润乾
RAQSOFT

报表提效-人无我独有

润乾软件出品

生活总是这样无奈吗

需求多变，报表总也做不完，累！

- 很多技术力量投入报表开发，却还是疲于应付
- 技术高手用来做报表，感觉很浪费
- 缺少低成本高效率的应对方案

报表工具花了不少钱

开发效率怎么还这么差

大牌敏捷BI系统也上过N套

竟然无济于事

报表为什么做起来这么辛苦

亲，报表为什么会没完没了？

报表的业务稳定性天生很差，业务开展会不断刺激出新的查询统计需求

那就没有办法消灭没完没了的问题吗？

没有！报表多变常态，无法消除，只能适应；但可以利用工具实施低成本应对方法!

那为什么上了专业报表工具、敏捷BI还是搞不定？

一般报表工具只能搞定呈现环节，但消耗工作量更大却是数据准备环节，报表工具通常就无能为力了!!!

那么数据准备怎么办？

也要工具化！全方位简化报表开发后，才能做到低成本高效率应对!

为什么呢?

图解：报表工具只能搞定报表呈现环节，但搞不定数据准备环节

报表层

报表呈现 → **报表工具**

数据准备 SQL/存储过程/JAVA → **硬编码** →

数据源层

传统报表应用结构

- 没有普适工具，编写维护困难
- 专业编码，对人员要求高
- 实现周期长，难以适应多变需求
- 耦合性高，无法独立开发
- 运维复杂，修改报表常常要重启应用

表急，
看解决方案
！

解决方案

在报表工具中增加数据准备环节，使得报表开发彻底工具化，实现高效率！

报表开发工具化

报表绘制与数据准备两个环境都可以使用报表工具开发

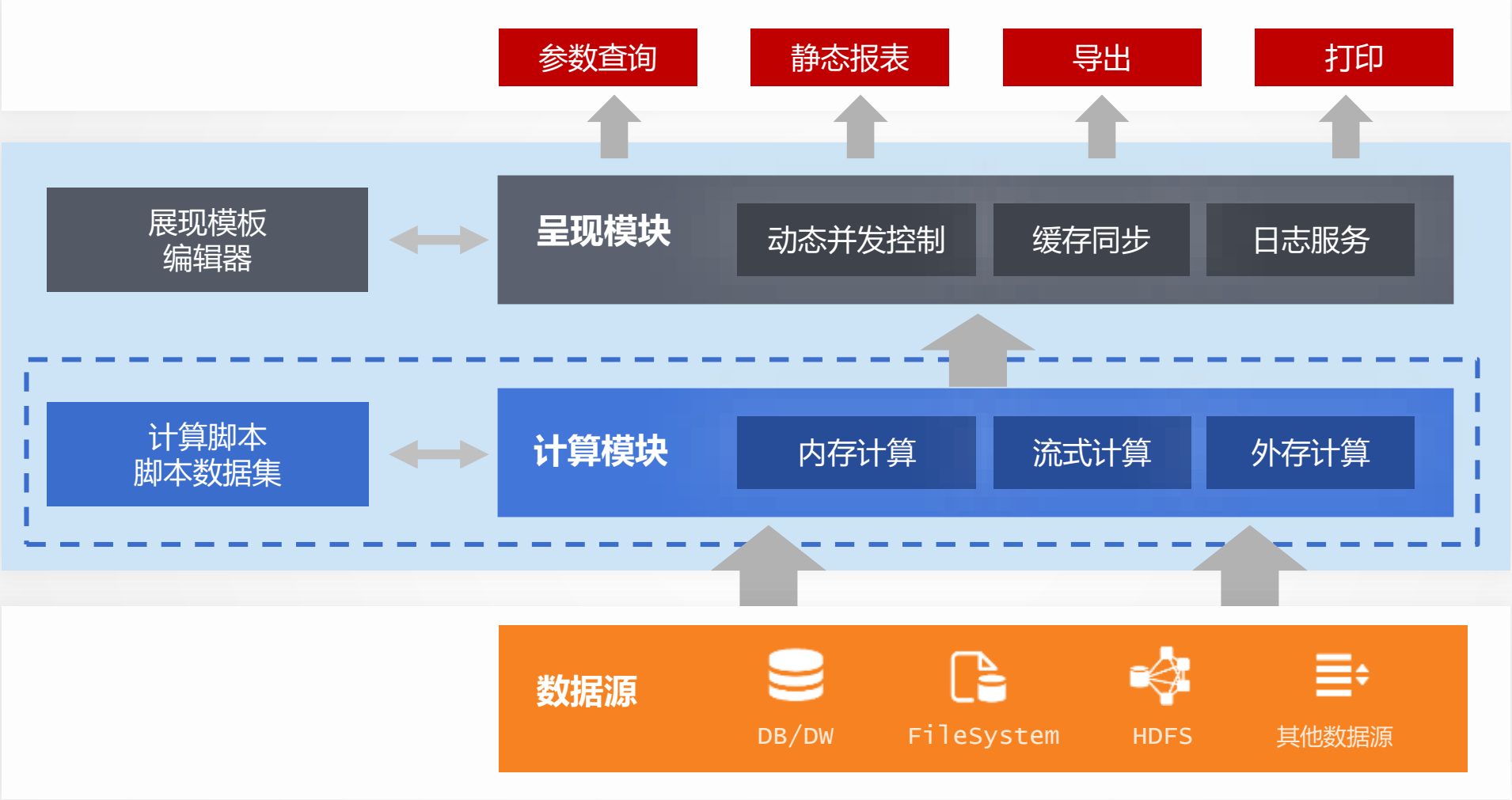
报表模块独立化

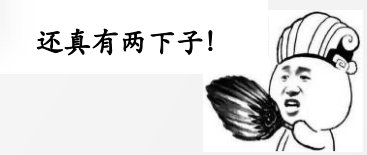
报表呈现及数据准备工作从应用程序和数据库中独立出来，单独维护，与应用解耦

开发人员广泛化

一般技术人员都能做，成本低，响应快

报表应用结构





报表模板（计算和呈现）置于应用系统之外，可独立团队开发，报表实施过程全面工具化

引入计算模块后

降低开发难度

提高运算性能

优化应用结构

比JAVA的优势

相对使用JAVA实现数据准备，润乾报表的计算模块（脚本）采用集合化语法，代码要比没有直接提供结构化计算的JAVA更加短小

写的更快更短

- 脚本基于Java提供了更高层的类库和方法

容易理解和排错

- 伪实代码的比例大约是只有1:1.5，大多数报表数据准备算法可以在一个屏幕内显示出来
- 一个页面内能看到更多代码，能更完整地理解代码的含义与排错

比SQL/存储过程的优势

某支股票最长连续涨了多少交易日

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

计算脚本

```
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

语法体系更容易描述人的自然思维！

思考：按照自然思维怎么做？

多样性数据源支持

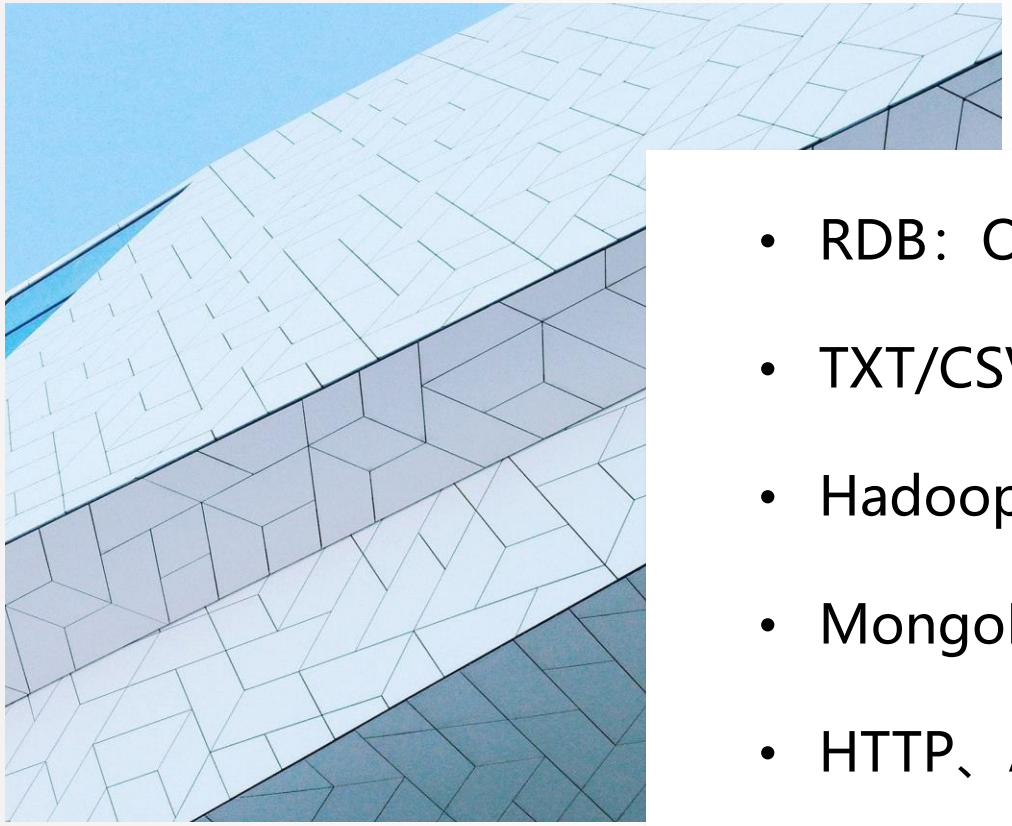
引入计算模块后，报表工具的计算能力和容量都能胜任多样性数据源

计算模块处理多样性数据源

无需入库，减少工作步骤

支持多层数据格式，减少开发工作量

多源支持与混算



- RDB：Oracle,DB2,MS SQL,MySQL,PG,....
- TXT/CSV，JSON/XML，EXCEL
- Hadoop：HDFS，HIVE，HBASE，SPARK
- MongoDB，REDIS，...
- HTTP、ALI-OTS
- ES，Kafka
- .....

引入计算模块后

降低开发难度

提高运算性能

优化应用结构

并行取数

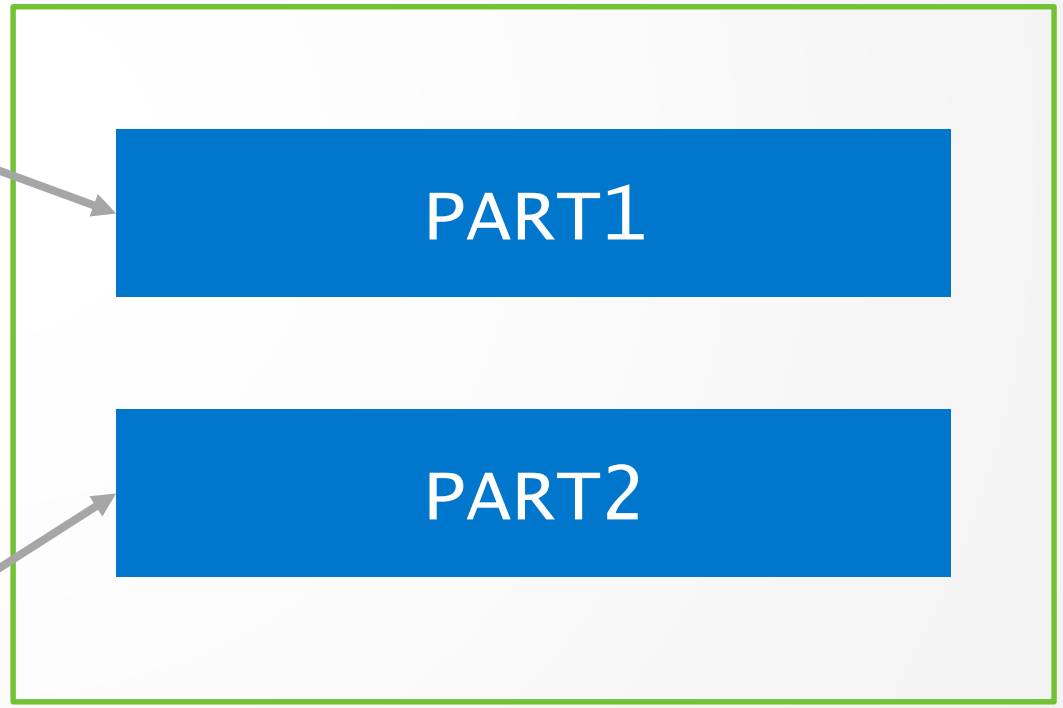
数据库JDBC性能较差，报表性能又严重依赖于取数环节；润乾报表可以采用多线程并行的方式同时建立多个数据库连接从数据库分段取数，可以获得数倍性能提升

	A	B	C
1	fork 4	=connect(db)	/分4线程，要分别建立连接
2		=B1.query@x(“select * from T where part=?”,A1)	/分别取每一段
3	=A1.conj()		/合并结果

控制SQL执行路径

- 数据库的透明性为用户带来方便的同时，使得优化SQL执行路径非常困难
- 润乾报表可以自由控制执行路径，部分运算可以移出数据库实施，方便完成性能优化

```
select t.*
 from (select *
        from (select syb.org_abbreviation as syb,
                max(xmb.org_abbreviation) as
xmb,
                sub.org_subjection_id as sub_id,
......
 group by l.requisition_id,
                l.note,
                l.requisition_type,
......
 left join cpl_rwdmx_view ve
        on ve.requisition_id = a.req_id) t
```

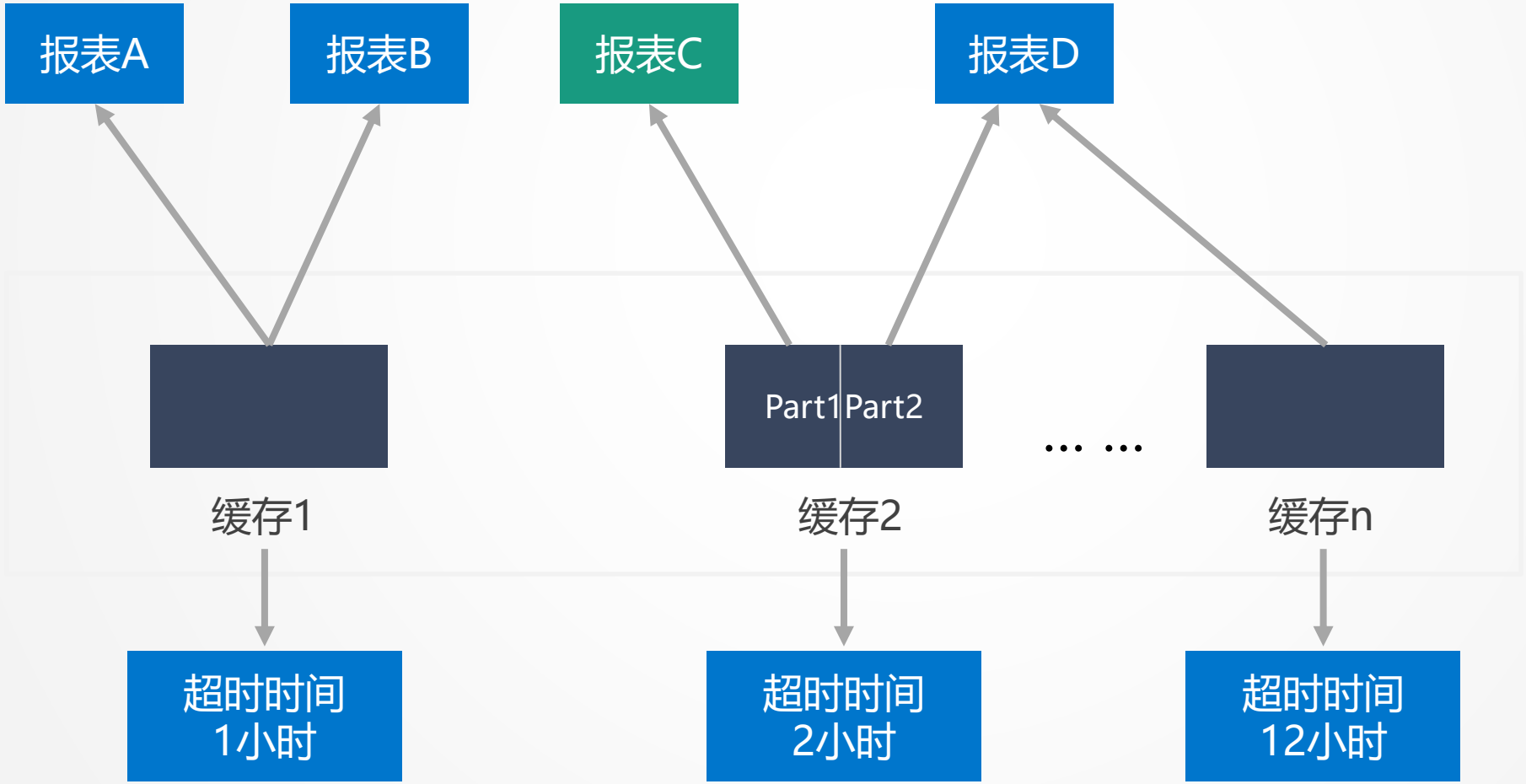


SQL – 442秒

计算脚本+SQL – 41秒

可控缓存

润乾可以实现报表的**部分缓存**、多个报表之间**缓存复用**、以及不同缓存的**不同生存周期**



海量大清单报表

传统报表模式

全部取出后再呈现，用户体验恶劣

全内存计算的报表可能溢出

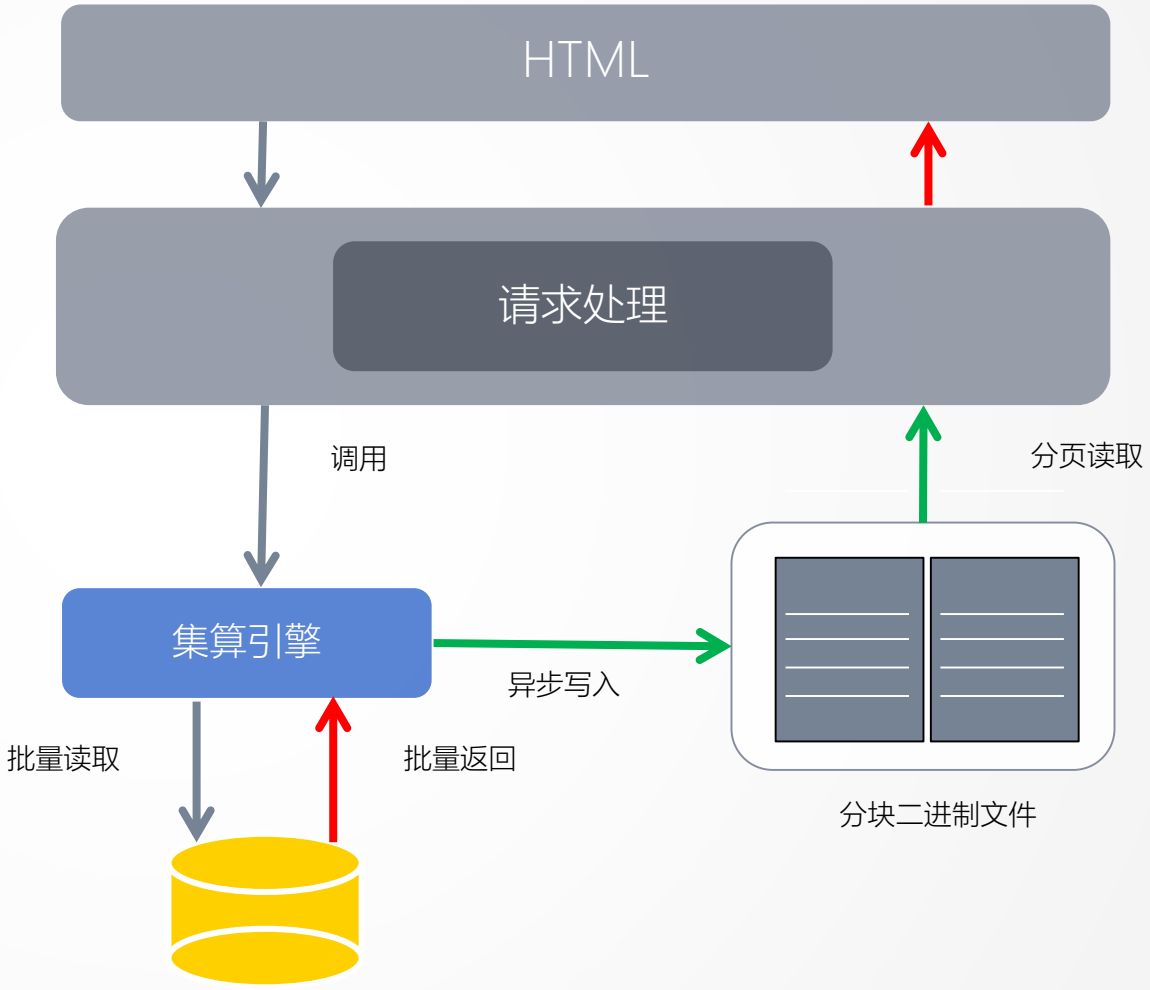
由数据库实现分页取出，性能很差

润乾报表实现异步分页取出

异步线程边取数边呈现，快速响应

本地缓存，按页随机取数

不依赖于数据库的分布取出能力



引入计算模块后

降低开发难度

提高运算性能

优化应用结构

解释执行降低应用耦合度

使用JAVA和计算模块（脚本）做报表数据源会有以下不同：

JAVA

模块化困难

Java程序必须和主应用一起编译打包，耦合度高

难以热切换

Java编写的报表数据准备算法有修改后会导致整个应用重新编译部署，很难做到热切换

类库少

JAVA程序在结构化和半结构化计算方面的类库较少

计算脚本

模块化简单

计算脚本和报表模板一起管理维护，从而使报表功能模块化

容易热切换

计算脚本是解释执行的语言，很容易做到热切换

类库多

更丰富的语法和类库，让结构化数据的计算更有效率

算法外置减少存储过程

采用存储过程实现数据准备算法，会造成报表与数据库的耦合问题

存储过程和报表的存放位置不同，导致对应难度很大

存储过程修改需要分配相应的数据库权限，存在安全隐患

存储过程容易被其他应用使用，造成多个应用间的耦合

使用计算脚本替代存储过程完成报表数据准备，会极大减少存储过程，算法外置后与报表模板一起存放管理，完全归属于应用本身，降低报表与应用其他部分或其他应用的耦合

数据外置减少中间表

由于数据量或计算复杂度原因，经常需要在数据库中创建中间表，中间表会带来如下问题：

数据库管理混乱

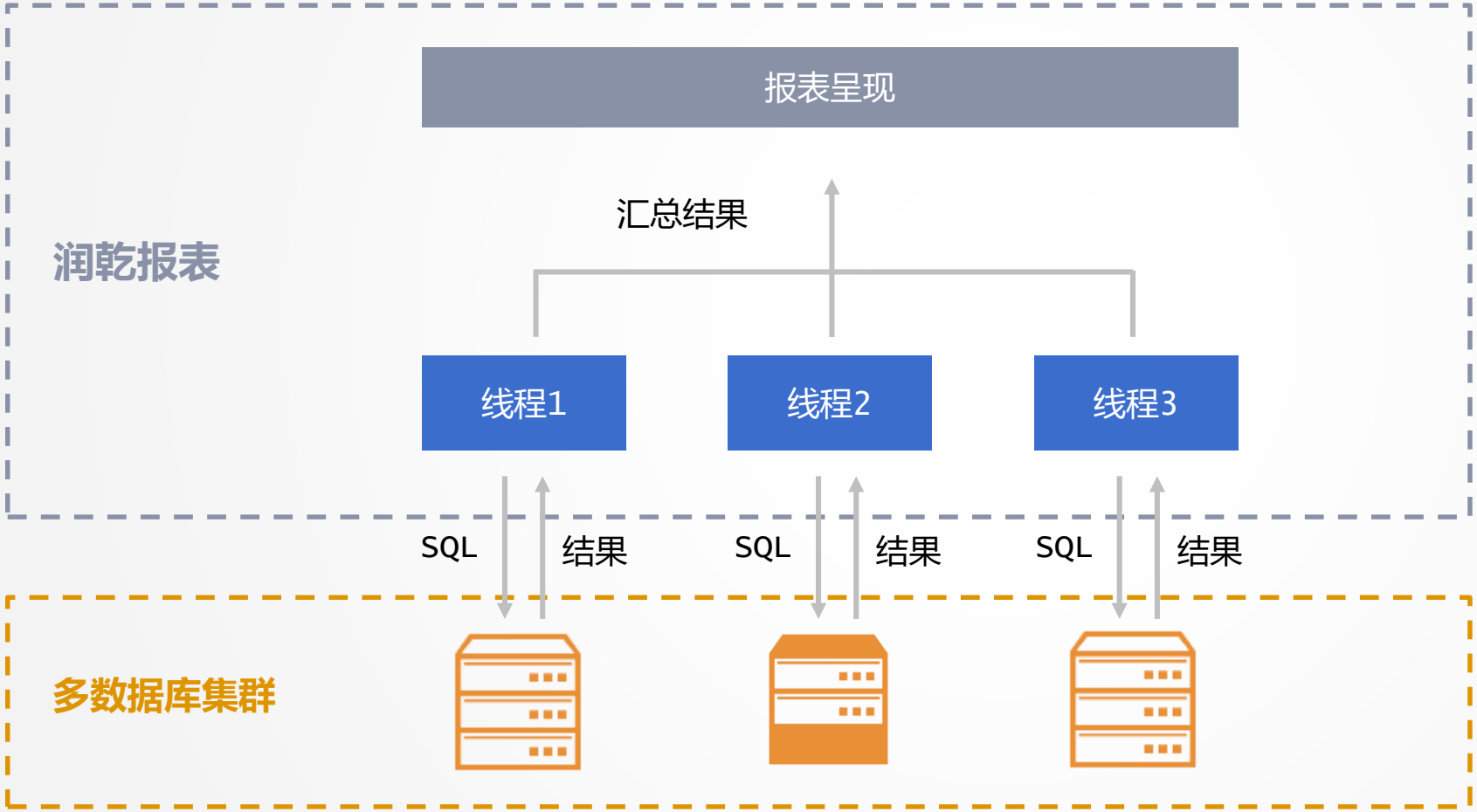
各个应用的累积大量中间表存储在线性结构的数据库中造成数据库管理混乱

数据库资源浪费

不用的中间表，仍然需要相应ETL过程向其更新数据，浪费数据库资源

将中间表数据外置存放到文件系统，便于管理，通过计算脚本获得计算能力，可以获得更高效的IO性能和计算能力，充分减少数据库中间表，梳理数据库结构

并行取数与跨库计算



润乾报表计算模块可以基于多库（同构异构均可）实施并行取数和跨库运算

彻底解决报表开发困难的征途

累积报表经验

建设报表知识库论坛。减少沟通中的误解，实现历史工作的复用

建设报表团队

模块独立后可建设培训独立于应用开发团队外的低成本报表队伍应对多变需求

独立报表模块

调整应用结构，梳理报表数据源，实现报表模块与数据库及应用解耦

加入计算模块

数据计算工具化
解放数据准备阶段人力

引入报表工具

呈现工具化
解放报表呈现阶段人力

创新技术

推动应用进步！