SENSORS Data

基于 Impala 构建实时用户行为分析引擎

fulili@sensorsdata.cn

SENSORS Data

大纲

- 什么是用户行为分析
- 整体架构及数据模型
- 实时导入的实现
- 查询性能优化

SENSORS Data

什么是用户行为

- 用户行为：谁在什么时候干了什么事情
  - Who: 参与者，即用户
  - When: 行为发生的时间
  - Where: 行为发生的地点，可以从 IP 地址、GPS 定位等信息中获取
  - How: 进行行为的方式，包括使用的设备、渠道、环境信息等
  - What: 具体做了什么，例如以 4888 的价格购买了一个 iPad
- 用户行为数据本质上是一种特化的日志数据

SENSORS Data

典型应用

1

运营监控

- 昨天的PV和UV有多少?
- 上个月销售表现如何?
- 近期活跃用户数变化趋势?

2

产品改进

- 用户粘性如何?
- 产品核心流程的转化如何?
- 新功能的使用情况怎样?

3

商业决策

- 是否要开展天津地区业务?
- 应该加大哪个渠道的广告投放?
- 哪个大区的地推团队表现最优异?

SENSORS Data

需求特点

- 需求特点：
  - 时间轴、大量维度、维度取值分散
  - 分析灵活性要求高，查询模式变化多
  - 要求实时响应
  - 查询频率较低
- 灵活性 > 及时性 > 时效性

SENSORS Data

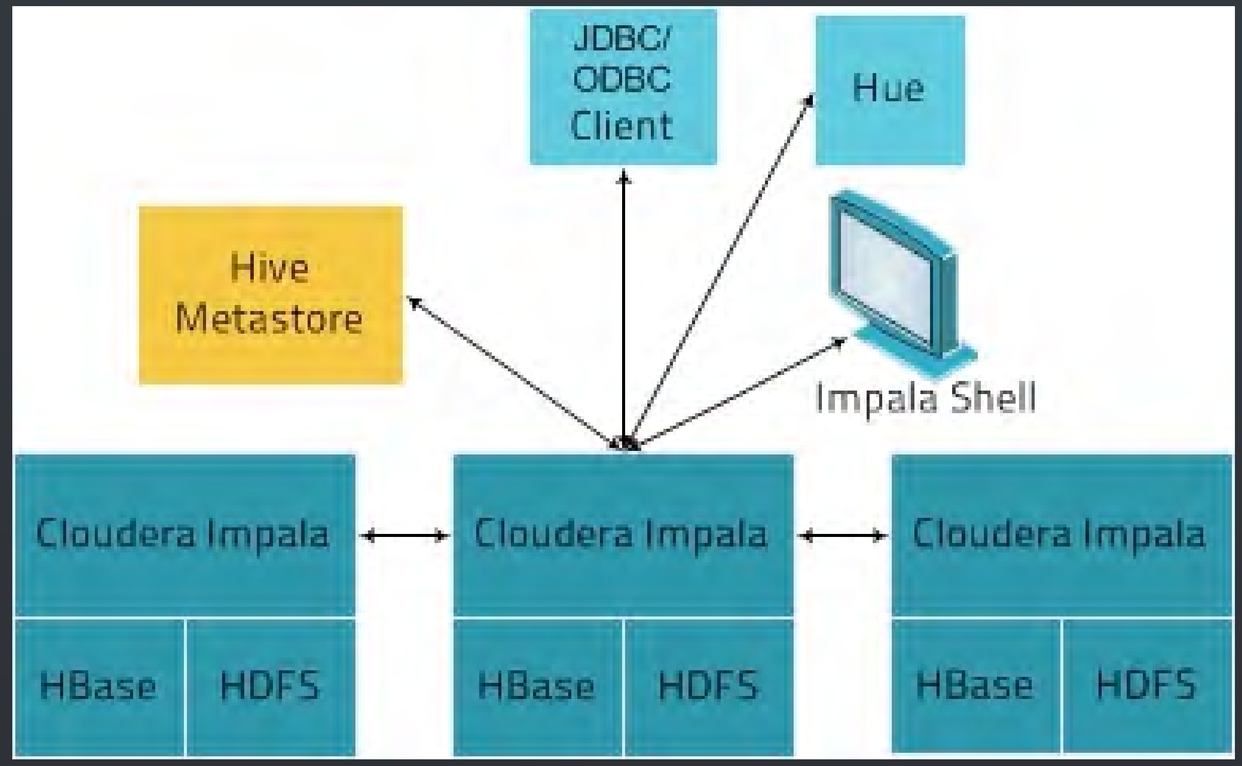
选择查询引擎

- 什么是合适的查询引擎
  - 足够灵活：支持 SQL
  - 足够快：实现交互式查询

SENSORS Data

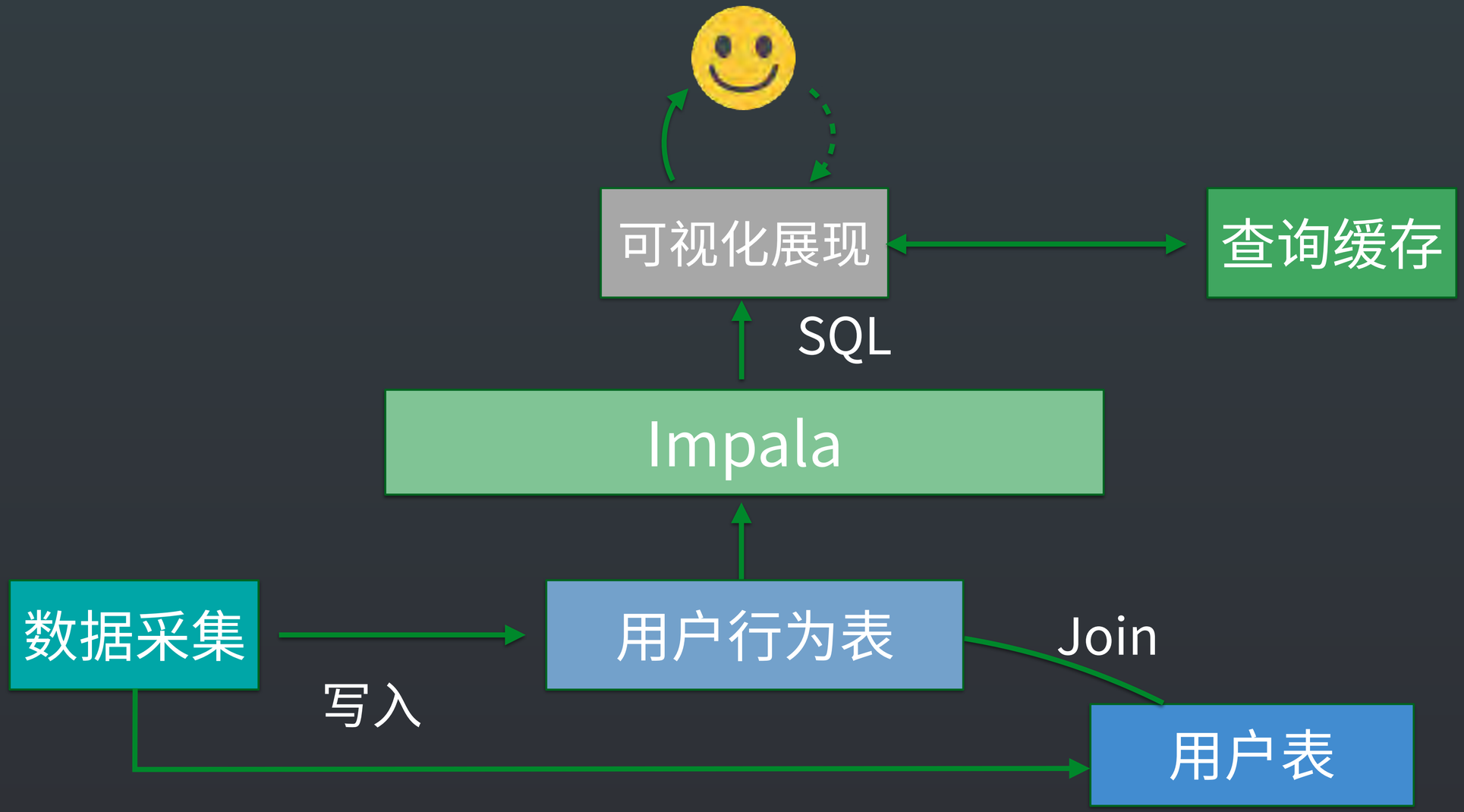
Impala 架构

- 大部分功能特性和 Hive 类似
- 基于 MPP 的查询引擎
- 较低的容错性
- 较高的内存需求
- 较高的查询效率



SENSORS Data

基于 Impala 的系统架构



SENSORS Data

简化的数据模型

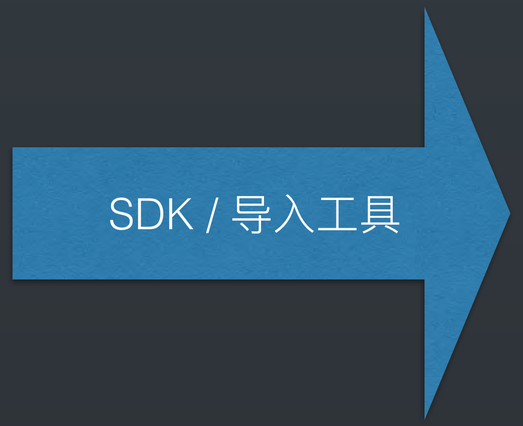
客户端操作

服务端日志

订单数据

注册数据

其它业务数据



用户行为表

用户表

SENSORS Data

用户行为表

时间	用户	事件	渠道	搜索关键词	价格
2015-03-01 00:00	123	注册	Baidu		
2015-03-01 00:01	123	登录			
2015-03-01 10:00	123	搜索		iPad	
2015-03-01 03:00	123	支付订单			4888

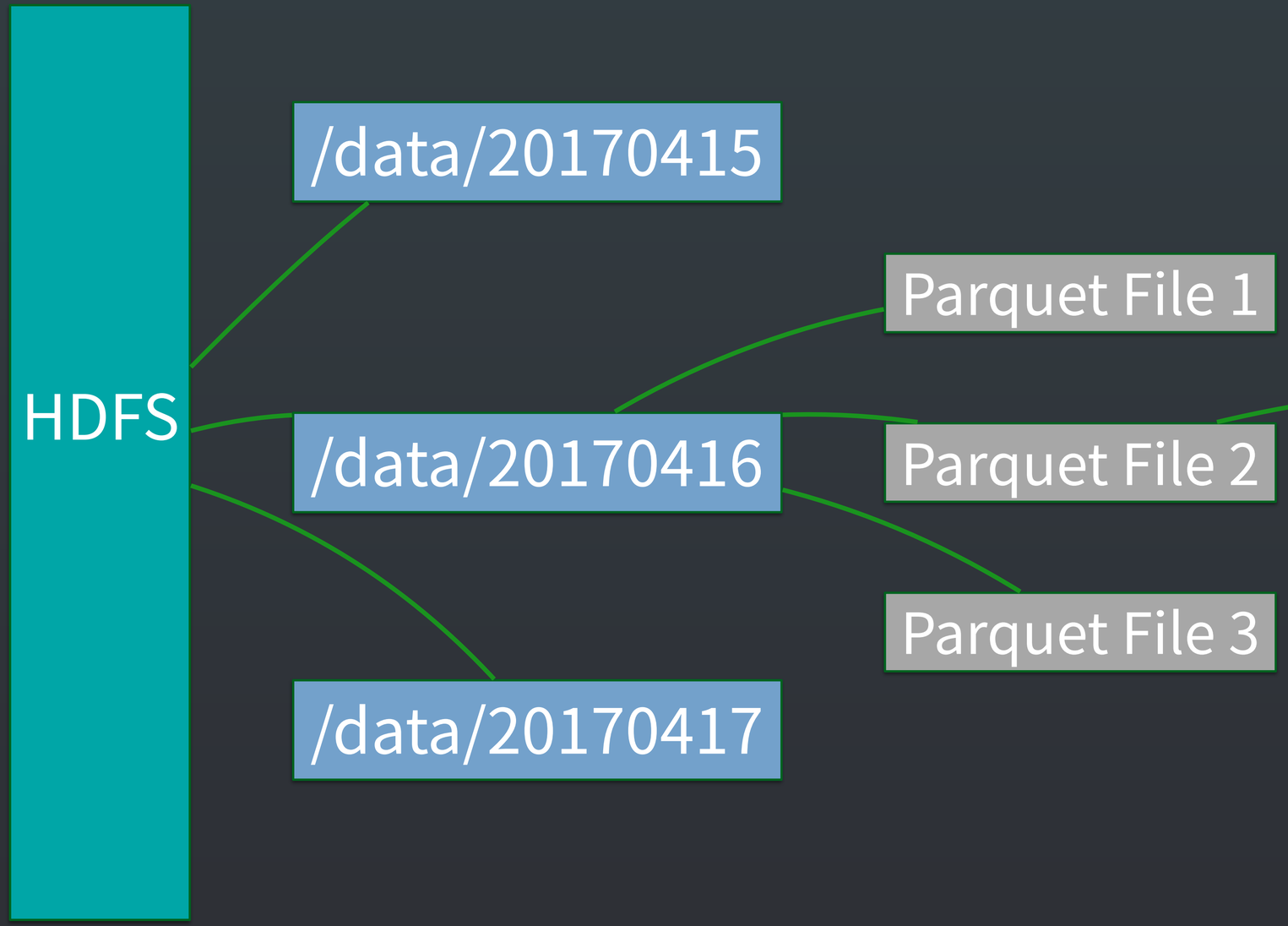
SENSORS Data

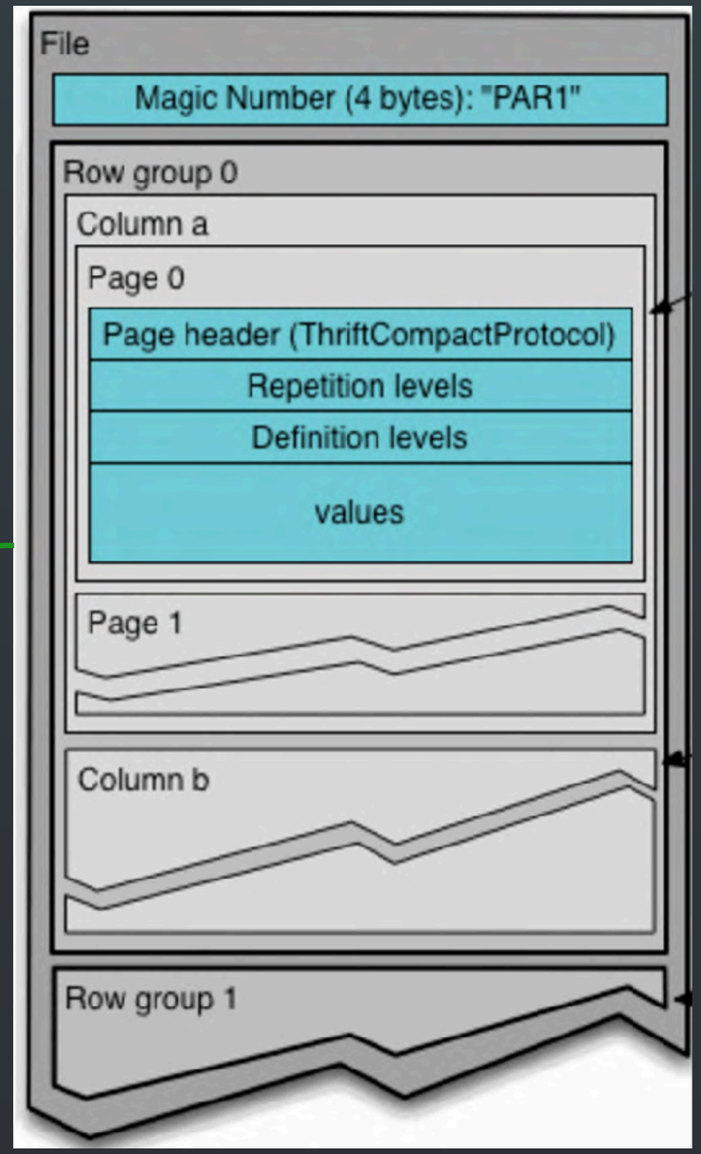
用户表

用户ID	性别	注册渠道	会员等级
123	男	Baidu	1
456	男	官网	2
789	女	头条	3
888	女	未知	4

SENSORS Data

数据存储格式





SENSORS Data

存储格式

- 使用 Parquet 作为主存储格式
  - 列存
  - 按时间分区
  - 局部排序
- 问题：只支持批量写入，无法追加，无法实现实时写入

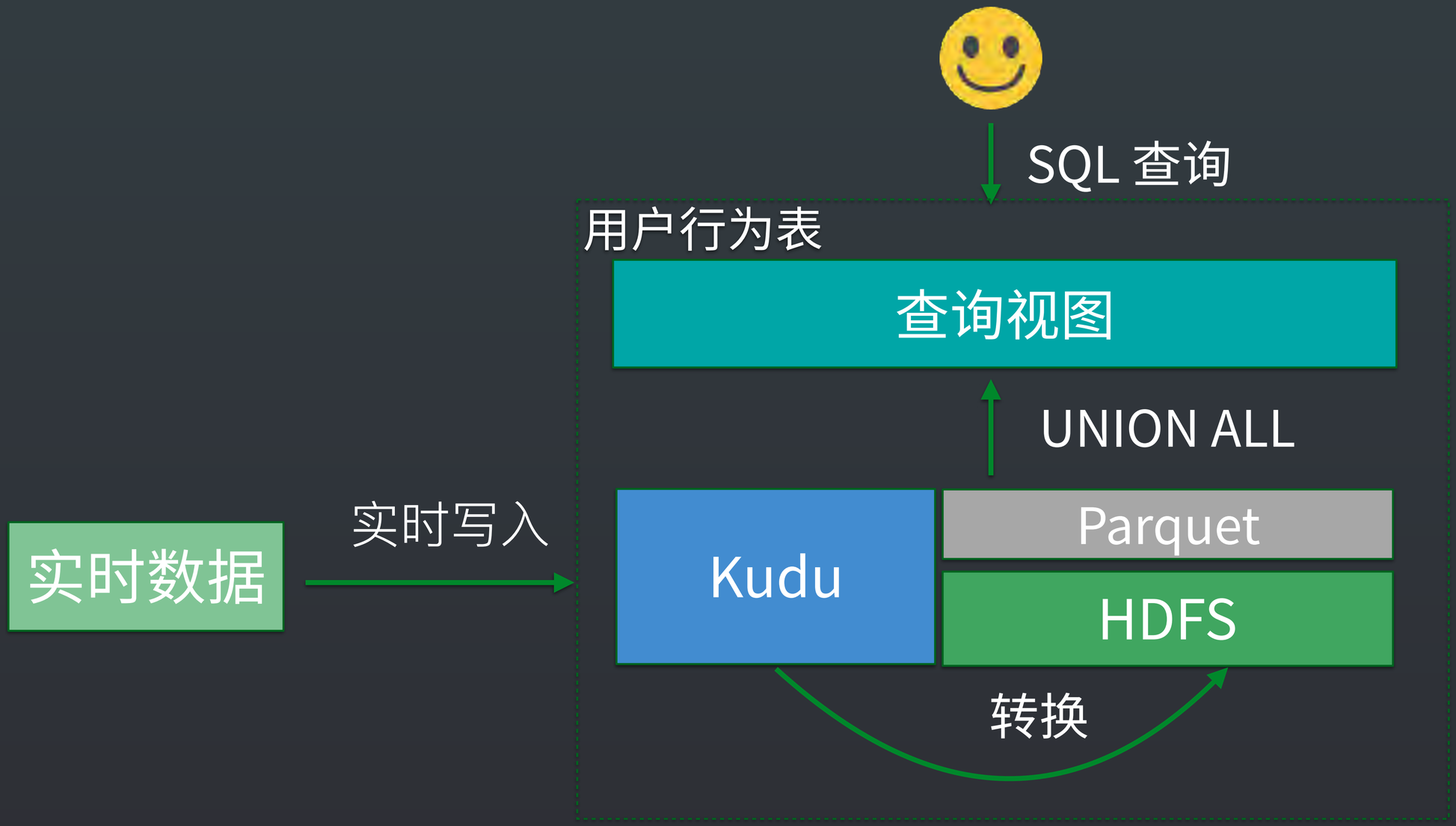
SENSORS Data

引入 Kudu

- Kudu：新一代面向实时分析的存储引擎
  - 底层使用类似 Parquet 的存储结构
  - 支持实时写入、实时更新及随机查询
  - 扫描性能比 Parquet 略差

SENSORS Data

Parquet 和 Kudu 的融合



SENSORS Data

Parquet 和 Kudu 的融合

- 同时使用两种存储格式
  - Kudu 存储实时数据、Parquet 存储历史数据
  - 定时进行数据转储 Kudu -> Parquet
- 使用视图进行无缝融合，对查询层完全透明
  - 优化 UNION ALL 的实现，消除不必要的数据拷贝

SENSORS Data

基本查询优化

- 合理的硬件：CPU、IO、网络、内存
- 写好 SQL
- 性能：3 节点，10秒内完成 10 亿条行为数据的分组聚合。

SENSORS Data

实现查询抽样

- 为什么要做抽样：节约成本、提高效率
- 查询抽样 vs 采集抽样：
  - 存储便宜，应该尽量存储最全的数据
  - 抽样分析的局限性，例如细分维度的分析

SENSORS Data

实现查询抽样

- 抽样逻辑：按照用户抽样，以保证一个用户的行为的完整性。
- 抽样实现：
  - 根据用户 ID 的 Hash 值得到抽样编号
  - Parquet / Kudu 数据按照抽样编号排序
  - Impala 进行查询扫描时根据需要只扫描需要部分的数据

SENSORS Data

实现转化漏斗

- 复杂查询的一个例子：转化漏斗
  - 任意定义的转化逻辑
  - 任意选择的查询时段
  - 得到每个用户精确的转化信息

SENSORS Data

实现转化漏斗

- 计算逻辑
  - 从用户行为表抽取需要的事件数据
  - 事件数据按照用户 ID 分组，每个分组内按照时间排序
  - 进行具体的转化逻辑计算
- 实现方式
  - 实现自定义的分析函数（UDAnF），需要对 Impala 进行改造
  - 优化最耗时的步骤：全局排序 -> 预排序 + 归并排序

SENSORS Data

Join 优化

- 事件表：累计千亿，每天增加数亿
- 用户表：累计数亿，每天增加数十万
- 使用场景：最近 7 天不同会员等级的用户的累计成交量

SENSORS Data

Join 优化

- 优化逻辑：
  - 使用每天的活跃用户数据构建 Bloom Filter
  - Join 之前先用 Bloom Filter 对用户表进行过滤
- 优化效果：Join 右表的数据量从数亿降到数千万

SENSORS Data

总结

- 简化的数据模型
- 使用支持 SQL 的通用查询引擎：Impala + Kudu
- 针对应用场景进行针对性的性能优化

SENSORS Data

神策，帮你实现数据驱动!