computing, fast response to large concurrent access requests

SQL Conversion
Built-in SQL conversion function, converting standard SQL into various database SQL, fully compatible with GP, TD, ORACLE etc.

Front Desk Display
Multidimensional analysis, OLAP, reports, query, large screen, mobile terminal, manage dashboard

Backstage database and data warehouse
A large number of cold data are post-positioned, without the need for the prepositioned database to store the full amount of data repeatedly, and the structure change is very small.

esProc Data Layering Strategy according to Temperature

Hot data: Completely in memory
Completely in memory storage of hot data with high frequency access.Support spare-tire-format memory cluster and lateral expansion, low demand for single-machine memory capacity

Warm data: local files
Warm data accessed in medium frequency is stored in local binary file, which has large capacity and high performance.

Cold data: database, data warehouse
A large number of cold data are post-positioned, the prepositioned database does not need to store the full amount of data repeatedly, and the structure changes very little.


- The end -
THANK YOU