实时计算产品案例——实时数据仓库

实时计算产品经理 付空

实时数据仓库

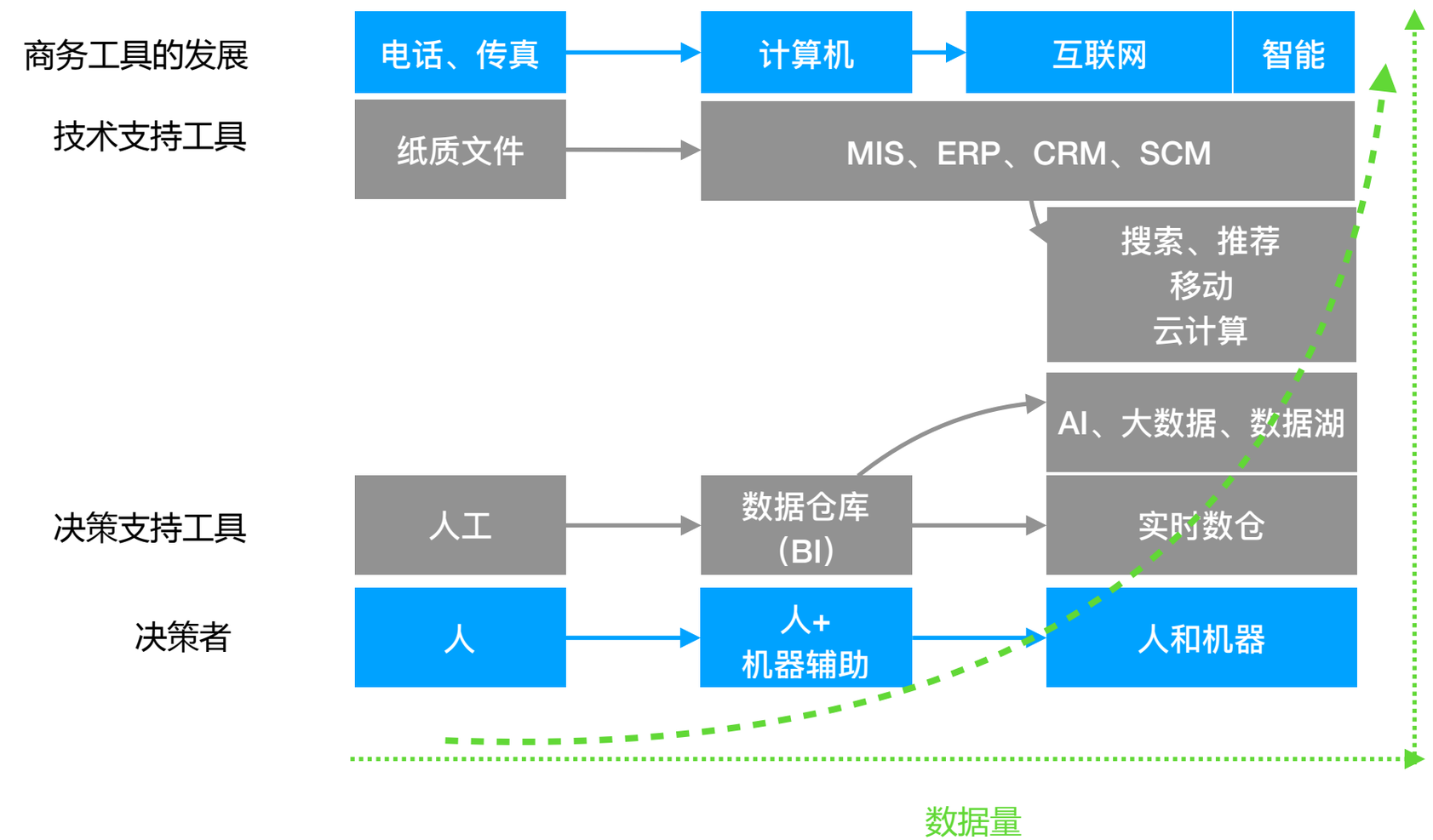
数据仓库是一个面向主题的（Subject Oriented）、集成的（Integrate）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策。

数据仓库是伴随着企业信息化发展起来的，在企业信息化的过程中，随着信息化工具的升级和新工具的应用，数据量变的越来越大，数据格式越来越多，决策要求越来越苛刻，数据仓库技术也在不停的发展。

数据仓库的趋势：

实时数据仓库以满足实时化&自动化决策需求；

大数据&数据湖以支持大量&复杂数据类型（文本、图像、视频、音频）；



实时数据仓库

数据仓库工具的发展

数据仓库有两个环节：数据仓库的构建与数据仓库的应用。

早期数据仓库构建主要指的是把企业的业务数据库如ERP、CRM、SCM等数据按照决策分析的要求建模并汇总到数据仓库引擎中，其应用以报表为主，目的是支持管理层和业务人员决策（中长期策略型决策）。
随着业务和环境的发展，这两方面都在发生着剧烈变化。

业务和环境的变化
随着IT技术走向互联网、移动化，数据源变得越来越丰富，在原来业务数据库的基础上出现了非结构化数据，比如网站log，IoT设备数据，APP埋点数据等，这些数据量比以往结构化的数据大了几个量级，对ETL过程、存储都提出了更高的要求；
互联网的在线特性也将业务需求推向了实时化，随时根据当前客户行为而调整策略变得越来越常见，比如大促过程中库存管理，运营管理等（即既有中远期策略型，也有短期操作型）；
同时公司业务互联网化之后导致同时服务的客户剧增，有些情况人工难以完全处理，这就需要机器自动决策。比如欺诈检测和用户审核。
总结来看，对数据仓库的需求可以抽象成两方面：实时产生结果、处理和保存大量异构数据。

注：这里不讨论数据湖技术。

实时数据仓库

数据仓库工具的发展

原始状态

计算&存储大量异构数据

构建：大数据离线计算

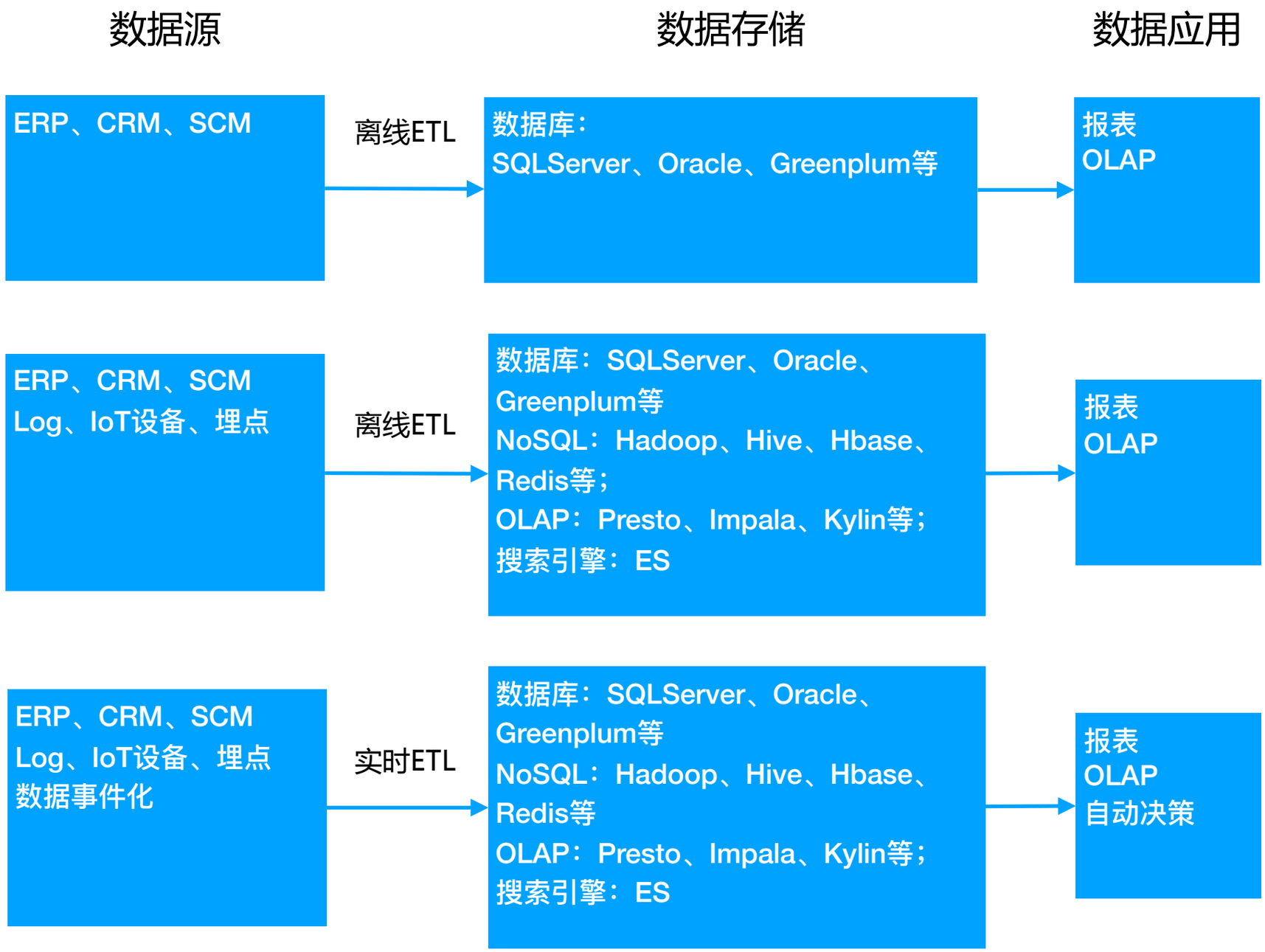
存储：NoSQL存储

实时产生结果

构建：数据事件化、实时ETL

存储：OLAP引擎

应用：OLAP、自动决策



实时数据仓库

数据仓库方法论

面向主题

从公司业务出发，是分析的宏观领域，比如供应商主题、商品主题、客户主题和仓库主题

元数据

为多维数据分析服务

数据立方体

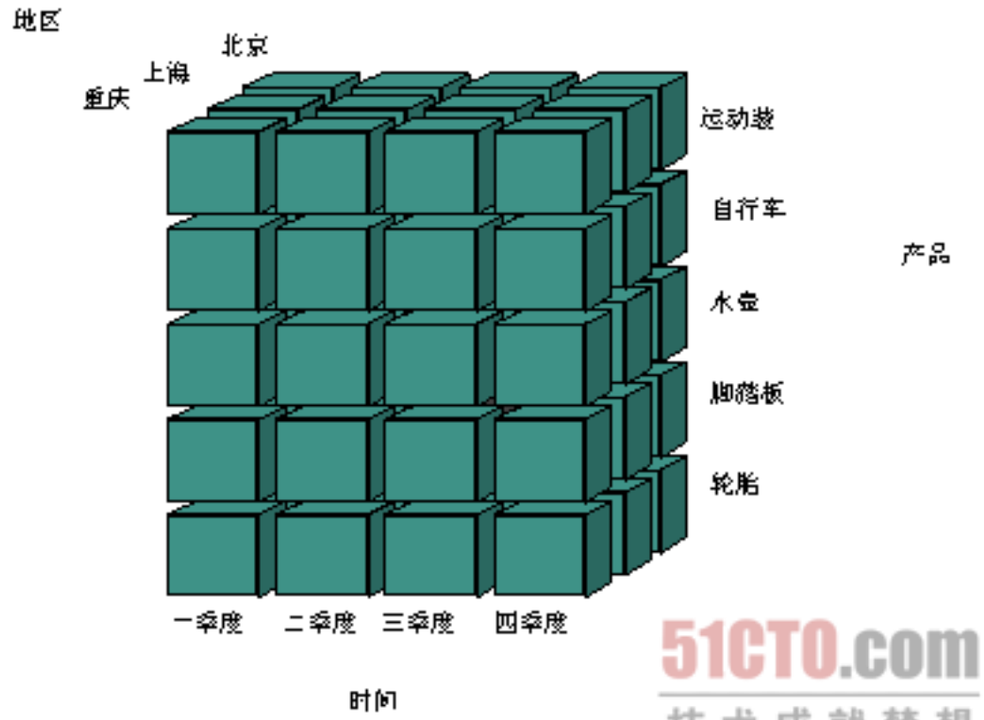
上卷、下钻、切片、旋转

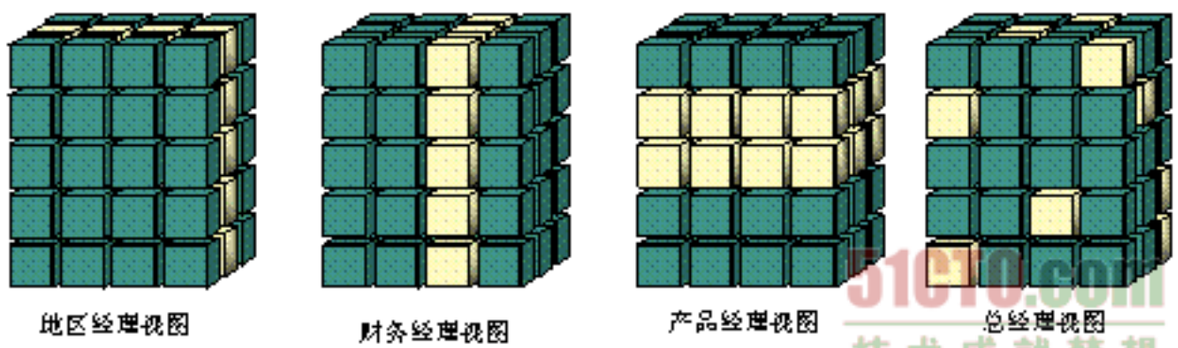
反范式数据模型

星型数据模型

实事表

维度表





注：图像来自51CTO网站

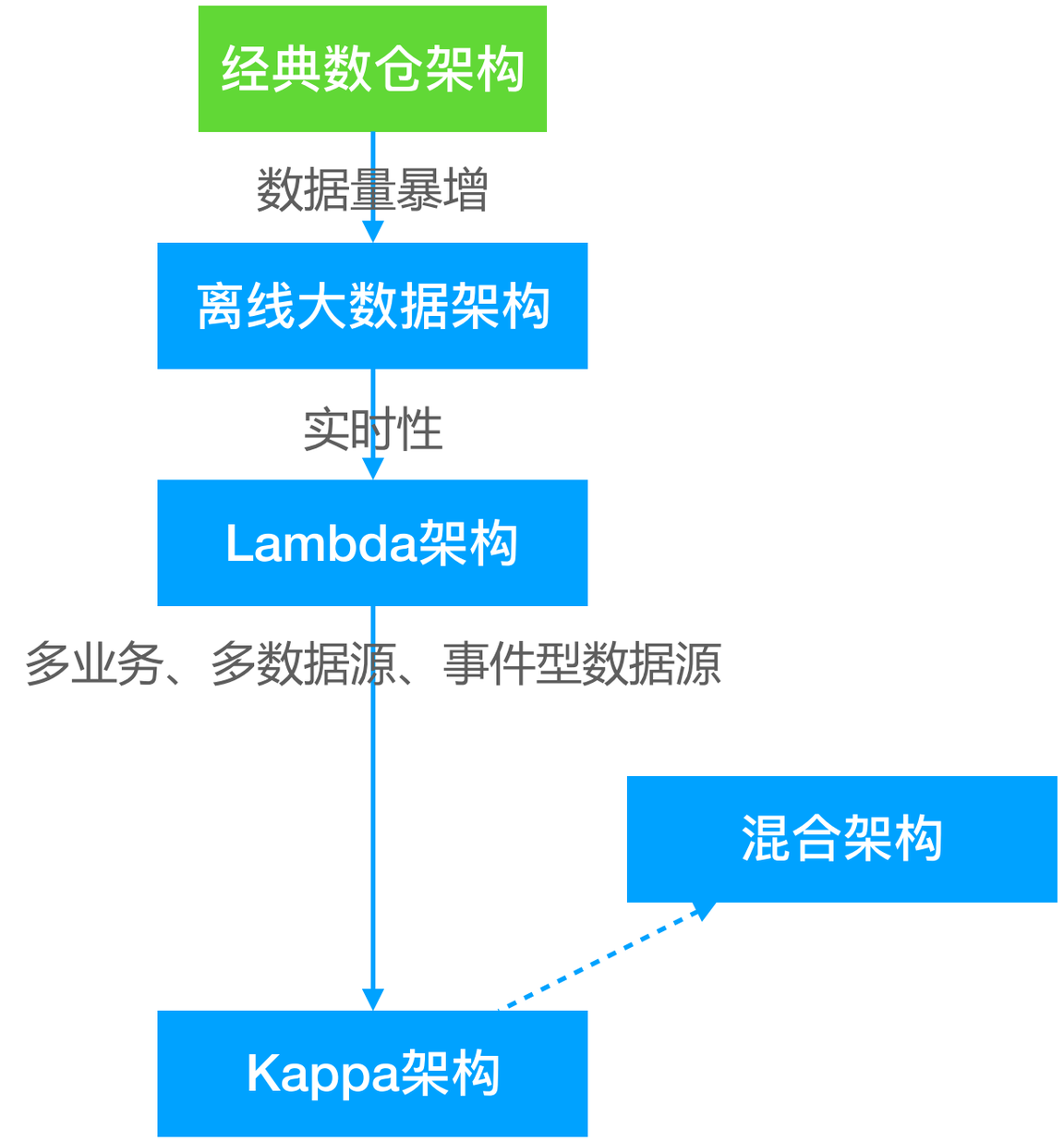
实时数据仓库

数据仓库架构的演变

数据仓库概念是Inmon于1990年提出并给出了完整的建设方法。随着互联网时代来临，数据量暴增，开始使用大数据工具来替代经典数仓中的传统工具。此时仅仅是工具的取代，架构上并没有根本的区别，可以把这个架构叫做离线大数据架构。

后来随着业务实时性要求的不断提高，人们开始在离线大数据架构基础上加了一个加速层，使用流处理技术直接完成那些实时性要求较高的指标计算，这便是Lambda架构。

再后来，实时的业务越来越多，事件化的数据源也越来越多，实时处理从次要部分变成了主要部分，架构也做了相应调整，出现了以实时事件处理为核心的Kappa架构。



实时数据仓库

数据仓库架构的演变

离线大数据架构

数据源通过离线的方式导入到离线数仓中。

下游应用根据业务需求选择直接读取DM或加一层数据服务，比如mysql 或 redis。

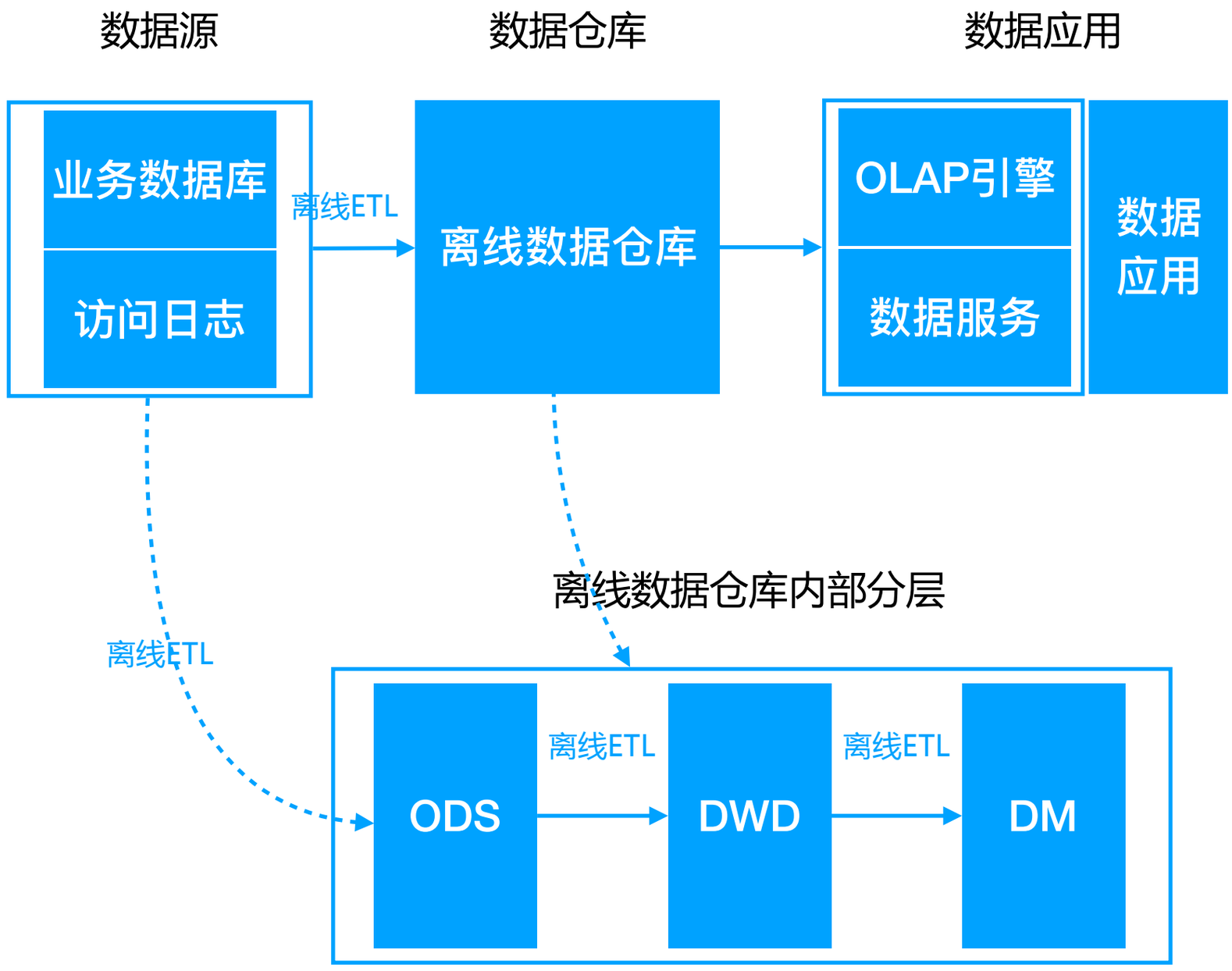
数据仓库分为三层：

ODS，操作数据层，保存原始数据；

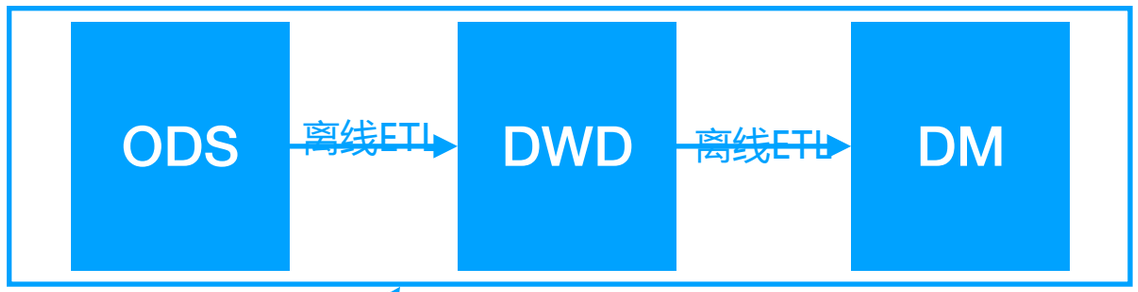
DWD，数据仓库明细层，根据主题定义好事实与维度表，保存最细粒度的事实数据；

DM，数据集市/轻度汇总层，在DWD层的基础之上根据不同的业务需求做轻度汇总；

典型的数仓存储是HDFS/Hive，ETL可以是MapReduce脚本或HiveSQL。



实时数据仓库



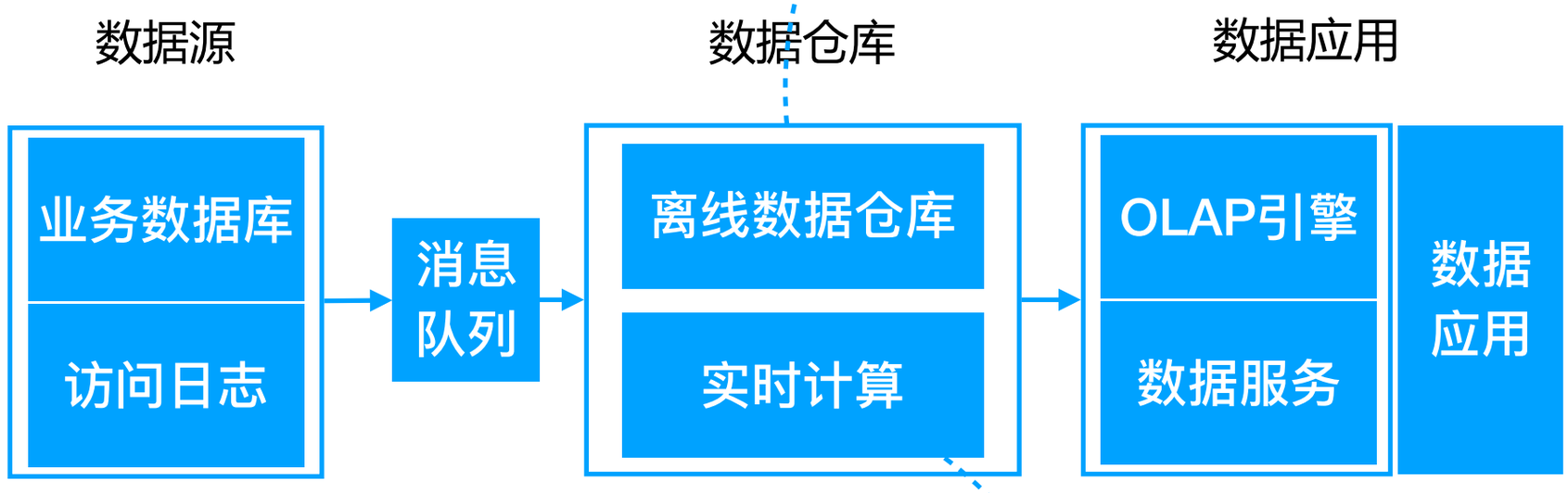
离线数据仓库内部分层

数据仓库架构的演变

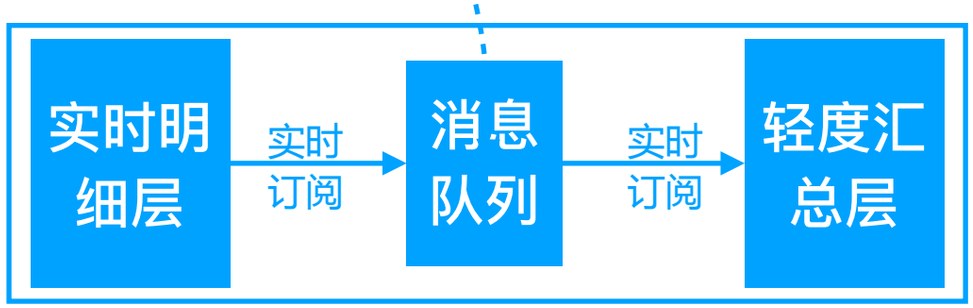
Lambda架构

为了计算一些实时指标，就在原来离线数仓的基础上增加了一个实时计算的链路，并对数据源做流式改造（即把数据发送到消息队列），实时计算去订阅消息队列，直接完成指标增量的计算，推送到下游的数据服务中去，由数据服务层完成离线&实时结果的合并。

注：流处理计算的指标批处理依然计算，最终以批处理为准，即每次批处理计算后会覆盖流处理的结果。



流处理引擎不完善时的折中



实时计算内部分层

可直接完成计算，也可以像离线那样分层，取决于指标复杂度分层的话中间用消息队列桥接

Lambda架构问题：

1.同样的需求需要开发两套一样的代码

这是Lambda架构最大的问题，两套代码不仅仅意味着开发困难（同样的需求，一个在批处理引擎上实现，一个在流处理引擎上实现，还要分别构造数据测试保证两者结果一致），后期维护更加困难，比如需求变更后需要分别更改两套代码，独立测试结果，且两个作业需要同步上线。

2.资源占用增多：同样的逻辑计算两次，整体资源占用会增多（多出实时计算这部分）

实时数据仓库

数据仓库架构的演变

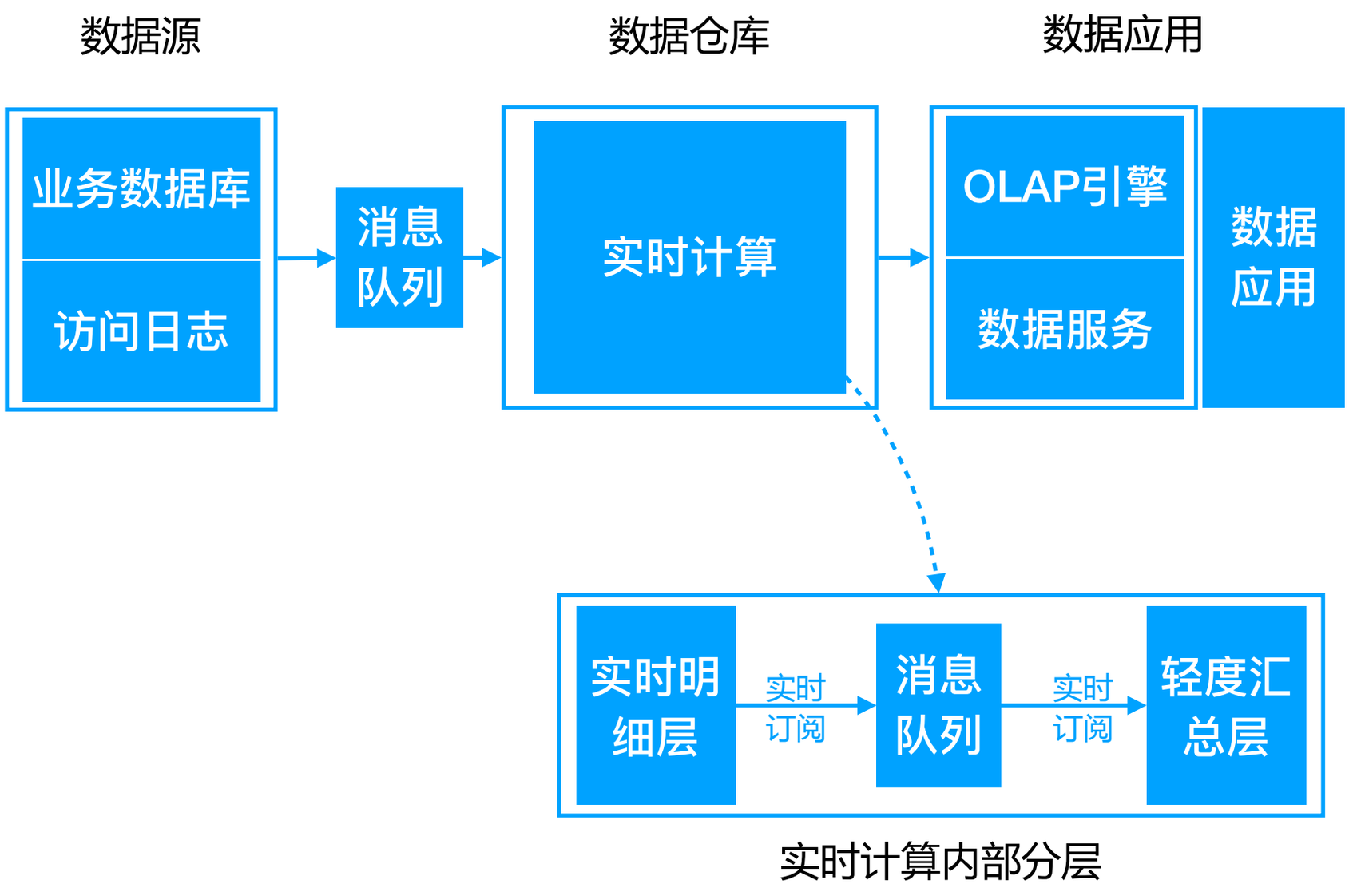
Kappa架构

Lambda架构虽然满足了实时的需求，但带来了更多的开发与运维工作，其架构背景是流处理引擎还不完善，流处理的结果只作为临时的、近似的值提供参考。后来随着Flink等流处理引擎的出现，流处理技术很成熟了，这时为了解决两套代码的问题，LickedIn 的Jay Kreps提出了Kappa架构

Kappa架构可以认为是Lambda架构的简化版（只要移除lambda架构中的批处理部分即可）。

在Kappa架构中，需求修改或历史数据重新处理都通过上游重放完成。

Kappa架构最大的问题是流式重新处理历史的吞吐能力会低于批处理，但这个可以通过增加计算资源来弥补。



实时计算内部分层

可直接完成计算，也可以像离线那样分层，取决于指标复杂度分层的话中间用消息队列桥接

实时数据仓库

数据仓库架构的演变

Kappa架构的重新处理过程

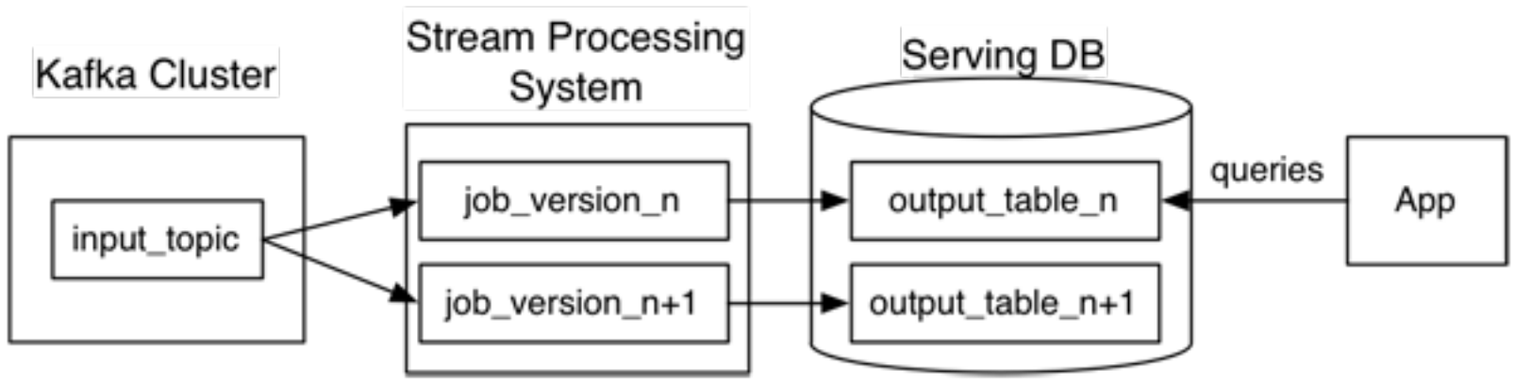
重新处理是人们对Kappa架构最担心的点，但实际上并不复杂：

1.选择一个具有重放功能的、能够保存历史数据并支持多消费者的消息队列，根据需求设置历史数据保存的时长，比如Kafka，可以保存全部历史数据。

2当某个或某些指标有重新处理的需求时，按照新逻辑写一个新作业，然后从上游消息队列的最开始重新消费，把结果写到一个新的下游表中。

3.当新作业赶上进度后，应用切换数据源，读取2中产生的新结果表。

4.停止老的作业，删除老的结果表。



实时数据仓库

数据仓库架构的演变

Lambda架构 与 Kappa架构对比

对比项	Lambda架构	Kappa架构
实时性	实时	实时
计算资源	批和流同时运行，资源开销大	只有流处理，仅针对新需求开发阶段运行两个作业，资源开销小
重新计算时吞吐	批式全量处理，吞吐较高	流式全量处理，吞吐较批处理低
开发、测试	每个需求都需要两套不同代码，开发、测试、上线难度较大	只需实现一套代码，开发、测试、上线难度相对较小
运维成本	维护两套系统（引擎），运维成本大	只需维护一套系统（引擎），运维成本小

实时数据仓库

数据仓库架构的演变

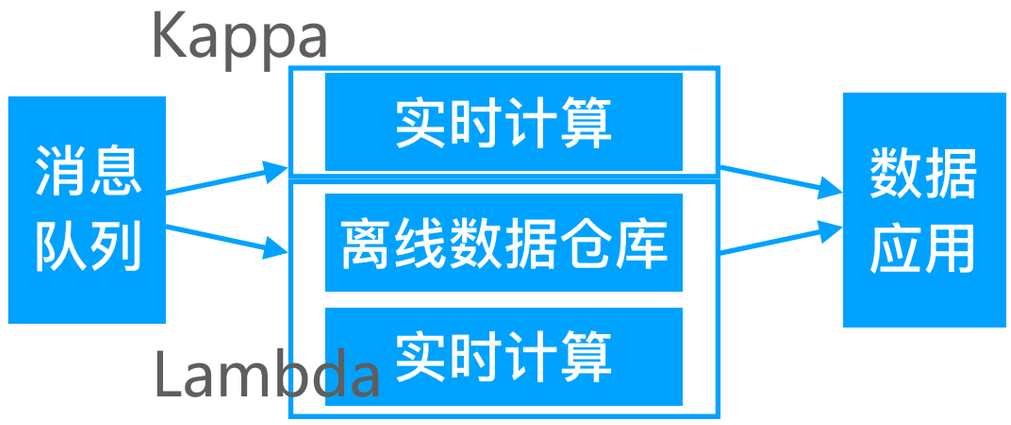
实时数仓-Lambda架构 与 Kappa架构总结

前面介绍了Lambda架构与Kappa架构的含义及优缺点，在真实的场景中，很多时候并不是完全规范的Lambda架构或Kappa架构，可以是两者的混合，比如大部分实时指标使用Kappa架构完成计算，少量关键指标（比如金额相关）使用Lambda架构用批处理重新计算，增加一次校对过程。（1）

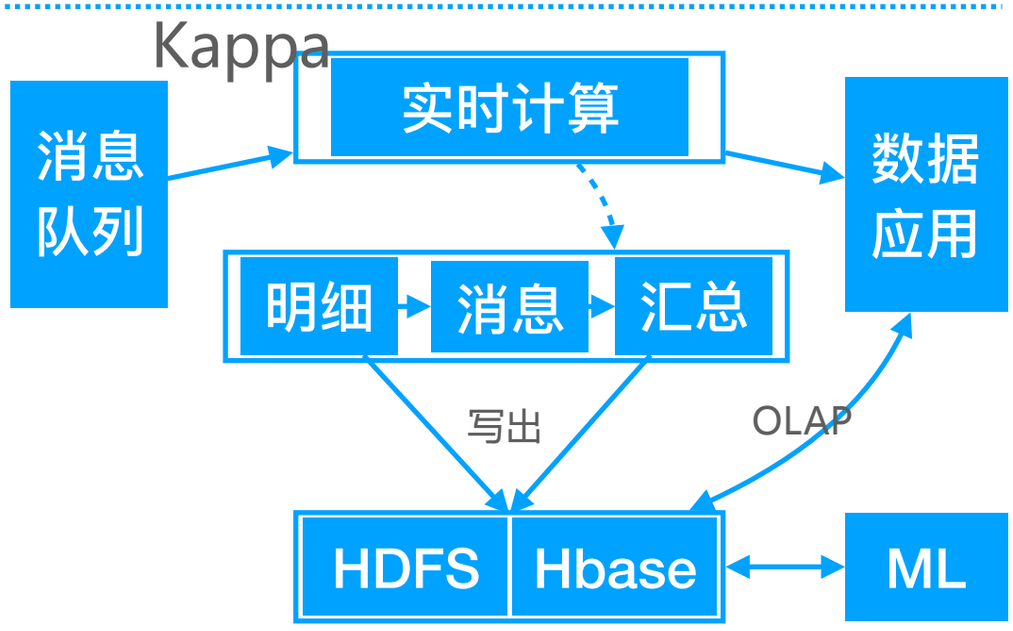
Kappa架构并不是中间结果完全不落地，现在很多大数据系统都需要支持机器学习（离线训练），所以实时中间结果需要落地对应的存储引擎供机器学习使用，另外有时候还需要对明细数据查询，这种场景也需要把实时明细层写出到对应的引擎中。（2）参考后面的案例

另外，随着数据多样性的发展，数据仓库这种提前规定schema的模式显得越来难以支持灵活的探索&分析需求，这时候便出现了一种数据湖技术，即把原始数据全部缓存到某个大数据存储上，后续分析时再根据需求去解析原始数据。简单的说，数据仓库模式是schema on write，数据湖模式是schema on read。（3）

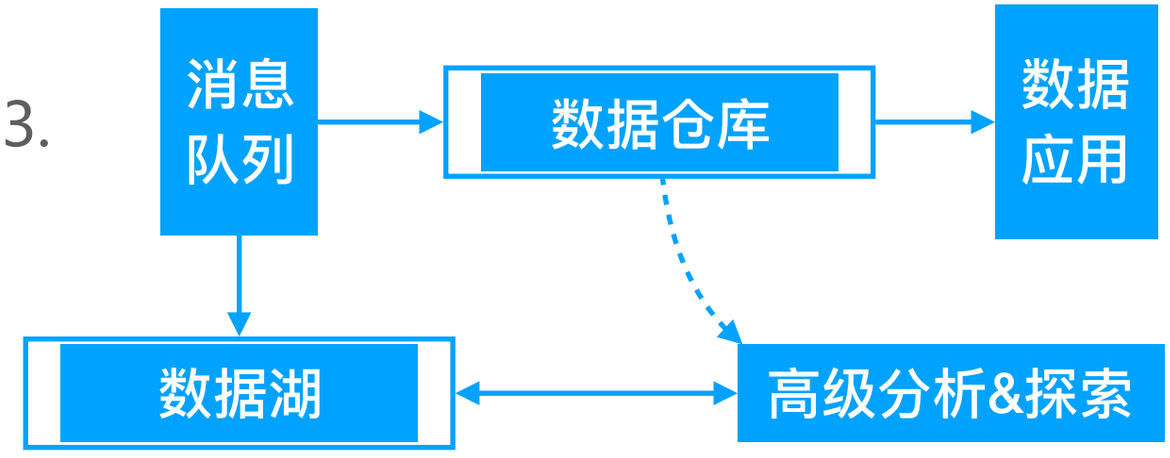
1.



2.



3.



实时数据仓库

实时数仓案例

菜鸟仓配实时数据仓库

本案例参考自菜鸟仓配团队的分享，涉及全局设计、数据模型、数据保障等几个方面。

整体设计如右图，基于业务系统的数据，数据模型采用中间层的设计理念，建设仓配实时数仓；计算引擎，选择更易用、性能表现更佳的实时计算作为主要的计算引擎；数据服务，选择天工数据服务中间件，避免直连数据库，且基于天工可以做到主备链路灵活配置秒级切换；数据应用，围绕大促全链路，从活动计划、活动备货、活动直播、活动售后、活动复盘五个维度，建设仓配大促数据体系

注：特别感谢缘桥同学的无私分享。

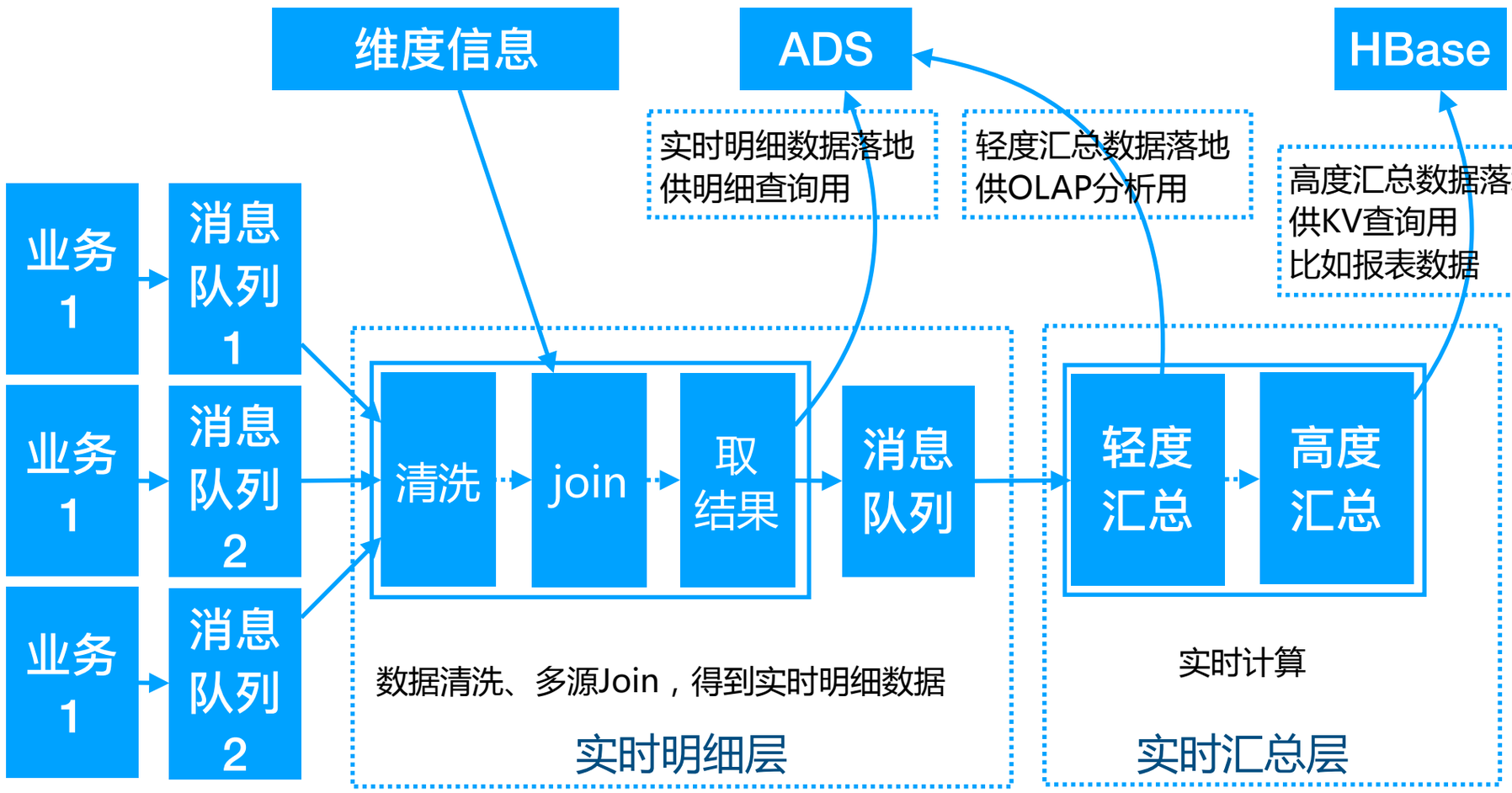
实时数据仓库

实时数仓案例

菜鸟仓配实时数据仓库

数据模型

不管是从计算成本，还是从易用性，还是从复用性，还是从一致性……，我们都必须避免烟囱式的开发模式，而是以中间层的方式建设仓配实时数仓，与离线中间层基本一致，我们将实时中间层分为两层。



实时数据仓库

实时数仓案例

菜鸟仓配实时数据仓库

数据模型

第一层DWD公共实时明细层

实时计算订阅业务数据消息队列，然后通过数据清洗、多数据源join、流式数据与离线维度信息等的组合，将一些相同粒度的业务系统、维表中的维度属性全部关联到一起，增加数据易用性和复用性，得到最终的实时明细数据。这部分数据有两个分支，一部分直接落地到ADS，供实时明细查询使用，一部分再发送到消息队列中，供下层计算使用；

第二层DWS公共实时汇总层

以数据域+业务域的理念建设公共汇总层，与离线数仓不同的是，这里汇总层分为轻度汇总层和高度汇总层，并同时产出，轻度汇总层写入ADS，用于前端产品复杂的olap查询场景，满足自助分析；高度汇总层写入Hbase，用于前端比较简单的kv查询场景，提升查询性能，比如产出报表等；

注：
1.ADS是一款提供OLAP分析服务的引擎。开源提供类似功能的有，Elastic Search、Kylin、Druid等；
2.案例中选择把数据写入到Hbase供KV查询，也可根据情况选择其他引擎，比如数据量不多，查询压力也不大的话，可以用mysql
3.因主题建模与业务关系较大，这里不做描述

实时数据仓库

实时数仓案例

菜鸟仓配实时数据仓库

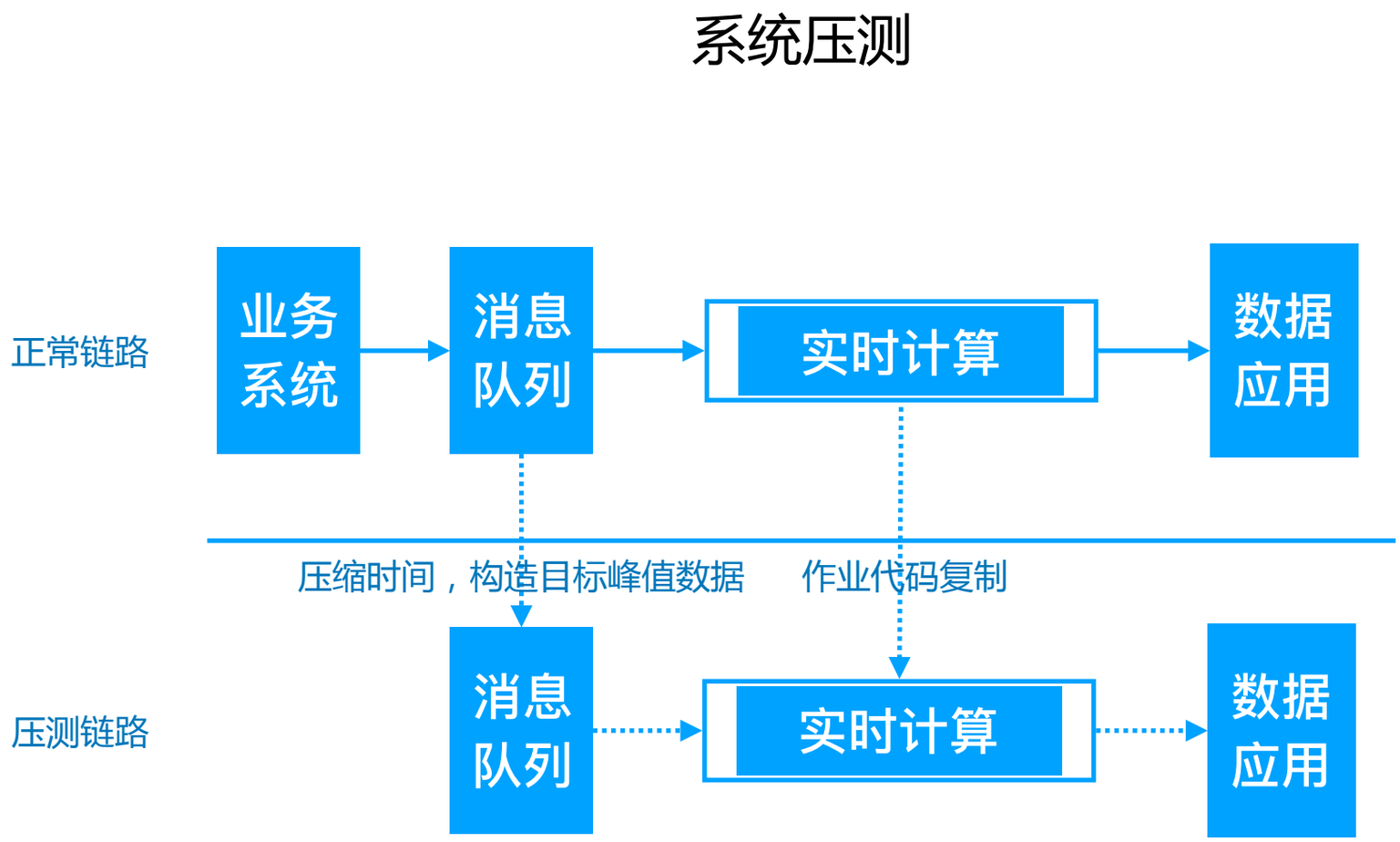
数据保障

集团每年都有双十一等大促，大促期间流量与数据量都会暴增。

实时系统要保证实时性，相对离线系统对数据量要更敏感，对稳定性要求更高。

所以为了应对这种场景，还需要在这种场景下做两种准备：

1.大促前的系统压测；
2.大促中的主备链路保障；



压测的主要目的是产出实时计算在大促过程所需资源及其配置

实时数据仓库

实时数仓案例

菜鸟仓配实时数据仓库

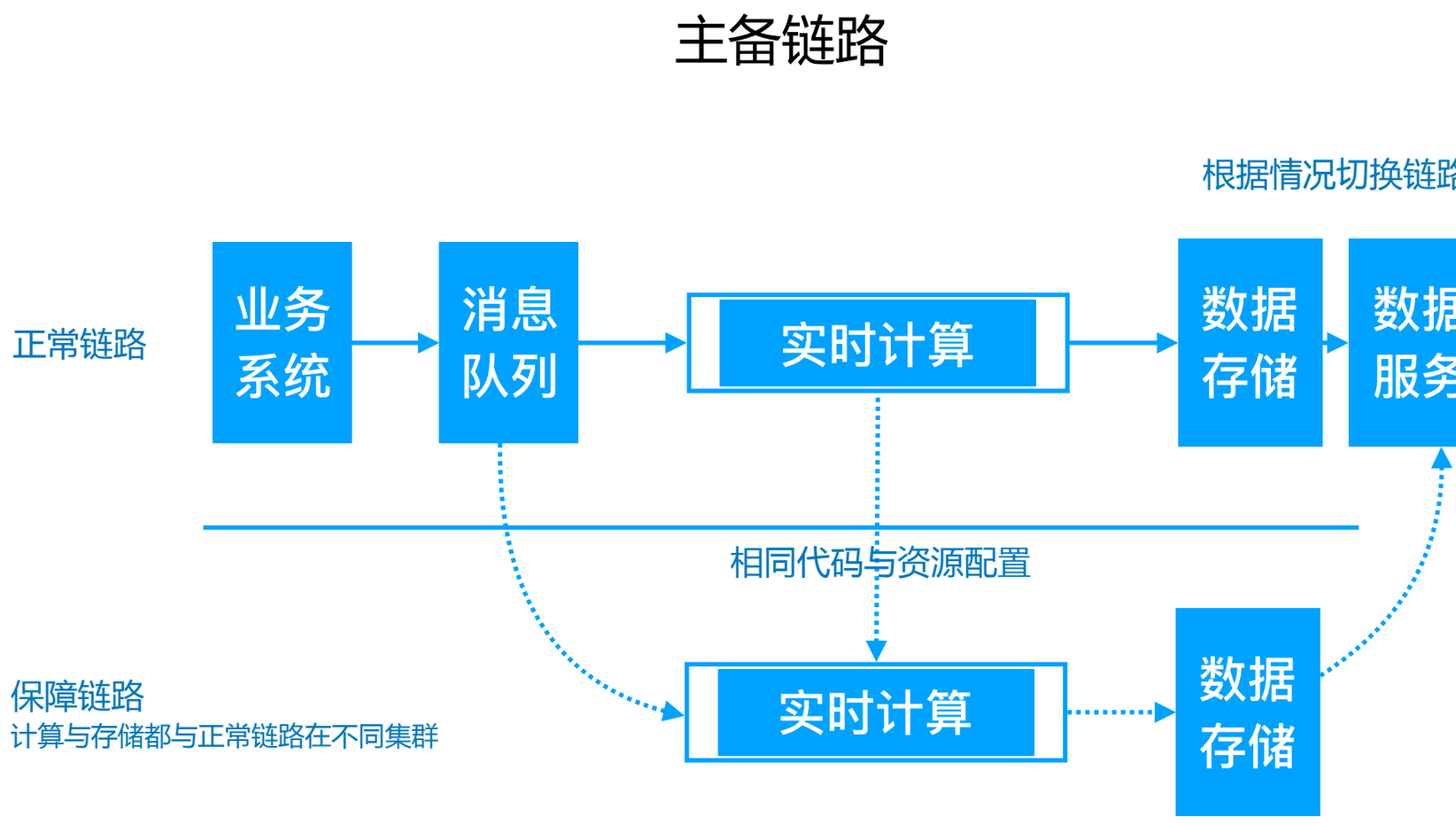
数据保障

集团每年都有双十一等大促，大促期间流量与数据量都会暴增。

实时系统要保证实时性，相对离线系统对数据量要更敏感，对稳定性要求更高。

所以为了应对这种场景，还需要在这种场景下做两种准备：

1.大促前的系统压测；
2.大促中的主备链路保障；



主备链路保障的目的是在主链出现问题能通过备链提供服务，可以只针对高优先级的作业做主备链路，并且不限于一条备链。

实时数据仓库

数据仓库

实时数仓与离线数仓的对比

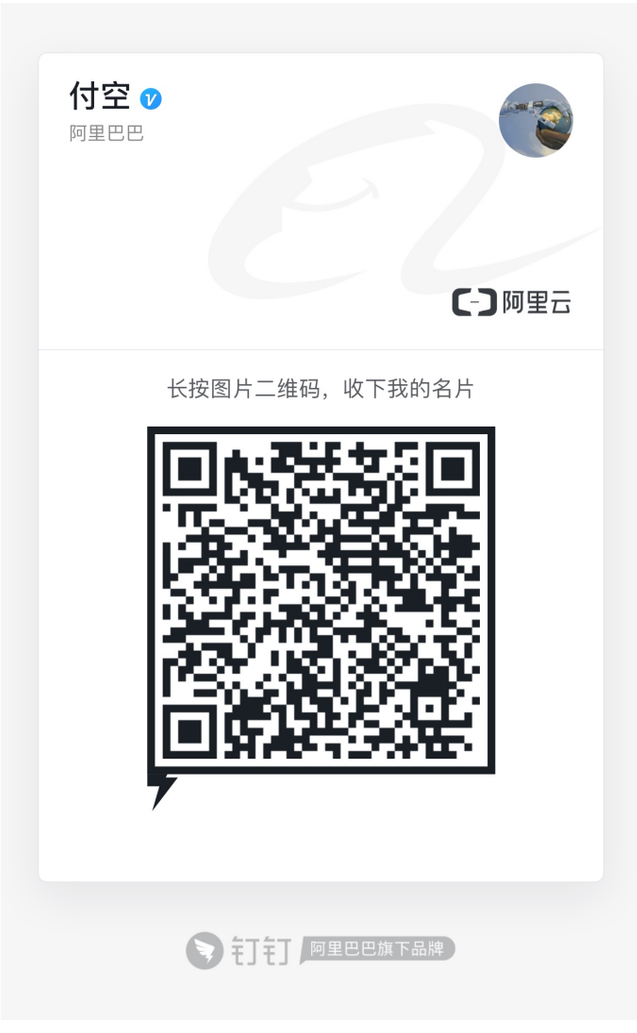
在看过前面的叙述与菜鸟案例之后，我们看一下实时数仓与离线数仓在几方面的对比：

首先，从架构上，实时数仓与离线数仓有比较明显的区别，实时数仓以Kappa架构为主，而离线数仓以传统大数据架构为主。Lambda架构可以认为是两者的中间态。

其次，从建设方法上，实时数仓和离线数仓基本还是沿用传统的数仓主题建模理论，产出事实宽表。另外实时数仓中实时流数据的join有隐藏时间语义，在建设中需注意。

最后，从数据保障看，实时数仓因为要保证实时性，所以对数据量的变化较为敏感。在大促等场景下需要提前做好压测和主备保障工作，这是与离线数据的一个较为明显的区别。

如何有需要，欢迎沟通



没有钉钉可加微信