# 概述

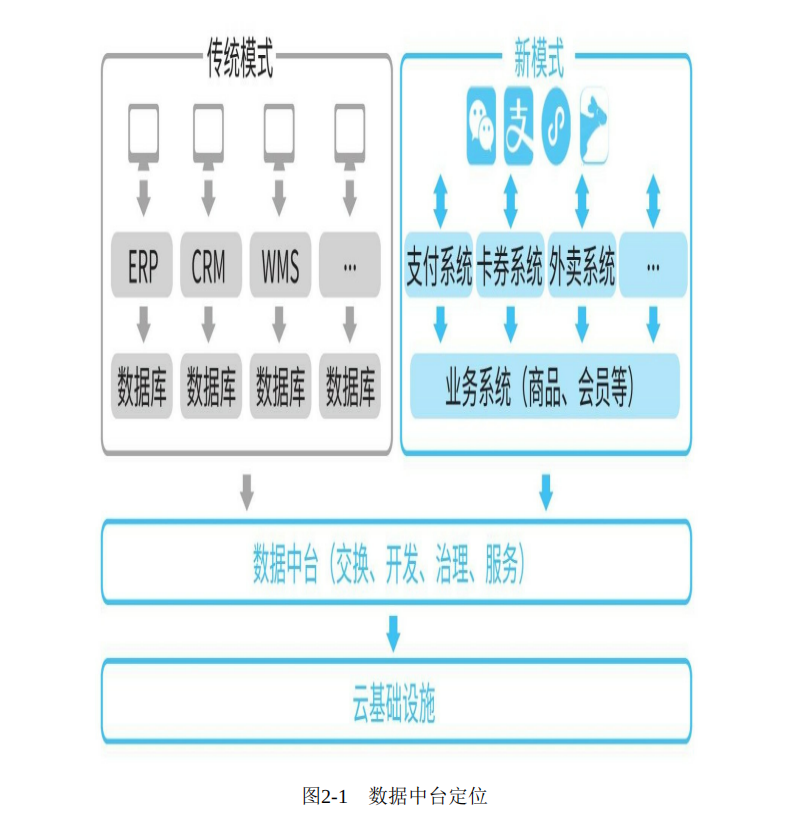
数据中台的需求不是来源于外部，而是来自内部，来自企业对自身未来 发展的担忧。数据中台是增援未来，是以发展的观点解决企业面临的问题，面对不确定的未来，企业无法确认今天的数据未来会怎么用，会产生什么样的价值，所以才需要数据中台。现在把数据源源不断地接进 来，源源不断地进行资产化、服务化，未来当企业看清楚业务场景，把 对数据的需求输入数据中台时，才知道原来数据可以这样使用，才知道 怎么去适配。数据中台是对未来场景的能力支撑，是增援未来的能力。

数据中台是一套可持续“让企业的数据用起来”的机制，是一种战略选择和组织形式，是依据企业特有的业务模式和组织架构，通过有形的产品和实施方法论支撑，构建的一套持续不断把数据变成资产并服务于业务的机制。

# **数据构成**

**企业全量数据的数据空间大致由三个维度构成**：自主生产和消费的数据、外部数据（含单向外部获取数据和单向对外提供数 据）、内外部交互数据。

# **数据中台定位**



# **数据中台核心能力**

数据中台必备的4个核心能力：汇聚整合、提纯加工（安全的访问机制、完善的质量体系、可扩展标签、面向业务主题、智能的数据映射）、服务可视化、价值变现。



# **数据中台与业务中台关系**

**业务中台与数据中台的区别与联系**：业务中台是抽象业务流程的共性形成通用业务服务能力，而数据中台则是抽象数据能力的共性形成通用数据服务能力。业务中台中沉淀的业务数据进入到数据中台进行体系化的加工，再 以服务化的方式支撑业务中台上的应用，而这些应用产生的新数据又流转到数据中台，形成循环不息的数据闭环。

# **数据中台与数据仓库关系**

**数据中台包含数据仓库**，是远大于数据仓库的。

# **数据中台技术需求**

**技术需求**：比如： 要保持原来的报表需求，仍需要保持批量离线计算的能力（Hadoop、 Oracle RAC）； 针对准实时的指标统计和实时推荐，需要**实时流式计算**的能力 （Storm、Spark Streaming、Flink）； 针对决策类业务如海量人群的圈人需求和ad-hoc需求，需要即席计算能力（Greenplum、Elasticsearch、Impala）； 针对高并发业务场景（如用户画像），需要在线计算能力（MySQL、 Redis、Oracle）。 因此，企业需要一个统一的数据中台来满足**离线／实时**计算需求、各种查询需求（实时查询和ad hoc），同时在将来新数据引擎（更快的计算框架，更快的查询响应）出现时，又不需要重构目前的大数据体系。

# **数据中台建设方法论体系**

数据中台建设方法论体系，需要从组织、保障、准则、内容、步骤5个层面全面考虑，以确保数据中台建设和实施能如期完成。

·1种战略行动：把用数据中台驱动业务发展定位为企业级战略，全局谋划。

·2项保障条件：通过宣导统一组织间的数据认知，通过流程加速组织变革。

·3条目标准则：将数据的可见、可用、可运营3个核心准则始终贯穿于 中台建设的全过程，保障建设在正确轨道上。

·4套建设内容：通过技术体系、数据体系、服务体系、运营体系建设保 证中台建设的全面性和可持续性。

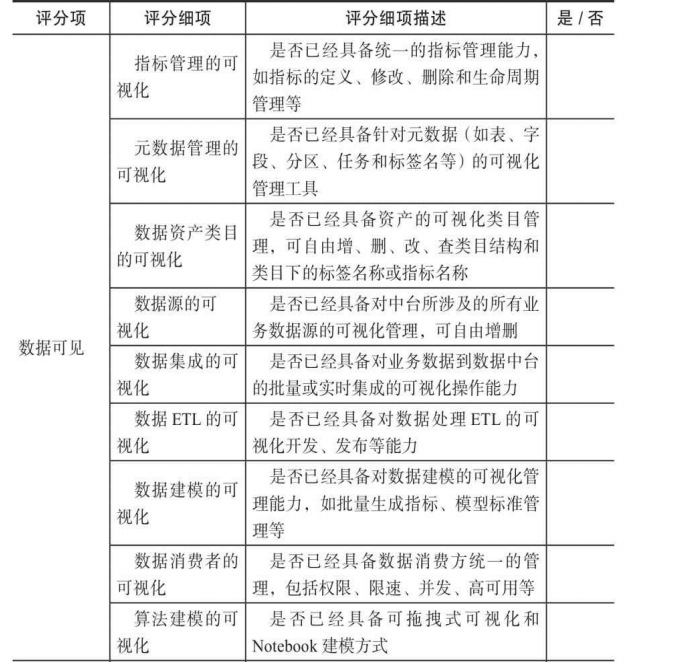
·5个关键步骤：通过理现状、立架构、建资产、用数据、做运营5个关键行动控制中台建设关键节点的质量。

# **数据中台四个意识**

**数据中台需要注意**：数据采集意识（各种采集、广泛采集）、数据标准化意识（之所以需要进行数据治理，是因为数据不标准。如果希望数据发挥价值，就需要保持统一数据标准的意识，只有不同部门、不同业务对于数据的理解都一致了，才能减少因数据口径不一导致的资源浪费。）、数据使用意识、数据安全意识。

# **数据中台建设准则**

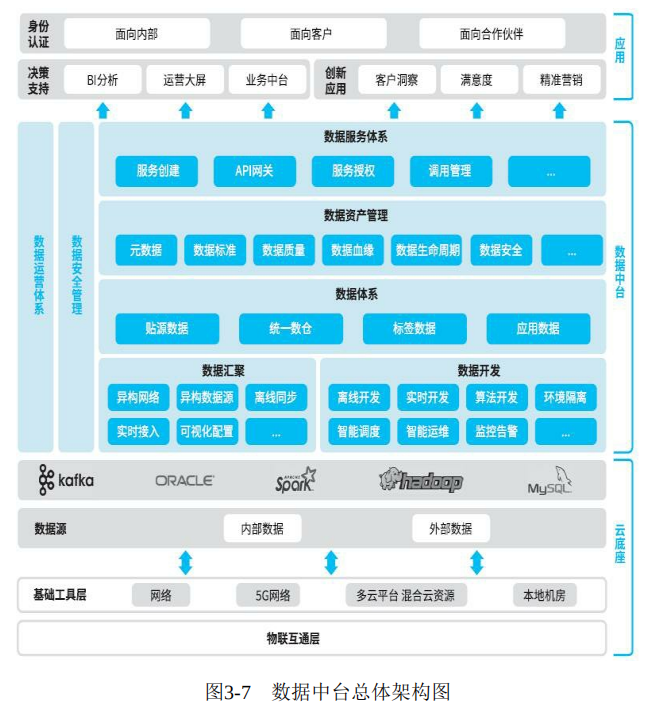
**建设准则**：







# **数据中台架构**



**数据汇聚**是数据中台数据接入的入口。数据中台本身几乎不产生数据，所有数据来自于**业务系统、日志、文件、网络**等，这些数据分散在不同的网络环境和存储平台中，难以利用，很难产生业务价值。数据汇聚是数据中台必须提供的核心工具，把各种异构网络、异构数据源的数据方便地采集到数据中台中进行集中存储，为后续的加工建模做准备。数据

汇聚方式一般有**数据库同步、埋点、网络爬虫、消息队列**等；从汇聚的时效性来分，有**离线批量汇聚**和**实时采集。**

**数据开发**通过数据汇聚模块汇聚到中台的数据没有经过处理，基本是按照数据的原始状态堆砌在一起的，这样业务还是很难使用。数据开发是一整套数据加工以及加工过程管控的工具，有经验的数据开发、算法建模人员利用数据加工模块提供的功能，可以快速把数据加工成对业务有价值的形式，提供给业务使用。数据开发模块主要面向开发人员、分析人员，提供离线、实时、算法开发工具，以及任务的管理、代码发布、运维、监控、告警等一系列集成工具，方便使用，提升效率。

**数据体系:**有了数据汇聚、数据开发模块，中台已经具备传统数据仓库（后面简称：数仓）平台的基本能力，可以做数据的汇聚以及各种数据开发，就可以建立企业的数据体系。之前说数据体系是中台的血肉，开发、管理、使用的都是数据。大数据时代，数据量大，增长快，业务对数据的依赖也会越来越高，必须考虑数据的一致性和可复用性，垂直的、烟囱式的数据和数据服务的建设方式注定不能长久存在。不同的企业因业务不同导致数据不同，数据建设的内容也不同，但是建设方法可以相似， 数据要统一建设，笔者建议数据按照贴源数据、统一数仓、标签数据、 应用数据的标准统一建设。

**数据资产管理:**通过数据体系建立起来的数据资产较为偏技术，业务人员比较难理解。资产管理是以企业全员更好理解的方式，把企业的数据资产展现给企业全员（当然要考虑权限和安全管控），数据资产管理包括对数据资产目录、元数据、数据质量、数据血缘、数据生命周期等进行管理和展示，以一种更直观的方式展现企业的数据资产，提升企业的数据意识。

**数据服务体系：**前面利用数据汇聚、数据开发建设企业的数据资产，利用数据管理展现 企业的数据资产，但是并没有发挥数据的价值。数据服务体系就是把数 据变为一种服务能力，通过数据服务让数据参与到业务，激活整个数据中台，数据服务体系是数据中台存在的价值所在。企业的数据服务是千变万化的，中台产品可以带有一些标准服务，但是很难满足企业的服务诉求，大部分服务还是需要通过中台的能力快速定制。数据中台的服务 模块并没有自带很多服务，而是提供快速的服务生成能力以及服务的管 控、鉴权、计量等功能。

**运营体系和安全管理：**通过前面的数据汇聚、数据开发、数据体系、数据资产管理、数据服务体系，已经完成了整个数据中台的搭建和建设，也已经在业务中发挥一定的价值。运营体系和安全管理是数据中台得以健康、持续运转的基础， 如果没有它们，数据中台很可能像个一般项目一样，会在搭建起平台、建设部分数据、尝试一两个应用场景之后而止步，无法正常地持续运营，不能持续发挥数据的应用价值。这也就完全达不到建设数据中台的目标。

# **数据中台应用场景**

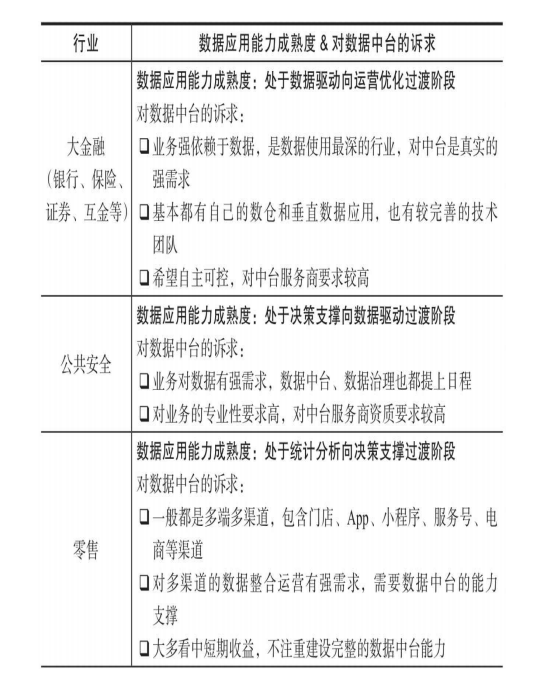
第一，能够追溯数据资产的形成过程，包括涵盖了哪些数据来源，经过了怎样的加工环节，涉及哪些业务环节和部门等；

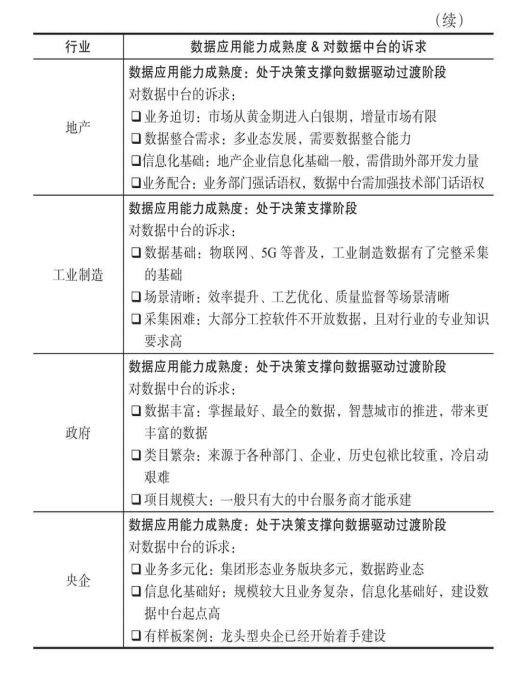
第二，能及时获取到数据资产当前的状态，尤其是数据质量和安全情况，如更新频率、合规性、空值率等；

第三，能够知道数据资产被哪些业务调用了，以通过建立数据闭环了解和追溯数据资产所带来的业务价值；

第四，能够对整个数据中台从数据采集到数据应用的整个链路建立监控体系，便于及时发现和排除故障，保障数据资产的稳定性；

第五，建立丰富的数据内外部共享和服务渠道，实现数据价值的释放和交换。只有同时满足上述五个条件时，企业才有足够的信息来源来支撑整个数据资产的运营及迭代优化。





# **数据中台调研案例**



# **数据采集**

## 数据采集、汇聚的方法和工具

1. **线上行为采集**：线上行为的主要载体可以分为传统互联网和移动互联网两种，对应的形态有PC系统、PC网页、H5、小程序、App、智能可穿戴设备等。在技术上，数据采集主要有**客户端SDK埋点**和**服务端SDK埋点**等方式。其中 客户端SDK埋点主要是通过在终端设备内嵌入埋点功能模块，通过模块提供的能力采集客户端的用户行为，并上传回行为采集服务端。
2. **线下行为采集**：线下行为数据主要通过一些硬件来采集，如常见的Wi-Fi探针、摄像头、传感器等。随着设备的升级，各种场景中对智能设备的应用也越来越多，安防、客户监测、考勤等都开始深入到生活中。常见的主要有Wi-Fi信号采集、信令数据采集、图像视频采集以及传感器探测等。
3. **互联网数据采集**：网络爬虫又称为网页蜘蛛，是一种按照既定规则自动抓取互联网信息的程序或者脚本，常用来做网站的自动化测试和行为模拟。Google、搜狗、百度等提供的互联网信息检索能力，都是基于它们内部自建的网络爬虫，在遵守相关协议的情况下，**不断爬取互联网上的新鲜网页信息，对内容进行处理后**提供相应的检索服务。
4. **内部数据汇聚**，**数据汇聚不同于数据采集**，数据采集有一定的**数据生产属性**，将终端的用户行为信息通过特定的方法记录后，通过中间系统的流转写入目标存储中。当然，也能通过某种形式在某个数据源中落地，如数据库或日志文件等，然后通过数据汇聚的能力实现数据采集和存储。

## **数据的类型**

**从数据组织形式来分**，数据主要分成三类：

·**结构化数据**：规则、完整，能够通过二维逻辑来表现的数据，严格遵 循数据格式与长度规范，常见的有数据库表、Excel等二维表。

·**半结构化数据**：数据规则、完整，同样严格遵循数据格式与长度规 范，但无法通过二维关系来表现，常见如JSON、XML等形式表达的复 杂结构。

·**非结构化数据**：数据结构不规则或不完整，不方便用二维逻辑表来表现，需要经过复杂的逻辑处理才能提取其中的信息内容，如办公文档、 图片、图像和音视频等。

从时效性和应用场景来分，数据汇聚可以分成离线和实时两类：

·**离线**：主要用**于大批量数据的周期性迁移**，对时效性要求不高，一般采用分布式批量数据同步的方式，通过连接读取数据，读取数据过程中可以有全量、增量的方式，经过统一处理后写入到目标存储。

·**实时**：主要**面向低时延**的数据应用场景，一般通过增量日志或通知消息的方式实现，如通过读取数据库的操作日志（RedoLog、BinLog）来 实现相应的实时处理，业界常见的Canal、MaxWell、StreamSets、NiFi等框架和组件都有较多的实际应用。

## **数据采集工具**

在数据能力建设过程中，很多企业结合自身的场景和最佳实践也开源了 一些优秀的汇聚工具，如Sqoop、DataX、Canal等，适用场景不同，也各有优缺点。

1. Canal Canal Server模拟MySQL Slave的交互协议，伪装自己为MySQL的Slave 向Master发送dump协议，Master收到请求后开始推送binary log，Canal 解析byte流产出解析后的增量数据。**主要优点是流程架构非常清晰，部署和配置等相对简单**，同时可以额外做一些配置管理、开发改造的工作。**Canal的主要缺点是Server中的Instance和Client之间是一对一的消费，不太适用于多消费和数据分发的场景。**
2. Sqoop 是目前市面上相对通用的一种解决方案，是在**结构化数据和HDFS之间进行批量数据迁移**的工具。整体框架以Hadoop为核心，底层使用 MapReduce程序实现，MapReduce天生的特性保证了并行化和高容错率，任务运行在Hadoop集群上，减少了服务器资源的使用情况。其主要**优势**是，**在特定场景下，数据交换过程会有很大的性能提升**。主要**缺点**是，**处理过程定制程度较高，目前主要通过在命令行中配置参数来调整数据同步操作**行为，在用户的一些自定义逻辑和数据同步链路监控方面比较薄弱。除此之外，**任务运行完全依赖于MapReduce**，功能**扩展性**方面受到比较明显的约束和限制。
3. DataX DataX是阿里巴巴开源的一套插件式离线数据交换工具，以实现各种异 构数据源之间的高效数据交换为目标而设计，提供数据交换作业全链路 的流量监控，将作业本身的状态、数据流量、数据速度、执行进度等信息进行展示，提供脏数据探测功能，支持传输过程中对传输报错（如类型转换错误）进行策略化处理。由于它是基于进程内读写直连的方式，高并发数据交换场景下对机器内存要求比较高。除此之外，**DataX不支持非结构化数据的同步**，目前支持结构化数据源、半结构化数据源、非结构化数据源，但是非结构化数据源中需要存储的是一张逻辑意义上的二维表，例如CSV格式的文本信息，本质上还是结构化数据。

# **数据交换产品**

从上文的介绍中可以了解到，这些工具都无法很好地满足企业复杂的数 据交换场景。从数据类型来看，有结构化数据和非结构化数据；从实效性来看，有**实时数据交换和离线数据交换**。另外，数据交换应该是后续数据作业的起点，因此，相应的交换任务调度及状态要能够有效地与上下游形成依赖，借助统一调度的能力构建数据作业流。**数据交换中心的首要目的是屏蔽底层工具的复杂性，以可视化配置的方式提供给企业用户；其次需要考虑，为了解决数据孤岛，需要满足异构存储、异构数据类型的交换需求**；同时，还要考虑不同时效要求下的数据互通。因此，数据交换平台需要**屏蔽系统底层协议、传输安全、特性组件等信息**，让开发人员在数据接入过程中无须关注数据格式转换、数据路由、数据丢失等，只需要关注与业务本身的数据交换部分。企业信息化建设的多种数据源类型，可以通过同步模块的数据源进行统一管理，方便用户快速通过**可视化页面执行数据汇聚工作**。 在构建数据交换中心的实践过程中，基于异构数据源、异构厂商集群、数据应用时效性和相关技术栈等因素考虑，采取了不同的同步策略：**离线数据同步和实时数据同步**。同时，在两种同步服务的产品形态上，可以采用相同的可视化同步配置策略，以降低用户操作成本。

1. **数据源管理：**数据源管理主要是管理数据所用的存储，用于平台在做数据交换时，可以方便地对外部存储进行相应的管理。数据源可以是已有系统存储业务数据的地方，作为数据中台的数据来源，也可以是数据应用场景，为应用场景提供结果数据存储的地方。根据业务系统以及数据应用场景的不同，数据源也有不同的选择。例如，**广告场景对时效性要求很高**，相应的，对数据源读性能的要求就会很高，有些场景对于大批量数据的多维分析有需求，因此数据源需要**支持大批量数据的多维分析能力**。针对这些场景，涉及的数据源会有很多种，大致可以分成：

·**关系型数据库**：如Oracle、MySQL、SQL Server、Greenplum等。

·**NoSQL存储**：如HBase、Redis、Elasticsearch、Cassandra、MongoDB、 Neo4J等。

·**网络及MQ**：如Kafka、HTTP等。

·**文件系统**：如HDFS、FTP、OSS、CSV、TXT、Excel等。

·**大数据相关**：如Hive、Impala、Kudu、MaxCompute、ADB、LibrA、 ELK等。

1. **离线数据交换：**离线数据交换是针对数据时效要求低、吞吐量大的场景，解决大规模数据的批量迁移问题，其实现原理是将不同数据源的交换抽象为从源头数据源读取数据的读取插件，以及向目标端写入数据的写入插件，理论上可以支持任意类型数据源的数据交换工作。采用插件化方式构建，将数据源读取和写入抽象成读取插件、写入插件。非结构化的数据也可以通过扩展插件方式进行交换，**其场景主要是以文件或数据块的方式**进行交换，因此只需要适配源或目的存储的相应插件及数据处理的机制，如文件传输，数据块保存为特定格式的文件，即可以满足相应的需求。

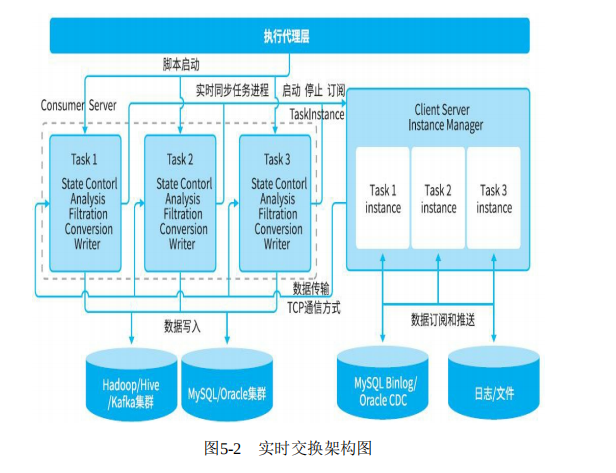
·**读取插件**：数据采集模块，负责采集数据源的数据，将数据发送给数据交换核心模块。

·**写入插件**：数据写入模块，不断从数据交换核心模块取数据，并将数据写入到目的端。

·**数据交换核心模块**：用于连接读取插件和写入插件，作为两者的数据传输通道，并处理缓冲、流控、并发、数据转换等核心技术问题。

离线数据同步技术具有以下亮点：

1. **前置稽核** 在源端数据同步开始前，可以进行数据质量规则校验，根据配置规则的阻塞、告警等策略控制数据同步是否运行。
2. **数据转换** 数据转换是指将各类非标准数据转换成标准数据格式，并且将转换后的数据推送到大数据平台指定的位置或库表。在数据同步、传输过程中，存在用户对于数据传输进**行定制化的场景**，包括**字段截取、替换、编码转换**等操作，可以借助ETL的T过程（Transform）实现。 在配置数据同步作业的字段映射关系时，可以对每个字段定义转换 （Transform）函数，例如字符串截取dx\_substr、字符串替换 dx\_replace、字符串过滤dx\_filter，还支持用户用Groovy自定义转换逻辑。
3. **跨集群数据同步** 由于**采用插件化**的设计思路，数据同步模块可支持不同集群间的数据同步。例如，从A集群上把数据同步到B集群上，**只需要开发A集群的 Reader和B集群的Writer**，便可以新建数据同步作业对数据进行跨集群迁 移。
4. **全量同步** 全量数据同步分为表全量同步和库全量同步（整库同步）两种方式。**表全量同步**每次读取表中全量数据并写入；**库全量同步策略**是把库中所有表进行数据同步，要求源端和目的端的表名称、结构相同，允许目标表不存在，**不存在时自动创建目标表**。
5. **增量同步** 增量同步分为**新增、覆盖和更新**三种策略。新增策略主要通过在目的端创建新分区或者直接追加写数据实现。覆盖和更新策略在同步配置时选择唯一键，根据唯一键对比同步中的数据和目的端数据，结合增量策略来判断数据是覆盖还是更新。
6. **实时数据交换** 实时数据交换主要负责把数据库、日志、爬虫等数据实时接入Kafka、 Hive、Oracle等存储中，便于后续进行实时计算或供业务查询分析使用，整体技术架构如图5-2所示。 实时同步有两个核心服务：**数据订阅服务（Client Server）**、**数据消费服务（Consumer Server）**。



数据订阅服务主要包含数据的订阅和读取、任务实例的启停控制等功能**，Client Server采用插件式设计思路**，可以支持扩展不同类型的数据订阅读取。

数据消费服务主要包含**任务状态控制**、**数据解析**、**数据过滤**、**数据转换、数据写入**等功能，通过TCP通信方式和数据订阅方式进行数据读取和传输，经过任务配置的过滤、转换等功能写入到目的端数据源中。数据消费服务也采用**插件式设计思路**，可以支持目的端扩展不同类型的数据源写入。

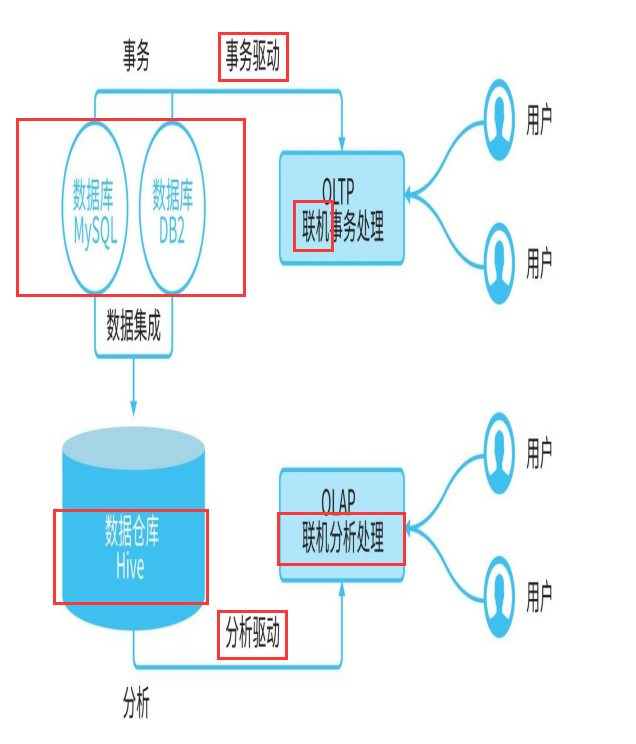
# **数据存储**

数据存储的选择需要考虑以下几个方面：

**数据规模、数据生产方式、数据应用方式**

**在线和离线**

**LOAP和LOTP**



**OLTP（On-Line Transaction Processing，联机事务处理）是专注于面向事务的任务的一类数据处理**，通常涉及在数据库中插入、更新或删除少量数据，主要处理大量用户下的大量事务。一般都是**高可用**的在线系统，**以小的事务以及小的查询为**主，评估其系统的时候，一般看其每秒执行的事务及查询的数量。在这样的系统中，单个数据库每秒处理的事务往往超过几百甚至几千个，Select语句的执行量每秒几千甚至几万个。典型的OLTP系统有**电子商务系统**、**银行、证券**等，如美国eBay的业务数据库就是很典型的OLTP数据库。

**OLAP，也叫联机分析处理（On-Line Analytical Processing）系统**，有的时候也叫**DSS（决策支持系统）**，就是我们说的数据仓库。常用于报表分析场景，相对于OLTP，对**准确性（如id-mapping）、事务性和实时性要求较低**。



**MPP（Massive Parallel Processing，大规模并行处理）架构**

1. 分布式系统
2. NoSQL数据库
3. 云数据库

# **数据开发**

**数据价值提炼工厂。**

**数据开发涉及的产品能力主要包括三个部分，分别是离线开发、实时开发和算法开发。**



## **计算能力**

**计算能力根据场景抽象分成四大类**：**批计算、流计算、在线查询和即席分析**。



**（1）批计算**

主要用于**批量数据**的**高延时**处理场景，如离线数仓的加工、大规模数据的清洗和挖掘等。目前大多是利用**MapReduce、Hive、Spark**等计算框架 进行处理，其特点是数据吞吐量大、延时高，**适合人机交互少的场景**。

**MapReduce与Spark的比较**

MapReduce由于设计上的一些限制，导致处理性能较慢，针对这个问题，业界也有很多优化方案及替代产品，但真正发展起来的，目前主要 有Spark。

**Spark的优点：**

**数据处理技术**：Spark将执行模型抽象为通用的**有向无环图（DAG）**执行计划，这可以将多个Stage串联或者并行执行，而无须将Stage的中间结果输出到HDFS中。

·**数据格式和内存布局**：Spark RDD能支持粗粒度写操作，而对于读操作，RDD可以精确到每条记录，这使得RDD可以用来作为分布式索引。

·**执行策略**：MapReduce在数据Shuffle之前花费了大量的时间来排序，Spark支持基于Hash的分布式聚合，调度中采用更为通用的任务执行DAG，每一轮的输出结果都可以缓存在内存中

**（2）流计算**

也叫实时流计算，对于数据的加工处理和应用有较强的**实效性要求**，常见于**监控告警场景**，例如实时分析网络事件，当有异常事件发生时能够及时介入处理。例如，阿里巴巴“双11”的可视化大屏上的数据展现是根据浏览、交易数据经过实时计算后展现在可视化大屏上的一种应用。这 类场景目前应用较多的计算框架主要有**Flink、Spark Streaming和Storm** 等。

**流计算的常见应用场景如下：**

·**流式ETL**：集成流计算现有的诸多数据通道和SQL灵活的加工能力，对流式数据进行实时清洗、归并、结构化处理。同时，对离线数仓进行有效补充和优化，为数据的实时传输提供可计算通道。

·**流式报表**：实时采集、加工流式数据，实时监控和展现业务和客户的各类指标，让**数据化运营实时化**。

·**监控预警**：对系统和用户的行为进行实时检测和分析，实时监测和发现危险行为。

·**在线系统**：实时计算各类数据指标，并利用实时结果及时调整在线系 统的相关策略，在**内容投放**、**无线智能推送**等领域有大量的应用。

**（3）在线查询**

主要用于数据结果的在线查询、条件过滤和筛选等，如数据检索、条件过滤等。根据不同的场景也会有多种选择，如营销场景**对响应延时要求高**的，一般会采集缓存型的存储计算，如**Redis、Tair**等；对**响应延时要求正常**的，可以选择**HBase和MySQL**等；需要进行条件过滤、检索的，可以选择**Elasticsearch**等。**企业一般对在线查询的需求比较旺盛**，因此可能会有多套在线计算的能力提供服务。

**在线查询的常见应用场景如下：**

·**画像服务**：根据对象标识提供具体的查询服务，如通过Redis可以提供低延迟、高并发的查询服务能力；通过HBase可以提供大规模数据的查询服务能力，**征信查询**就是类似的服务。

·**搜索的应用场景**：提供搜索引擎的能力，为用户提供模糊匹配、意图识别检索等能力，快速检索需要的内容，如常见的文档搜索、商品搜索等。

·**圈人场景**：通过一些特定的条件规则，可以快速筛选出业务所需要的群体，为后续的运营、营销等工作的开展提供支撑。

**（4）即席分析**

主要用于**分析型场景**和**经验统计**。一般而言，企业80%的数据处理需求是**在线查询和即席分析**。针对不同维度的分析，有多种方式可以提供，提前固定计算的维度、根据需求任意维度的交叉分析**（ad-hoc）**等都是常见的场景。目前也有很多相应的产品、框架来支撑这方面的应用，如 **Kylin、Impala、ClickHouse、Hawk**等。

**多维交叉分析，其大部分是聚合型操作，如group by、sum、avg、count等。**

**查询方式如下：**

·**ROLAP**：以关系数据库为核心，以关系型结构进行多维数据的表示和存储，结合**星型模式和雪花模式**实现。

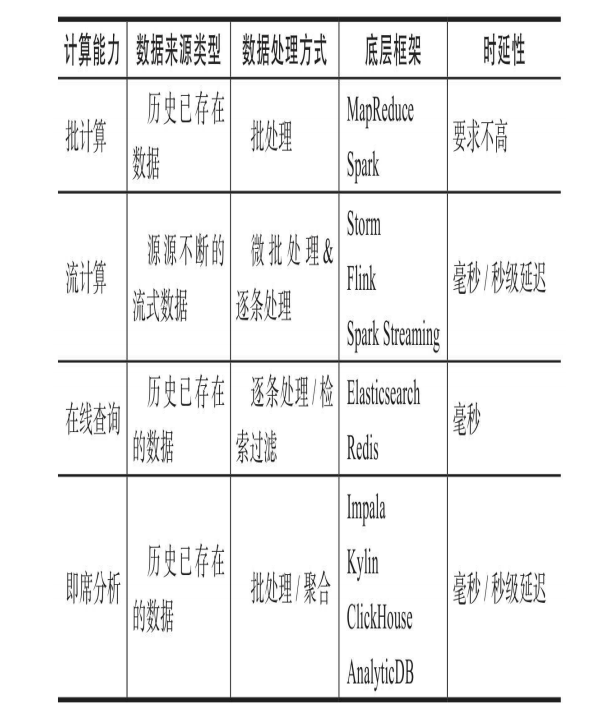
·**MOLAP**：基于多维数据组织的实现，以多维数据组织为核心，形成“立方块”的结构，通过对“立方块”进行各类处理来产生多维数据报表。

**即席分析的常见应用场景如下：**

·交互式数据分析：企业运营人员在日常工作中经常需要通过SQL从各个维度对当前业务进行分析，提供分析结果以便开展后续工作。离线计算的场景等待时间较久，用户体验不好，即席分析可以比较好地规避这个问题。

·群体对比分析场景：在业务中经常会有A/B测试场景，针对不同的群体，从各个维度对比分析也是即席分析经常支撑的场景。

**四种计算能力的对比：**



## **离线开发**

离线开发套件封装了大数据相关的技术，包括**数据加工、数据分析、在线查询、即席分析**等能力，同时也将任务的**调度、发布、运维、监控、告警**等进行整合，让开发者可以直接通过浏览器访问，不再需要安装任何服务，也不用关心底层技术的实现，只需专注于业务的开发，帮助企业快速构建数据服务，赋能业务。

**（1）作业调度**

在数据开发过程中，经常需要配置作业的上游依赖作业，这样作业之间便会组成一个**有向无环图**（DAG，Directed Acyclic Graph），同时会配置作业的开始调度时间。

1. **基线控制**

在大数据离线计算中，由于作业执行时间较长，经常会遇到急着用数据 却发现数据还没出来的情形。重新跑需要几个小时，时间已然来不及。 因此本书提出一种基线控制方法，用于统一管理数据处理作业的完成时间、优先级、告警策略，保障数据加工按时完成。调度模块会根据最先到达、最短执行时间原则，动态调整资源分配及作业的优先级，让资源利用效率最大化。

同时采用算法对作业完成时间进行智能预测。根据预测，当作业无法正常产出且动态调整无法完成时，调度中心会及时通过监控告警通知运维值班人员提前介入处理，为大数据作业执行留出充裕的时间。

1. **异构存储**

当前，企业内部的计算存储引擎呈现多元化趋势。例如，国内某大型企业同时使用Oracle、IQ、HANA、Hadoop、LibrA等多种数据库，涉及关系型DB、MPP、大数据数仓等多种不同类型。离线开发中心针对每种类型的计算引擎会**开发不同的组件**，例如，针对Oracle开发Oracle插件，针对Hadoop体系分别开发出Hive、Spark、MapReduce等插件。用户只需要**新建各种类型作业**，例如Oracle、IQ、HANA、Hive、Spark、 LibrA等，在执行时自动根据作业的类型寻找相应插件来运行作业。

1. **代码校验**
2. **多环境级联**
3. **推荐依赖**
4. **数据权限**

**Hadoop自身无数据权限系统**，由不同厂商各自去实现，目前主要有两种策略：

·RBAC（Role-Based Access Control，基于角色的访问控制）：比如 Cloudera用的是Sentry，华为的FusionInsight也是类似的机制。

·PBAC（Policy-Based Access Control，基于策略的访问控制）：比如Hortonworks用的Rang

**数据权限管理目标就是构建统一的权限管理系统来支持多种**引擎，可以直接在此系统上进行各种引擎的权限申请、审批和管理，无须接触底层引擎的权限管理系统。在适配不同引擎时，仍旧**采用插件化**的设计思路，针对每种权限管理系统开发一种插件，并支持用户通过二次开发来扩展插件。

## **实时开发**

数据的价值在于数据的**在线化**。实时开发套件是对**流计算能力**的产品封装。实时计算起源于对数据加工时效性的严苛需求：数据的业务价值随着时间的流逝会迅速降低，因此在数据产生后必须尽快对其进行计算和处理。通常而言，

实时计算具备以下三大特点：

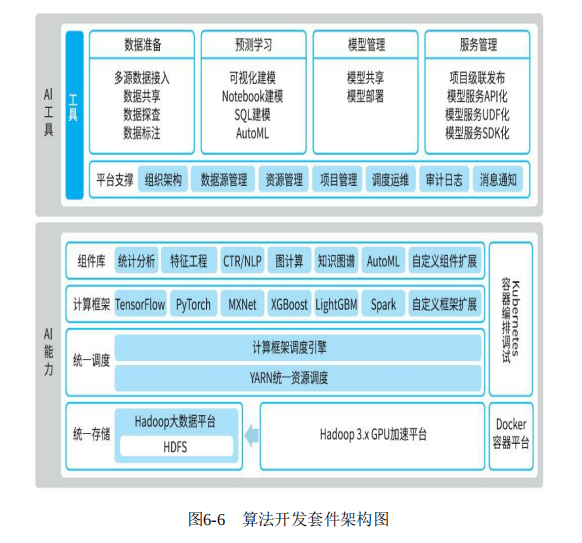
·**实时且无界（unbounded）的数据流**：实时计算面对的计算是实时的、流式的，流数据是按照时间发生的顺序被实时计算订阅和消费的。并且，由于数据产生的持续性，数据流将长久且持续地集成到实时计算系统中。

·**持续且高效的计算**：实时计算是一种“事件触发”的计算模式，触发源就是上述的无界流式数据。一旦有新的流数据进入实时计算，实时计算立刻发起并进行一次计算任务，因此整个实时计算是持续进行的高效计算。

·**流式且实时的数据集成**：流数据触发一次实时计算的计算结果，可以被直接写入目的存储中，例如，将计算后的报表数据直接写入MySQL 进行报表展示。因此，流数据的计算结果可以类似流式数据一样持续写 入目的存储中。

基于**Storm、Spark Streaming、Apache Flink**构建的一站式、高性能实时大数据处理能力，广泛适用于实时ETL、实时报表、监控预警、在线系统等多种场景。让用户彻底规避繁重的底层流式处理逻辑开发工作，助力企业向实时大数据计算升级转型。实时开发涉及的核心功能点包括**元数据管理、SQL驱动式开发、组件化配置**以及多计算引擎。

## **算法开发**



（1） 可视化建模

（2） Notebook建模

（3）数据集管理

（4) 核心算法组件

（5）多算法框架

# **数据同步（小记）**

·业务系统的数据库数据同步;（Sqoop、DataX）

·系统日志数据同步;(开源工具：Flume、logstash、fluentd

、商业工具：有阿里的loghub、日志易的日志采集汇聚)

·非结构化数据同步； （Lsyncd）

·互联网公开数据爬取。（Nutch）

# **数据体系特征**（重点）

中台数据体系应具备以下特征：

·**覆盖全域数据**：数据集中建设，覆盖所有业务过程数据，业务在中台数据体系中总能找到需要的数据。

·**结构层次清晰**：**纵向的数据分层**，**横向主题域**、**业务过程划分**，让整个层次结构清晰易理解。

·**数据准确一致**：定义**一致性指标**，**统一命名**、**统一业务含义、统一计算口径**，并有**专业团队负责建模**，保证数据的准确一致。

·**性能提升**：统一的规划设计，选用合理的数据模型，清晰地定义并统一规范，并且考虑使用场景，使整体性能更好。

·**降低成本**：数据体系的建设使得数据能被业务共享，这避免了大量烟囱式的重复建设，节约了计算、存储和人力成本。

·**方便易用**：易用的总体原则是越往后越能方便地直接使用数据，把一些复杂的处理尽可能前置，必要时做适当的冗余处理。比如在数据的使用中，可以通过维度冗余和事实冗余来提前进行相关处理，以避免使用 时才计算，通过公共计算下沉、明细与汇总共存等为业务提供灵活性。

**四个数据分层：**

·**贴源数据层**ODS（Operational Data Store，又称操作数据层）：对各业务系统数据进行采集、汇聚，尽可能保留原始业务流程数据，与业务系统基本保持一致，仅做简单整合、非结构化数据结构化处理或者增加**标识数据日期描述**信息，不做深度清洗加工。

·**统一数仓层DW**（Data Warehouse）：又细分为明细数据层DWD（Data Warehouse Detail）和汇总数据层DWS（Data Warehouse Summary），与传统数据仓库功能基本一致，对**全历史业务过程数据进行建模存储**。对来源于业务系统的数据进行重新组织。业务系统是按照业务流程方便操作的方式来组织数据的，而统一数仓层从**业务易理解的视角**来重新组织，定义**一致的指标、维度，各业务板块、业务域**按照统一规范独立建设，从而形成统一规范的标准业务数据体系。

·**标签数据层TDM（Tag Data Model）**：面向对象建模，对跨业务板块、跨数据域的特定对象数据进行整合，通ID-Mapping把各个业务板块、各个业务过程中的同一对象的数据打通，形成**对象的全域标签体系**，方便深度分析、挖掘、应用。

·**应用数据层ADS**（Application Data Store）：按照业务的需要从统一数仓层、标签数据层抽取数据，并面向业务的特殊需要加工业务特定数据，以满足业务及性能需求，向特定应用组装应用数据

**注意**：建设过程中数据的读取也有严格的规范要求。按照规范，**贴源数据层直接从业务系统或日志系统中获取数据**。贴源数据层的数据只被**统一数仓层使用**，统一数仓层数据只被标签层和应用层使用。**贴源数据层、统一数仓层只保存历史数据**以及被标签层、应用层引用，不直接支撑业务，所有业务使用的数据均来源于标签层和应用层。

## **贴源数据层建设——全域数据统一存储（重点）**

贴源数据层数据不仅是业务数据库中产生的数据，跟企业相关的所有数据都应该汇聚到贴源数据层，包括**业务系统数据、业务运行的日志数据、机器运转产生的日志数据、网络爬虫或者其他方式获取的外部数据。**

**特点：并不对业务数据做过多的清洗加工，尽可能保留数据的原始状态。**

**（1）贴源数据表设计**

贴源数据层中的数据表与对应的业务系统数据表**原则上保持一致**，数据结构上几乎不做修改，所以参考业务系统数据表结构来设计贴源数据层表结构即可，结构设计上没有太多的规范要求。

* 贴源数据层表的命名采用**前缀+业务系统表名**的方式
* 贴源数据层表的字段名与业务系统字段名**保持一致**
* 对于一些数据量较大的业务数据表，如果采用增量同步的方式，则要**同时**建立**增量表和全量表**，增量表利用后缀标识。比如，ODS\_系统简称\_业务系统表名\_delta，汇聚到增量表的数据通过**数据加工任务合并生成全量表数据**。
* 对于日志、文件等半结构化数据，不仅要存储原始数据，为了方便后续的使用还要对**数据做结构化处理**，并存储结构化之后的数据。原始数据可以按**行存储在文本类型**的大字段中，然后再通过**解析任务**把数据解析到**结构化数据表中**。

**（2）贴源数据层实现**

贴源数据层一般采用数据同步工具实现数据的同步落地。具体的实现步

骤如下：

1. 确定业务系统源表与贴源数据层目标表；

2）配置数据字段映射关系，目标表可能会增加采集日期、分区、原系统标识等必要信息，业务相关内容不做转换；

3）如果是增量同步或者有条件地同步部分数据，则配置数据同步条件；

4）清理目标表对应数据；

5）启动同步任务，往贴源数据层目标表导入数据；

6）验证任务是否可以正确运行，并且采集到准确数据；

7）发布采集任务，加入生产调度，并配置相关限速、容错、质量监控、告警机制。

## **统一数仓层建设——标准化的数据底座（重点）**

统一数仓层站在业务的视角，**不考虑业务系统流程**，从**业务完整性的角度重新组织数据**。统一数仓层的目标是建设一套覆盖全域、全历史的企业数据体系，利用这套数据体系可以还原企业任意时刻的业务运转状态。只要能达到这个目标，利用范式建模、维度建模、实体建模中任意一种建模方法都是可以的，这里主要介绍维度建模，因为它更适合大数据时代数据量巨大的特点。

**关注指标**：统一数仓层建设过程以维度建模为理论基础，构建**总线矩阵**，划分**业务板块**，**定义数据域**、**业务过程**、**维度**、**原子指标、修饰类型、修饰词、时间周期、派生指标**，进而确定**维度表、事实表**的模型设计。

·**业务板块**：根据业务的属性划分出的相对独立的业务板块，业务板块 是一种大的划分，各业务板块中的业务重叠度极低，数据独立建设，比如**地产板块、金融板块、医疗板块**等。

·**模型设计**：以建模理论为基础，基于维度建模总线架构，构建一致性的维度和事实，同时设计出一套表命名规范。

·**数据域**：数据域是统一数仓层的顶层划分，是一个**较高层次的数据归类标准**，是对企业业务过程进行抽象、提炼、组合的集合，面向业务分析，一个**数据域**对应一个**宏观分析领域**，比如**采购域、供应链域、HR域**等。数据域是抽象、提炼出来的，并且不轻易变动，既能涵盖当前所有业务需求，又能在新业务进入时无影响地将其分配到已有的数据域中，只有当所有分类都不合适时才会扩展新的数据域。数据域是有效归纳、组织业务过程的方式，同时方便定位指标/度量。

·**业务过程**：业务过程是一种企业的业务活动事件，且是企业经营过程中不可拆分的**行为事件**，比如**下订单**、**银行转账**、账号注册都是业务过程。业务过程产生度量，并且会被转换为最终的事实表中的事实。**业务过程一般与事实表一一对应**，也有一对多或者多对一的特殊情况，比如累计快照事实表就会把多个业务过程产生的事实在一张表中表达。

**·修饰词**：修饰词指除统计维度以外的对指标进行**限定抽象的业务场景词语**，修饰词隶属于一个修饰类型。比如，在日志域的访问终端类型下，有**修饰词PC、无线端**。修饰类型的出现是为了方便管理、使用修饰词。

**·原子指标**：原子指标是针对某一业务事件行为的度量，是一种**不可拆分的指标**，具有明确业务含义，比如**支付金额**。原子指标有确定的字段名称、数据类型、算法说明、所属数据域和业务过程。原子指标名称一 般采用“**动作+度量**”方式命名，比如**支付金额**、**注册用户数**。

·**派生指标**：派生指标可以理解为对**原子指标业务统计范围**的圈定，比如最近1天北京买家支付金额（“最近1天”是时间周期，“北京”是修饰 词，修饰词“买家”是维度）。**派生指标=1个原子指标+多个修饰词+时间修饰词**。

·**计算方法**：指标的数学计算方式，比如**汇总、平均、最大、最小等**。

·**维度表**：维度是观察事物的角度，提供某一业务过程事件所涉及的用于过滤及分类事实的描述性属性，用于描述与“谁、什么、哪里、何时、为什么、如何”（5W1H）有关的事件。比如“早上小王在小卖部花费5元钱购买了一个面包”，以购买为业务过程进行分析，可从这段信息中提取三个维度，即**时间维度**（早上）、**地点维度**（小卖部）和**商品维度**（面包）。维度表是统一设计的，在整个数据仓库中共享，所有数据域、业务过程都需要用到维度，都可以在公共维度表中获取相关维度属性。

**·事实表**：事实是观察事物得到的事实数据，事实涉及来自业务过程事件的度量，基本都是以**数量值**表示，比如一次购买行为就可以理解为是一个事实，5元就是事实信息。在**确定数据域与业务过程**后，就可以根据**业务过程涉及的维度、度量及粒度**，设计相关的事实表。**事实表不跨数据域**，根据需要，一个事实表可能对应**同数据域**下**一个或者多个业务过程**。事实表又分为明细事实表和汇总事实表。明细事实表记录事务层面的事实，保存的是原子数据，数据的粒度通常是**每个事务一条记录**， 明细事实表数据被插入，**数据就不再进行更改**，其更新方式为**增量更新**。汇总事实表是把明细事实聚合形成的事实表，包括以具有规律性 的、可预见的时间间隔来记录事实的周期快照事实表和以不确定的周期 来记录事实的累计快照事实表。

**·粒度**：粒度是指统一数仓层数据的细化或综合程度，对各事实表行实际代表的内容给出明确的说明，用于确定某一事实表中的行表示什么。确定维度或者事实之前**必须声明粒度**，因为每个维度和事实都**必须与定义的粒度保持一致**。原子粒度是最低级别的粒度，是对业务过程最详细的刻画，原子粒度事实必须保留。

**·一致性指标定义**：指标归属到具体数据域，定义指标的**含义、命名、类型、计算方法**，确保指标的全局一致性。

**（1）数据域划分**

**数据域**需要**抽象提炼**，并且长期维护和更新，但**不轻易变动**。在划分数据域时，**既能涵盖当前所有的业务需求**，又能在**新业务进入时无影响地将其插进已有的数据域中或者扩展新的数据域**。

步骤：

**第一阶段：数据调研**。

**·业务调研**：确定项目要涵盖的**业务领域和业务线**，以及各个业务线可以细分成哪几个**业务模块**，各业务模块具体的**业务流程**是怎样的，通过跟业务专家访谈或进行资料文档收集，梳理主要业务流程、业务边界、专业术语等。

**·数据调研**：调研全部数据目录信息，梳理数据流与业务过程关联关系。

**第二阶段：业务分类。**

**·业务过程提取**：根据调研结果抽取出全部业务过程。

·**业务过程拆分**：将组合型的业务过程拆分成一个个不可分割的行为事件，如**下单**、**支付、收货、退款**。

·**业务过程分类**：按照业务分类规则，将相似特征的业务过程分为一类，且每一个业务过程只能唯一归属于一类。

**第三阶段：数据域定义。**

**·业务分类确认**：对业务分类结果再次确认，避免分类范围中出现业务特征明显与其他业务过程无关的情况。

**·数据域定义**：根据业务分类的规律总结出划分业务范围的标准定义。

**·数据域命名：**为每个数据域起一个专属名称，并附上英文全称和简称。

**第四阶段：总线矩阵构建 。**

**·关系梳理**：明确每个数据域下有哪些业务过程，并梳理出业务过程与哪些维度相关。

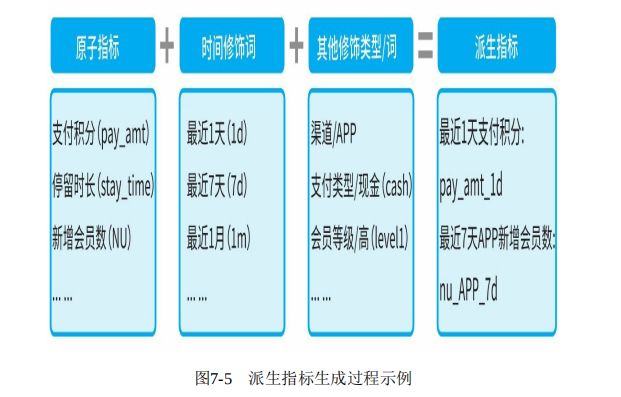
**·矩阵构建**：定义一张二维矩阵，将数据域下的业务过程与维度信息如实记录下来



1. **指标设计**

指标就是在企业业务运转过程中产生的度量事实，一致性指标设计是为了在企业内外部使指标的**命名、计算方法、业务理解达到一致**，避免不同部门同一个指标的数据对不上或者对同一个指标的数据理解不一致。

一致性指标的定义为，描述原子指标、修饰词、时间周期和派生指标的含义、类型、命名、算法，被用于模型设计，是建模的基础。例如：



**特别注意：**

**一致性指标设计**是**事实表模型设计的来源**，有了一致性的指标定义，在设计事实表模型时引用定义好的一致性指标，可达到指标的一致性与标志性。

1. **维度表设计**

维度是**维度建模**的基础和灵魂，维度表设计得好坏直接决定了维度建模的好坏。维度表包含了**事实表所记录的业务过程度量的上下文和环境**，它们除了记录**5W**等信息外，通常还包含了很多描述属性字段。每个维度表都包含单一的主键列。维度设计的核心是确定维度属性，**维度属性是查询约束条件**（SQL where条件）、分组（SQL group语句）与报表标签生成的基本来源。维度表通常有多列或者说多个属性。维度表通常比较宽，是扁平型非规范表，包含大量细粒度的文本属性。实际应用中，包含几十甚至上百个属性的维度并不少见。维度表应**该尽可能包括一些**

**有意义的文字性描述**，以方便下游用户使用。维度属性尽可能丰富。维度属性设计中会有一些反规范化设计，把相关维度的属性也合并到主维度属性中，达到易用、少关联的效果。维度表设计主要包括**选择维度**、确定**主维表**、梳理关联维表、定义维度属性等过程。

**1）选择维度：** 维度作为维度建模的核心，在企业级数据仓库中必须**保证维度的唯一性**。维度一般用于查询约束条件、分组、排序的关键属性，维度既可以从报表需求中分析获取，也可以从与业务人员的交谈中发现。

**2）确定主维表**：主维表一般直接从业务系统同步而来，是分析事实时所需环境描述的最基础、最频繁的维度属性集合。比如用户维表从业务系统的用户基本信息表中产出。

**3）梳理关联维表：**数据仓库是业务源系统的数据整合，不同业务系统或者同一业务系统中的表间存在关联性。根据对业务的梳理，确定哪些表和主维表存在关联关系，并选择其中的某些表用于生成维度属性。如**商品与类目、SPU、卖家、店铺**等维度存在关联关系。

**4）定义维度属性**：从主维表或关联维表中选择维度属性或生成新的维度属性，过程中尽量生成更丰富、更通用的维度属性，并维护和描述维度属性的层次及关联关系。如商品维表，商品属于类目，类目属于行业。

1. **事实表设计**

事实表是统一数仓层建设的主要产出物，统一数仓层绝大部分表都是事实表。一般来说事实表由两部分组成：**一部分是由主键和外键组成的键值部分**，另一部分是用来描述业务过程的事实度量。**事实表的键值部分确定了事实表的粒度**，事实表通过粒度和事实度量来描述业务过程。事实表的**外键**总是对应某个**维度表的主键**，实际建设和试用过程中，为了提升事实表的易用性和性能，不仅会**存储维度主键**，还会把关键的维度属性存储在事实表中。这样事实表就包含表达粒度的键值部分、事实度量及退化的维度属性。一切数据应用和分析都是围绕事实表来展开的，稳定的数据模型能大幅提高数据复用性。

在Kimball的维度建模理论中主要定义了**事务事实表**、**周期快照事实表**、**累积快照事实**表三种类型的事实表。

·**事务事实表：**事务事实表描述**业务过程事务层面**的事实，每条记录代表一个**事务事件**，保留事务事件活动的原始内容。事务事实表中的数据在事务事件发生后记录，一般记录后数据就**不再进行更改**，**其更新方式为增量更新**。事务事实表相对其他事实表保存的数据粒度更细，可以通过事务事实表对事务行为进行详细分析。

·**周期快照事实表**：周期快照事实表以具有规律性、可预见的时间间隔产生快照来记录事实，每行代表某个时间周期的一条记录，记录的事实是时间周期内的聚集事实值或状态度量。周期快照事实表的内容一般在所表达的时间周期结束后才会产生，**一般记录后数据就不再更改**，**其更新方式为增量更新**。周期快照事实表一般是建立在事务事实表之上的聚集，**维度比事务事实表少，粒度比事务事实表粗**，但是由于对事实进行了多种形式的加工从而产生了新的事实，故一般事实会比事务事实表多。

·累计快照事实表：累积快照事实表覆盖一个事务从开始到结束之间所有的关键事件，覆盖事务的整个生命周期，通常具有多个日期字段来记录关键事件时间点。周期快速事实表涉及的多个事件中任意一个的产生都要做记录，由于周期快照事实表涉及的多个事件的首次加载和后续更新时间是不确定的，因此在首次加载后允许对记录进行更新，一般采用全量刷新的方式更新。 **累计快照事实表一般用于追踪某个业务的全生命周期及状态转换，**比如**交易业务，涉及下单、支付、发货、确认收**货，这些相关事件在不同的事务事实表中，通过事务事实表很难看到不同事件之间的转化及状态变化，通过累计快照事实表可把相关事件串起来放在一条记录中，这样就很容易解决了。不管哪种类型的事实表，设计方法都类似，事实表设计可以遵循以下步骤：

第一步：确定业务过程。

企业业务是由一个个业务过程组成的，事实表就是为了记录这些业务过程产生的事实，以便**还原任意时刻的业务运转状态**。所以设计事实表，第一步就是确定实施所要表达的是哪一个或者几个业务过程。笔者理解业务过程是企业活动事件，比如**注册、登录、下单、投诉**等都是业务过程，最基本的是每一个业务过程对应一张事实表，这样最容易理解。但是实际开发过程中，业务过程和事实表会存在多对多的关系。

第二步：定义粒度。

不管事实表对应一个还是多个业务过程，粒度必须是确定的，每个事实表都有且只能有唯一的粒度，粒度是事实表的每一行所表示的业务含义，是事实的细节级别。在实际设计过程中，**粒度与主键等价**，粒度更偏向业务，而主键是站在技术角度说的。虽然粒度在最终的事实表中很难被体现，但是定义粒度是必不可少的步骤，这样可避免整个事实表的业务含义模糊。

第三步：确定维度。

定义粒度之后，事实表每一行的业务含义就确定了。那么业务人员会站在哪些角度来描述事实度量？这就要确定维度了，常见的**维度有时间、区域、客户、产品、员工**等。维度依附于粒度，是粒度的环境描述。

第四步：确定事实。

事实就是事实表度量的内容，也就是业务过程产生的事实度量的计算结果，**比如注册量、登录次数、交易金额、退款量**等。事实表的所有事实度量都与事实表所表达的业务过程相关，所有事实必须满足第二步所定义的粒度。

第五步：冗余维度属性。

事实表的设计要综合考虑数据来源和使用需求，在满足业务事实记录的同时也要满足使用的便利性和性能要求。大数据时代，事实表记录数动辄亿级，甚至数十亿、数百亿，维表也有可能达到亿级甚至更多。利用标准维度模型会经常出现维表与事实表关联的情况，这种对亿级表的关联计算，在性能上是灾难性的。为了满足业务需求，降低资源消耗，建议适当冗余维度属性数据到事实表，直接利用事实表就可以完成绝大部分业务的使用需求，这样下游使用时可减少大表关联，提升效率。所以大数据时代，适当进行维度冗余是可取的。

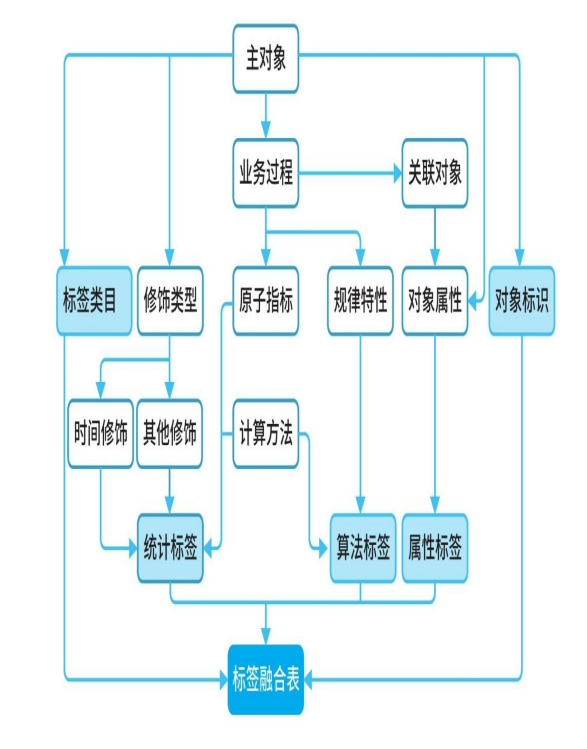
注意**：维度属性冗余与模型的稳定性是有矛盾的**，因为维度的属性是有可能改变的，如果属性已经冗余到事实表中，那么维度属性就与事实一起被记录到事实表中。如果后续维度属性值改变，由于事实表已经生成，事实表的内容基本不会再做改变，这样就会出现已记录的维度属性与真实的维度属性不一致，导致数据错误的情况。属性的冗余是一种优化建议，冗余带来的收益与弊端要综合考虑。

## **标签数据层建设——数据价值魅力所在（重点）**

统一数仓层是按照数仓的维度规范建模，对业务数据进行了重新组织标准化。但是**同一个对象的各种信息分散在不同的数据域并且有不同的数据粒度**。比如客户数据，**基本信息**在客户域按照客户粒度组织，**交易信息**在交易域按照订单粒度组织，**社交信息在社交域**按照关系对粒度组织，这导致很**难了解一个客户的全面信息**，要通过各种关联计算才能满足业务的需要，数据使用成本较高。而获取、分析客户的全面数据，是多个业务的共同需求，这可以通过建设标签数据层来满足。大数据的典型应用基本都是**通过建立标签体系来支撑的**，大数据的核心价值和魅力 通过标签数据的多样应用得到充分体现。

标签数据层是**面向对象建模**，把一个对象各种**标识打通归一**，把**跨业务板块、数据域**的对象数据在同一个粒度基础上组织起来打到对象上。

标签归属到一个对象，标签按照产生和计算方式不同可分为**属性标签**、**统计标签、算法标签**。



**·对象**：是客观世界中研究目标的抽象，可以是现实存在的，也可以是虚拟的，是具备独立特征的个体，比如自然人、产品、账户等。

**·对象标识**：对象的标识符用以标识一个对象，**一般是各种ID**，比如手机号、身份证、登录账号。

**·标签**：利用原始数据，通过一定的加工逻辑产出，能够为业务所直接使用的**可阅读、易理解、有业务价值**的数据。

**·标签类目**：是标签的分类组织方式，是标签信息的一种结构化描述，目的是**管理、查找标签，一般采用多级类目**。

**·属性标签：**属性是对实体基本性质的刻画，**属性的变化非常缓慢**，有些甚至永远固定，属性是一类实体区别于另一类实体的差异所在。属性标签是根据人类对实体的长期认知得出的，**比如性别、年龄、体重**。

**·统计标签：**统计标签是特定场景下，维度和度量的组合。构建出实体所在场景的维度、度量矩阵，就可以根据经验和实际业务需要组装统计标签，比如**日均登录次数**、最近30天交易额。

**·算法标签**：算法标签是**不可以直接获取的**，需要通过复杂逻辑分析推理得出，是通过分析对象在多个场景下发生多个事件的规律性得出的相关结论，比如**信用指数**、**购买能力**、**品牌偏好**。

**·标签融合表**：以对象为核心把属性标签、统计标签、算法标签组装起来得到的表，是标签数据层落地的产出物。标签融合表设计要考虑标签的类目结构进行合理组织。

步骤：

1）确定对象：可把对象分 为“人”“物”“关系”三大类。“人”往往具有主动性和智慧，能主动参与社会活动，主动发挥推动作用，往往是关系的发出者。“物”往往是被动的，包括原料、设备、建筑物、简单操作的工具或功能集合等，是关系的接收者。“人”和“物”是实体类的对象，即看得到、摸得着的对象，而“关系”属于一种虚拟对象，是对两两实物实体间的联系的定义。

2）对象ID打通：要完成对象的ID打通，一般会给每个对象设置一个超级ID，比如**SUPER-ID**作为唯一识别该对象的标识码，业务系统中不同的对象标识ID都通过一定的算法规则与这个 SUPER-ID打通，进而完成对象所有业务标识ID的打通。

ID打通，首先必须有ID-ID之间的**两两映射打通关系**，通过ID-ID之间两 两映射关系表，将多种ID之间的关联打通。比如手机号、身份证号码可以打通，手机号、邮箱账号可以打通，这样通过手机号就可以把身份证号码和邮箱账号也打通了。完全孤立的ID是无法进行打通的。通过ID- ID间的两两映射，打通整个ID关系，看似简单，实则计算复杂，计算量非常大。

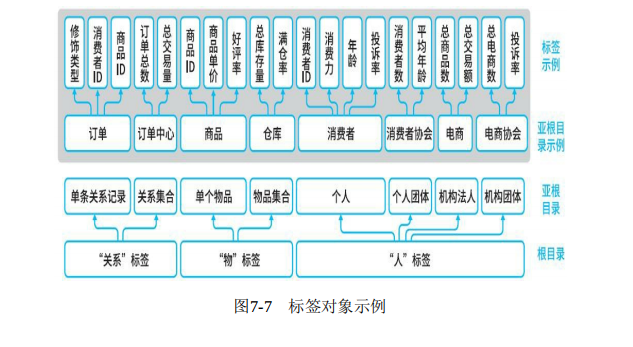
**大数据领域中的ID-Mapping技术**就是用机器学习算法来取代 野蛮计算，解决对象数据打通的问题。基于输入的ID关系对，利用机器学习算法做稳定性和收敛性计算，输出关系稳定的ID关系对，并生成一个SUPER-ID作为唯一识别该对象的标识码。

一般来说，**ID打通是标签体系建设的前提**，没有ID打通就无法收集到一个对象的全面信息，也就无法对这个对象进行全面标签化刻画。

3）标签类目设计：通过建立对象标签类目体系来对对象

的标签进行分类管理。

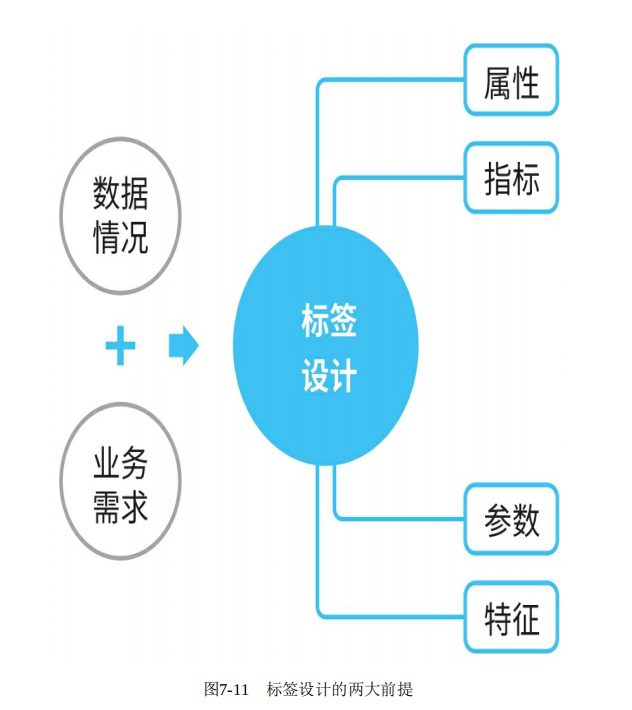
构建标签类目体系首先需要确定**根目录**。根目录就是**上文提到的对象**， 因此有三大类根目录：人、物、关系。根目录就像树根一样直接确定这 是一棵什么树。如果根目录是人，即这个标签类目体系就是人的标签类目体系，每个根目录都有一个识别列来唯一识别具体对象。人这种大类下包括自然人和企业法人两种**亚根**，同时自然人群体或企业法人群体也可以认为属于人的对象范畴内，也是亚根。自然人实例可以有消费者（人）、员工（人）、加盟商（人）等，因此可以形成消费者（人）的 标签类目体系、员工（人）的标签类目体系、加盟商（人）的标签类目体系。同样法人也可以细分为实体公司（人）、营销公司（人）、运输公司（人）等。自然人群体可以细分为消费者协会（人）、卖家联盟（人）、直播圈（人）等。法人群体可以细分为电商协会（人）、国际品牌公司联盟（人）等。从最大的“人”根目录、到“自然人/法人/自然人 群体/法人群体”亚根，再到实例“用户/员工/加盟商”，都属于根目录的范 畴。根据类似的方式，也可以将物细分为“物品”“物体”“物品集合”“物体集 合”等亚类，各亚类下也可以细分根；关系也可以细分“关系记录”“关系 集合”。电商行业中的物品可以细分为“商品”或“服务”等，进而构建商 品标签类目体系、服务标签类目体系。各亚根示例在图7-7所示虚框中 的下半部分，上半部分为标签示例。



标签类目体系是对业务所需标签采用类目体系的方法进行设计、归属、分类。类目体系本身是对某一类目标物进行分类、架构组织，分类通常使用一级类目、二级类目、三级类目等作为分类名，将item分入合适的类目中，每个具体的item都是叶节点。某大型互联网集团构建的商品类目体系，是对海量商品进行行业类目梳理的经典成功案例，其对所售商品先进行一级分类，分为美妆、女装、母婴、数码、鞋包等；美妆一级分类下有基础护理、彩妆、美发、美体等二级分类；基础护理二级分类下又细分为卸妆、洁面、化妆水、乳液面霜等三级分类，卸妆三级分类 下再下挂具体的卸妆油、卸妆液等具体商家商品。

1. 标签设计：标签设计就是设计合适的标签并将其挂载到标签类目。

将**数据提炼转化为标签**的过程就叫标签化，也就是标签设计过程。一个好的标签设计，等于已经完成了好的数据服务50%的工作，标签设计考验的是理解、抽象、提炼、提升业务场景的数据能力。标签设计要充分考虑两大前提条件。



## **应用数据层建设——灵活支撑业务需求（重点）**

应用数据层是按照业务使用的需要，组织已经加工好的数据以及一些面 向业务的特定个性化指标加工，以满足最终业务应用的场景。应用数据 层一般也是采用维度建模的方法，但是为了满足业务的个性需求以及性能的要求，会有一些反规范化的操作，所以应用数据层并没有非常规范的建设标准。

**(1)应用数据表设计**

应用数据层的建设是强业务驱动的，业务部门需要参与到应用数据层的建设中来。推荐的工作方式是，**业务部门的业务专家把业务需求告知数据部门的数据工程师**，然后在建模过程中深入沟通，这样最终形成的应用数据层的表设计才能既满足业务需求又符合整体的规范。因此应用数据层的特点就是考虑使用场景，其有以下几种结构：

1. 应用场景是多维的即席分析，一般为了减少连接、提升性能会**采用大宽表**的形式组织。
2. 如果是特定指标的查询，可以采用K-V表形式组织，涉及此类表的时候需要深入了解具体的查询场景，例如是否有模糊查询，以便于选择最适合的数据结构。
3. 有些场景下一次要查询多种信息，也可能会用复杂数据结构组织.
4. **应用数据表实现**

应用数据层建设步骤如下：

1. 调研业务应用对数据内容、使用方式、性能的要求，需要明确业务应用需要哪些数据，数据是怎么交互的，对于请求的响应速度和吞吐量等有什么期望。这个时候需要参与沟通的可能不仅仅是业务部门的业务专家，业务系统的研发人员也需要参与讨论。
2. 盘点现有统一数仓层、标签数据层数据是否满足业务数据需求，如果满足则直接跳到第3步；如果有个性化指标需求，统一数仓层、标签 数据层数据无法满足，则进行个性化数据加工。
3. 组装应用层数据。组装考虑性能和使用方式，比如应用层是多维的自由聚合分析，那就把统一数仓层、标签数据层以及个性化加工的指标组装成大宽表；如果是特定指标的查询，可以考虑组装成K-V结构数据。
4. **应用数据场景化支撑**

随着数据技术的发展，数据应用场景已经不限于做BI分析出报表，而是在更广的业务领域发挥价值，比如根据**客户兴趣做推荐、根据客户的历史行为做搜索优化**，也有可能快速获取客户信息服务业务。这些不同的使用场景，对数据的组织方式和底层的存储计算技术的要求是不同的，应用层的模型设计要考虑业务需要和技术环境。应用数据层加工的结果数据集，要根据不同的使用场景，同步到不同的存储介质，以达到业务对不同吞吐量和响应时间的需要.

示例：





# **数据资产管理**

## **现状**

·**缺乏统一的数据视图**：数据资源分布在企业的多个业务系统中，分布 在线上和线下，甚至分布在企业的外部。由于缺乏统一的数据视图，数据的管理人员和使用人员无法准确快速地找到自己需要的数据。数据管理人员也无法从宏观层面掌握自己拥有哪些数据资产，拥有多少数据资产，这些数据资产分布在哪里，以及变化情况怎样等。

·**数据基础薄弱**：大部分企业的数据基础还很薄弱，存在数据标准混乱、数据质量参差不齐、各业务系统之间数据孤岛化严重、没有进行数据资产的萃取等现象，阻碍了数据的有效应用。

·**数据应用不足**：受限于数据基础薄弱和应用能力不足，多数企业的数据应用刚刚起步，主要在精准营销、舆情感知和风险控制等有限场景中进行了一些探索，数据应用的深度不够，应用空间亟待开拓。

**·数据价值难估**：企业难以对数据对业务的贡献进行评估，从而难以像运营有形资产一样运营数据。产生这个问题的原因有两个：一是没有建立起合理的数据价值评估模型；二是数据价值与企业的商业模式密不可分，在不同应用场景下，同一项数据资产的价值可能截然不同。

**·缺乏安全的数据环境**：数据的价值越来越得到全社会的广泛认可，但随之而来的是针对数据的犯罪活动日渐猖獗，数据泄露、个人隐私受到侵害等现象层出不穷。很多数据犯罪是由安全管理制度不完善、缺乏相应的数据安全管控措施导致的。

**·数据管理浮于表面**：没有建立一套数据驱动的组织管理制度和流程， 没有建设先进的数据管理平台工具，导致数据管理工作很难落地。

## **数据资产管理的四个目标**

数据资产管理是数据中台面向企业提供数据能力的一个窗口，数据资产中心将企业的数据资产统一管理起来，实现数据资产的可见、可懂、可 用和可运营。

**·可见**：通过对数据资产的全面盘点，形成数据资产地图。针对数据生产者、管理者、使用者等不同的角色，用数据资产目录的方式共享数据资产，用户可以快速、精确地查找到自己关心的数据资产。

**·可懂**：通过**元数据管理**，完善对**数据资产的描述**。同时在数据资产的建设过程中，注重数据资产业务含义的提炼，将数据加工和组织成人人可懂的、无歧义的数据资产。具体来说，在数据中台之上，需要将数据资产进行标签化。标签是面向业务视角的数据组织方式。

**·可用：**通过统一数据标准、提升数据质量和数据安全性等措施，增强数据的可信度，让数据科学家和数据分析人员没有后顾之忧，放心使用数据资产，降低因为数据不可用、不可信而带来的沟通成本和管理成本。

**·可运营：**数据资产运营的最终目的是让数据价值越滚越大，因此数据资产运营要始终围绕资产价值来开展。通过建立一套符合数据驱动的组织管理制度流程和价值评估体系，改进数据资产建设过程，提升数据资产管理的水平，提升数据资产的价值。

## **数据治理**

**六个目标**

·提升数据质量，帮助做出基于数据的更高效、更准确的决策；

·构建统一的、可执行的数据标准；

·良好地响应数据生产者、消费者、数据处理技术人员等数据利益相关 者的需求，如保护好客户（数据生产者）的数据隐私和数据安全；

·培训组织内所有的管理层和员工，让大家采用共同的解决数据问题的 办法；

·实现可重复的数据管理流程，并确保流程透明；

·实现数据的可持续运营、数据资产的增值。

**六个原则**

·标准化原则：数据标准化是实现高价值数据、支撑以数据为基础的相 关业务的先决条件。组织必须制定可参考、可落地的标准。当发生争议 的时候，有权威的标准可供仲裁参考。

·透明原则：除了一些需要保密的安全措施之外，数据治理相关的文 件、数据问题的发现等，都应该是公开透明的，相关人员应该清楚正在 发生的事情，以及事情发生后应如何按照原则处理。

·数据的认责与问责：数据治理必须解决无人问责的问题，比如将很多 岗位列为负责人，最终却没有人真正负责。数据的认责是数据治理的先 决条件，数据的问责和考核制度是确保数据治理工作真正落地的制度保 障。

·平衡原则：在大数据时代，时时刻刻都在涌现海量数据。在进行数据 治理工作的过程中，必须在代价和收益之间取得平衡。往往没有必要追 求百分之百的数据质量，而对于历史遗留数据，数据标准也不可能对其 进行完全约束。很多时候，对于企业来说，数据可商用是平衡原则的重 要参考。

·变更原则：随着市场和业务的不断发展，数据标准、元数据、数据质 量等要求并不是一成不变的，既要控制数据的变更流程，也要主动适应 这些变化，推动标准更新。

·持续改进原则：业务在不断变化，数据在持续产生，数据治理非朝夕 之功，需要持续推动，不断改进，形成长效机制。

### 数据标准管理

**数据标准可分为以下几类：**

·业务术语标准

·参考数据和主数据标准

·数据元标准

·指标数据标准

**数据标准管理的可行方案：**

数据中心落地：根据数据标准要求建设数据中心（数据仓库或者数据 中台），源系统数据与数据中心做好映射，保证传输到数据中心的数据为标准化后的数据。**这种方式的可行性较高，是绝大多数组织的选择。**

### 数据模型管理

数据模型按不同的应用层次分成概念数据模型、逻辑数据模型、物理数 据模型3种。

**概念模型**是一种面向用户、面向客观世界的模型，主要用来描述世界的概念化结构，与具体的数据库管理系统无关。

**逻辑模型**是一种以概念模型的框架为基础，根据业务条线、业务事项、业务流程、业务场景的需要，设计的面向业务实现的数据模型。逻辑模型可用于指导在不同的数据库管理系统中实现。逻辑数据模型包括网状 数据模型、层次数据模型等。

**物理模型**是一种面向计算机物理表示的模型，描述了数据在存储介质上的组织结构。物理模型的设计应基于逻辑模型的成果，以保证实现业务需求。它不但与具体的数据库管理系统有关，而且还与操作系统和硬件有关，同时考虑系统性能的相关要求。

数据模型管理的关键活动包括：

·定义和分析企业数据需求；

·定义标准化的业务用语、单词、域、编码等；

·设计标准化数据模型，遵循数据设计规范；

·制定数据模型管理办法和实施流程要求；

·建设数据模型管理工具，统一管控企业数据模型。

数据模型是数据资产管理的基础，一个完整、可扩展、稳定的数据模型 对于数据资产管理的成功起着重要的作用。通过数据模型管理可以清楚 地表达企业内部各种业务主体之间的数据相关性，使不同部门的业务人 员、应用开发人员和系统管理人员获得关于企业内部业务数据的统一完 整视图。

### 元数据管理

元数据是数据的户口簿。想想一个人的户口簿是什么，是这个人的信息登记册：上面有他的姓名、年龄、性别、身份证号码、住址、原籍、何时从何地迁入等，除了这些基本的描述信息之外，还有他和家人的血缘关系，比如父子、兄妹等。所有这些信息加起来，就构成了对这个人的全面描述，而这些信息都可以称为这个人的元数据。

**通过这张关于数据的地图，可以知道：**

·有哪些种类的数据；

·有哪些信息系统、哪些数据库、哪些表、哪些字段；

·数据全量是多少，每日增量是多少；

·数据分布在哪里；

·数据之间有什么流向关系；

**元数据分为以下类型。**

·**技术元数据**：库表结构、字段约束、数据模型、ETL程序、SQL程序等。

·**业务元数据**：业务指标、业务代码、业务术语等。

·**管理元数据**：数据所有者、数据质量定责、数据安全等级等。

元数据的管理包含元数据的增删改查、变更管理、对比分析、统计分析等。

·**元数据的增删改查**。通过对不同的角色赋予相应的权限，实现元数据在组织范围内的信息共享。值得注意的是，对元数据的修改、删除、新增等操作，必须经过元数据管理员的**审核流程**。

**·元数据变更管理**。对元数据的变更历史进行查询，对变更前后的版本 进行比对等。

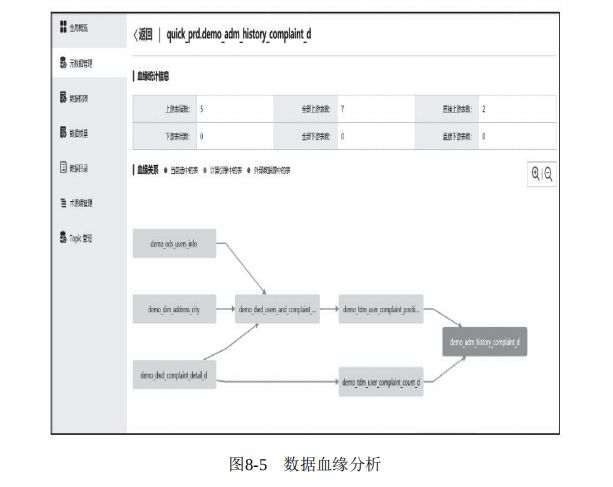
·**元数据对比分析**。对相似的元数据进行比对。比如，对近似的两张表进行对比，发现它们之间的细微差异。

·元数据统计分析。用于统计各类元数据的数量，如各类数据的种类、数量、数据量等，方便用户掌握元数据的汇总信息。

**元数据的应用**

那么有了元数据以后，能做什么呢？

1. **元数据浏览和检索，**通过提供直观的可视化界面，让用户可以按不同类型对元数据进行浏览和检索。通过合理的权限分配，元数据浏览和检索可以大大提升信息在 组织内的共享。
2. **数据血缘和影响性分析**，数据血缘和影响性分析主要解决“数据之间有什么关系”的问题。因其重要价值，有的厂商会从元数据管理中将其单独提取出来，作为一个独立的重要功能。但是考虑到数据血缘和影响性分析其实是来自于元数据信 息，所以还是放在元数据管理中来描述。血缘分析指的是获取到数据的血缘关系，以历史事实的方式记录数据的来源、处理过程等。 以某张表的血缘关系为例，其血缘分析展示如图8-5所示。 数据血缘分析对于用户具有重要的价值，比如当在数据分析中发现问题 数据的时候，可以依赖血缘关系，追根溯源，快速定位到问题数据的来 源和加工流程，减少分析的时间和难度。 数据血缘分析的典型应用场景：某业务人员发现“本月客户增长情况”报 表数据存在明显不合理的情况，于是向数据部门提出异议，技术人员通过元数据血缘分析发现，“本月客户增长情况”报表受到上游DWD（Data Warehouse Detail，明细数据层）6张不同的数据表的影响。通过这种方式，技术人员可以快速定位到问题的源头，低成本地解决问题。除了血缘分析之外，还有影响性分析，它能分析出数据的下游流向。当系统进行升级改造的时候，如果修改了数据结构、ETL程序等元数据信息，依赖数据的影响性分析，可以快速定位出元数据修改会影响到哪些下游系统，从而减少系统升级改造带来的风险。从上面的描述可以知 道：数据影响性分析和血缘分析正好相反，血缘分析指向数据的上游来源，而影响性分析指向数据的下游。



1. **数据冷热度分析，**冷热度分析主要是对数据表的被使用情况进行统计，如表与ETL程序、表与分析应用、表与其他表的关系情况等，从访问频次和业务需求角度出发，进行数据冷热度分析，用图表展现表的重要性指数。数据的冷热度分析对于用户有着巨大的价值，其典型应用场景有：如果观察到某些数据资源处于长期闲置，没有被任何用户查看，也没有任何应用去调用它的状态，用户就可以参考数据的冷热度报告，结合人工分析，对冷热度不同的数据做分层存储，以便更好地利用HDFS资源，或者评估是否对失去价值的这部分数据做下线处理，以节省数据存储空间。

### 主数据管理

### 数据质量管理

目前业内认可的数据质量标准有如下几类。

1. 准确性： 描述数据是否与其对应客观实体的特征一致。 举例：用户的住址是否准确；某个字段规定应该是英文字符，在其位置上是否存在乱码。
2. 完整性：描述数据是否存在缺失记录或缺失字段。 举例：某个字段不能为null或空字符。
3. 一致性： 描述同一实体同一属性的值在不同的系统中是否一致。 举例：男女是否在不同的库表中都使用同一种表述。例如在A系统中，男性表述为1，女性表述为0；在B系统中，男性表述为M，女性表述为 F。
4. 有效性： 描述数据是否满足用户定义的条件或在一定的取值范围内。举例：年龄的值域在0~200之间。另一个枚举的有效性例子是银行的币种代码。
5. 唯一性： 描述数据是否存在重复记录。举例：身份证号码不能重复，学号不能重复。
6. 及时性： 描述数据的产生和供应是否及时。 举例：生产数据必须在凌晨2：00入库到ODS（Operational Data Store， 操作数据层）。
7. **稳定性**：描述数据的波动是否稳定，是否在其有效范围内。 举例：产品质量抽样统计的合格率，不会有超过20%的波动范围。
8. **连续性**：描述数据的编号是否连续。 举例：有关部门处理环保违法案件，案件的编号必须是连续的。

9）**合理性**：描述两个字段之间逻辑关系是否合理。举例：企业注销时间必须晚于注册时间，自然人的死亡时间必须晚于出 生时间。

### 数据安全管理

### 数据价值管理

### 数据共享管理

### 生命周期管理

### 标签管理

### 数据资产门户

数据资产目录的组织方式灵活多样，常见的有**按业务域组织、按数据来源组织、按数据类型组织**。

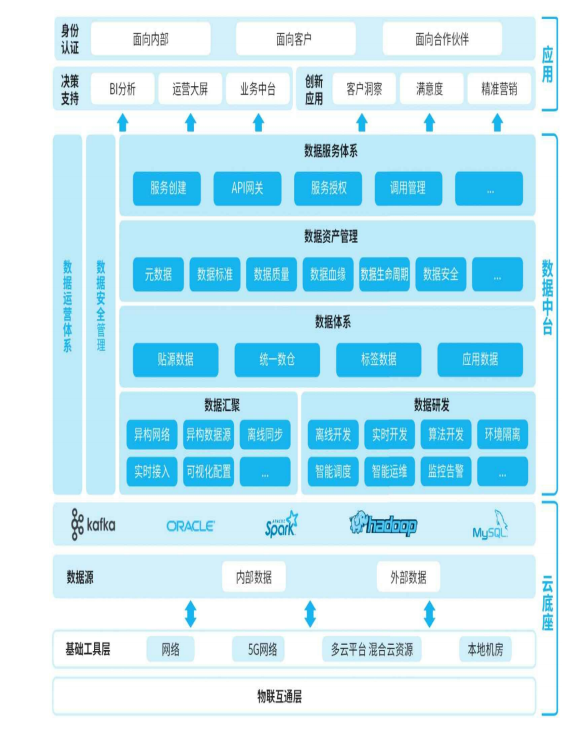
根据用户角色的不同，数据资产目录有多种展现视角，概括来讲，有3 类用户角色：数据资产开发者、数据资产管理者和数据资产使用者。

·**数据资产开发者**关注当前开发的数据资产是否有重复，是否有准确的定义，通过数据资产目录，数据资产开发者可以将自己负责开发的数据资产发布到合适的资产目录下。

·**数据资产管理者**必须掌握数据资产的全局情况，包括拥有哪些数据资产、数据资产分布在哪里、数据资产的质量情况、数据资产的使用情况等。数据资产管理者通过对数据资产的合理授权，控制数据资产的使用。

·**数据资产使用者**关心数据是什么、数据在哪里、如何获取到数据。通过数据资产目录和获取到的合理授权，数据资产使用者能快速定位到自己需要的数据资产，掌握数据资产的存在形式是什么（结构化还是半结构化），如何获取到自己想要的数据，评估现有的数据资产能否满足所建应用的需要。

# **数据服务体系**



按照数据与计算逻辑封装方式的不同，数据服务可分为以下三类：

·**基础数据服务**：它面向的对象是物理表数据，主要面向的场景包括数据查询、多维分析等，通过自定义SQL的方式实现数据中台全域物理表 数据的指标获取和分析。

·**标签画像服务**：它面向的对象是标签数据，主要面向的场景包括标签圈人、画像分析等，通过界面配置方式实现数据中台全域标签数据跨计算、存储的统一查询分析计算，加快数据应用的开发速度。

·**算法模型服务**：它面向的对象是算法模型，主要面向的场景包括智能营销、个性化推荐和金融风控等，主要通过界面配置方式将算法模型一键部署为在线API，支撑智能应用和业务。



四类：

**查询服务**

**分析服务**

**推荐服务**

**圈人服务**

# **附记：需要重点关注的几种工具及技术**

大数据工具：Cloudera

系统日志同步工具：loghub、Flume、logstash

数据开发类工具：

开源：基于Hadoop的**HUE、zeppelin、kettle**，

商业的有IBM的InfoSphere DataStage，Cloudera的CDH和CDP，国内有阿里的**DataWorks**，腾讯的**TBDS**，网易的猛犸，数澜的数栖。另外，有些产品的源代码是开源的，但是商业维护是收费的，如Talend的Data Integration。算法类工具开源的有Jupyter，商业的有阿里云的PAI、腾讯的智能钛、第四范式的Sage AI等。这些工具都能不同程度地兼容 Python、Spark MLlib、TensorFlow、MXNet等各类算法库和算法框架