集算器

数据计算中间件

CONTENTS

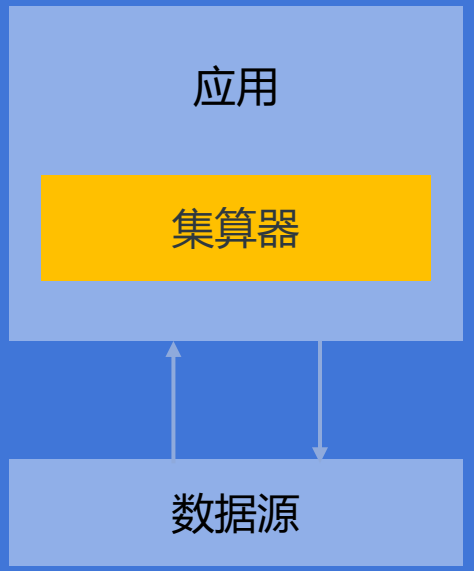
01

集算器是什么

集算器是什么

- 数据计算中间件（DCM），位于数据源和应用之间提供通用数据计算与处理服务
- 独创的SPL（Structured Process Language）语法

- 简洁语法、敏捷开发
- 自有计算能力，不依赖数据库
- 无缝嵌入应用
- 多样数据源支持



集算器解决什么

面向**业务逻辑/微服务开发**、**报表数据准备**等数据计算场景

- 长SQL嵌套N层，存储过程几十K，三个月后自己都看不懂
- Java代码无穷长，写着改着累死人；每次改变要重启，业务用户受不了
- DB/NoSQL/文本/Json/Web几十种数据源，做梦都想跨源混合算
- 数据量大了冷热数据分库后，再想全量（T+0）查难死人
- 过度依赖存储过程，应用架构难调整难扩容
- 数据库里表太多，存储计算资源耗尽，想删删不了
- 报表没完没了做不完，人员成本投入何时休

集算器对标技术

数据库

- 沉重和封闭
- 难以进行跨库/无库运算
- 数据入库费时费力费资源
- 无法嵌入使用
- SQL/存储过程开发调试难

JAVA

- 缺少必要结构化数据计算类库
- 实施计算编码难度高
- 代码冗长，难写难维护
- 不支持热切换

PYTHON

- pandas不是专业结构化数据计算包，复杂运算较为繁琐
- python集成性弱，很难无缝嵌入到应用中

某保险公司-库外存储过程

原来基于Vertica/MySQL等数据源时，应用尽量使用单源；万不得已需要多源混算只能使用Java完成，十分繁琐

用户评价

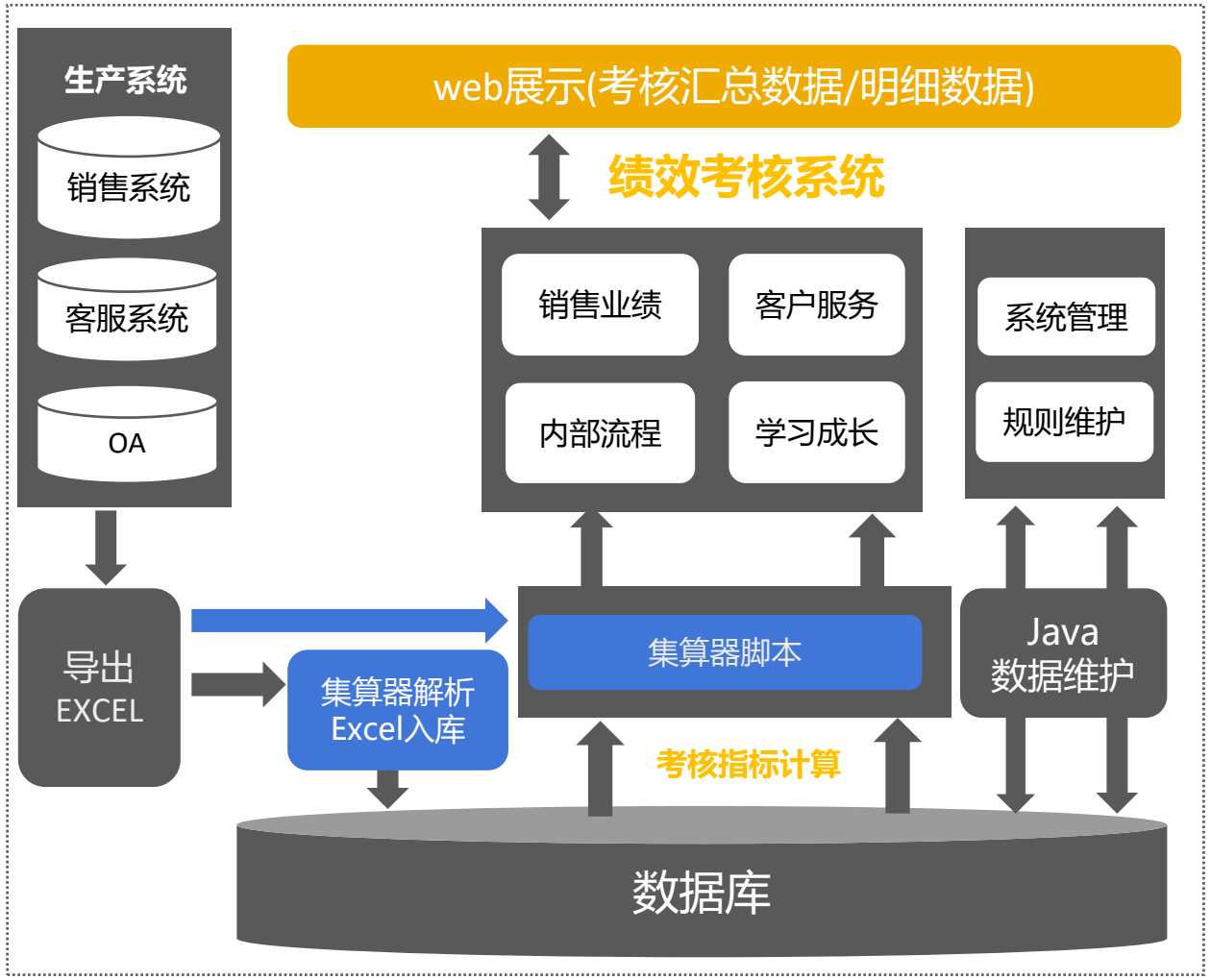
“The best use for us is to pass parameters to the Vertica database.”

“ 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.”

引入集算器后，使Vertica支持存储过程，方便跨源计算

某银行-计算中间件

原来使用Java和存储过程计算考核数据，开发周期长、性能低、维护困难



应用效果

降低开发成本

全部44种Excel解析由原来的30人天下降到6人天

提升开发效率

每种Excel解析代码量由100行变为3行

降低维护成本

除代码量简短易维护外；脚本热部署即修改即生效

某大型产权交易所-数据集市

改造前由于新老系统数据格式、规范不同导致系统间无法互通

实现系统间数据互通，统一管理，计算复用

应用效果

✓ **异构数据源清洗**

多源混算基于多个异构源同时ETL

✓ **文件系统存储**

文件存储节约数据库成本，更灵活

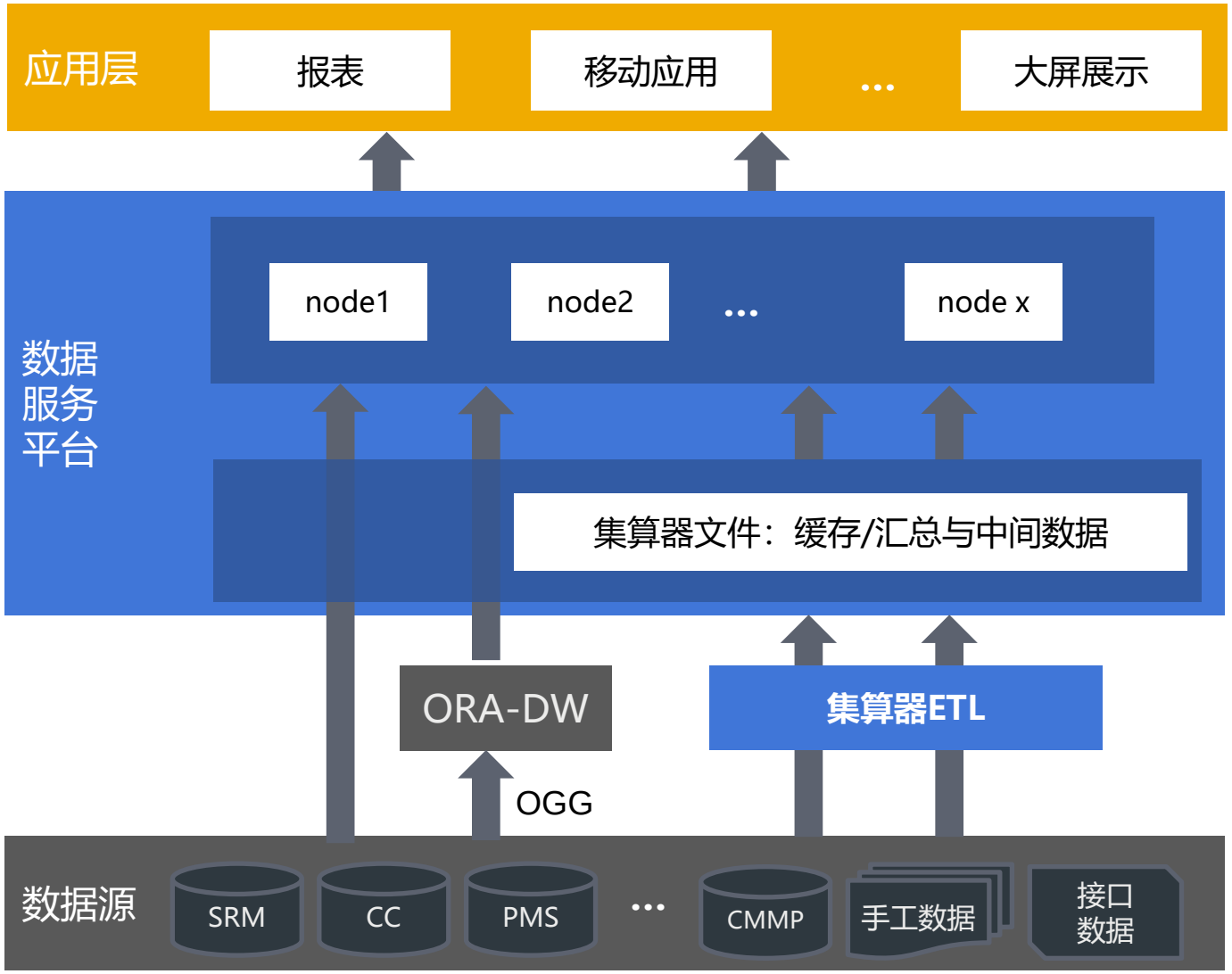
✓ **计算逻辑复用**

一个业务逻辑统一计算避免不一致

✓ **易于横向扩容**

节点可横向扩展提升算力

某航空-数据服务平台



✓ **统一数据管理**

屏蔽底层数据差异，规范了数据管理

✓ **多源混合计算**

多数据源直接混算无需同库

✓ **历史数据归档**

文件存储历史数据，基于文件分析

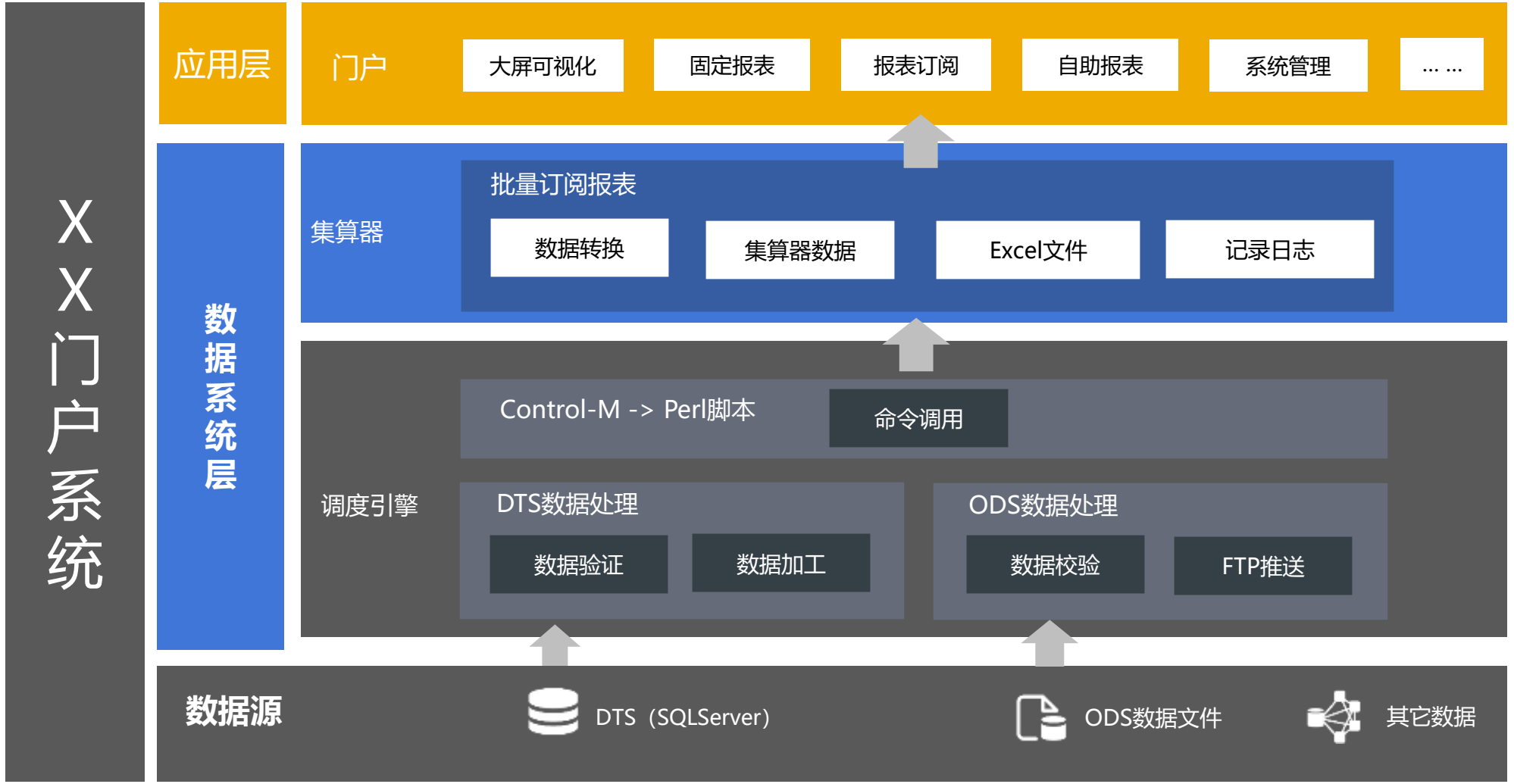
✓ **提升计算性能**

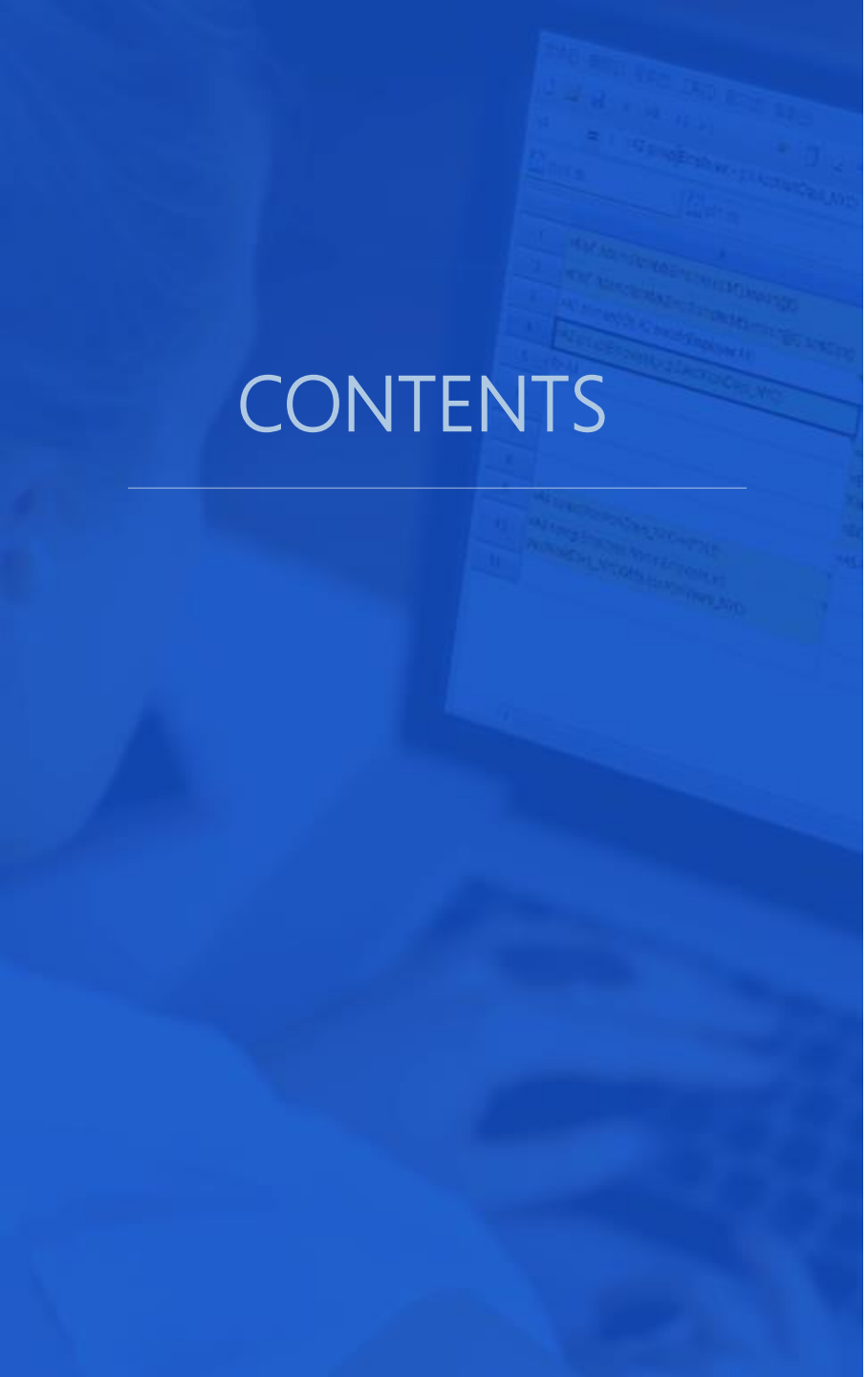
数据计算性能提升了5-20倍

✓ **易于横向扩容**

节点横向扩展提升算力

金融业-订阅系统


XX门户系统
应用层
门户
大屏可视化
固定报表
报表订阅
自助报表
系统管理
… …
数据系统层
集算器
批量订阅报表
数据转换
集算器数据
Excel文件
记录日志
调度引擎
Control-M -> Perl脚本
命令调用
DTS数据处理
数据验证
数据加工
ODS数据处理
数据校验
FTP推送
数据源
DTS（SQLServer）
ODS数据文件
其它数据



CONTENTS

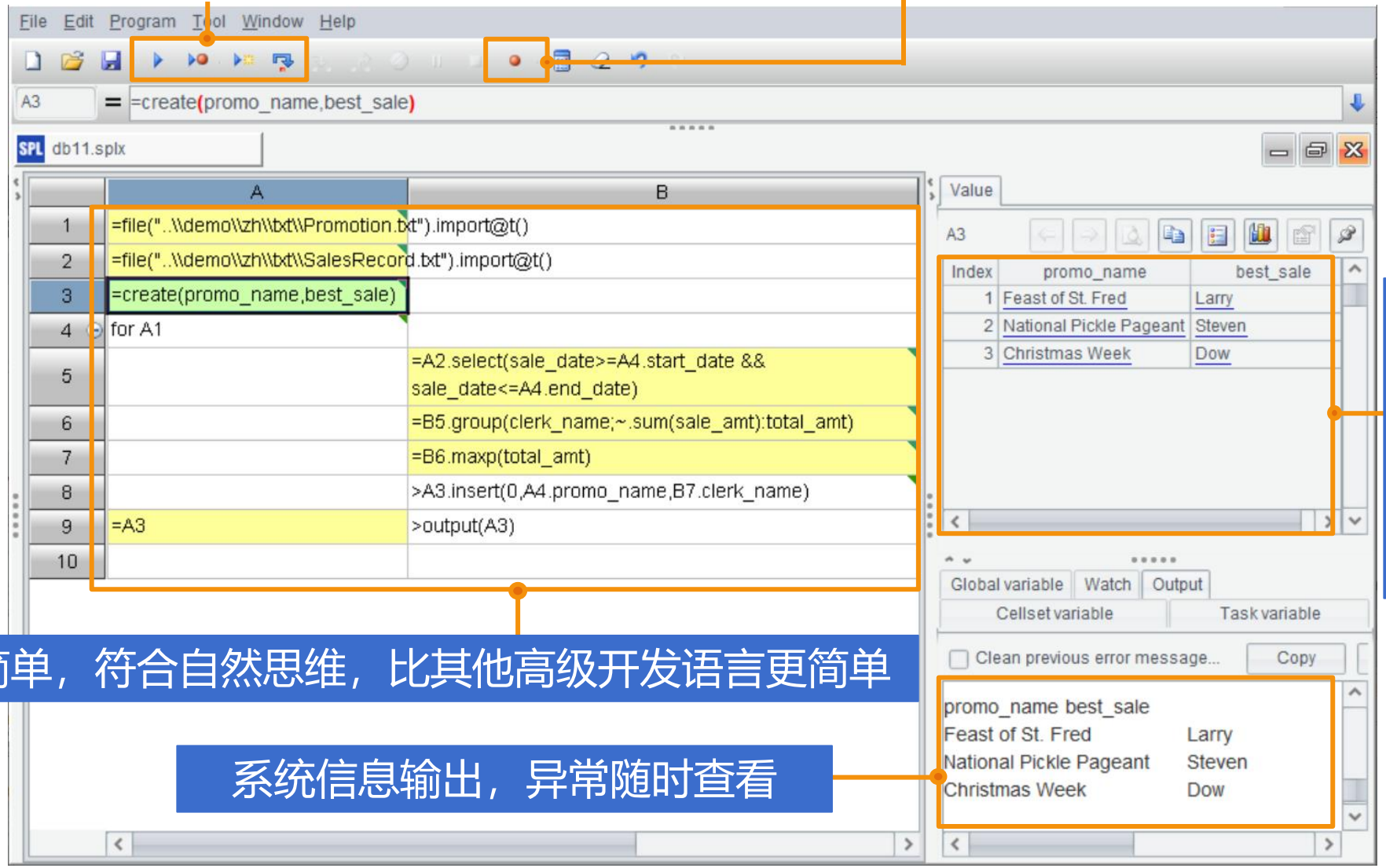
02

技术特性

简洁易用的开发环境

执行、调试执行、执行到光标、单步执行

设置断点



网格结果所见即所得，易于调试；方便引用中间结果

语法简单，符合自然思维，比其他高级开发语言更简单

系统信息输出，异常随时查看

敏捷语法

计算目标：某支股票最长连续涨了多少交易日

```
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解法

- SQL在使用窗口函数的情况下嵌套三层完成；
- 前面读懂了吗？

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

SPL解法

其实这个计算很简单，按照自然思维：先按交易日排序（行1），然后比较当天收盘价比前一天高就+1，否则就清零，最后求个最大值（行3）

专门设计的语法体系

SPL特别适合复杂过程运算

	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.c	umber			
12		=B10.count(~>=10000)	/number of big customer			
13		=B12/B11				
14		=A5.insert(0,B6,B7,B8,B9,B11,B12,B13)				

天然分步、层次清晰、直接引用单元格名**无需定义变量**

丰富的运算类库

专门针对结构化数据表设计

	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(~.sum(amount))	
8		=B7.count()	
9		=B7,count(~>=10000)	

分组、循环

	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			

集合运算

	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			

排序、过滤

	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/c		
9			
10			

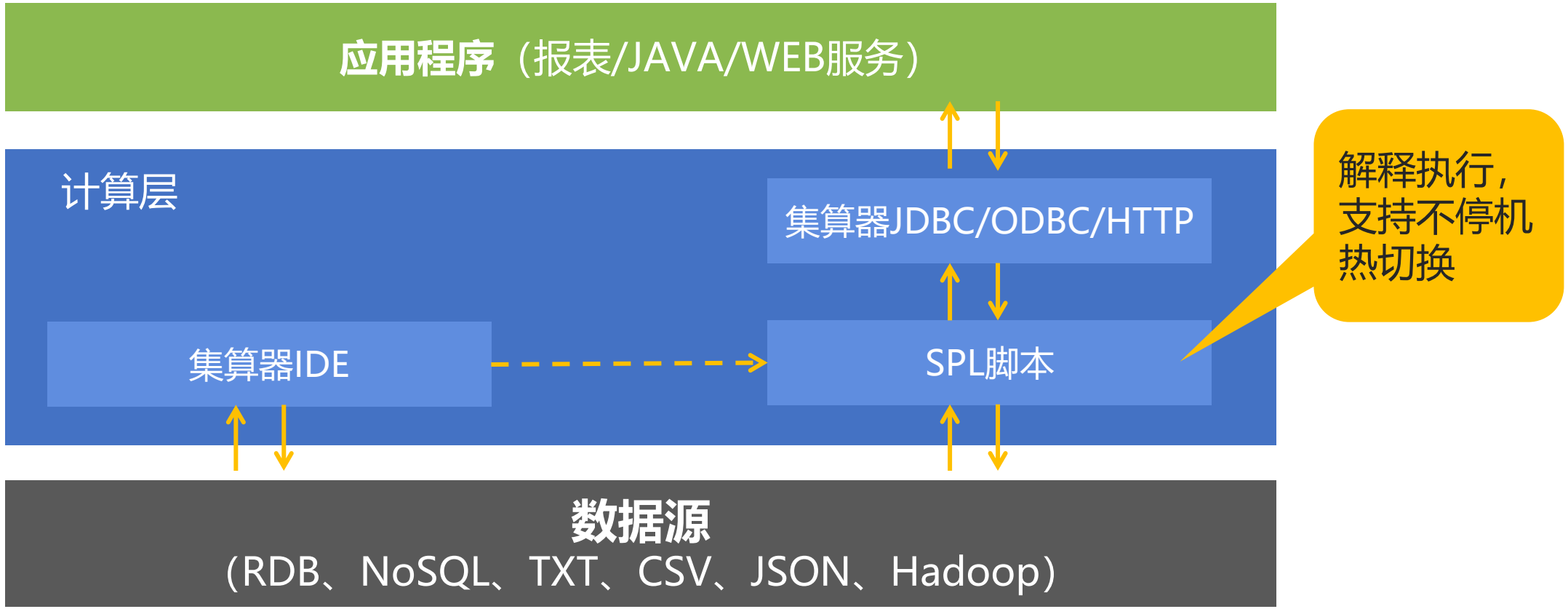
有序集合

多样性数据源

- 直接使用多个数据源混合计算，无需后台先将数据统一（ETL）后再计算
- 支持SQL查询文件、NoSQL等数据源

集成性

集算器使用JAVA开发，提供标准应用接口可无缝集成到应用中



DCM评价标准

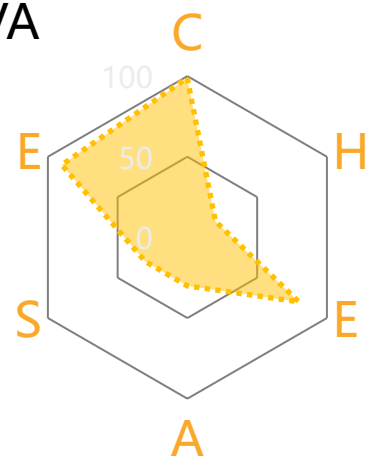
CHEASE

C	H	E	A	S	E
兼容性	热部署	高性能	敏捷性	扩展性	集成性
Compatible	Hot-deploy	Efficient	Agile	Scalable	Embeddable

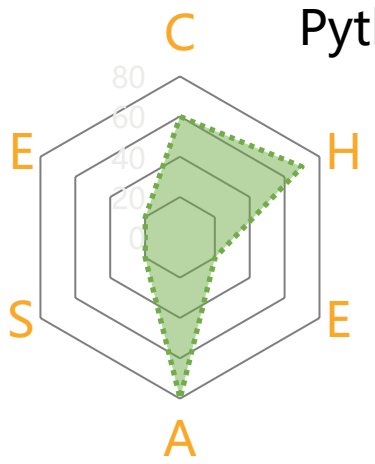
延伸阅读：DCM：中间件家族迎来新成员

集算器SPL能力圈

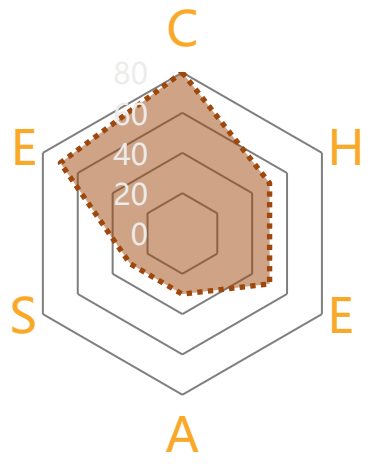
JAVA



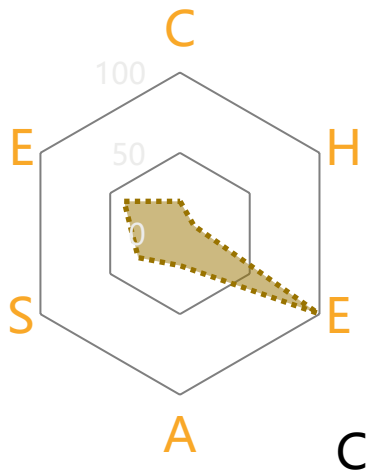
Python

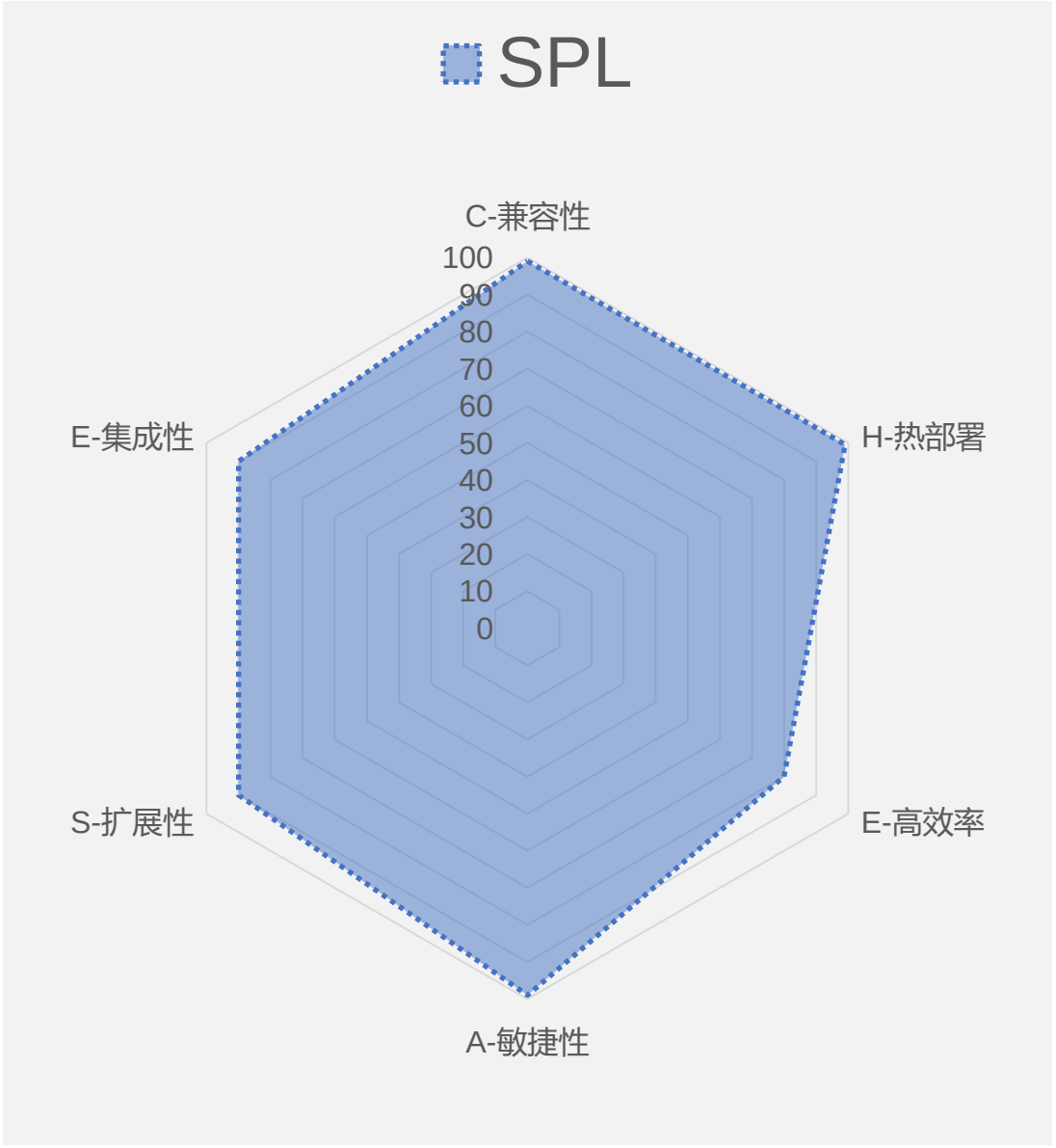


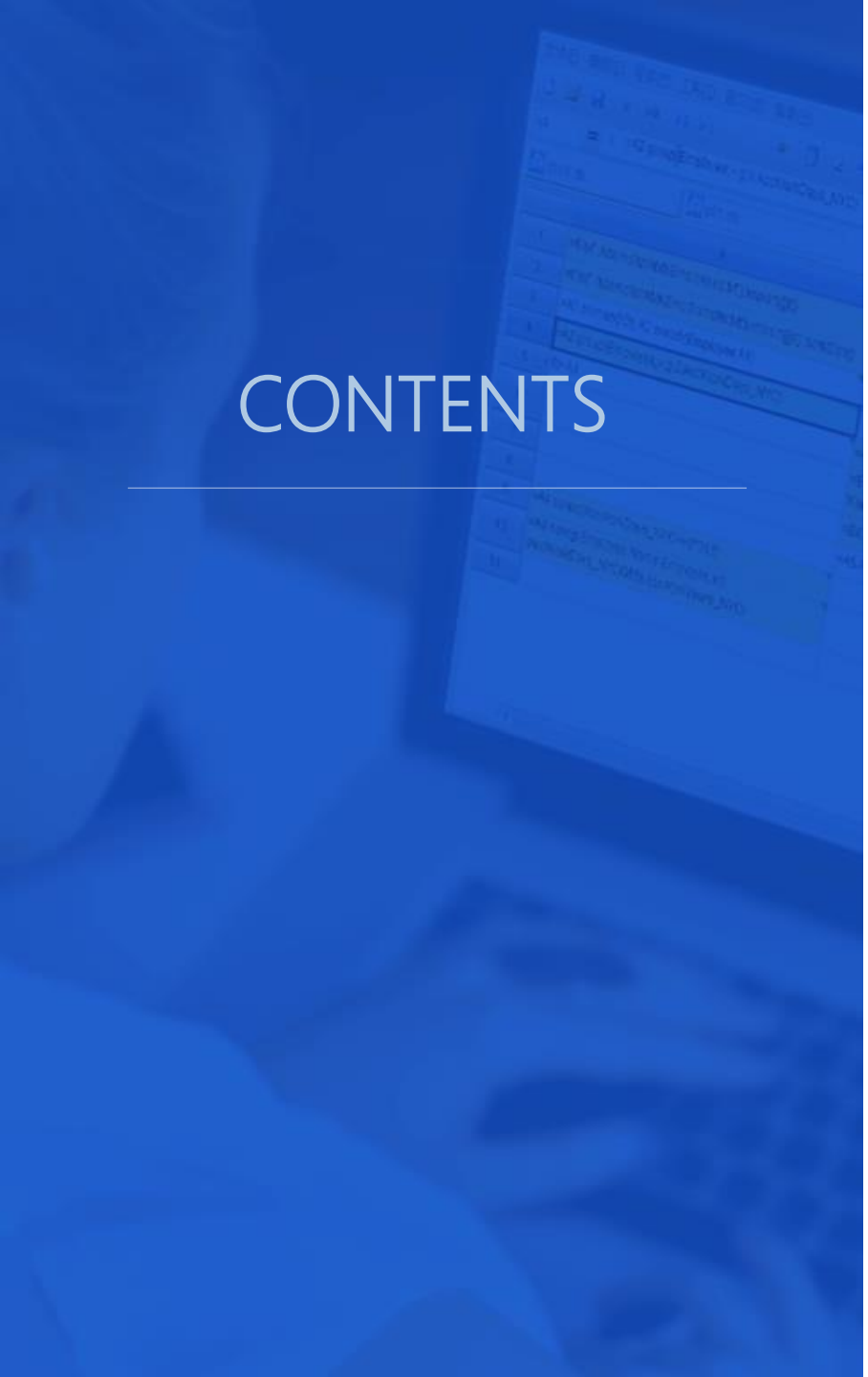
SQL



C++







CONTENTS

02

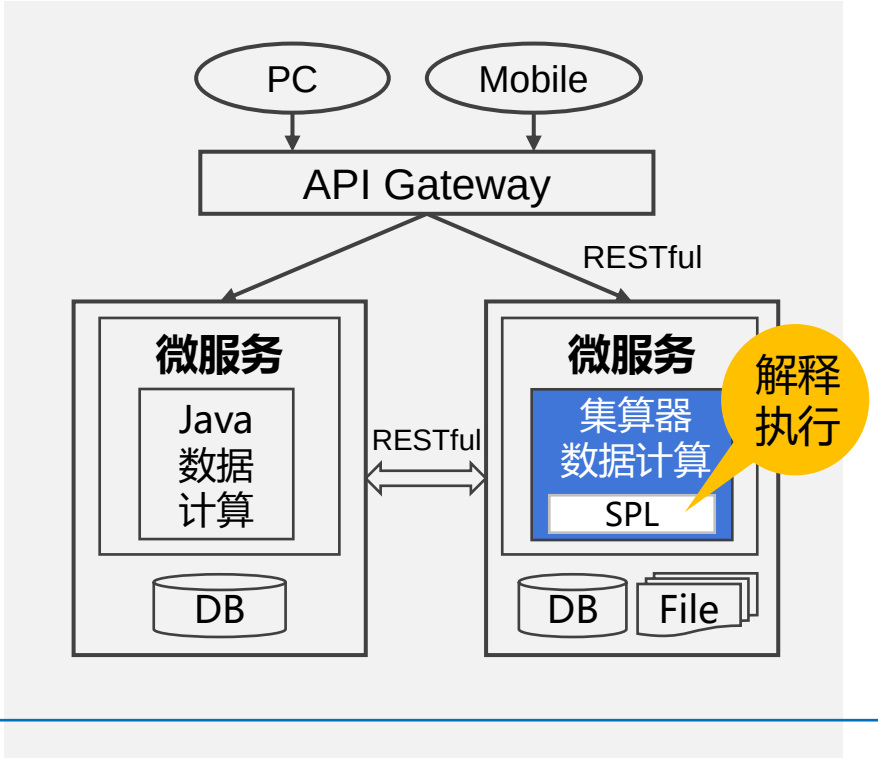
疼点解决

替代JAVA/ORM实现业务逻辑

现状

- 微服务等主流架构要求在应用端实施数据处理
- 数据库很难嵌入前端应用使用，只能硬编码
- JAVA缺乏足够的结构化计算类库导致数据处理开发困难，无法热切换

解决方案



- ✓ SPL替代JAVA/ORM在（微服务）应用内实施数据计算
- ✓ 丰富计算类库与敏捷语法简化开发
- ✓ 体系开放可以实时处理任意数据源数据
- ✓ SPL解释执行，天然支持热切换
- ✓ 高效算法与并行机制保证计算性能

延伸阅读：开源 SPL，ORM 的终结者

多样性数据源及混合计算

现状

- 企业数据源种类越来越多，经常需要跨源混合查询
- 数据库跨数据源混合计算功能弱（通常仅支持同构），性能低
- 应用端硬编码实现繁琐，后续计算困难

解决方案

- ✓ SPL支持多数据源混合计算
- ✓ 支持SPL和SQL语法
- ✓ 无需物理同库，数据实时性强
- ✓ 同构/异构数据源实时混算
- ✓ 可充分发挥各类数据源自身的优势
- ✓ 嵌入应用内使用，为应用提供多源混算能力

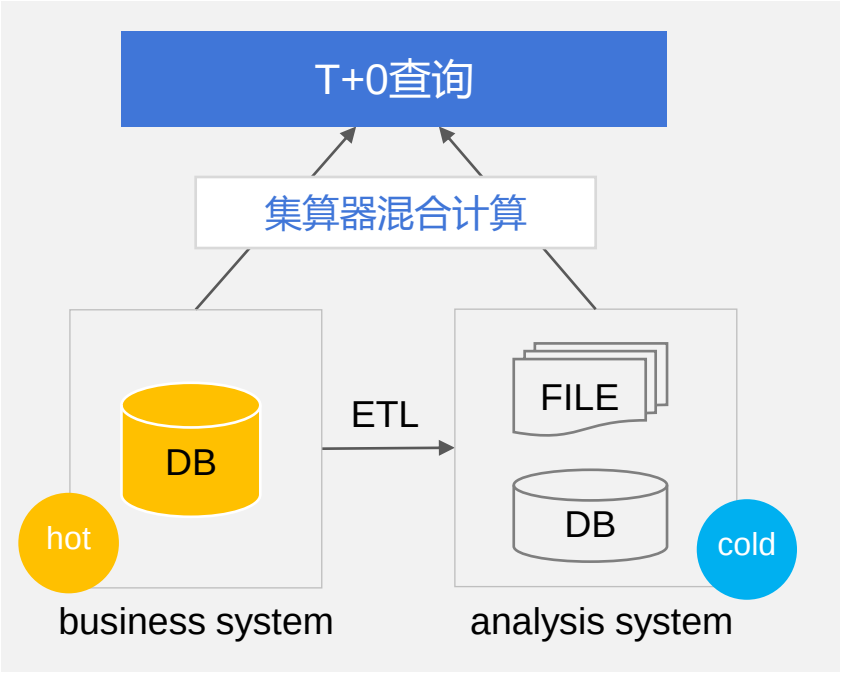
延伸阅读：多数据源混合计算用什么技术

实现T+0查询

现状

- 使用生产库进行交易和查询随着数据量增大会影响业务
- 冷热数据分离（分库）后只能查询T+N的数据，很难进行全量T+0查询
- T+0需要但不能被满足，业务体验差

解决方案



- ✓ 借助SPL多源混算能力从冷热数据源同时取数计算实现T+0
- ✓ 支持异构源实现
- ✓ 冷数据可以存储在文件系统中，可获得更高计算性能和更低存储成本
- ✓ 利用SPL的文件计算能力实施冷热数据混合计算

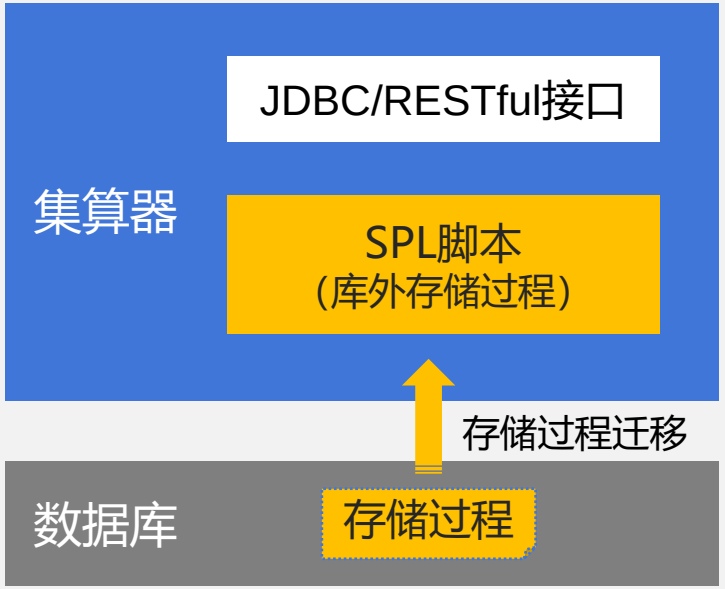
延伸阅读：开源 SPL 轻松应对 T+0

替代存储过程

现状

- 存储过程编辑调试困难，缺乏移植性，
- 编译存储过程对权限要求过高，安全性差
- 存储过程被多个应用共用还会造成应用间紧耦合

解决方案



- ✓ SPL天然适合复杂多步数据计算
- ✓ SPL脚本天然可移植
- ✓ 脚本只要求数据库的读权限，不会造成数据库案例问题
- ✓ 不同应用的脚本分别存储于不同目录，不会造成应用间耦合

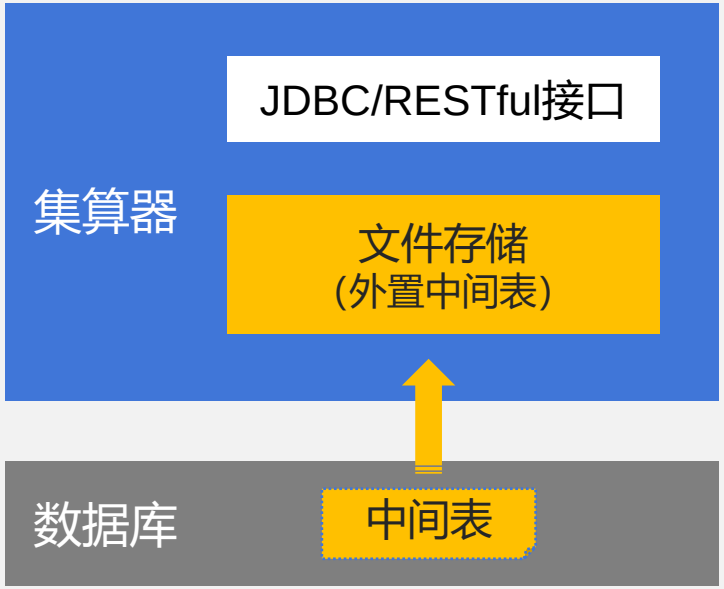
延伸阅读：爱恨交加的存储过程该往何处去

消灭数据库中间表

现状

- 为了查询效率或简化开发会在数据库中生成大量中间表
- 中间表空间占用大，导致数据库过于冗余臃肿
- 不同应用使用同一个中间表会造成紧耦合，中间表管理困难（很难删除）

解决方案



- ✓ 中间表存储在数据库中是为了利用数据库进行后续计算，使用文件存储后使用SPL实施后续计算
- ✓ 外置中间表（文件）更易于管理，采用不同目录存储不会引起应用耦合性问题
- ✓ 中间表外置可以为数据库充分减负

延伸阅读：开源 SPL 消灭数以万计的数据库中间表

应对报表没完没了

现状

- 报表/BI工具只能解决呈现环节的问题，对数据准备无能为力
- 数据准备使用SQL/存储过程/JAVA硬编码，开发维护困难，成本高
- 报表需求客观上没完没了，数据准备是导致高开发成本的主要因素

解决方案

- ✓ 在报表呈现和数据源之间增加计算层来解决数据准备问题
- ✓ SPL简化报表数据准备，弥补报表工具计算能力不足，全面提升报表开发效率
- ✓ 报表呈现和数据准备两个环节都能快速响应以低成本应对没完没了的报表需求

延伸阅读：开源 SPL 优化报表应用应对没完没了


THANKS