



StarRocks MV AMA

2023.12



物化视图基础概念

物化视图概念与价值

- 维基百科的定义： a **materialized view** is a database object that contains the results of a query.

物化视图字面上有两层含义

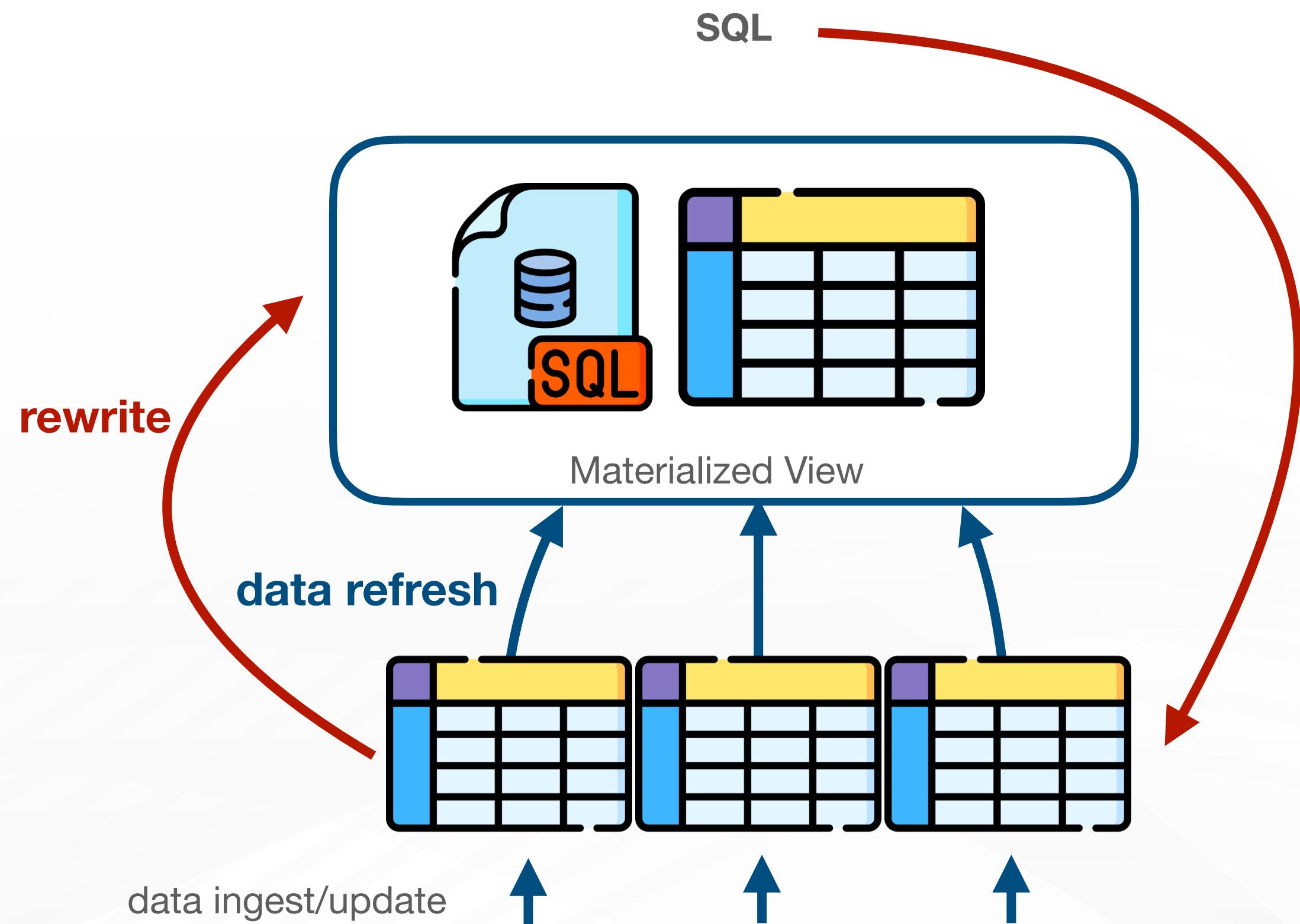
- “Materialized” 物化意味着物化加速
- “View” 表示建模

StarRocks 物化视图的关键技术

- 预计算：
 - 内表/外表/嵌套
 - 单表/多表
- 刷新：
 - 分区级别增量
 - 定时/手动/触发
- 改写：
 - 优化器自动查询改写

```

CREATE MATERIALIZED VIEW mv1
PARTITION BY dt
REFRESH EVERY (INTERVAL 1 HOUR)
PROPERTIES("resource_group" = "rg1")
AS
SELECT c_city, count(*)
FROM lineorder t1
JOIN customer t2
ON t1.lo_custkey = t2.c_custkey GROUP BY c_city
  
```



数据分层建模

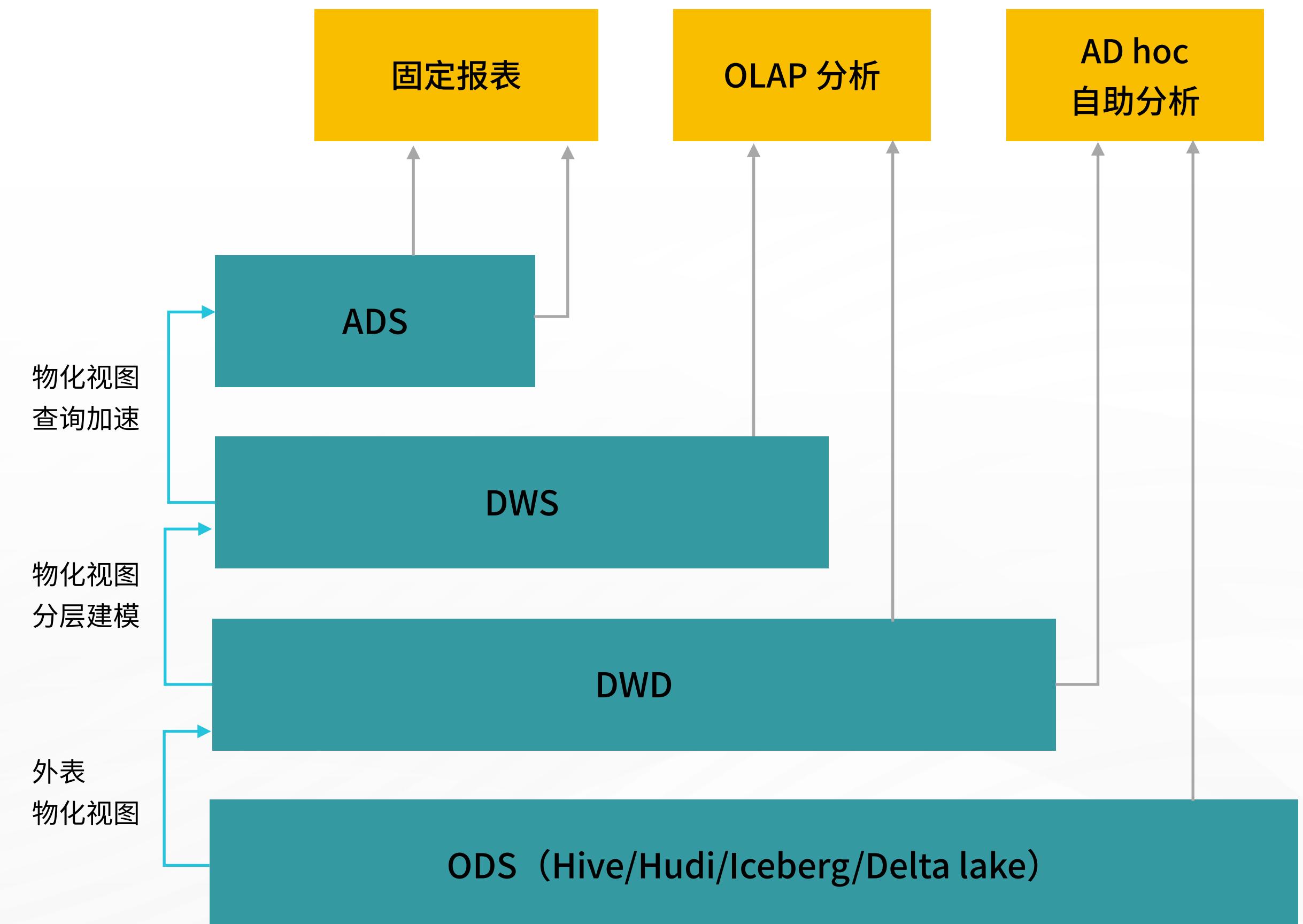
- 简化数据加工，无需维护外部组件
- 自动管理数据间依赖，分区粒度刷新

透明查询加速

- 自动利用物化视图查询改写透明加速
- 支持 Aggregate、Join、Union 等查询

湖仓融合

- 基于外部 Catalog 建立物化视图
- 免去复杂的 ETL 数据准备工作





物化视图典型场景

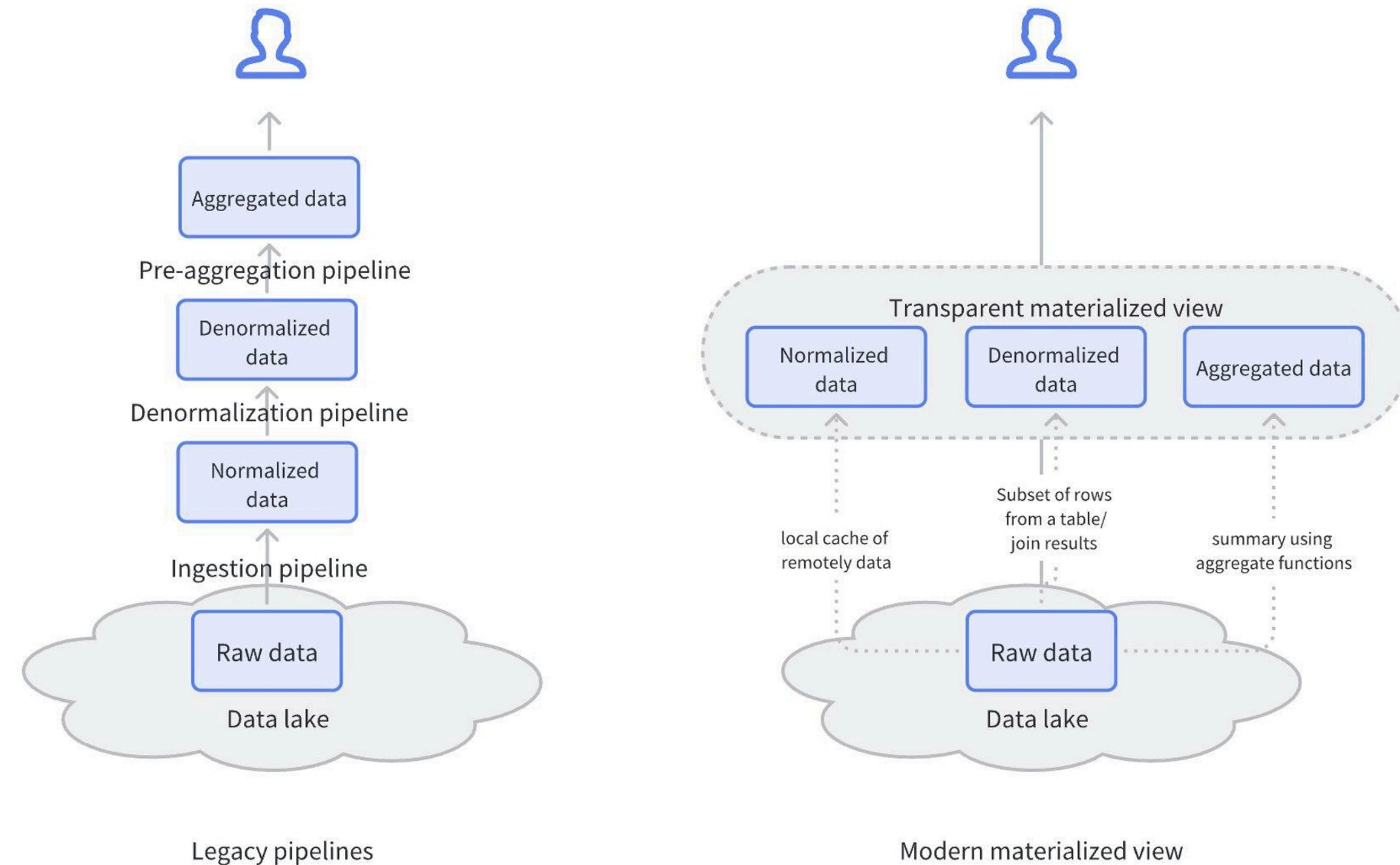


原来想要加速一个查询该怎么办?

- 业务反馈查询速度差强人意
- 跟业务沟通计算逻辑
- 更改某些层表的schema
- 更改ETL任务，重新提交
- 业务更改查询至新表

如果用StarRocks能怎么办?

- 发现某些查询速度差强人意，定位热表
- 定位查询慢节点，创建物化视图
- 查询自动改写，业务无感知自动提速



原来湖上查询速度不理想怎么办?

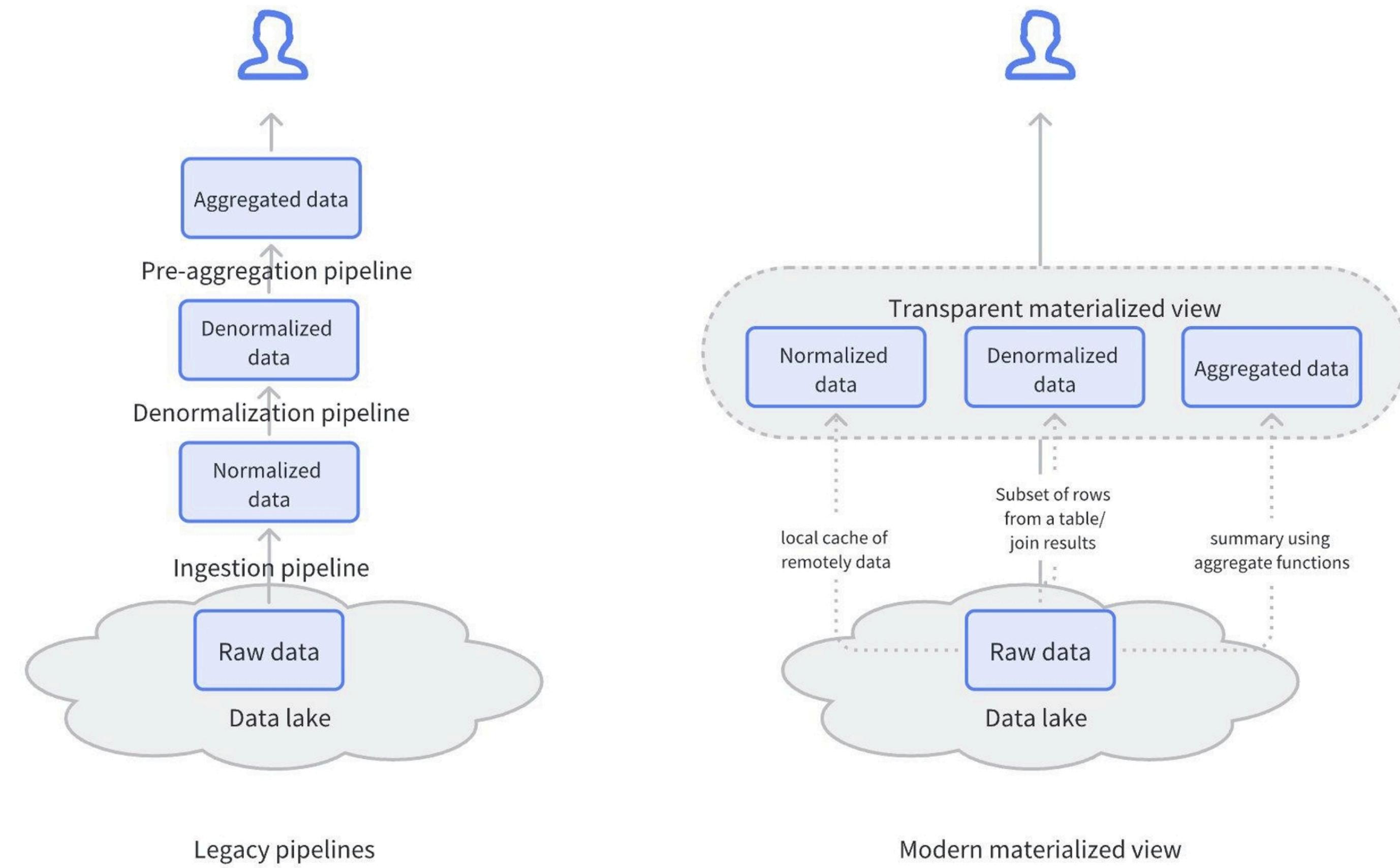
- 在湖上进行数据加工，提前计算，降低数据扫描量
- 导入至OLAP数据库内

这样有什么弊端

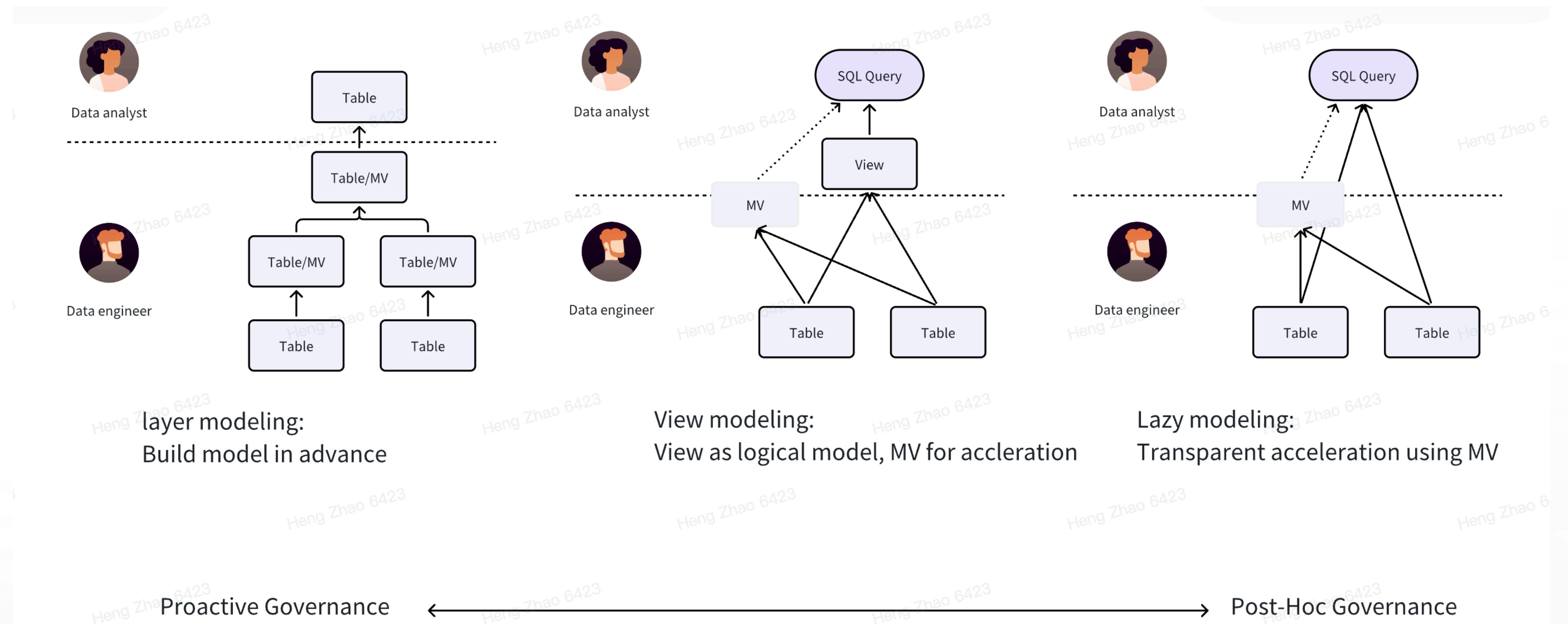
- 维护导入任务、计算口径、数据冗余

如果用StarRocks能怎么办?

- 一般查询，直接查湖，Datacache提速
- 性能并发要求高的查询，湖上物化视图，维护加工和数据导入/更新逻辑
- 自动透明查询改写



三种建模方式与物化视图的关系



- Layer modeling是一个事前治理的结构，需要数据工程师对数据和业务都有了解，能够完成数据加工的任务
- Lazy modeling是一个事后的建模，通过Adhoc的查询沉淀profile，后置的对慢查询进行透明加速
- View modeling是一个折衷的方案，通过业务人员进行逻辑视图的初步构建，然后可以对应的进行mv加速



物化视图使用案例

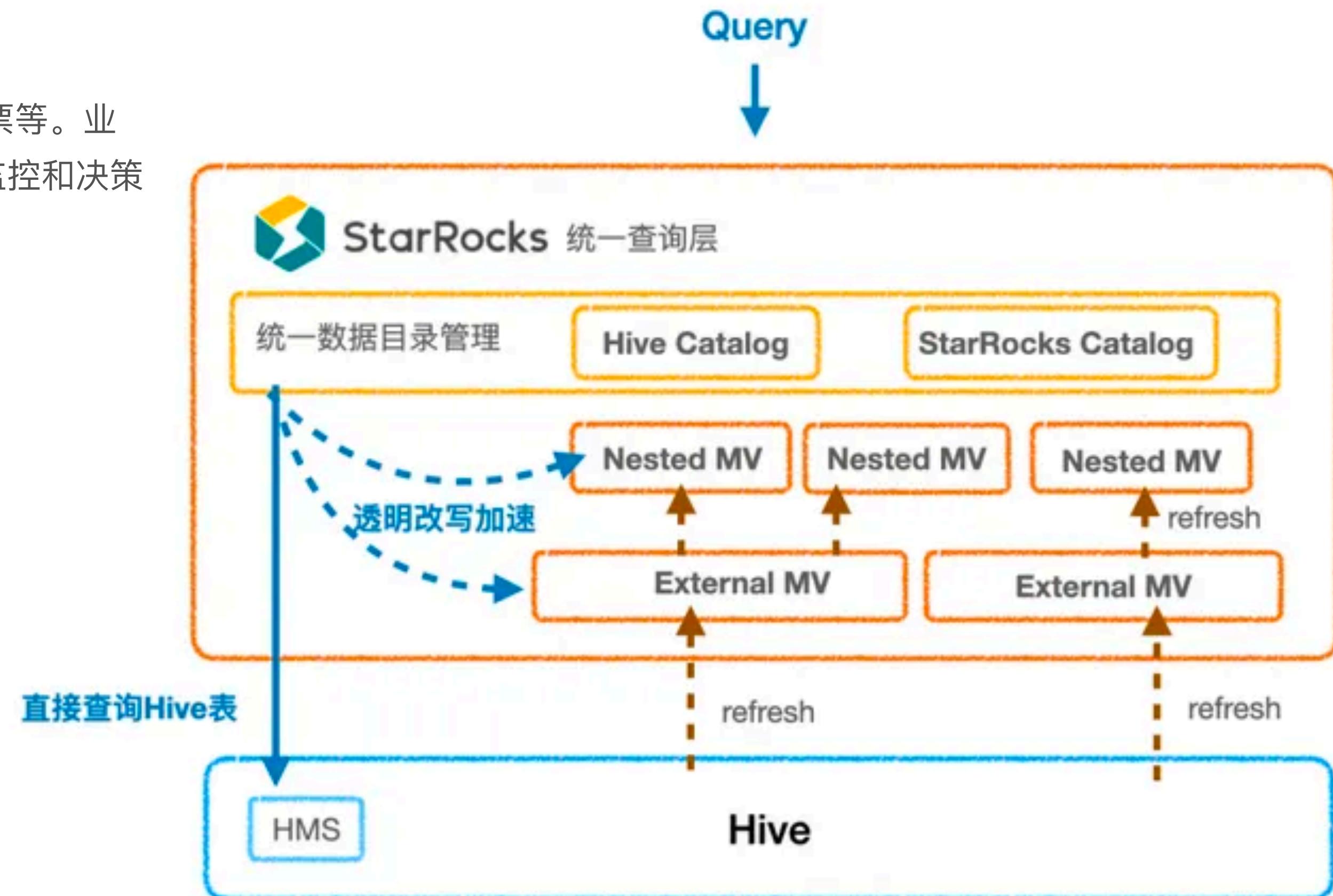
业务场景和痛点

- 统一报表平台Artnova
承载了集团所有 BU 的报表业务，如酒店、机票、商旅、度假、市场、火车票等。业务人员可以通过在平台内配置自定义报表来获取所需的业务数据，辅助业务监控和决策

- 业务难点：SQL复杂、底表大、高并发
- 技术难点
 1. 数据搬迁和管理
 2. 按需透明加速

实际效果

- 湖上直查，开启Data cache，利用向量化计算，IO合并，延迟物化等技术，7倍以上性能提升，Trino兼容性99%
- 通过Catalog+MV的方案，对特定慢查询创建MV，StarRocks Hive 外表的报表平均提速达到 10 倍，与 StarRocks 内表提速相当。目前已超过 10W 查询从 Trino 切到 StarRocks Hive 外表上



物化视图案例分享：实时、离线统一的湖仓一体

业务场景和痛点

- 实时数仓到湖仓一体的统一

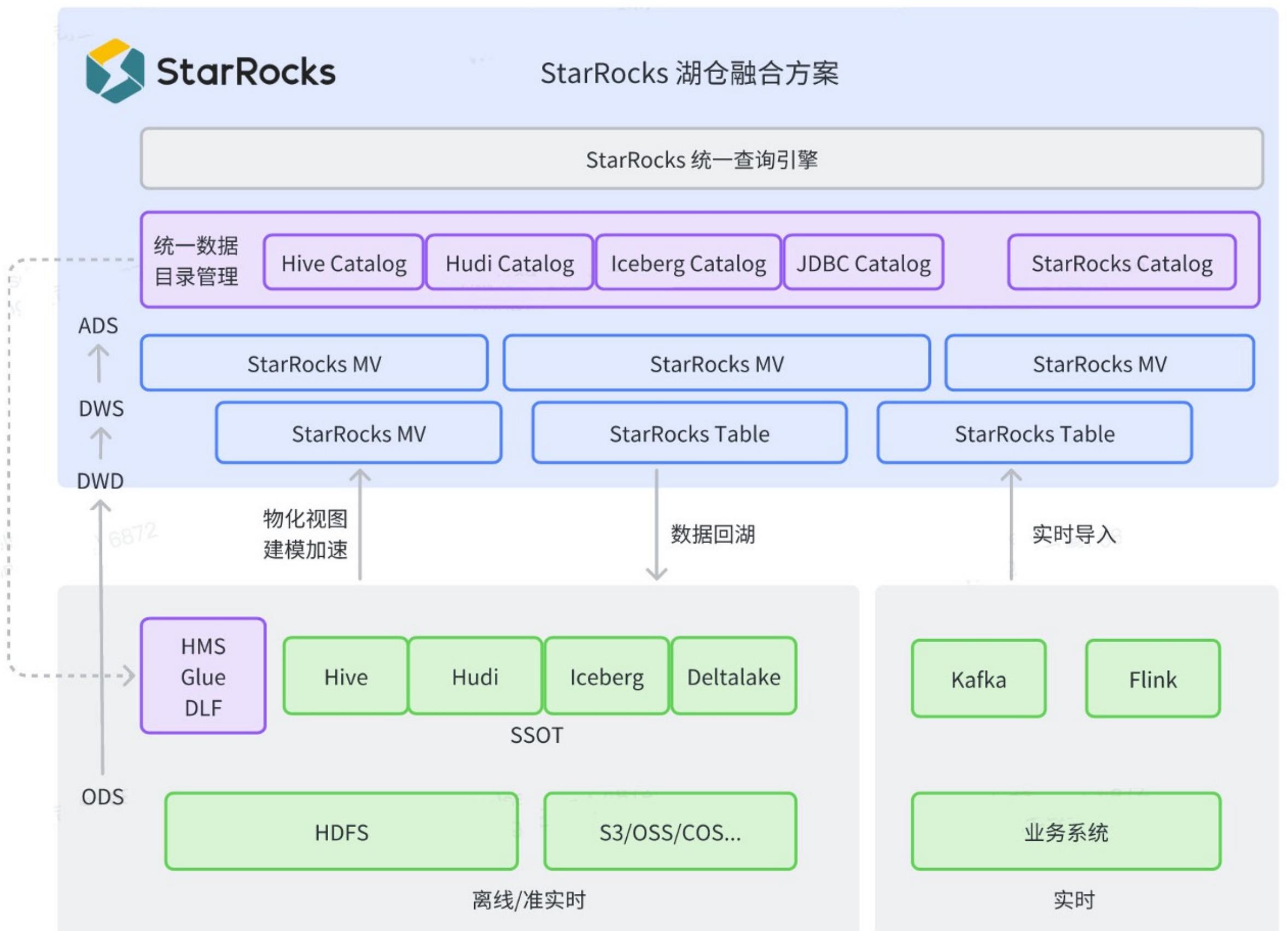
实时数据通过Flink已经统一到StarRocks。离线数据在Hive中，通过Impala查询，聚合数据通过Kylin创建Cube加速查询。

- ### • 痛点

1. Kylin中几百个cube，单个cube上亿级别，半夜开始刷新早上9点刷完，构建时间太长
 2. 命不中的就会降级为hive非常慢

- ## • 迁移方案和效果

1. 直接基于Hive catalog的物化视图，用物化视图打成大宽表，如果性能不行就再上卷。
 2. 外表物化视图的第一层就直接采用kylin的spark构建的SQL，只需要简单修改Spark的SQL，就可以完成宽表构建，部分cube直接抛弃，用外表查询满足，最重留下50个mv
 3. 构建时间从7-9个小时下降到1.5小时，不需要额外的spark资源，复用现有的StarRocks集群资源
 4. 查询效果大部分在1s以内，最大的2年的数据多表直接join，大概7-8s 完成
 5. 迁移以后 指标的开发从 3.5天 下降到 1.5天

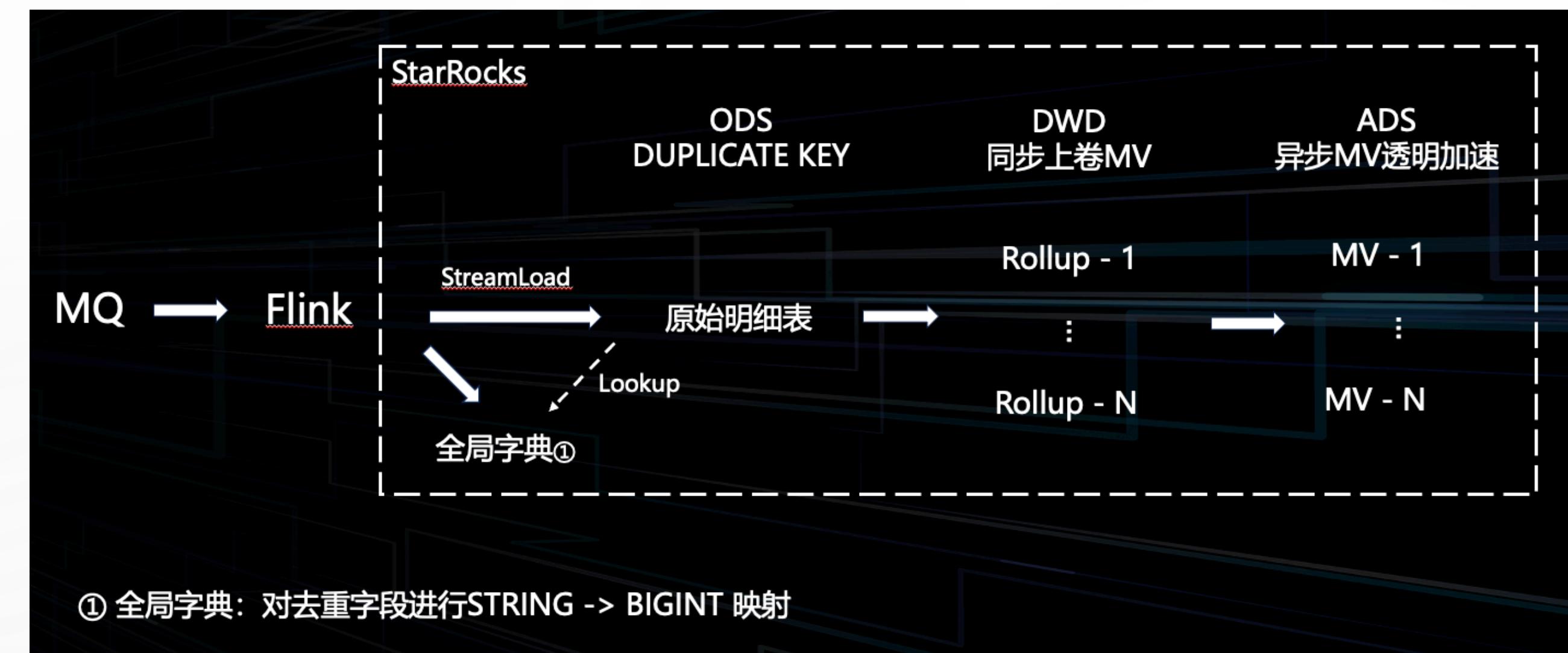


业务场景和痛点

- 实时业务洞察
网约车实时看板是公司最关键的业务监控工具之一，包含着最核心的业务指标，如实时呼叫量、冒泡数量和 GMV
- 业务难点：高基数高并发精确去重

数据链路

- 数据接入：实时从flink导入base表和字典表，通过内部字典表降低复杂度
- 增量聚合层：实时数据通过同步MV，实现count distinct(expr)的透明加速
- 透明加速层：通过异步物化视图构建



增量聚合

```
CREATE MATERIALIZED VIEW mv_dwd AS
SELECT dt, time_slice(`table_time`, INTERVAL 5 minute, floor) AS
`ts`, call_city, source_type, bitmap_union(to_bitmap(order_id))
FROM base_table
GROUP BY dt, ts, call_city, source_type;
```

透明加速

```
CREATE MATERIALIZED VIEW `mv_ads`
PARTITION BY (dt)
DISTRIBUTED BY HASH(`ts`) BUCKETS 1
REFRESH ASYNC START("2023-06-28 21:00:00") EVERY(INTERVAL 30
SECOND) PROPERTIES (
"partition_refresh_number" = "3"
)
AS SELECT
`dt`,
time_slice(`table_time`, INTERVAL 5 minute, floor) AS `ts`,
`call_city`,
`source_type`,
count(DISTINCT `order_id`) AS `order_num`
FROM `base_table` GROUP BY
`dt`,`ts`,`call_city`, `source_type`;
```

