> 我是谁，我从哪里来，我要到哪里去，我会做什么，我能做什么？我时常反思这些问题，才不至于在快速发展的社会中迷失。
>
> 作为数据从业者，我们也需要探查数据的本质，并对其进行追踪、登记、管理，才不至于在海量数据中迷失。

今天这篇文章将会详细介绍描述数据的数据：元数据，并给出具体的落地实施方案。

一、元数据是什么

1、定义：

描述数据的数据，本质上还是数据。

2、解读：

数据本身带有的技术属性与其在业务运行中的业务属性，我们称其为元数据，例如：表数据量、占用空间、字段信息、业务描述、负责人、优先级等。

3、作用：

元数据通过全局统一的数据描述信息及系统化管理，统一数据标准，促进数据集成和共享，打通企业内部数据孤岛，提升数据管理和应用效率。

二、元数据的组成

> 元数据的边界范围及其划分方式，尚未有统一标准。
>
> 以下内容仅代表目前我方团队总结的最佳实践，求同存异，欢迎讨论。

目前常见元数据分类包括：技术元数据、业务元数据、操作元数据、管理元数据、行为元数据、运营元数据、服务元数据。

每个分类下面还有繁多的属性，但是究其本质，我们可以将元数据根据属性来源划分为两类：

1、数据本身的特定属性，为**技术元数据。**

2、业务赋予的可变属性，为**业务元数据。**

1、技术元数据

不可手动编辑，自动获取

主要服务于开发人员，帮助明确数据存储、结构、权限等信息，为数据开发和系统集成奠定基础。

服务于业务人员，通过数据血缘理清数据关系，定位业务流程，辅助业务开展。

技术属性主要包括以下几类信息：

1）基础信息

表的schema信息以及字段信息等，包含以下字段：

库名称、库类型、表名称、表数量、表注释、表分区字段、表分区数量、字段名称、字段类型、字段长度、字段注释、字段默认值、主键信息、外键信息、索引信息等。

2）存储信息

本地存储中的文件信息，包含以下字段：

文件路径、文件数量、文件大小、文件类型，压缩格式等。

3）调度信息

离线与实时任务中的信息，包含以下字段：

任务名称、任务类型、任务路径、调度时间、调度SQL、调度逻辑等。

4）血缘信息

数据加工、流转过程产生的数据与数据之间的关系，包含以下内容：

数据节点、流出节点、中间节点、流入节点、节点属性等。

2、业务元数据

业务赋予，手动登记

通过明确业务属性，统一数据的业务含义，保持团队认知一致，进而为数据分析和应用更好的提供支撑。

业务元数据包括以下几类信息：

1）业务信息

业务描述、业务部门、业务系统、负责人等。

2）标准化信息

用于统一认知，消除歧义，包含以下字段：

指标名称、指标层级、指标口径、维度信息、计算方式、映射信息、转换规则等。

3）数据质量信息

针对当前数据进行的质量监控内容，包含以下字段：

质量监控名称、监控内容、监控级别、监控规则、告警方式等。

4）权限信息

访问权限、角色权限、用户权限、安全等级等。

5）服务信息

当前数据对外提供服务的方式，包含以下字段：

服务方式（接口、报表、sdk等）、服务内容、接口信息、负责人等

> 我方团队并不认可将数据变更记录、任务执行日志等纳入元数据的范围。
>
> 元数据只应包含属性信息，不包含行为记录。

三、我们为什么需要元数据

1、数据定位模糊，理解困难

数据开发过程中，我们常常会迷失在底层海量数据中，无法快速定位目前所需数据。

定位到数据后，还需花费大量时间理解当前数据，理解渠道包括但不限于：询问同事、查看数据详情、查

询数据权限、查看底层存储、定位影响分析等。

综上所述，在使用数据时，我们往往需要花费大量时间去定位并理解当前数据。

2、数据管理能力低下

数据管理能力是企业实现数据资产化的重要前提。

业务快速发展，数据量成指数级递增。与此同时却没有一个有效的管理手段，数据散落在各地，存储成本与使用成本上升，导致企业数字化转型、数据化运营无法顺利开展。

3、数据孤岛，各自为战，标准不一

数据部门的职责之一是汇集各方数据，进行集中管理。

在此过程会发现各来源方的数据标准不一，规则混乱，且存在重复建设。部门间互相割裂，都有对数据独到的理解与使用，此时数据孤岛便产生了。

出现此情况的原因在于部门间各自为战，缺少统一的元数据管理对数据标准，业务含义等进行同步，从而统一认知，避免数据孤岛的出现。

> 数据孤岛也称为数据烟囱，可无论是“烟囱”还是“孤岛”，总要有“破局”的时候。

4、集成度低，东奔西跑

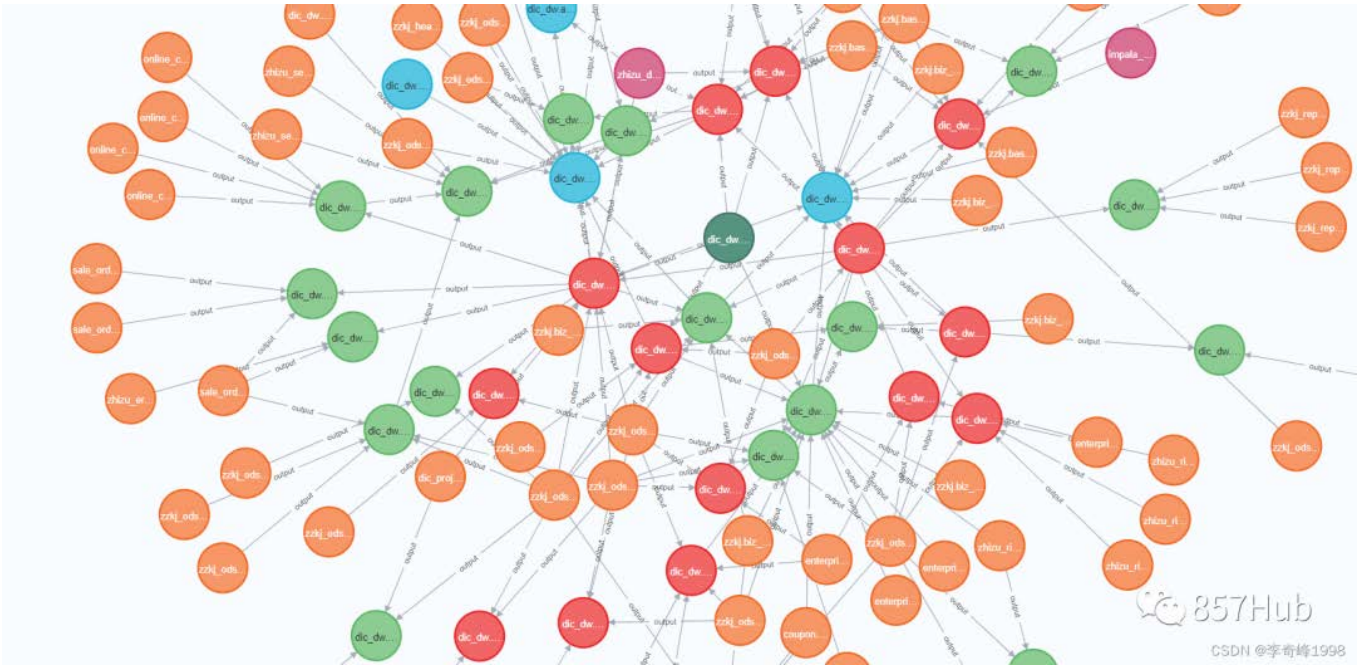
开发过程中，我们需要切换各个开发工具之间进行数据查看与操作。例如通过数据库工具提交SQL操作，通过AirFlow进行任务调度，通过Kafka进行管道操作等。

在多个开发工具中定位与操作数据，过程较为繁琐。如果能够打通多个平台，将信息集中展示并且统一操作入口，可以使开发更高效。

5、数据依赖混乱

大数据开发作为数据汇集与数据服务提供方，庞大的数据与混乱的数据依赖导致管理成本与使用成本飙

升。



6、数据治理推动困难

数据治理工作涉及范围较广，且是一个持续不断的过程。需要多部门，全流程打通。而数据治理的开展，

操作，过程把控，结果验证，需要一个统一的元数据管理平台进行辅助。

四、元数据可以做什么

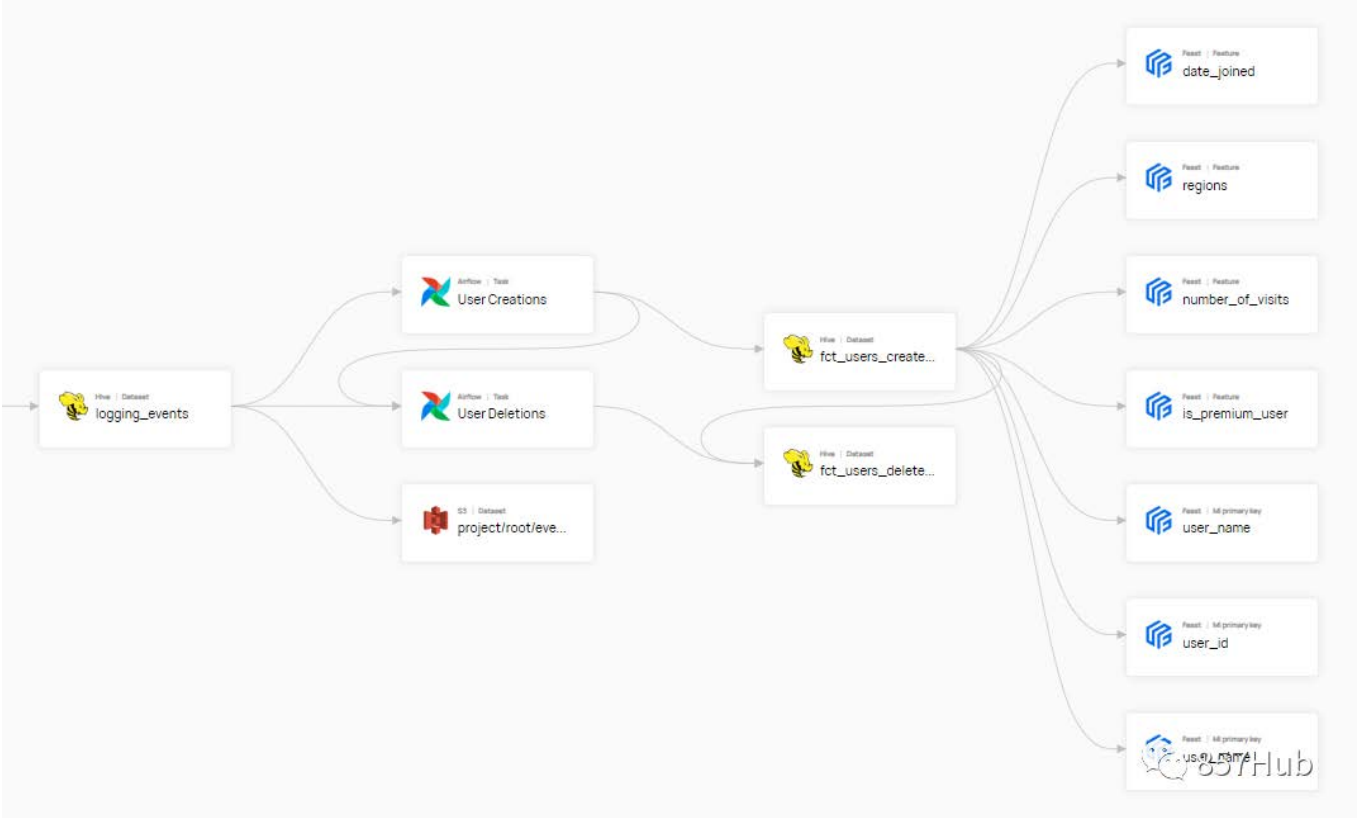
1、快速定位，理解数据

通过全文检索以及分类筛选，快速定位目前所需数据，并根据已有元数据信息进行理解。

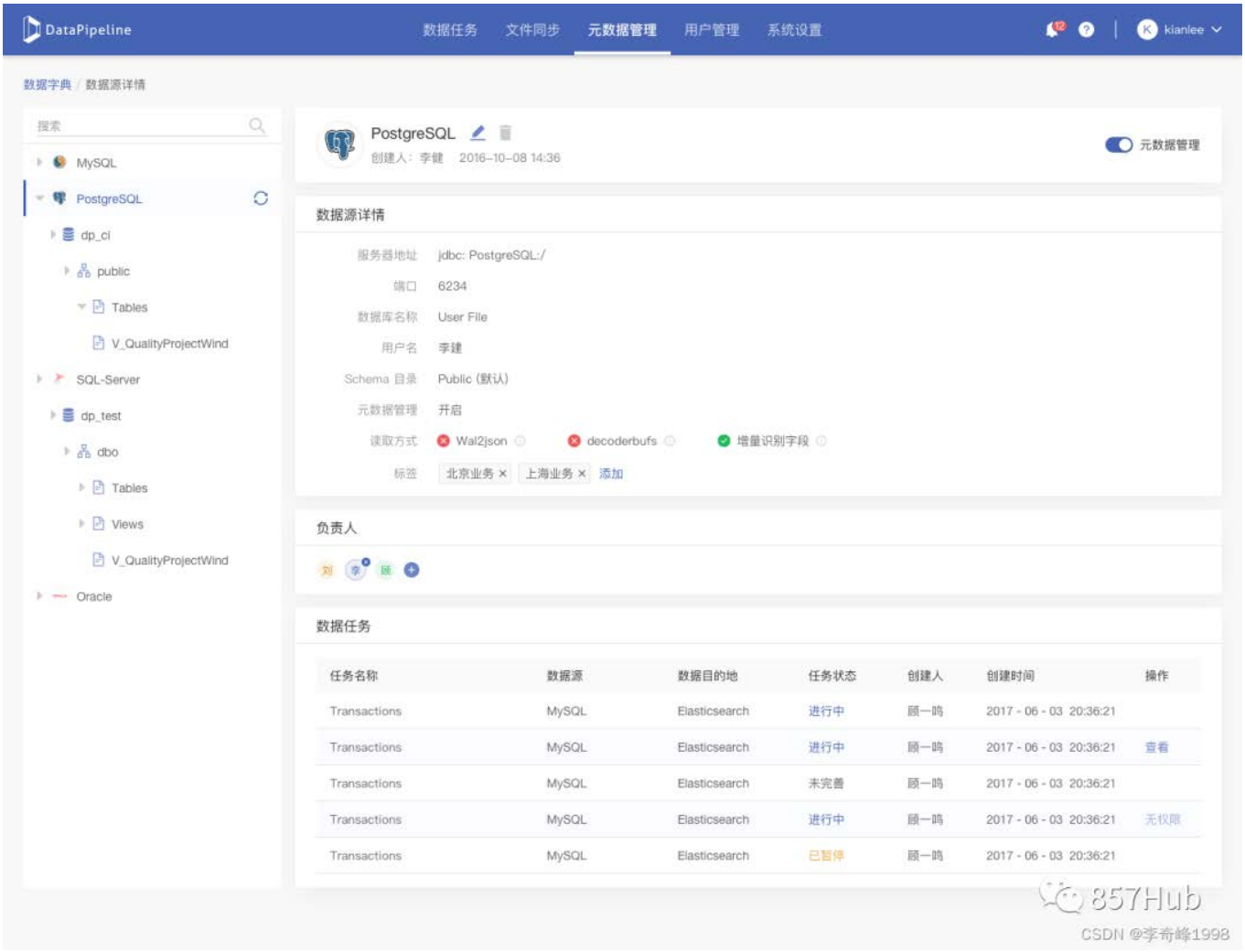
2、数据血缘，流程定位，追踪溯源

数据血缘是在数据的加工、流转过程产生的数据与数据之间的关系。

通过构建数据血缘，进行数据关系探查，用于跟踪数据流经路径，追踪溯源。



3、统一管理，赋能业务



通过元数据管理平台可以对数据进行为之有效的管理，赋能与数据开发与业务使用，加速企业数字化转型。

4、打通孤岛，对齐认知，统一标准

梳理并登记各业务部门的元数据信息，并进行同步。快速对齐各部门的数据认知，统一标准，消除歧义。避免数据重复建设的情况。

5、数据集成，快速开发

打通各类开发组件，并且通过元数据管理平台，以数据视角触发各种操作，例如表级调度，权限修改，schema信息修改，提交SQL任务等。

6、推动数据治理

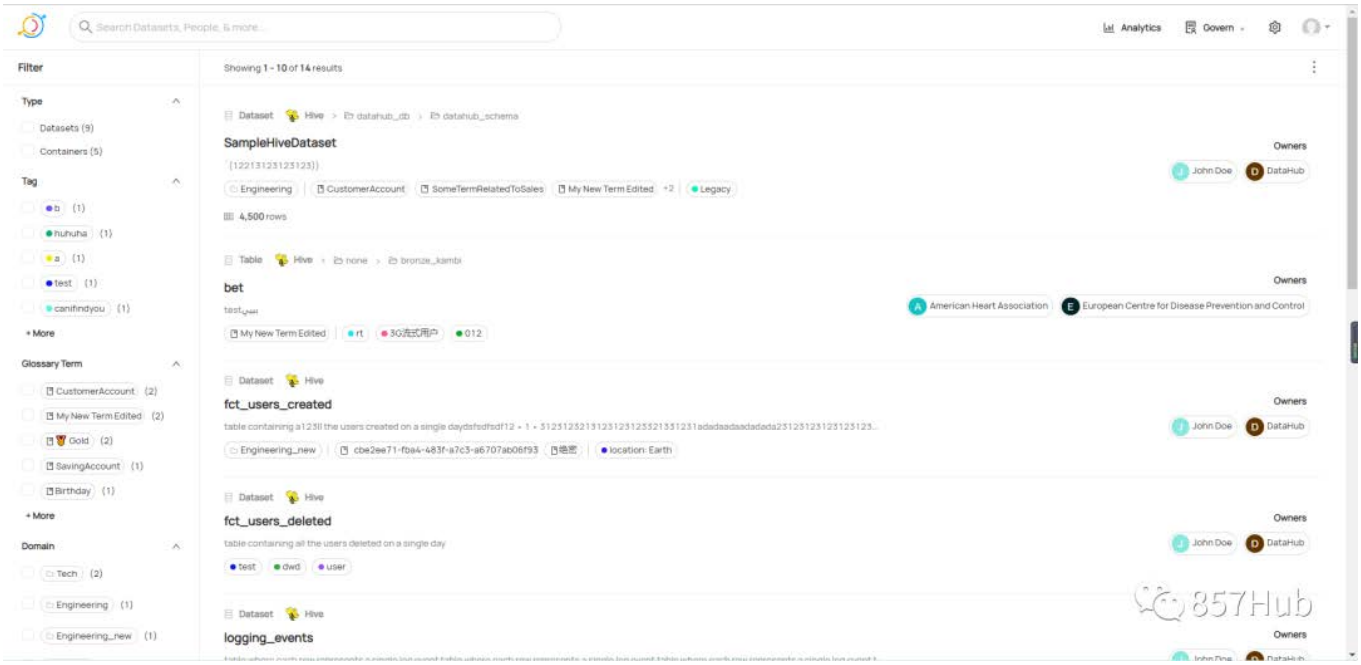
通过元数据管理平台，串联数据链路中的开发人员、管理人员和业务人员，打通多部门，全流程。推动开展数据治理、把控过程、结果验证。

五、元数据管理落地方式

目前业内常见的元数据管理落地方式，主要有以下三种：

1、采用开源系统：

Metacat、Datahub、Atlas等



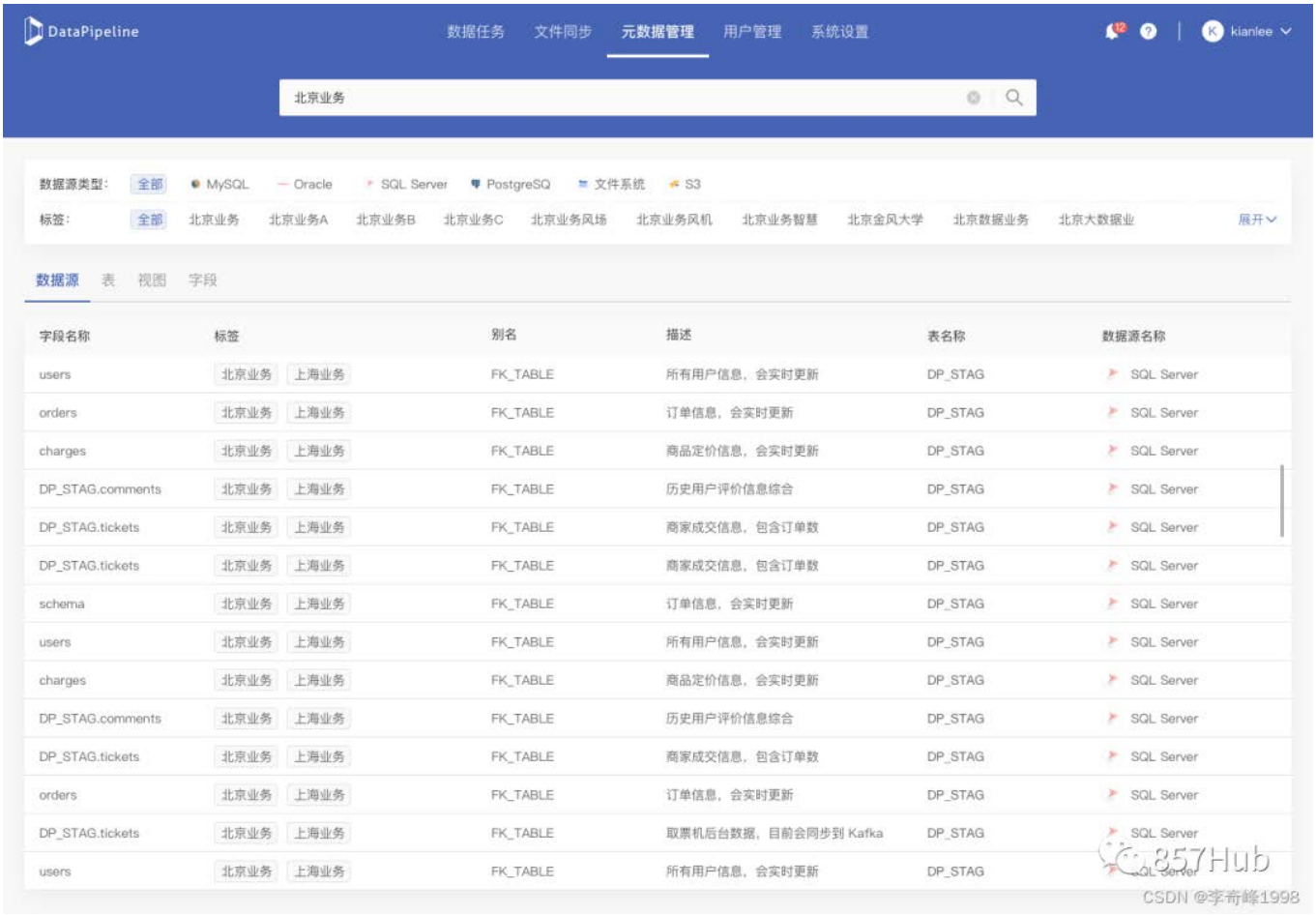
采用开源系统最大的优点是投入成本较低，但是缺点主要包括：

1、适配性较差，开源方案无法完全匹配公司现有痛点。

2、二开成本高，需要根据开源版本进行定制化开发。

2、厂商收费平台：

esensoft，DataPipeline等



此类数据平台中会内置元数据管理系统，功能较为全面，使用方便。但是同样也有以下缺点：

1、贵

2、需要**ALL IN**平台，为保障数据血缘的使用，数据业务需要全部迁移到厂商平台中。

3、自建

通过设计元模型、构建采集器、后端、前端自建元数据管理系统，此方案开发投入较大，但是有以下优点：

1、因地制宜，可根据核心痛点定制化开发元数据及数据血缘系统。

2、技术积累，对于开发人员来说，从0-1开发数据血缘系统，可以更深刻的理解数据业务。

3、平台解耦，独立于数据平台之外，数据血缘的开发不会对正常业务造成影响。

> 接下来我们讲讲如何自建元数据管理系统与落地实施。

六、元数据管理落地实施

0、挖掘痛点，推动实施

为什么要把**挖掘痛点，推动实施**这步放在首位，且步骤序号为0呢？

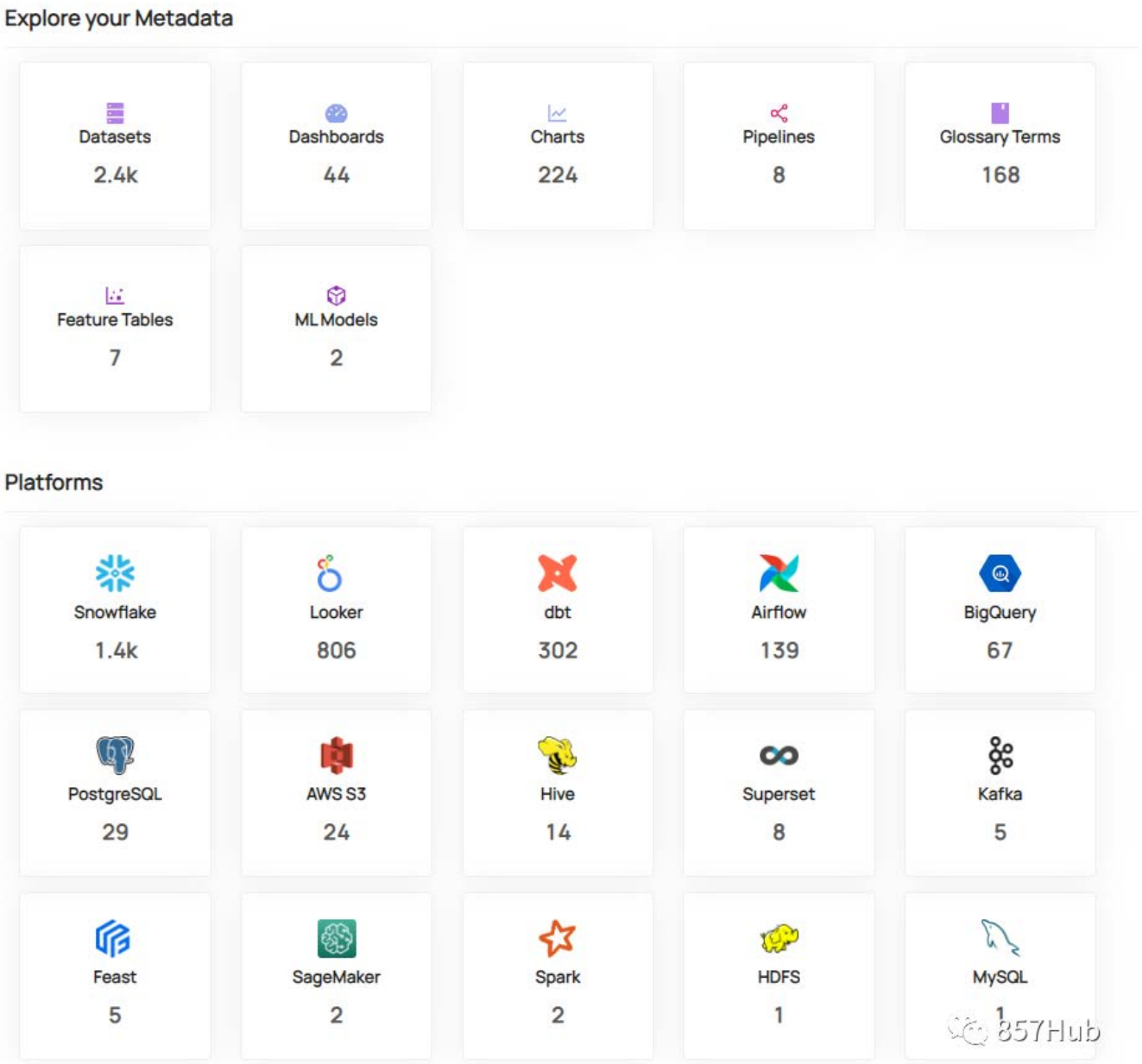
首先很多公司因为数据量少，团队规模小，没有体会到数据管理混乱带来的困扰，也就没有元数据管理的建设需求。

其次因为公司没有元数据建设经验，上级部门不重视等原因所以也无法进行元数据的落地实施。

所以在元数据管理的建设之前，一定要深度挖掘当前数据开发与业务推进中的数据管理痛点，并且通过自身的专业能力推广元数据管理的优势，进而推动落地实施。

此步骤序号为0，意思就是我们自身能够意识到元数据管理的重要性还远远不足，在实施层面还要引起上级重视，推动部门协作，切记不可闭门造车。

1、明确需求，确定边界



进行元数据管理系统构建之前，需要进行需求调研，明确系统主要功能，从而确定元数据模型的最细粒度，元数据采集的边界范围，系统的应用方式等。

根据本文第二章 元数据的组成 可以得出，元数据中属性信息众多，在开发时如果将所有属性信息都进行获取且登记，一方面开发成本较大，其次实施过程也较为繁琐。

所以前期需要明确需求，确定系统主要功能以及元数据采集的边界，后续才可以根据业务痛点给出更加准确的解决方案，提高**ROI（投入产出比）。**

以我方团队为例，在系统设计阶段，我们在收集了目前痛点以及梳理现有系统以及组件后，明确了系统初期建设时的三个核心业务：

1. 数据定位、展示、快速理解。

2. 数据血缘，流程定位，追踪溯源

3. 数据集成，快速开发

所以在初期开发时，我方团队可以只针对以上业务进行元数据信息的采集与登记，后续根据迭代需求进行扩展。

2、元数据建模

元数据本质上也是数据，开发时需要对其进行数据建模。元数据模型我们称其为**元模型**。

一个统一的元模型可以规范元数据的范围边界，统一元数据格式与存储方式。

目前数据领域存在多种类型的数据库系统，例如：

- 关系型数据库：Mysql、Oracle
- MapReduce数据库：Hive
- 键-值模型数据库：redis、Memcached
- 列族模型数据库：Hbase
- 文档模型数据库：MongoDB
- 图模型数据库：Neo4j、Nebula
- 搜索引擎：Elasticsearch、Solr
- MPP数据库：Greenplum、Doris

各类数据库都有其独特的存储模型与适用场景，但在元数据层面，所有数据都需要统一的元模型，无论数据库是关系型的，还是非关系型的。在元数据层面它们都代表一个个数据实体。

元数据建模一般采用**E-R模型**，将元数据抽象为 "实体"（Entity）、"属性"（Property）、"关系"（Relationship） 来表示元数据各属性以及不同层级之间的关系。

元数据信息的保存采用结构化数据库即可，目前我方团队采用Mysql保存元数据信息。

> 数据血缘信息采用Neo4j图数据库存储。

3、元数据采集

此处我认为是元数据管理系统的开发难点，也是最难推进的。

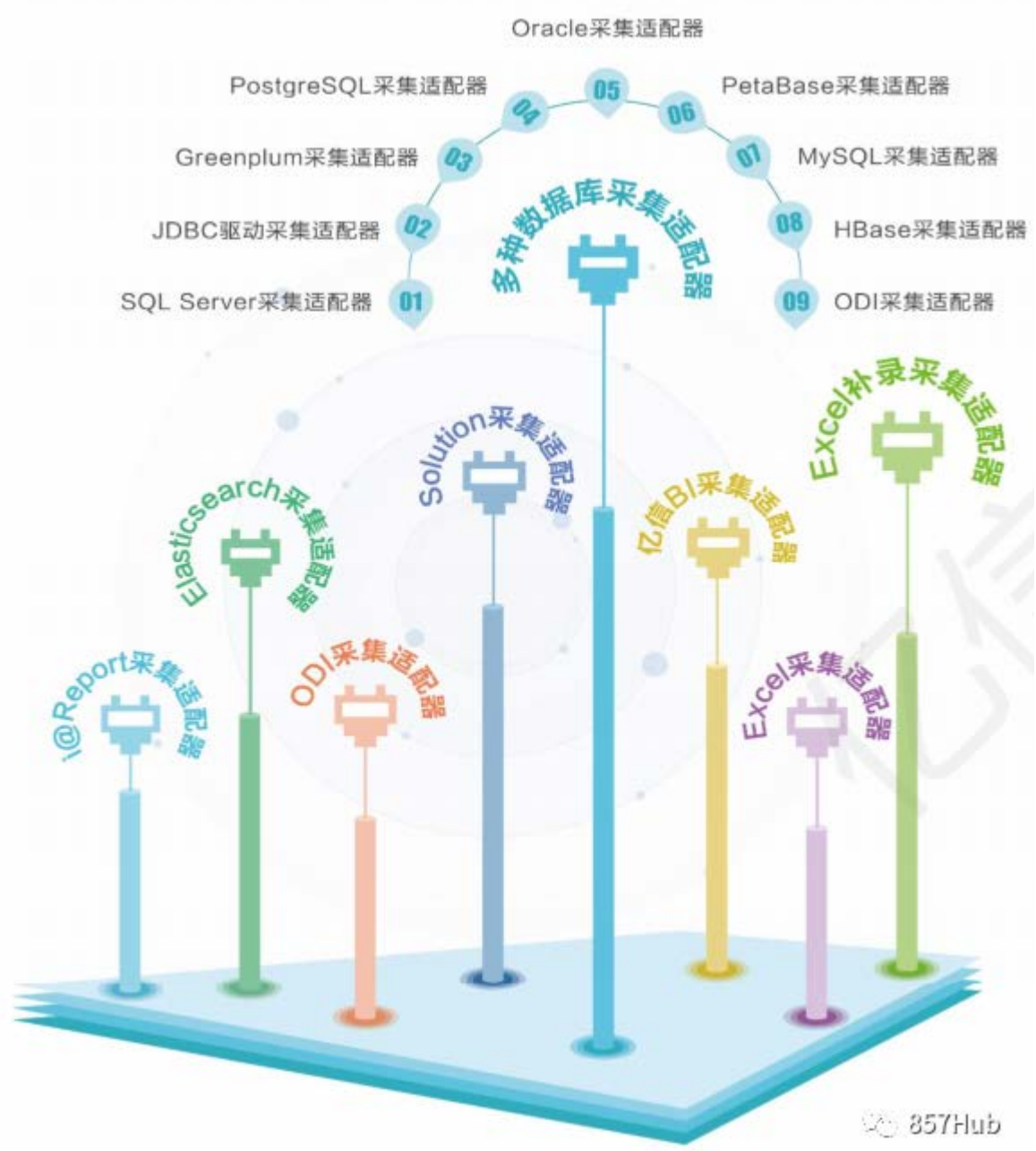
首先我们需要打通各数据组件采集技术元数据，这是**技术难点。**

其次我们需要数据开发与使用人员配合进行业务元数据的登记，这是**管理难点。**

1）自动化采集【技术元数据】

如果想要真正做一个数据集成度高的元数据管理系统，在目前大数据繁多的组件下，我们需要打通的组件包括但不限于：

- 中间件：Kafka、RabbitMq、RockitMq等
- 数据库：Hive、Kudu、Hbase、Mysql、Doris、Greenplum、ElasticSearch、PostgreSQL等、
- 调度系统：DolphinScheduler、AirFlow、Oozie等、
- 底层存储：HDFS、AWS S3 等
- 计算引擎：Flink、Spark、Impala等
- BI报表：Tableau、Superset等



目前我方团队已打通组件列表如下：

1. 通过JDBC打通Impala获取Hive与Kudu的库表基础信息
2. 通过文件方式打通Tableau获取中所有报表以及自定义SQL信息
3. 通过SDK方式打通HDFS获取存储信息
4. 通过API方式打通DolphinScheduler获取当前表的调度信息以及抽取SQL
5. 通过SDK方式打通Kafka获取当前表的来源Topic相关信息
6. 通过API方式获取Spark任务的执行信息和状态

2）手动登记【业务元数据】

由于企业缺乏统一的数据标准，数据孤岛现象明显，所以即使采集到技术元数据，也只是方便开发人员进

行开发与运维，无法做到真正的统一管理。

此时需要采用人工方式对现有数据进行梳理并登记业务元数据，以实现统一管理。但是在登记过程中，往往会出现很多问题，例如数据标准不一致、业务含义不清晰、相关人员不配合等情况，上述问题主要由于企业忽视数据管理所导致。

所以在元数据管理落地之前，我们就需要挖掘目前数据管理的痛点，并且在公司内部推动元数据管理的落地实施，这样才能更好的推动数据的梳理与登记。

4、构建数据血缘

数据血缘构建详情，参考以下文章：

【实战讲解】数据血缘落地实施

5、管理系统开发

元数据管理系统的功能应包括但不限于以下几点：

1. 元数据概览：展示当前系统纳管的元数据概览，可以从元数据来源组件或项目类别查看。
2. 元数据检索：通过全文检索以及分类筛选的方式，快速定位目前所需的元数据信息。
3. 元数据详情展示：展示当前元数据详情信息，包括技术元数据与业务元数据。
4. 元数据编辑：用于登记业务元数据，并进行更新修改。

5. 数据血缘展示操作：用于前端展示数据血缘信息，并点选操作。
6. 元数据采集管理：用于管理元数据采集的组件连接信息

6、元数据驱动

我们团队目前落地元数据管理后，带来的改进有以下几点：

1) 数据集成开发

通过元数据打通调度系统、中间件、数据库操作等，实现数据的集成式开发，提高开发效率。

2) 元数据查询、定位、展示

通过元数据管理系统定位数据并展示，大大提升数据共享能力，数据部门人来人往的现状得到改善，咨询数据的消息也大幅减少。

3) 推动数据与业务梳理

建设元数据的过程中，通过推动各部门间进行数据梳理，将原先混乱的数据及业务进行整体排查，减少冗余数据，明确业务逻辑。

4) 辅助数据运维

- 通过对技术元数据中的数据总量进行排序筛选，对数据量极少的表进行下架整改，对数据量极多的表进行逻辑优化。
- 通过对技术元数据中的表文件数量进行排序，对文件数量极多的数据表进行小文件合并。
- 通过对技术元数据中的表文件大小进行排序，对占用空间极多的表开启压缩。

5) 规范数据开发

公司内部的开发规范例如库表、字段的命名规范，字段类型规范等。

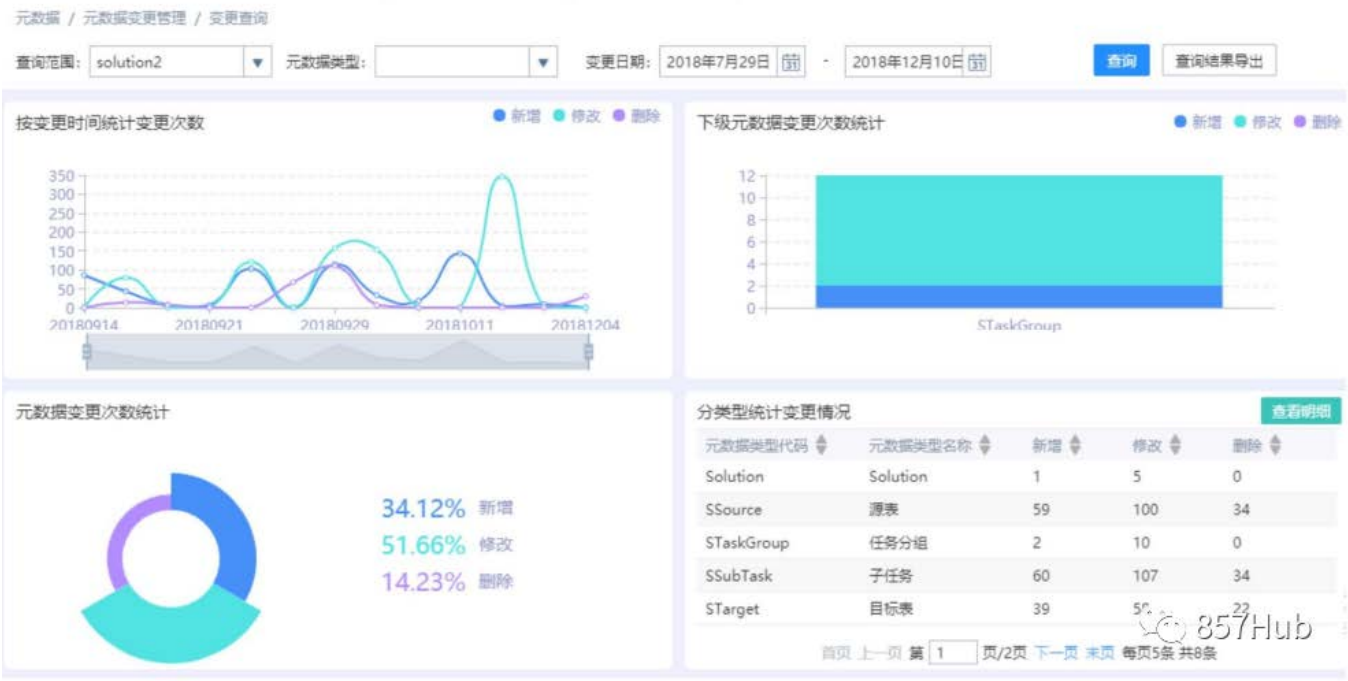
可以通过元数据对目前已有信息进行标准化校验，用于规范数据开发。

6) 梳理数据依赖关系

通过数据血缘梳理数据依赖关系，流程定位，追踪溯源。

7) 成果汇报、工作量统计

通过元数据信息的增减以及构建可视化展示页面，可以快速展示当前数据开发团队的成果，统计工作量等。



七、元数据评价标准

在推动元数据管理落地过程中，经常会有用户询问：元数据模型质量如何？采集信息是否全面？能否解决他们的痛点？做出来是否好用？

于是我也在思考，市面上元数据管理系统那么多，我们自建的核心优势在哪里，元数据的优劣从哪些层次进行评价，于是我方团队量化出了以下三个元数据评价的技术指标：

> 以下指标关注元数据，对于其管理系统的稳定性，易用性等不进行关注。

1、准确率

定义： 假设一张表实际的数据属性与元数据中采集的数据属性相符，既不缺失也不多余，属性值一致无误，则认为这个表的元数据是准确的，元数据准确的数据量占全部元数据量的比例即为元数据准确率。

准确率是元数据中最核心的指标，元数据的属性缺失或异常可能会造成数据不一致，从而导致业务开展顺利进行，严重则会导致生产故障。

我们在实践中通过两种途径，尽早发现有问题的血缘节点：

人工校验： 通过构造测试用例来验证其他系统一样，元数据的准确性问题也可以通过构造用例来验证。实际操作时，我们会从线上运行的任务中采样出一部分元数据，人工校验是否正确。

用户反馈： 全量元数据的准确性验证是个漫长的过程，但是具体到某个用户的某个业务场景，问题就简化多了。实际操作中，我们会与一些业务方深入的合作，一起校验元数据准确性，并修复问题。

2、覆盖率

定义： 当有数据资产采集至元数据中时，则代表元数据覆盖了当前数据资产。被元数据覆盖到的数据资产占所有数据资产的比例即为元数据覆盖率。

元数据覆盖率是比较粗粒度的指标。作为准确率的补充，用户通过覆盖率可以知道当前已经支持的数据资产类型，以及每种覆盖的范围。

在内部，我们定义覆盖率指标的目的有两个，一是我方比较关注的数据资产集合，二是寻找当前业务流程中尚未覆盖的数据资产集合，以便于后续优化。

当血缘覆盖率低时，元数据管理的应用范围一定是不全面的，通过关注元数据覆盖率，我们可以知晓元数据管理的落地进度，推进有序落地。

3、时效性

定义： 从数据资产新增和任务发生修改的时间节点，到最终新增或变更的血缘关系录入到血缘系统的端到端延时。

对于一些用户场景来说，血缘的时效性并没有特别重要，属于加分项，但是有一些场景是强依赖。不同任务类型的时效性会有差异。

例如：故障影响范围告警以及恢复，是对血缘实时性要求很高的场景之一。如果血缘系统只能定时更新T-1的状态，可能会导致严重业务事故。

提升时效性的瓶颈，需要业务系统可以近实时的将任务相关的修改，以通知形式发送出来，并由血缘系统进行更新。

八、杂谈

1、元数据为何越来越火

从百度指数可以看到，元数据管理的关键词近年来呈逐级上升的态势，且从857数据社区内的讨论中，元数据出现的频率与讨论热度也在众多数据话题中名列前茅。我也在思考为何元数据在近几年越来越火，且各大厂商都在积极推进落地。

我的看法是：目前大数据行业的发展已进入到成熟阶段，数据平台成熟度、大数据开源组件数量、从业人员水平相比从前均有较大提升。

同时由于海量数据的增长，企业应用数据能力的痛点凸显，数据管理能力低下，无法对目前已有的数据资源进行充分利用。

在此背景下，元数据由于其出色的数据洞察能力与数据管理能力，被越来越多的企业重视。

本篇文章目的就是帮助企业数据从业者理清元数据的来龙去脉，给出落地实施方案，从而推进数据行业更

好更快发展。

2、元数据和主数据的区别

元数据

元数据是描述数据的数据，主要为数据的技术属性以及业务属性。

主数据

主数据指的是企业核心业务对象，且在企业系统内部共享。从维度建模的角度来看，主数据一般存在企业的一致性维度表中，例如客户维度表、商品维度表、地区维度表等。

主数据具有4个主要特征：唯一性、有效性、稳定性、共享性。

区别与联系

主数据其实就是维度数据

元数据可以用来管理主数据

3、对于数据从业者的建议

个人认为，元数据后续的发展与行业占比会呈现稳步上升趋势

对于数据从业者来说，我们需要跳脱出开发者的视角，从全局角度去审视数据的发展，所以掌握元数据必不可少。

我也希望通过这篇文章，帮助更多数据从业者掌握元数据技能，从而提升自身的数据管理能力，应用能力，业务理解能力以及开发能力。

4、这篇文章的构建历程

起初是因为在建设数仓过程中，遇到数据管理的痛点，导致数仓推进效率降低，同时团队leader也想通过元数据管理加速公司的数字化转型，提高精细运营能力。

其次是因为857社区内部，也希望奇峰出一篇元数据的讲解文章，目的是为了更好的和面试官battle（多么单纯的目的啊）

同时上月发布的数据血缘文章在行业获得了一致好评，奇峰在文章中也承诺一周内更新《元数据管理落地实施》，没想到一拖就是20天，在大家的催更与自我驱动下，还是勉强完成了此篇文章。

最后：希望自己的思考能为大数据行业带来一些好的改变。