

# 实时数据更新与极速查询兼顾： Apache Doris Unique Key 写时合并实现揭秘

张晨

飞轮科技 资深技术专家

Apache Doris Committer

## 个人介绍



**张晨**

**SelectDB资深研发工程师**

**Apache Doris Committer**

拥有超过10年的大数据行业经验，曾先后在微软、百度、小米、蚂蚁任职，在OLAP系统、HSAP系统、分布式文件系统、对象存储领域都有深入的研发经历。



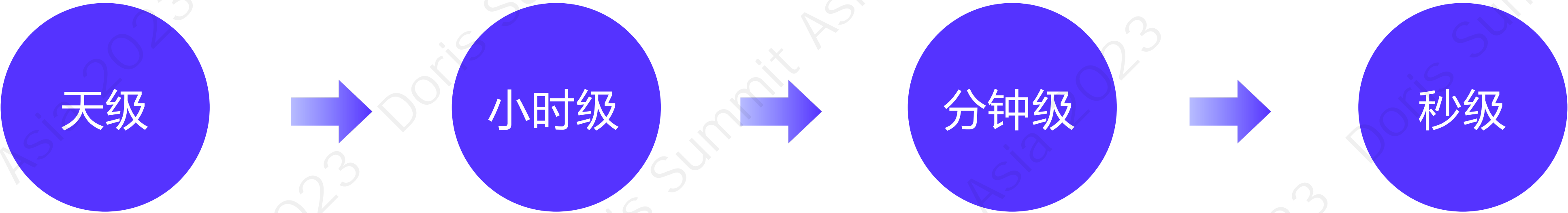
# 目录

1. OLAP 系统的实时更新需求介绍
2. OLAP 系统数据更新的挑战
3. 实时数据更新与极速分析如何兼得
4. Doris 2.0 数据更新能力全面解析
5. 用户实践

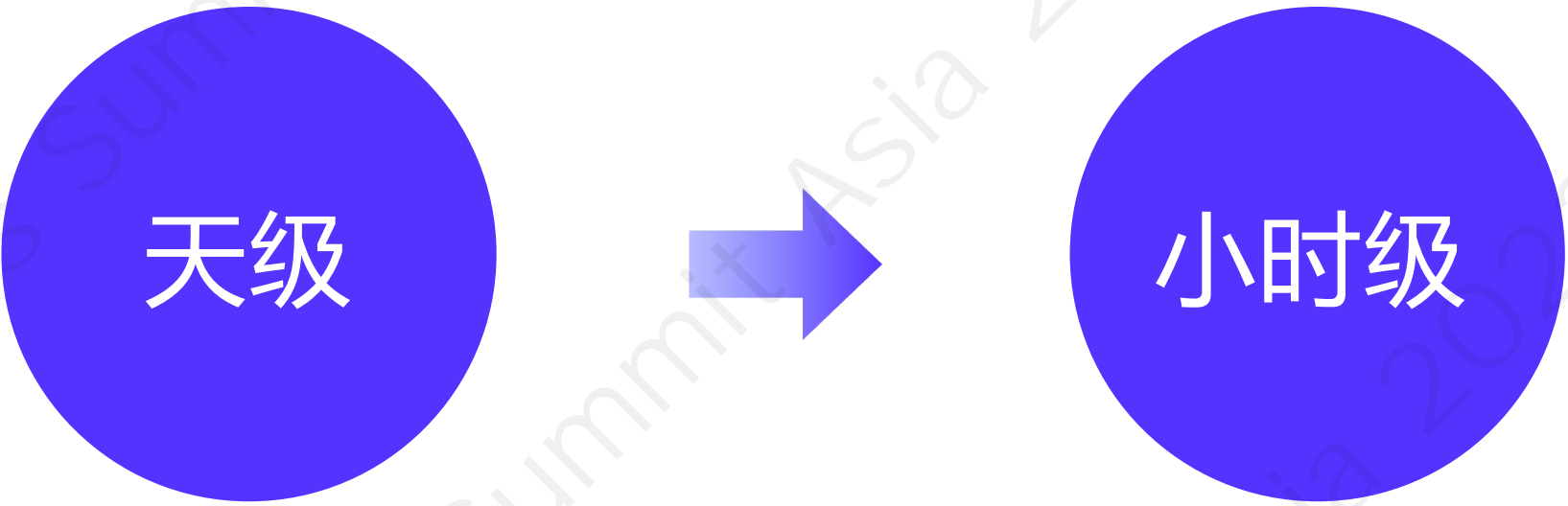
# 1 OLAP 系统的实时更新需求介绍

# 分析性能越来越快

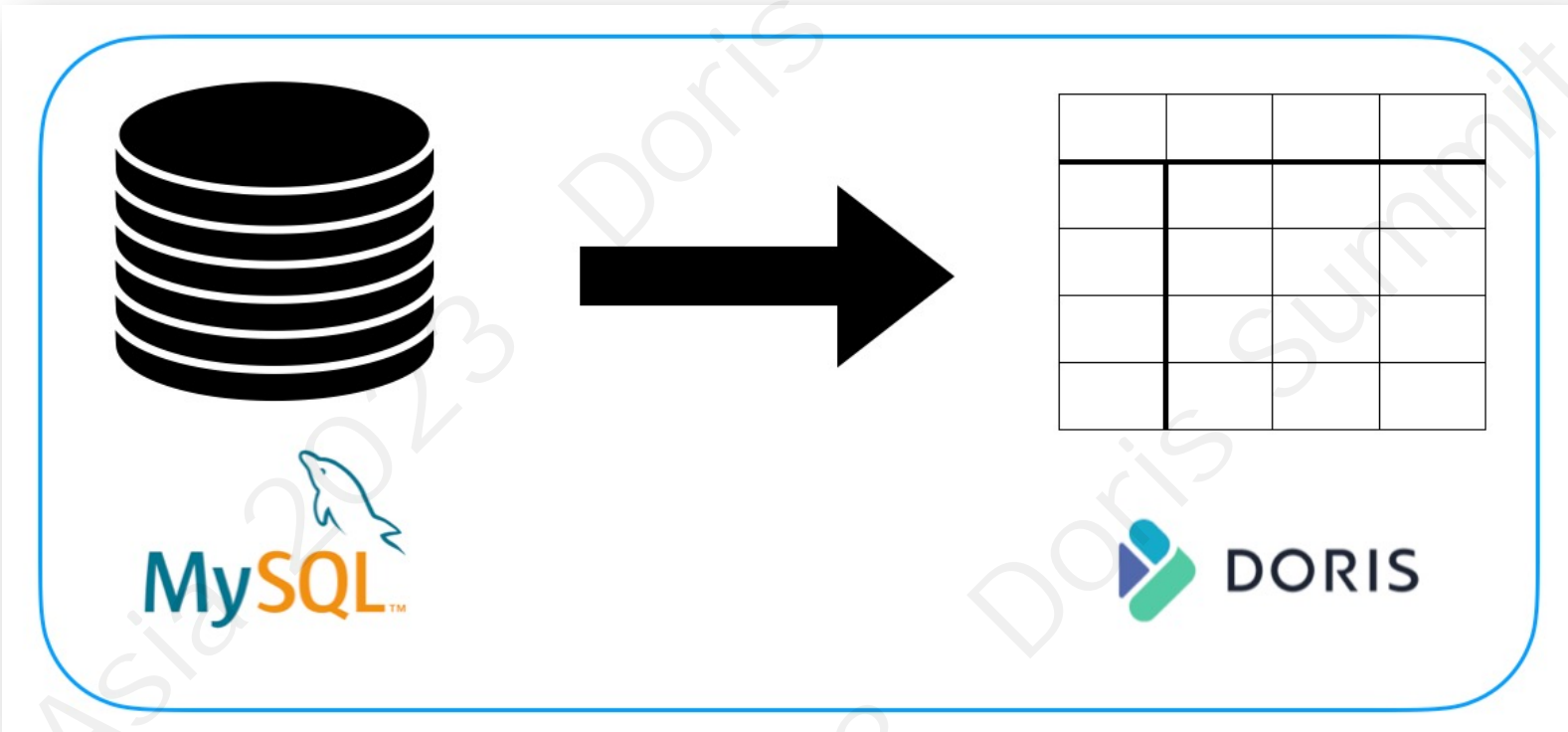
大数据量的分析延迟越来越低



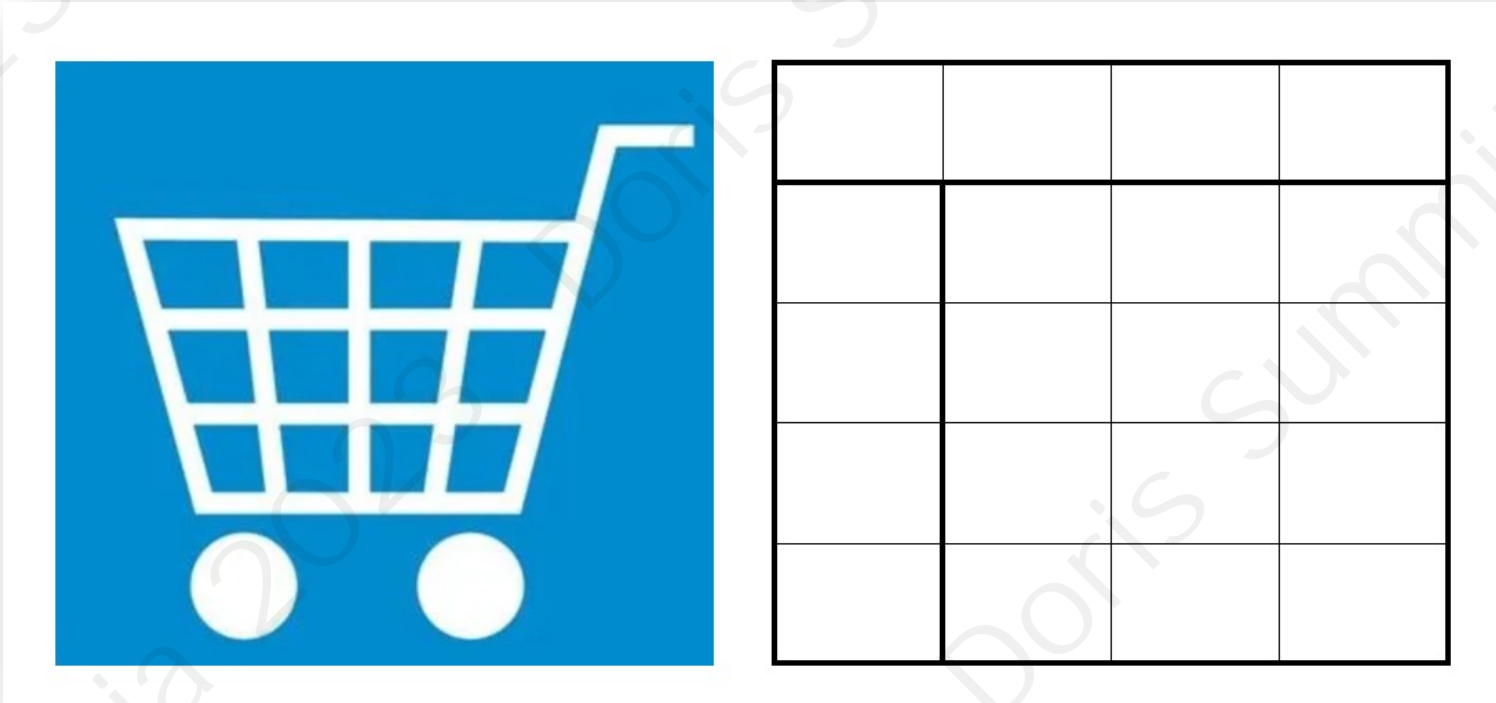
大数据量的更新延迟还有很大的提升空间



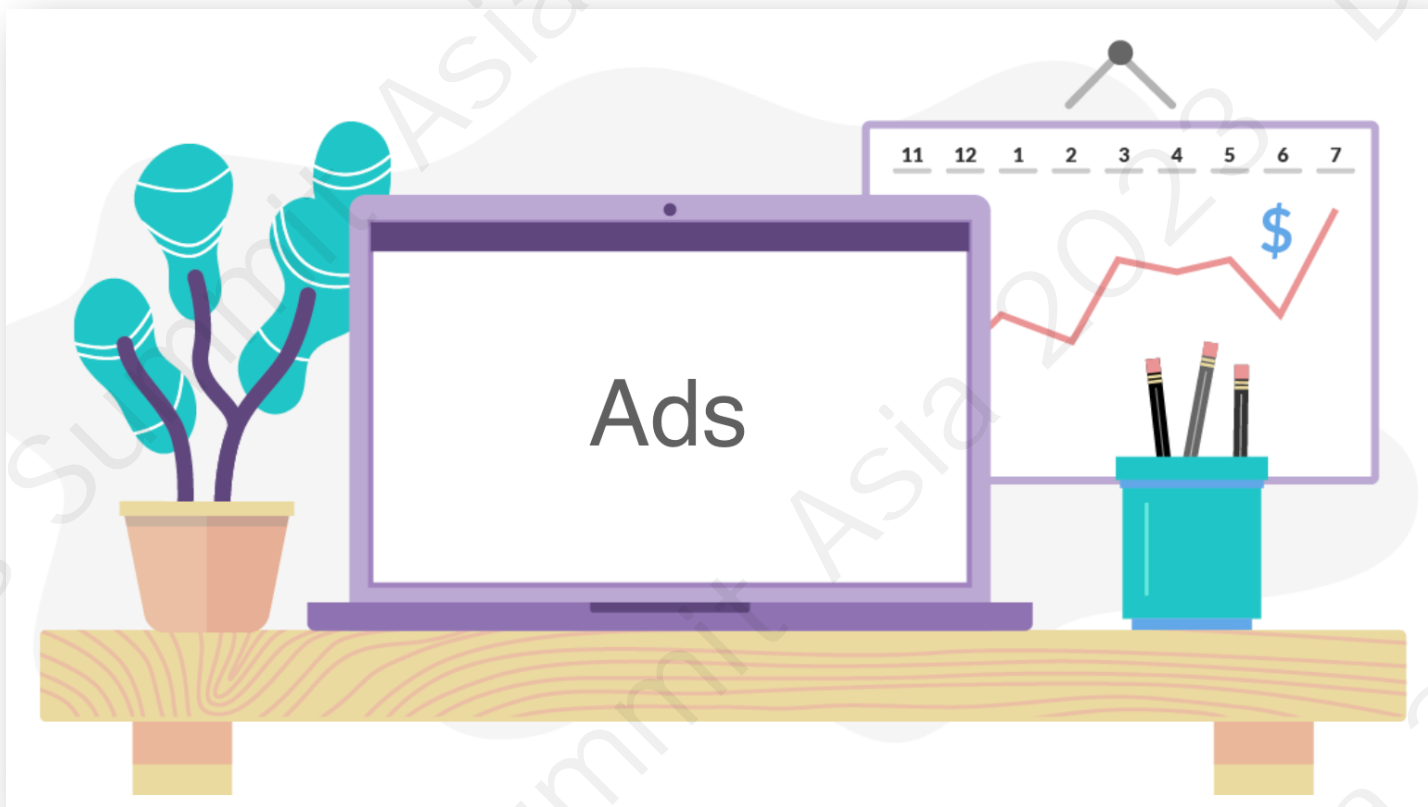
# OLAP 系统中的数据更新需求普遍存在



MySQL CDC



电商交易订单



广告效果投放



营销业务报表



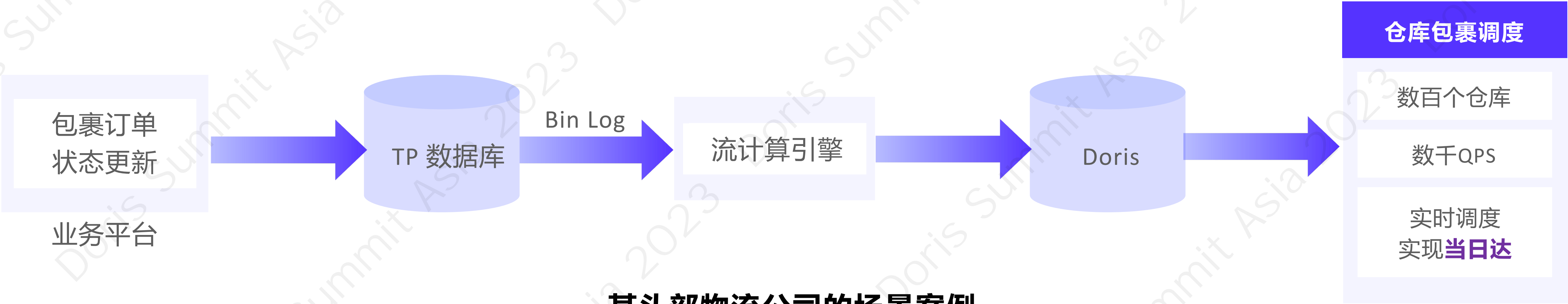
# 对数据更新的实时性要求越来越高

## 数据更新

- 10万条/s 的峰值写入流量
- 20并发同时写入
- 实时更新，数据新鲜度 10s 以内
- 保留半年以上数据，总数据量百亿条

## 数据查询

- 百毫秒级的查询延迟
- 数千QPS
- Ad-Hoc Query
- 几乎所有的查询都涉及多张宽表的join

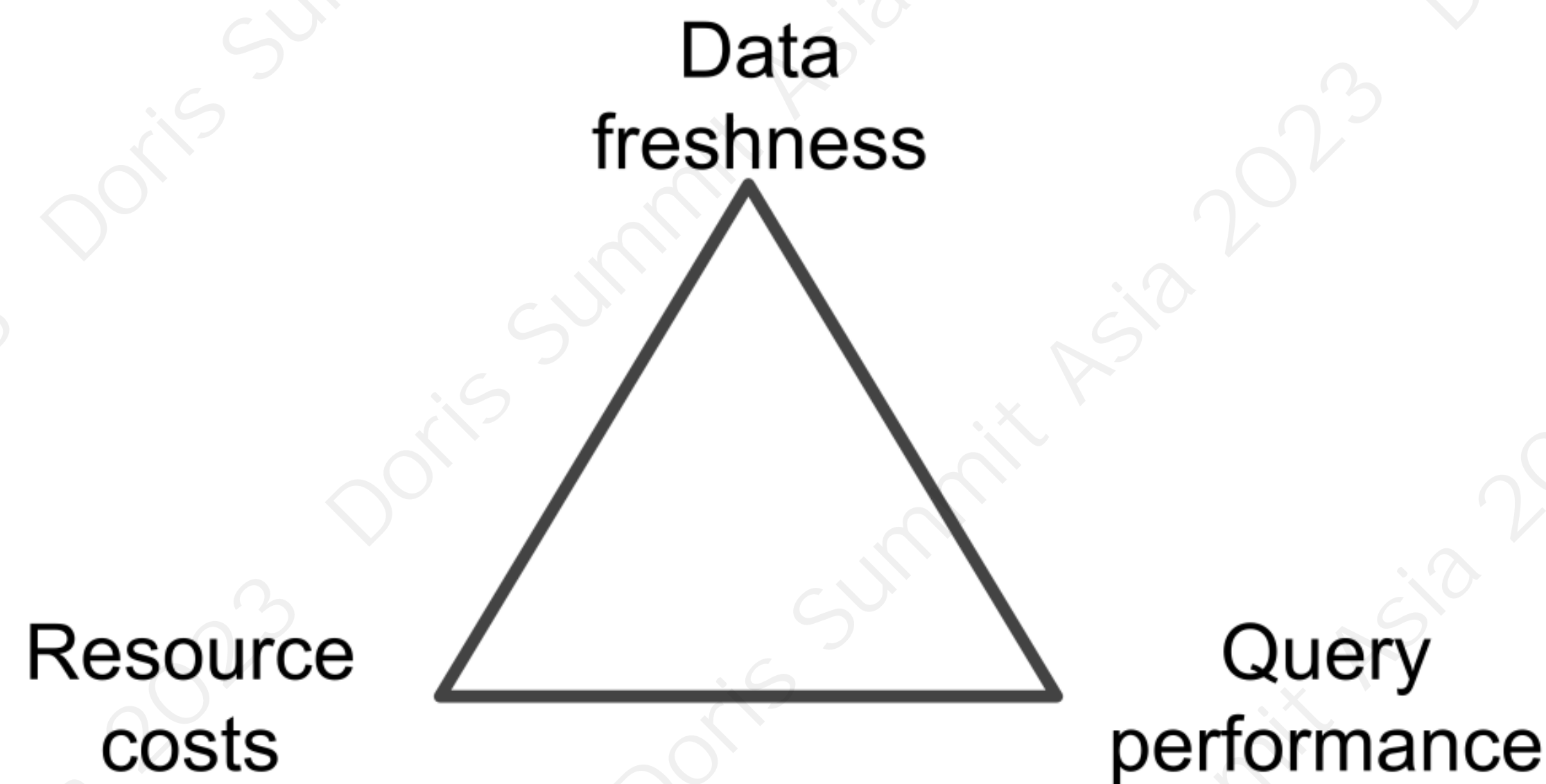


某头部物流公司的场景案例

## 2 OLAP 系统数据更新的挑战



# OLAP 数据更新的挑战



Napa: Powering Scalable Data Warehousing with Robust Query Performance at Google

# OLAP 数据更新的挑战

## OLAP系统往往采用列存

- 按列存储，可以只读取需要的列数据
- 按类型高效的进行编码和压缩
- 丰富的索引和裁剪
- 大大减少一次分析需要读取的数据量

## 列存数据更新的挑战

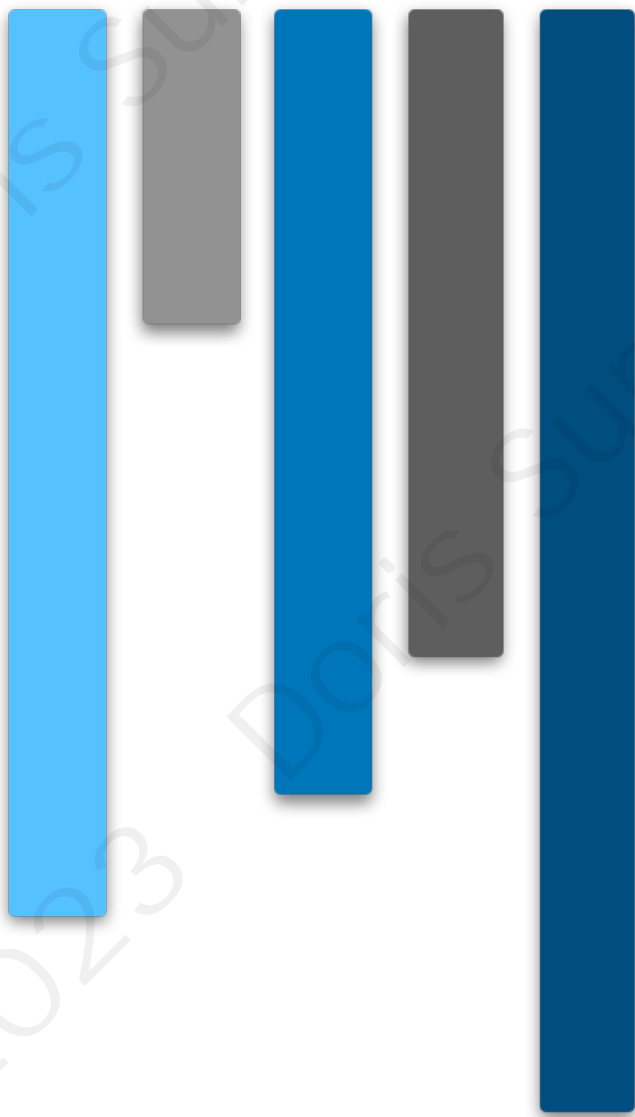
- 利用压缩、编码来减少文件大小，难以修改
- 丰富的索引，更新成本高
- 常见的更新方式（MoR，CoW）性能代价较大

Row-based Storage



Write Optimized  
(uncompressed,  
in-place updates)

Columnar Storage



Read Optimized  
(compressed,  
read-only pages)

# 大数据常见的数据更新方式：Merge-on-Read

## 行存系统（KV系统）

- 通过MVCC来实现高并发的写入支持
- 通过merge-on-read来支撑较高的点查QPS
- 代表系统如：LevelDB、RocksDB、HBase等

## 列存系统（OLAP系统）

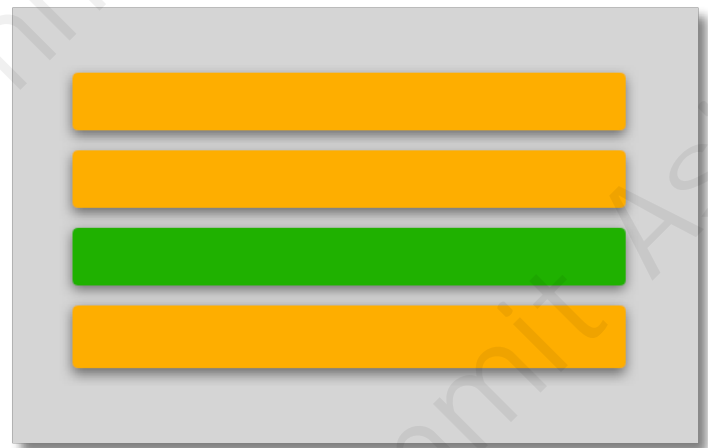
- 很多列存系统并不支持更新
- 支持merge-on-read的代表系统有：Hudi、Iceberg、ClickHouse 的 ReplacingMergeTree、Apache Doris的Unique Key

Row-based Storage

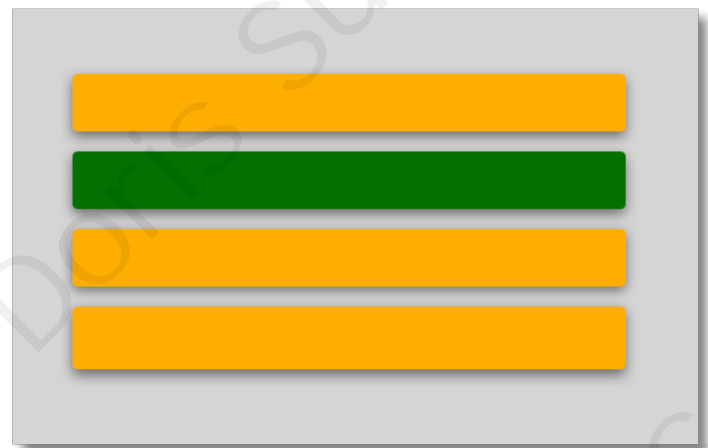
version 1



version 2

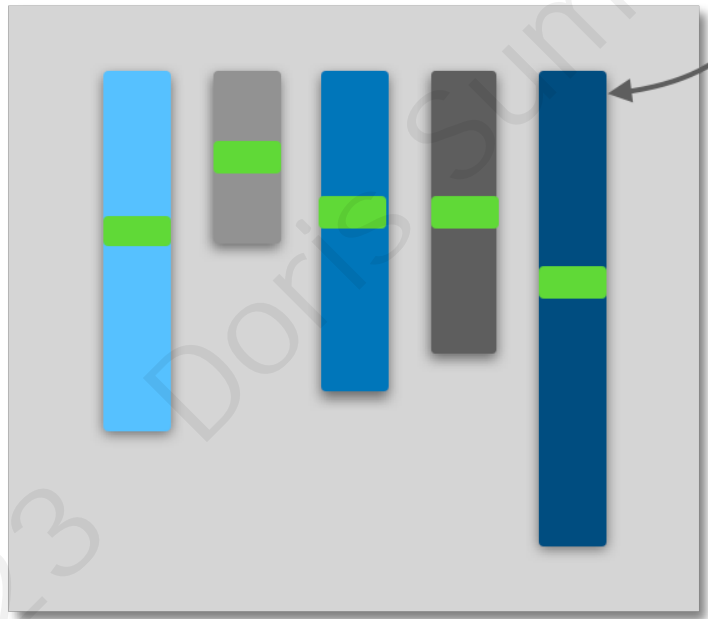


version 2

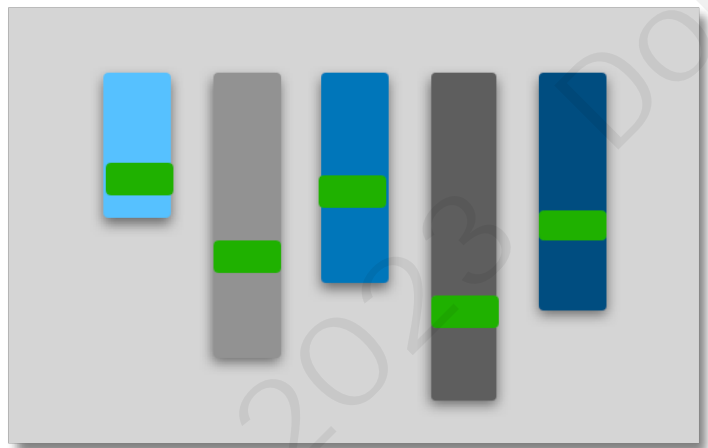


Columnar Storage

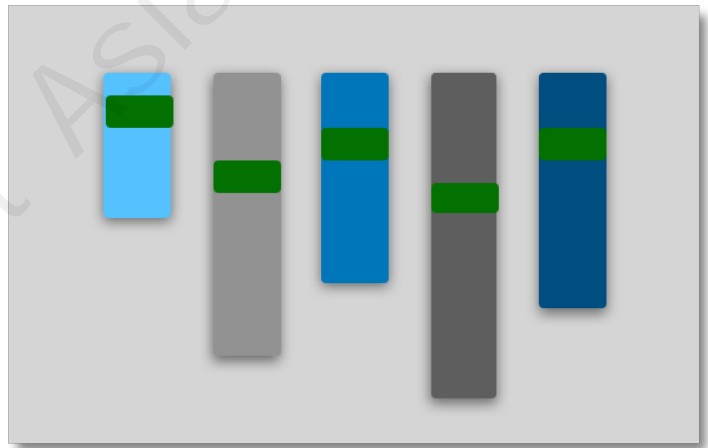
version 1



version 2



version 3



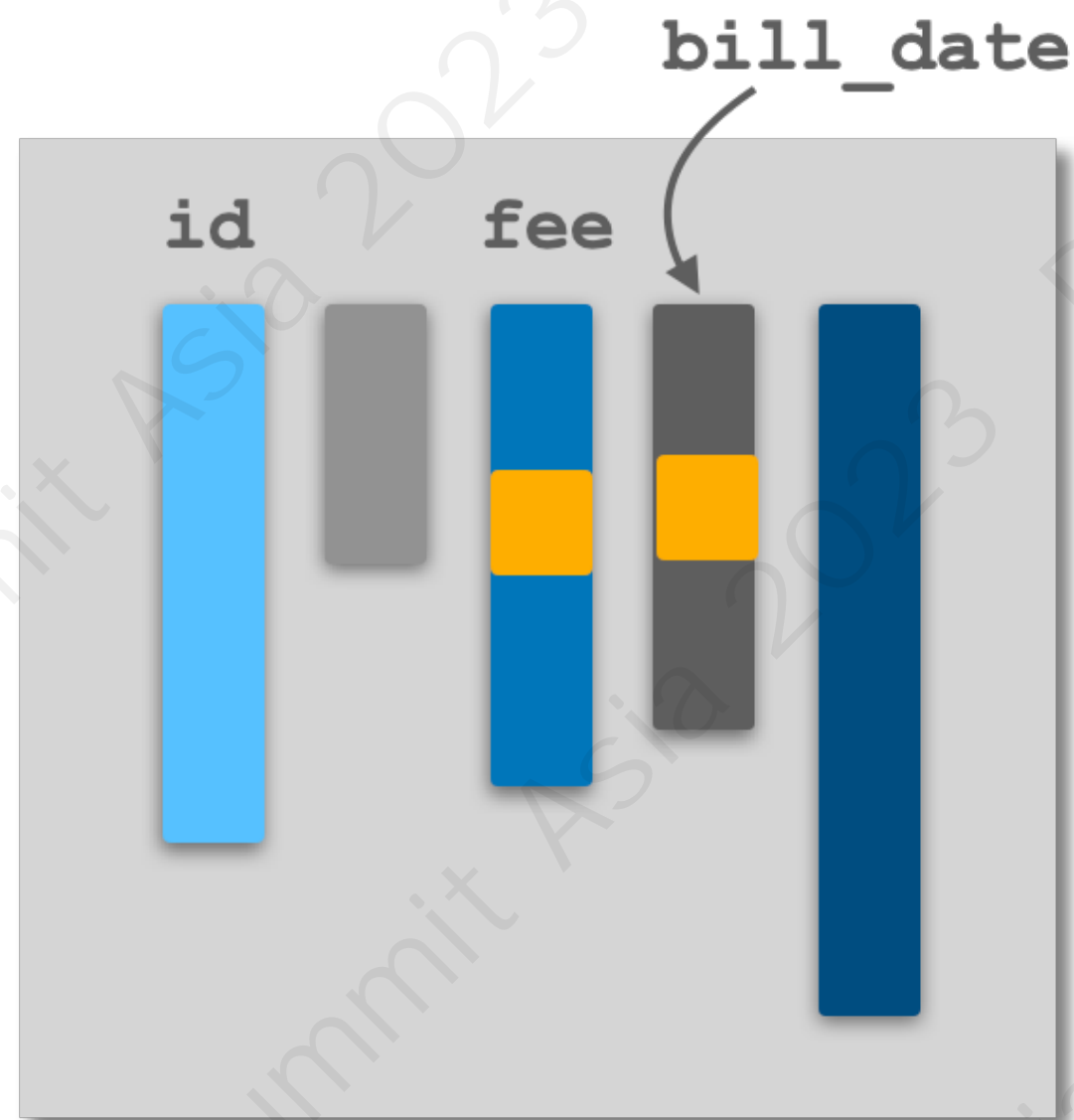


# OLAP 系统使用 Merge-on-Read 方案的性能问题

--例子：查询指定日期范围内所有账单开销的总额

```
SELECT SUM(fee) FROM ledger
```

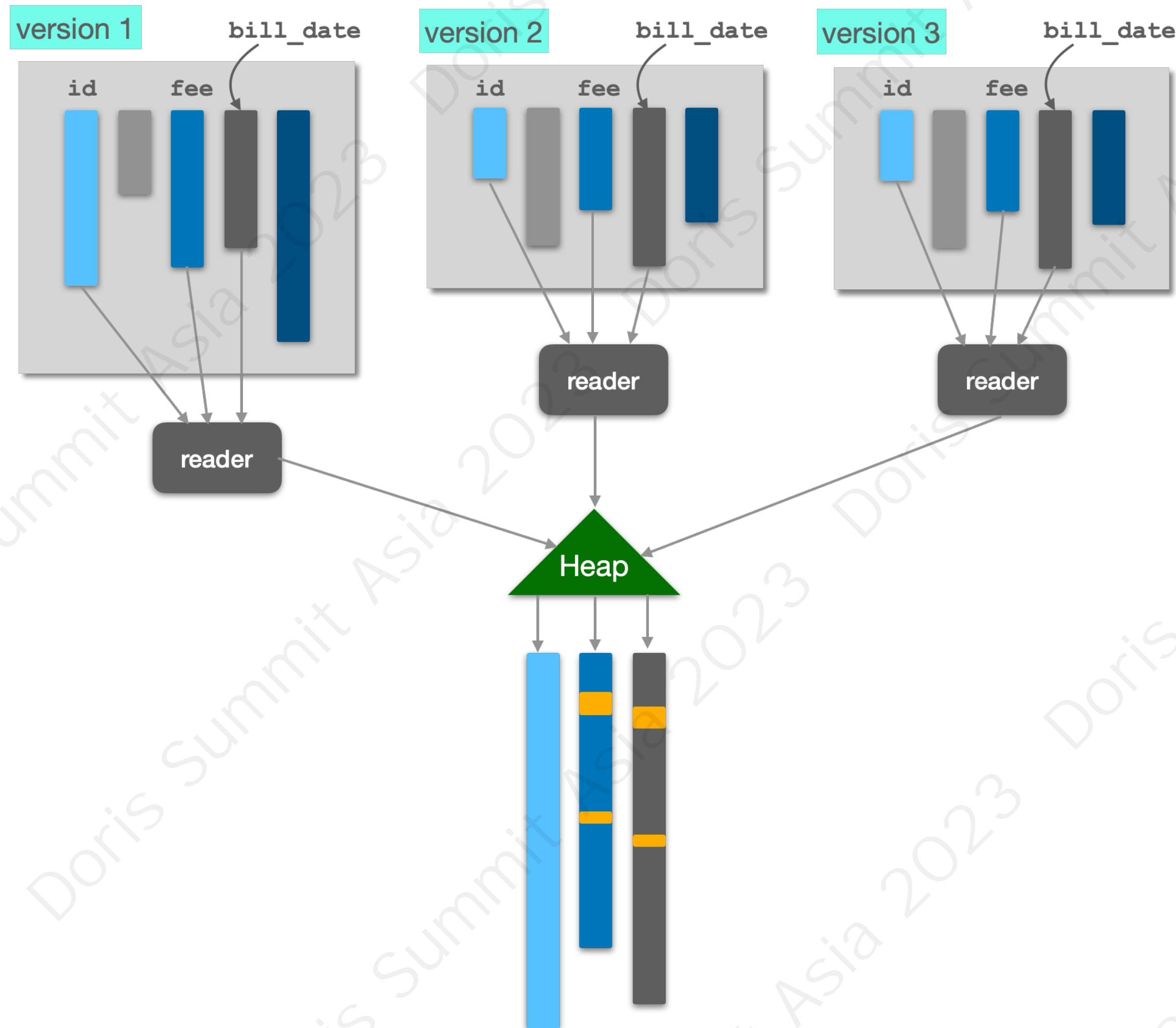
```
WHERE bill_date >= `2023-08-01 00:00:00` AND bill_date < `2023-08-02 00:00:00`
```



## 无数据更新时

- 只需要读取fee、bill\_date两列
- 通过bill\_date内建的zonemap可以快速的定位符合条件的数据和对应的行号范围
- 通得到的行号范围，在fee列中读取对应的数值求和

# OLAP 系统使用 Merge-on-Read 方案的性能问题



## 存在数据更新时

- 文件级别的zone map失效，因为version1, version2中存在过期数据，无法直接使用zonemap进行筛选
- 必须先对要读取的数据进行全部的一遍merge sort进行去重，才能得到最新的数据
- 最终在最新的数据上进行运算

## Merge-on-Read 的代价

- 需要额外读取Key列，用于数据的排序和去重
- 归并排序带来的额外的key compare代价，较高的CPU消耗
- 无法下推过滤条件，需要全量扫描涉及的所有列的所有数据

# 其他常见方案：DeltaStore

## Apache Kudu的DeltaStore更新方案

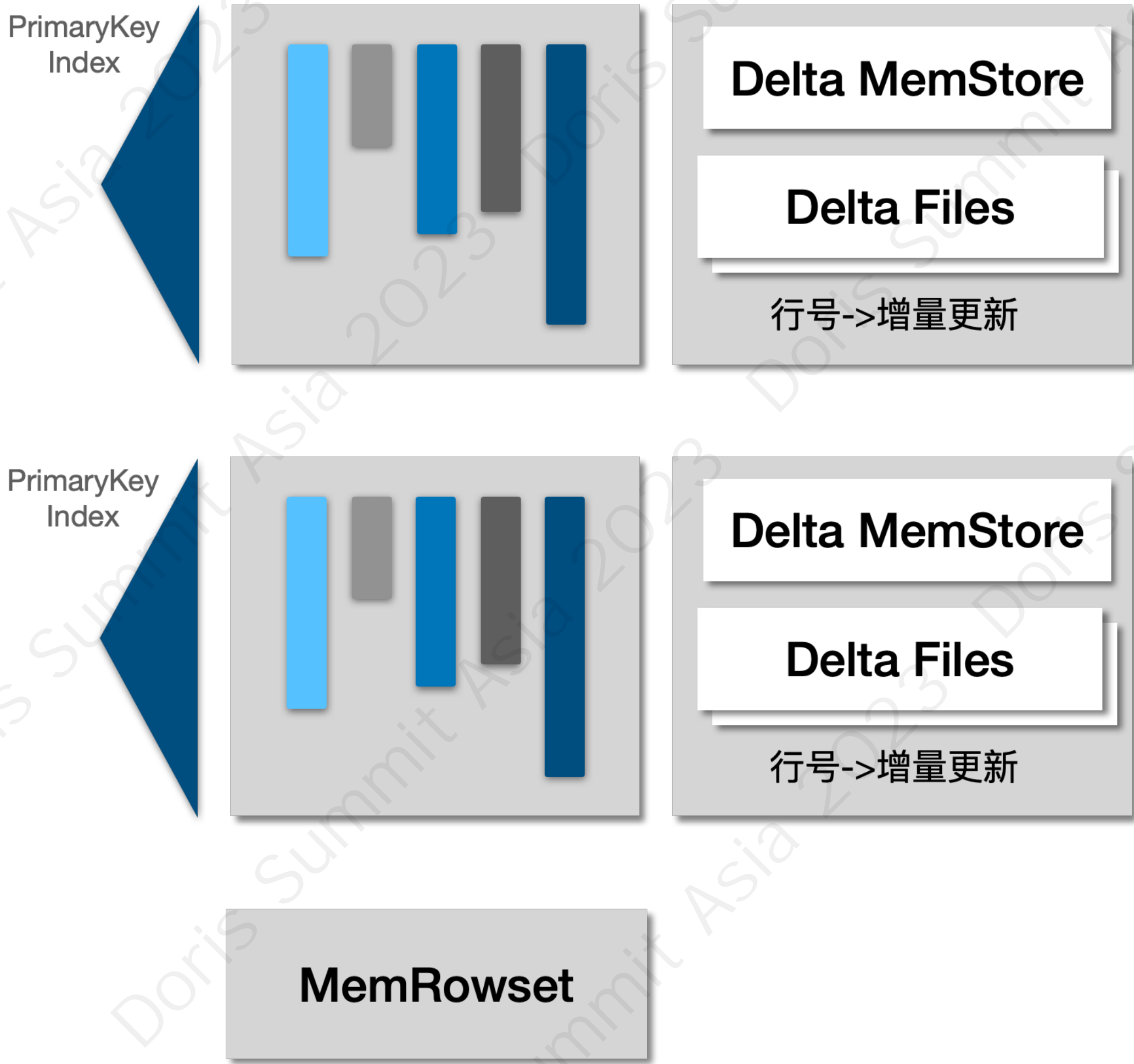
- 写入时先查索引，新增的key写入MemRowset
- 如果是已存在的key，就写入Delta MemStore中
- Delta MemStore和Delta Files中记录的都是行号和增量更新的数据间的映射关系

## 方案优点（对比Merge-on-Read）

- 避免了必须要读key列
- 避免了大量的key比较带来的CPU消耗
- 内存数据可查，数据新鲜度非常好

## 方案缺点

- 仍需在读时做数据合并，虽然按行号进行合并代价较低
- 在合并完成之前仍然无法下推过滤条件来减少数据量的读取，无法避免数据的全量扫描
- delta store中缺乏丰富的索引





## 其他常见方案：Copy-on-Write

Hudi/Iceberg 等数据湖系统支持通过Copy-on-Write的方式进行数据更新

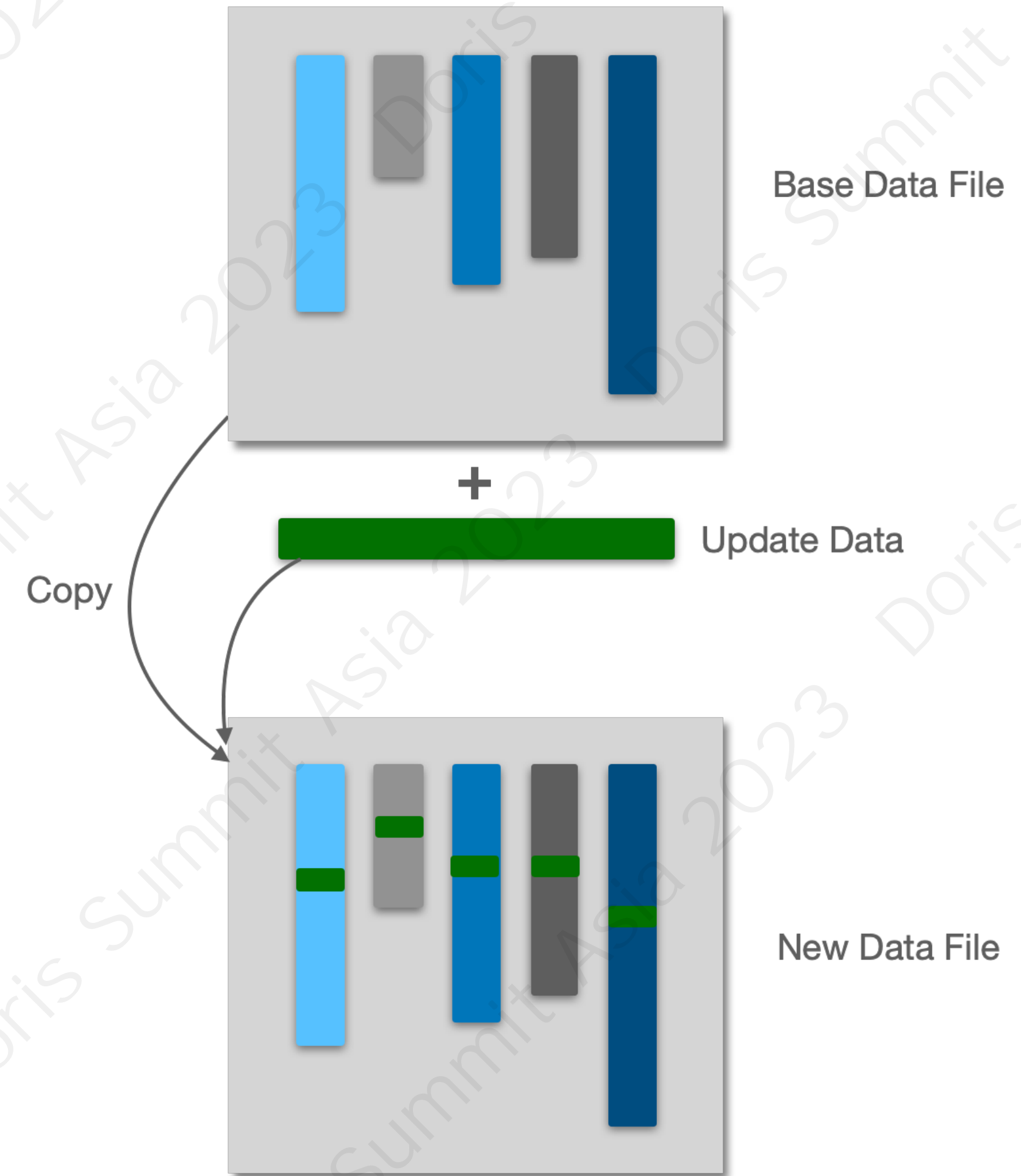
- 即使更新少量数据
- 也需要将Base数据全部读出来，更新后再写入到新的数据文件中

### 方案优点

- 最优的查询性能

### 方案缺点

- 最差的更新性能，严重的写放大



# 3 实时数据更新与极速分析如何兼得

# Doris 实时数据更新方案：Merge-on-Write

## 方案概述

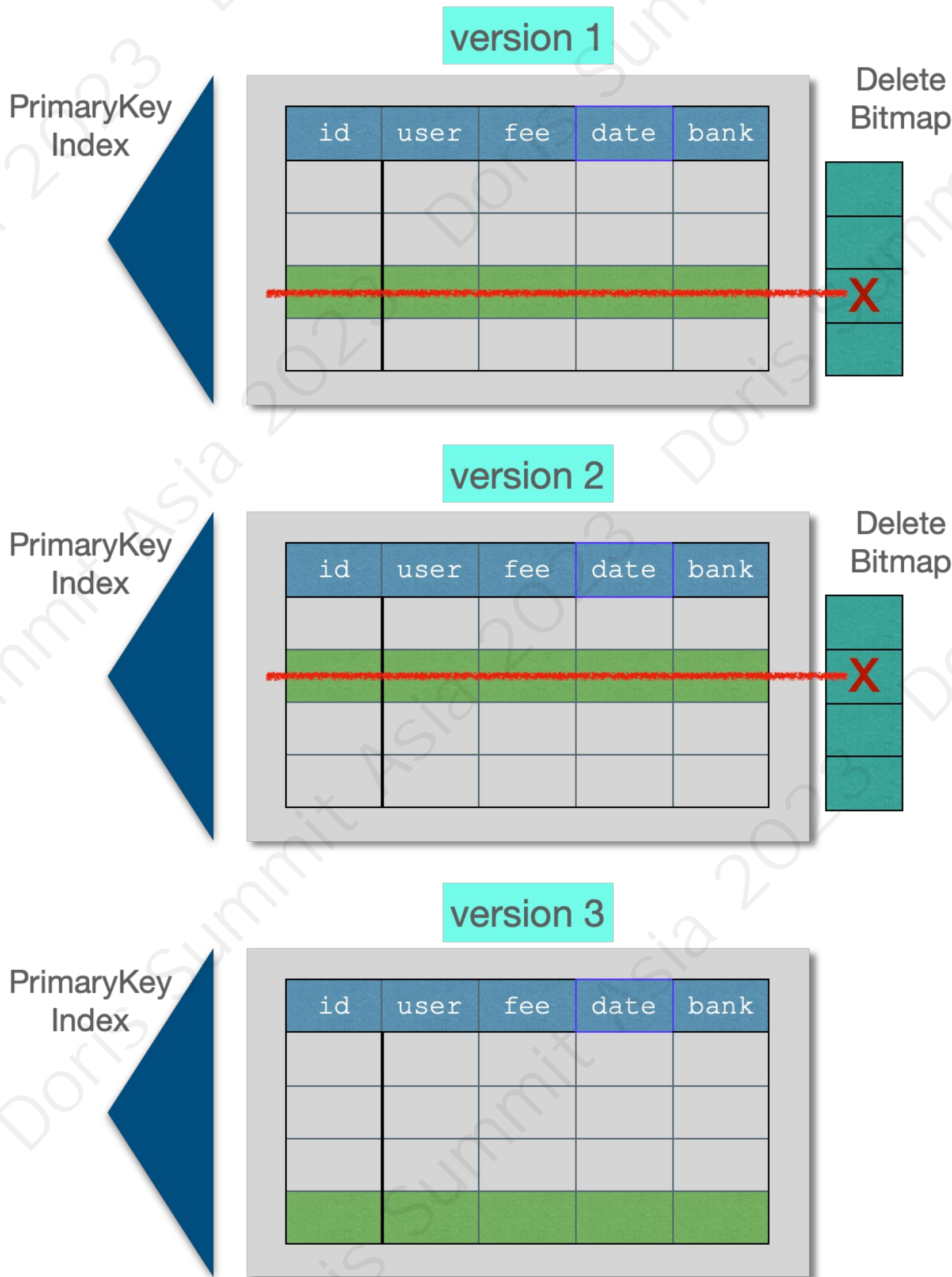
- 为每个文件增加一个主键索引
- 为每个文件增加一个主键的Bloom Filter
- 增加一个Delete Bitmap结构，记录该文件中被更新的数据行号，将它们标记删除
- 当写入新的数据，使用新写入的key查询历史数据（按版本从高到低逆序查询），查到了则在Delete Bitmap中将对应的行号标记删除

## 方案优点

- 接近最优的查询性能

## 方案缺点

- 对数据导入的性能有影响
- 复杂的写冲突处理逻辑





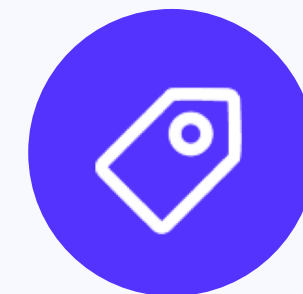
## Merge-on-Write 方案的挑战

### 并发写入冲突



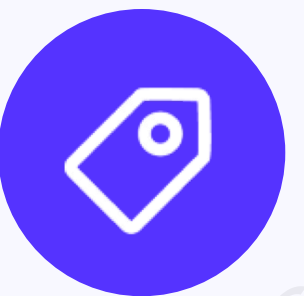
- 不仅需要标记删除Base数据
- 并发的写入数据，也需要互相进行标记删除，按照MVCC进行版本控制
- 超大的批量导入不应阻塞实时的小批高频写入
- Publish Timeout问题

### 各种写冲突



- Compaction
- Heavy Schema Change
- Clone Replica
- Add Invert Index

### 支持条件更新



- 用户自定义的版本列 (Sequence Column)
- 复杂的版本大小和覆盖关系判断

### 索引和BloomFilter的缓存效率



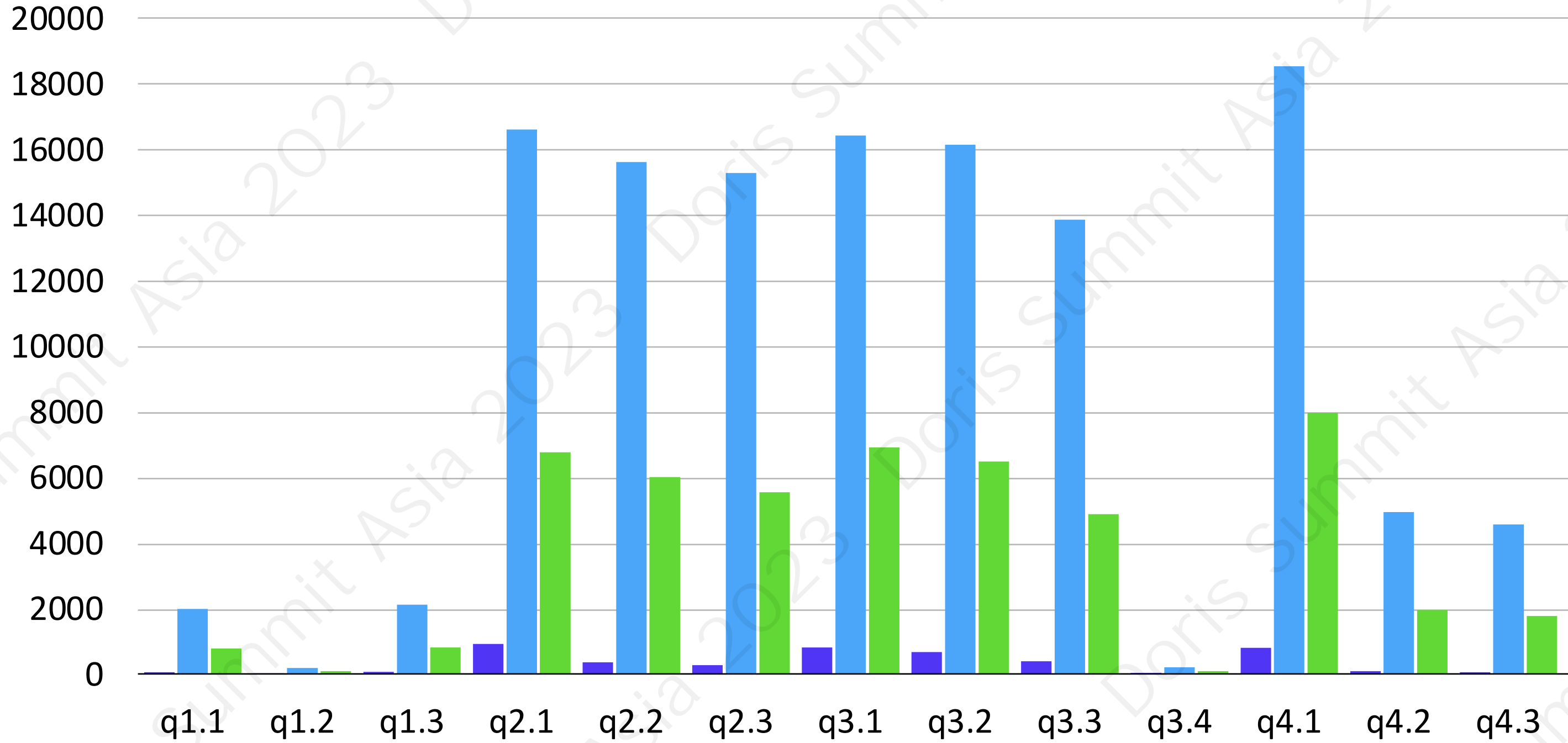
- 索引的查询效率会显著影响导入速度
- 缓存效率和命中率就非常关键

各个方案的对比

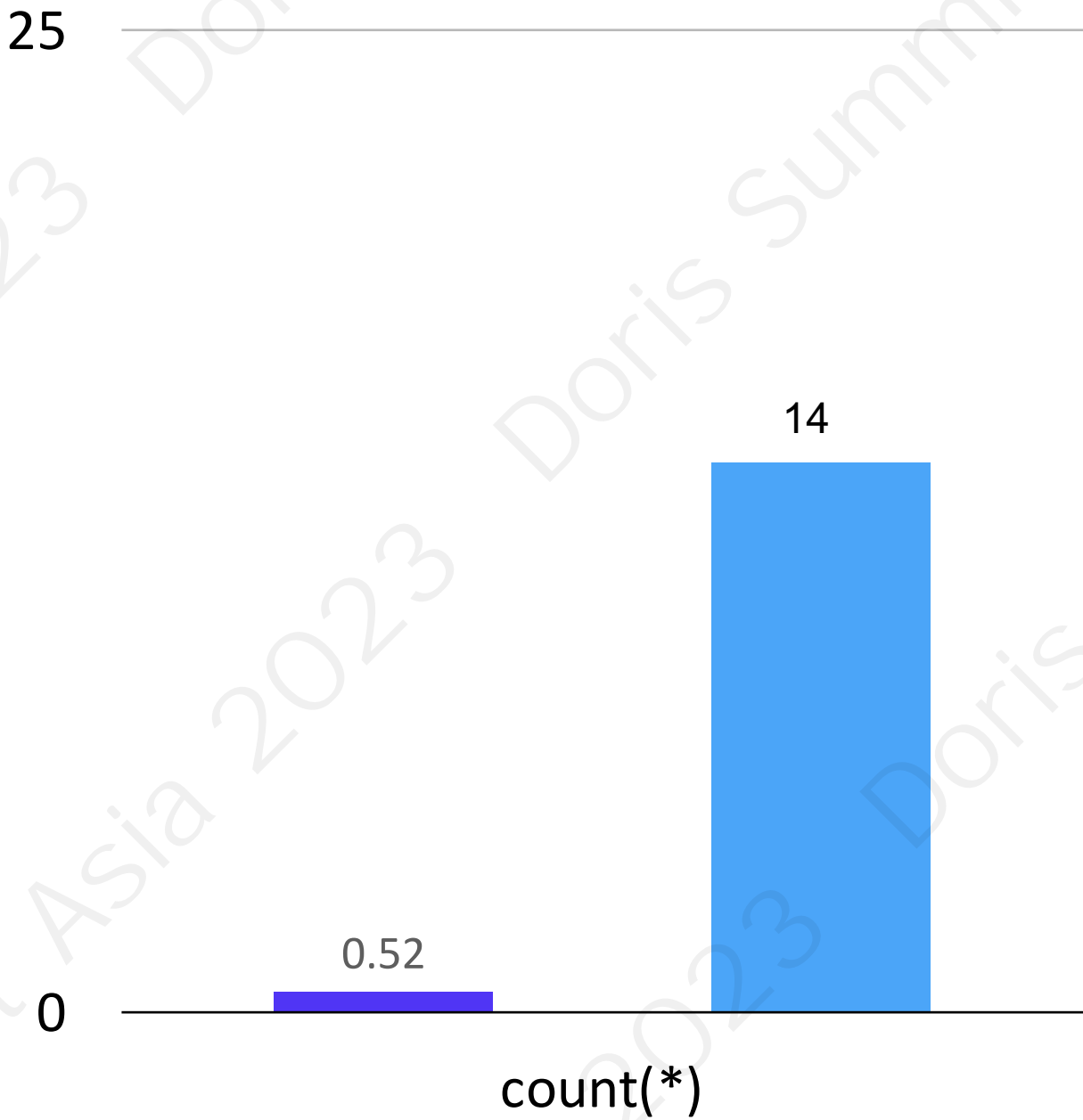
	Merge-on-Read	Merge-on-Write	Delta Store	Copy-on-Write
写入处理	无	查询索引 标记删除 并发冲突处理	查询索引 更新Delta Store	拷贝Base Data 更新后写入新的 Data File
写入代价	无	中	中	高
读取处理	额外读取key列 归并排序进行去重 对结果进行计算	通过 Delete Bitmap 过滤掉 被删除的数据	按行号合并Base和Delta 对合并后的最新数据进行 计算	无
读取代价	高	低	中	无
谓词下推	不支持	支持	不支持	支持
非Key列索引过滤	差	好	中	好

# Merge-on-Write 的查询优化效果

SSB-Flat 100G



Count(\*)



merge-on-write(32C64GB)  
merge-on-read(64C128GB)

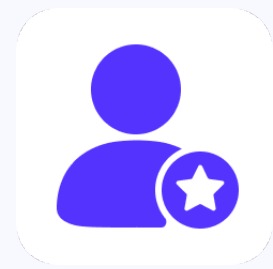
merge-on-read(32C64GB)

merge-on-write   merge-on-read



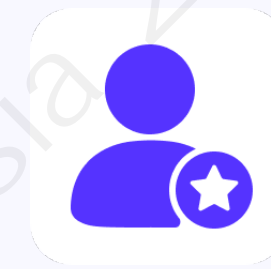
# 4 Doris 2.0 数据更新能力全面解析

## Doris 2.0 数据更新和删除能力



### 整行Upsert

- 基础的整行Upsert能力
- 支持高达十几万TPS的吞吐和秒级延迟



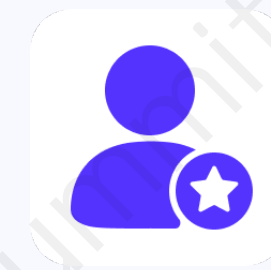
### 部分列更新

- 直接通过导入来更新部分字段
- 支持上万的TPS和高并发更新



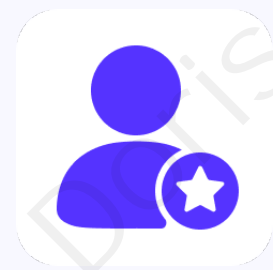
### 条件更新

- 按照用户给定的Sequence值来决定key的更新顺序
- 适用于数据乱序到达时的强一致需求



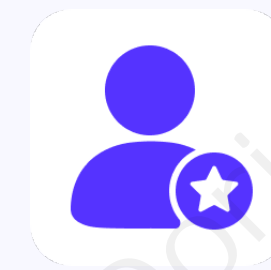
### 复杂的UPDATE语句支持

- 复杂的where条件和多表join支持
- 仅支持单并发的更新任务



### 条件删除

- 通过额外的Delete Sign列来标识要删除的key列
- 适用于批量删除key



### 复杂的DELETE语句支持

- 复杂的where条件和多表join支持
- 仅支持单并发的更新任务

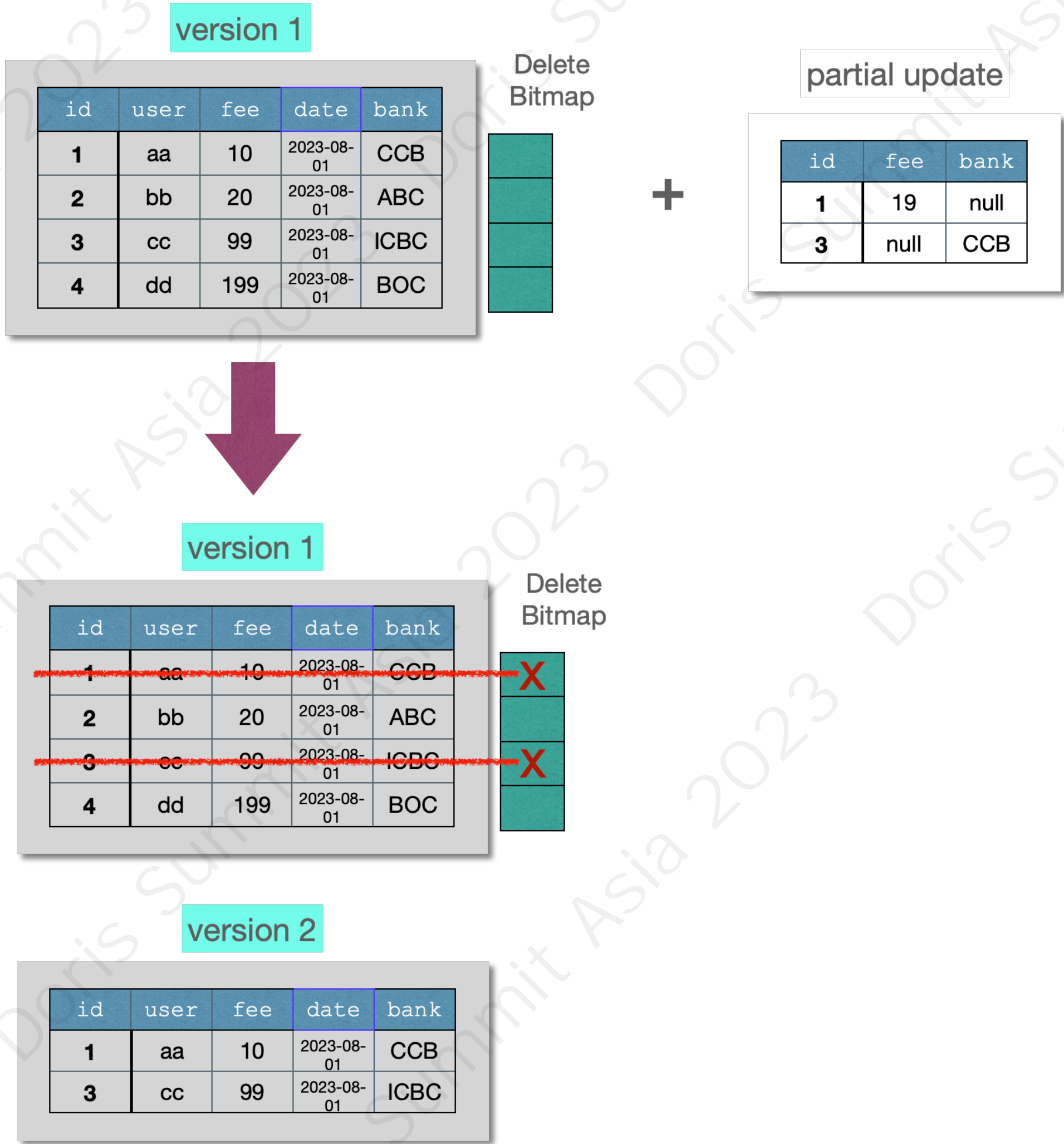
# 部分列更新

## 列更新广泛存在于多种场景中

- 数据修正
- 实时标签
- 宽表拼接
- ...

Doris2.0在MoW基础上，通过整行数据补齐，实现了高吞吐高并发的部分列更新能力支持

- 该功能最好配合 nvme SSD使用
- 高频更新，兼具极速的分析性能



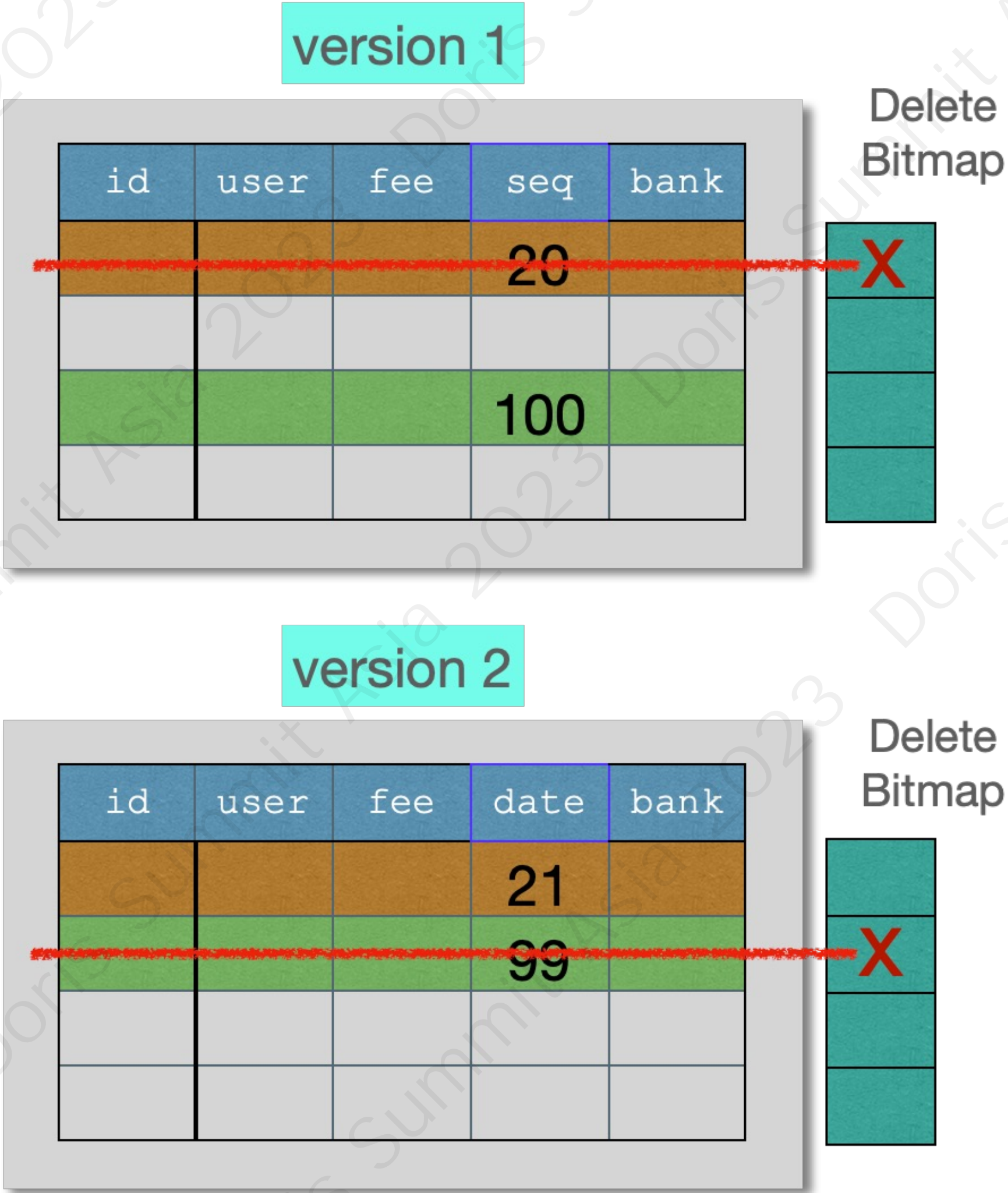


# 条件更新

当数据乱序到达时，仍然能按照原始的更新顺序生效

- 需要单独指定 Sequence Column
- 当数据乱序到达时，即使旧数据比新数据后写入Doris，系统会保证旧数据不会覆盖新的数据

```
CREATE TABLE test.test_table
(
  user_id bigint,
  date date,
  group_id bigint,
  modify_date date,
  keyword VARCHAR(128)
)
UNIQUE KEY(user_id, date, group_id)
DISTRIBUTED BY HASH (user_id) BUCKETS 32
PROPERTIES(
  "function_column.sequence_col" = 'modify_date',
  "replication_num" = "1"
);
```



## 复杂条件更新与删除

### UPDATE

-- 根据表 t2和 t3的 join结果更新表 t1中的数据

UPDATE t1

SET t1.c1 = t2.c1, t1.c3 = t2.c3 \* 100

FROM t2 INNER JOIN t3 ON t2.id = t3.id

WHERE t1.id = t2.id;

### DELETE

-- 根据表 t2和 t3的 join结果删除表 t1中的数据

DELETE FROM t1

USING t2 INNER JOIN t3 ON t2.id = t3.id

WHERE t1.id = t2.id;

# 5 用户实践



# Doris 2.0 高效数据更新

## 游戏行为分析场景

- 存量数据 300亿，单副本数十TB，表包含几十个字段，MoW
- 15 并发 flink connector 任务做 upsert，峰值吞吐 40w行/s

## 消费金融场景

- 宽表拼接的场景，使用MoW表的部分列更新
- 200 列，20 并发，每个并发更新 10 列
- 平均数据可见时间降低明显 27s -> 16s

## 物流运单分析场景

- 半年的物流运单数据分析，宽表 200 多个字段，MoW
- 8 并发 flink connector 任务做 upsert，10s checkpoint，吞吐 6w行/s

## 某客户 PoC 压力测试

- 40 并发 flink connector 任务做 upsert，10s checkpoint
- 导入稳定，吞吐可达 7.49w行/s

## 支付订单分析场景

- 支持月统计（数据规模千亿左右）、年统计（数据规模万亿左右）
- upsert 吞吐 10w行 /s



获取更多社区动态与最佳实践

### Apache Doris 官方平台:

- Apache Doris 官网: [doris.apache.org](https://doris.apache.org)
- Apache Doris GitHub: [github.com/apache/doris/](https://github.com/apache/doris/)

### 获取更多峰会资料:

- Doris Summit 峰会官网: [doris-summit.org.cn](https://doris-summit.org.cn)
- Doris Summit 峰会回放: <https://space.bilibili.com/1196172099/channel/collectiondetail?sid=1824324>