上海交通大学学位论文

基于知识图谱的数据治理平台构建及实现

姓　名： 陈煜

学　号： 520030910285

导　师： 蔡鸿明

学　院： 电子信息与电气工程学院

专业名称： 软件工程

申请学位层次： 学士

2024 年 6 月

A Dissertation Submitted to

Shanghai Jiao Tong University for Bachelor Degree

CONSTRUCTION AND IMPLEMENTATION OF A DATA GOVERNANCE PLATFORM BASED ON KNOWLEDGE GRAPH

Author: Chen Yu

Supervisor: Cai Hongming

School of Electronic Information and Electrical Engineering

Shanghai Jiao Tong University

Shanghai, P.R.China

June, 2024

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全知晓本声明的法律后果由本人承担。

学位论文作者签名：

日期：2024 年 6 月 3 日

上海交通大学

学位论文使用授权书

本人同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。

本学位论文属于：

☑**公开论文**

□**内部论文**，保密□1 年/□2 年/□3 年，过保密期后适用本授权书。

□**秘密论文**，保密__年（不超过 10 年），过保密期后适用本授权书。

□**机密论文**，保密__年（不超过 20 年），过保密期后适用本授权书。

（请在以上方框内选择打“√”）

学位论文作者签名：　　指导教师签名：

日期：2024 年 6 月 3 日　　日期：2024 年 6 月 3 日

摘 要

在当今数据驱动的商业环境中，数据不仅仅是一种资产，它也是企业竞争力的核心。有效的数据治理不仅可以提高数据的使用效率，而且对于提高企业的运营效率、增强决策的质量、保障数据安全和合规性至关重要。然而，随着数据量的爆炸式增长和数据类型的多样化，企业在数据管理上面临着前所未有的挑战。数据孤岛、数据质量不一、元数据不匹配等问题层出不穷，这些问题严重阻碍了数据的有效利用，增加了企业的运营成本，降低了决策的准确性。

在众多的数据治理工具和方法中，知识图谱因其独特的优势而成为解决上述问题的有力工具。知识图谱通过构建数据之间的关系网络，不仅可以实现数据的语义化表示，还能够提供数据之间的关联性分析，为数据的整合和质量控制提供了新的视角和方法。

因此，本文旨在探索和设计一种基于知识图谱的数据治理方法，该方法不仅能够从多种数据集中抽取和融合数据元数据，生成数据资产目录，而且还能够在知识图谱的基础上实现一个具有通用意义的数据治理平台。通过这个平台，企业可以更加高效地管理其数据资产，提高数据的可访问性和质量，支持更加精准的决策制定，最终推动企业的持续创新和增长。

本文具体完成的工作有：（1）设计并构建了面向数据治理的知识图谱；（2）完成了元数据融合方法；（3）开发了数据资产管理系统，实现数据的分类、编目和质量控制；（4）实现可视化交互界面。

关键词：知识图谱，数据治理，数据资产管理，元数据融合

ABSTRACT

In today's data-driven business environment, data is not just an asset; it is the core of corporate competitiveness. Effective data governance can not only improve the efficiency of data utilization but is also crucial for enhancing operational efficiency, improving decision-making quality, and ensuring data security and compliance. However, with the explosive growth of data volume and the diversification of data types, enterprises are facing unprecedented challenges in data management. Issues such as data silos, inconsistent data quality, and mismatched metadata are rampant, severely hindering the effective use of data, increasing operational costs, and reducing decision accuracy.

Among numerous data governance tools and methods, knowledge graphs have become a powerful tool for addressing the above problems due to their unique advantages. Knowledge graphs, by building relationships between data, can not only achieve semantic representation of data but also provide associative analysis between data, offering new perspectives and methods for data integration and quality control.

Therefore, this paper aims to explore and design a knowledge graph-based data governance method that not only extracts and integrates data metadata from various datasets to generate a data asset catalog but also implements a universally applicable data governance platform based on the knowledge graph. Through this platform, enterprises can more efficiently manage their data assets, improve data accessibility and quality, support more precise decision-making, and ultimately drive continuous innovation and growth.

Specifically, this paper has accomplished the following tasks: (1) Designing and building a knowledge graph for data governance; (2) Completing metadata fusion methods; (3) Developing a data asset management system to classify, catalogue, and control data quality; (4) Implementing a visual interactive interface.

Key words: knowledge graph, data governance, data asset management, metadata fusion

目 录

第一章 绪论

1.1 研究背景

在当今的商业环境中，数据已经从一种简单的资产转变为企业竞争力的核心。在这个被大数据和人工智能驱动的时代，企业的成功在很大程度上取决于其如何收集、管理和使用数据。有效的数据治理不仅可以提高数据的使用效率，而且对于提高企业的运营效率、增强决策的质量、保障数据安全和合规性至关重要。

然而，随着数据量的爆炸式增长和数据类型的多样化，企业在数据管理上面临着前所未有的挑战。首先，数据孤岛问题使得企业无法全面地理解和利用其数据资产。其次，数据质量的不一致性会导致决策的不准确性，增加运营成本。再次，元数据的不匹配问题会阻碍数据的整合和使用。这些问题严重阻碍了数据的有效利用，增加了企业的运营成本，降低了决策的准确性。

在这种情况下，加强数据治理显得尤为关键。数据治理是一个综合性的管理过程，旨在确保数据资产在整个生命周期内的质量、可用性、完整性、安全性和合规性。通过制定一系列政策、程序和控制来管理数据的获取、存储、分配和使用，有效的数据治理可以系统地提升数据地准确性和价值，解决数据孤岛问题，并促进数据的整合与共享，从而显著提高企业的决策效率和精准度。

在众多的数据治理工具和方法中，知识图谱因其独特的优势而成为解决上述问题的有力工具。知识图谱的核心优势在于其能够明确地定义实体之间的各种关系，以及实体与其属性之间的联系，从而允许更精确的数据分析和推理。通过这种方式，知识图谱帮助企业揭示数据中隐藏的模式，洞察潜在的业务机会或风险，并实现更为精准的预测和决策支持。并且这种结构化的数据表达使得复杂信息地检索变得更为直观和高效，知识图谱的应用还能促进数据治理中的数据标准化和元数据管理，对于维护数据一致性尤为重要，通过系统地管理和利用知识图谱，企业能够有效提升数据的整体质量和价值。

1.2 研究现状

本文拟结合知识图谱技术路线，实现基于知识图谱的数据治理平台，解决在数据治理中数据异构和元数据不匹配问题。因此，本文拟对知识图谱、元数据融合、数据资产管理进行调研，并简要介绍研究现状。

1.2.1 知识图谱

数据汇聚指的是从多个数据源收集信息并将其合并成一个统一的视图的过程，通常包含离线在线抽取以及数据转化(ETL)过程，在知识图谱的上下文中，数据汇聚是构建和维护知识图谱的重要步骤。

知识图谱是一个展示实体属性及其相互关系的语义网络。它以图形方式建模和表达知识，模拟了人类的认知结构，从而优化了海量信息的组织、管理和理解[1]。作为一种特殊的数据网络，语义网络为用户提供了一个查询平台，通过图形化方式展现经过处理和推理的知识。知识图谱因此成为了智能化语义搜索的基础和连接桥梁[2]。

知识抽取是构建知识图谱的初步阶段，主要挑战在于自动从异构数据源中提取信息以形成候选知识单元。这一过程涉及从无结构或半结构化数据中自动识别实体、关系及实体的属性等结构化信息的技术[3]。核心技术包括实体识别、关系提取和属性提取。

实体抽取的功能是在分词文本中提取出知识的对象——实体，通常需要用到命名实体识别技术（NER）。NER 技术最初主要采用基于规则的方法进行实体抽取。2012 年，Ling[4]等人借鉴 Freebase 的实体分类方法，归纳出 112 种实体类别，并基于条件随机场模型进行实体边界识别，最后采用自适应感知机算法实现了对实体的自动分类。

由于互联网内容的动态性和 Web2.0 技术的发展，传统的人工预定义实体分类系统难以满足当前的需求。开放域的实体抽取和分类技术[5]为这一挑战提供了解决方案。这种技术主要利用统计机器学习方法，根据实体在数据集中的上下文特征进行分类和聚类，从而有效地识别和组织实体。

关系抽取建立在实体抽取的基础之上，它通过分析文本来识别实体之间的联系，并将这些信息链接起来，形成一个网络化的知识结构。在关系抽取的早期研究中，

主要采用了通过手工构建的语法和语义规则，基于这些规则使用模式匹配技术来确定实体之间的关系。后来，研究人员逐渐采用统计机器学习方法[6]，半监督学习方法[7]，弱监督学习方法[8]。2007 年，Banko 等人[9]提出了 OIE 方法，基于自监督（self-supervised）学习方式, 通过使用少量的人工标注数据作为训练集，可以开发一个实体关系分类模型。这个模型随后被用于关系抽取，从而识别和分类文本中的实体关系。这种方法有效地利用了有限的标注资源来提高关系抽取的准确性。。

属性抽取的目的是从多样化的信息源中获取特定实体的属性数据，以便为每个实体构建详尽的描述。这一过程涵盖了从结构化和非结构化数据源中识别并收集实体相关属性的技术。可以从百科类网站获取大量实体属性数据，但还有大量实体属性数据隐藏在非结构化的数据中，可以通过基于百科类网站半结构化数据训练的模型抽取，也可以通过数据挖掘等方式进行抽取。

对于软件开发领域知识图谱的知识抽取，研究人员也提出了不同的方法。王飞等人[10] 通过采用递进学习（curriculum learning）的方法，可以通过精心安排训练数据中不同关系的顺序，来更充分和可靠地进行知识抽取。这种策略模拟了逐步学习的过程，从简单到复杂地逐步处理信息，从而提高了模型对复杂关系的识别和抽取能力；邢双双等人[11]通过无监督的方式进行软件术语列表抽取，采用改进的 Hearst Pattern[12]和启发式规则抽取双向边关系完成了软件知识图谱的知识抽取。

1.2.2 元数据融合

元数据是关于数据的数据，描述了数据的内容、格式、结构和管理信息，对于数据的检索、理解和使用至关重要。它为数据的发现、解释和有效使用提供了必要的上下文信息，包括了基本的描述性信息如标题、作者、发布日期等，以及访问权限、数据的来源、数据的质量和任何有助于理解数据起源的信息。

元数据融合是数据集成和知识管理领域中的一个重要问题，它涉及从多个数据源中提取、整合和管理元数据的过程。这一过程不仅要求技术上的融合，更要求语义上的一致性，即不同数据源的元数据在融合后能够保持其语义的准确性和完整性。元数据融合的目的在于创建一个统一的视图，以便用户能够跨越不同数据源检索和分析数据。但在实际操作中元数据融合面临的主要问题有标准化问题、数据质量问题、语义一致性问题等。

为解决元数据的多源、多样、分散及非结构化问题。王利亚等人利用自然语言处理（NLP）算法对这些数据进行去重、归一化和消歧，构建了健康大数据平台[13]，这样的平台能够提高数据的可用性和可靠性，为医疗决策和科学研究提供支持。

严承希等人[14]提出的元数据非规范性输入单元的概念进一步强调了元数据融合的复杂性。元数据的非规范性意味着它可能来自于不同的标准和格式，这就需要对其进行转换与复合。他们提出的冲突冗余检测思想及知识合并方法，为处理元数据判重及融合提供了一种策略，这有助于解决在融合过程中出现的冲突和重复问题。

1.2.3 数据资产管理

数据资产一词最初于 1974 年由 Peterson R E 提出[15]，在大数据时代，数据已成为企业中极其重要的财富资产。有效的数据资产管理能力成为企业竞争力的关键因素，决定着企业能否在激烈的市场竞争中取得成功。企业需要通过高效的数据收集、处理和分析方法，来充分利用这些数据资产，从而优化决策过程、提升运营效率和增强客户关系[16]。然而由于多数企业信息化建设初期未进行统一规划同时缺少统一的流程与规范，企业内部各业务部门都建立自己的“烟囱式”IT 系统[17]，数据资源分散在多个业务系统中，存在数据质量不高、管理难度大、数据冗余、共享不足、应用较少等多种问题，数据价值难以发挥的问题。

阿里在数据治理方面走在行业前沿[18]，通过数据的层次化和水平解耦，可以沉淀出公共的数据能力，从而实现数据运营和产品技术的从前台业务中分离。这种分离有助于成立一个独立的部门，该部门通过技术手段连接大数据的计算和存储能力，并通过业务理解来链接数据的应用场景。这样的平台不仅提升了数据处理的效率和灵活性，还能更好地支持企业在不同业务领域中的数据需求和创新，实现数据资产的最大化利用。通过服务重用，实现业务能力共享[19]。利用数据技术对大量数据进行收集、保存、处理和优化，并统一数据标准和测量，建立一个大数据资产层，以此为不同业务领域提供有效的支持。

Campos 等人的研究[20]强调了制造业中数据和大数据分析的特点，特别是在工业资产管理方面。对资产管理目的下分析系统架构的重要方面进行了详细讨论，并深入探讨了数据和大数据技术在相关领域的应用,详细介绍了机器学习和数据挖掘在数据资产管理中的应用以及数据分析可视化。

熊拥军等人的研究[21]提出基于数据中台理念规划图书馆数据资产管理架构，包括数据汇聚、数据开发、资产管理和数据服务四大模块，数据中台理念和图书馆数据资产管理深度融合，可增强图书馆数据的价值转化能力，提高图书馆数据创新应用能力，从而提升图书馆智慧化管理和服务水平。

1.2.4 研究现状总结

尽管知识图谱在多个领域如搜索引擎优化、推荐系统等已广泛应用，其在数据治理领域的利用却相对有限，目前的数据治理实践主要依赖于传统的数据集成技术和元数据管理方法，这些方法在处理异构数据源和保持数据一致性方面存在局限。因此，本文提出利用知识图谱的结构化特性，以及其在元数据融合和数据资产管理中的潜在优势，来提升数据治理的效率和质量。

1.3 研究意义

本研究意义在于探索和设计一种基于知识图谱的数据治理方法。这种方法不仅能够从多种数据集中抽取和融合数据元数据，生成数据资产目录，而且还能够在知识图谱的基础上实现一个具有通用意义的数据治理平台。我们期望通过这个平台，企业可以更加高效地管理其数据资产，提高数据的可访问性和质量，支持更加精准的决策制定，最终推动企业的持续创新和增长。

这种新的数据治理方法的开发和实现，将有助于解决当前企业在数据管理上面临的众多挑战，为企业提供一个更有效的工具来提升其数据驱动的决策制定能力。我们希望通过本研究，能够推动数据治理的理论和实践的发展，为企业的数据驱动的创新和增长提供支持。

1.4 研究内容

本课题主要研究内容如下：

（1）研究面向关系型数据库数据和半结构化的 json 等类型数据的数据汇聚，实现多种数据源的配置和抽取，设计和实现一个通用的数据抽取框架，能够适应不同的数据源和结构。探索如何自动识别和处理不同数据源中的数据异质性问题。

（2）研究元数据融合方法，设计元数据融合框架，支持多种融合算法，利用本体论和图算法优化融合过程，通过实验比较不同算法的效果，并优化选择。

（3）研究数据资产管理方式，实现可视化展现各类数据及数据关联的方式，开发数据资产管理系统，实现数据的分类、编目和质量控制，设计直观的数据可视化界面，方便用户理解和操作数据资产。

（4）结合典型场景，实现基于知识图谱的数据治理平台构建及应用,进行平台验证，设计知识图谱的架构，包括数据模型、存储、查询和更新机制。实施平台验证，通过实际案例分析平台的效能和可扩展性，以及在数据治理方面的实际效果

1.5 本文组织结构

本文将分为五个章节展开进行讨论，论文主要内容如下：

第一章，绪论。本文的绪论部分主要讨论了本课题的研究背景，知识图谱、元数据融合、数据资产管理的国内外研究现状，本课题的研究意义、研究内容和论文的行文框架。

第二章，需求分析及系统框架。这一章主要结合系统使用场景，分析了基于知识图谱的数据治理平台的业务需求，提出基于知识图谱的数据治理平台的功能性和非功能性需求，并对系统的整体框架做出了说明。

第三章，平台系统实现。这部分在需求分析的基础上，根据系统框架展开说明原型系统中使用的方法和各个层次的实现，主要包括元数据语义融合、数据抽取框架以及知识图谱构建三个部分，详细介绍了平台背后的技术细节。

第四章，原型展示及测试验证。本章展示了第三章实现的方法的结果以及原型的实现效果，之后验证了本课题系统设计的合理性和有效性，最终对展示结果进行讨论，并提出了今后的改进方向。

第五章，结论。本章总结了所做的工作和得到的成果，对基于知识图谱的数据治理平台提出了改进的方向，对未来的研究方向做出了展望。

1.6 本章小结

本文分析了数据治理的重要性和面临的挑战，特别强调了知识图谱在提高数据管理效率中的作用。通过探讨数据孤岛问题、数据质量不一致性和元数据不匹配等

问题，本章突出了知识图谱不仅在数据语义化表示上的优势，还在数据整合和质量控制上提供了新方法。此外，本章提出了一个基于知识图谱的数据治理平台的设计思想，旨在通过这一平台提升数据的可访问性和决策支持能力，最终推动企业持续创新和增长。这为后续章节关于具体的实现方法和系统框架提供了理论基础和研究方向。

第二章 需求分析及系统框架

本章将分析结合知识图谱的数据治理平台的需求并提出系统框架设计。分析数据资产管理的业务场景，并对该平台进行需求分析，设计出本软件系统的框架结构。

2.1 业务场景描述

以典型的线上购物平台为例，数据治理平台位于核心位置，起到协调和连接不同数据系统的作用，从数据采集到最终的数据应用，确保整个数据流的高效和安全管理。平台开始处理用户交互（如浏览、购买、评价）和后端操作（如库存管理、订单处理）产生的数据，通过初步清洗和分类保证数据质量，在这一环节中，数据治理平台负责管理数据的格式化、标准化及元数据管理，确保数据在存储时保持其完整性和可查询性，清洗和分类后的数据被传输到数据仓库或云存储系统中进行存储，此外数据治理平台还将与数据分析工具和计算平台相连接，利用这些工具对数据进行深入分析，最终数据治理平台确保正确的数据流向各种报告工具和仪表板，提供数据服务供 ERP、SCM、CRM 等企业信息系统使用。业务场景如图 2-1 所示。

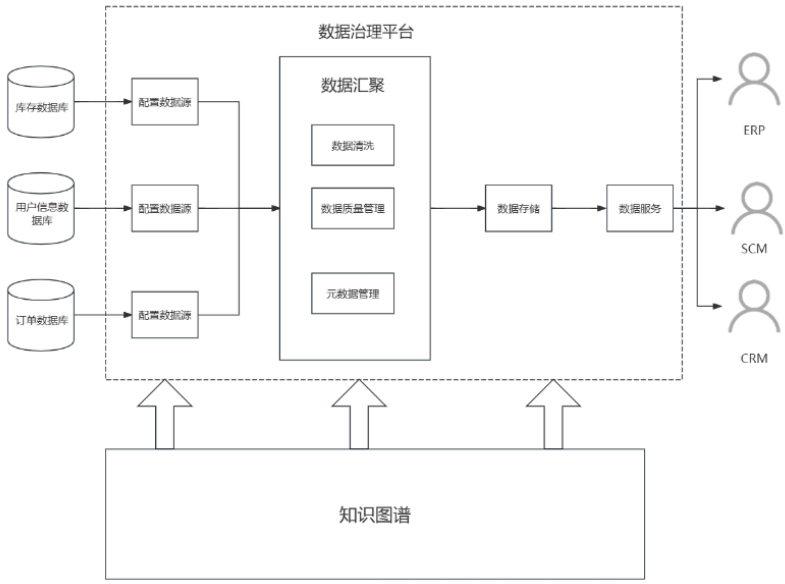

图 2-1 业务场景

知识图谱作为整个数据治理平台的支撑架构，知识图谱可以用于识别和关联来自不同数据源的相似实体，解决数据异构性问题，并构建统一视图提高数据整合的效率和准确性。在数据清洗过程中，利用知识图谱的关系和属性信息可以更精确地识别和纠正数据错误；在数据存储阶段 ，知识图谱不仅存储数据的结构和内容，还包括数据间的关系，为后续的数据查询和分析提供丰富的语义上下文，如解释消费者行为模式和产品推荐关联等。该数据治理平台不仅提高了数据的可用性和处理效率，还能通过智能算法推荐相关的数据资产和分析方法，为企业带来更全面的数据洞察和业务价值。

2.2 需求分析

根据基于知识图谱的数据治理平台的应用业务场景，本文分析了系统的功能性需求，系统的用例图如图 2-2 所示。

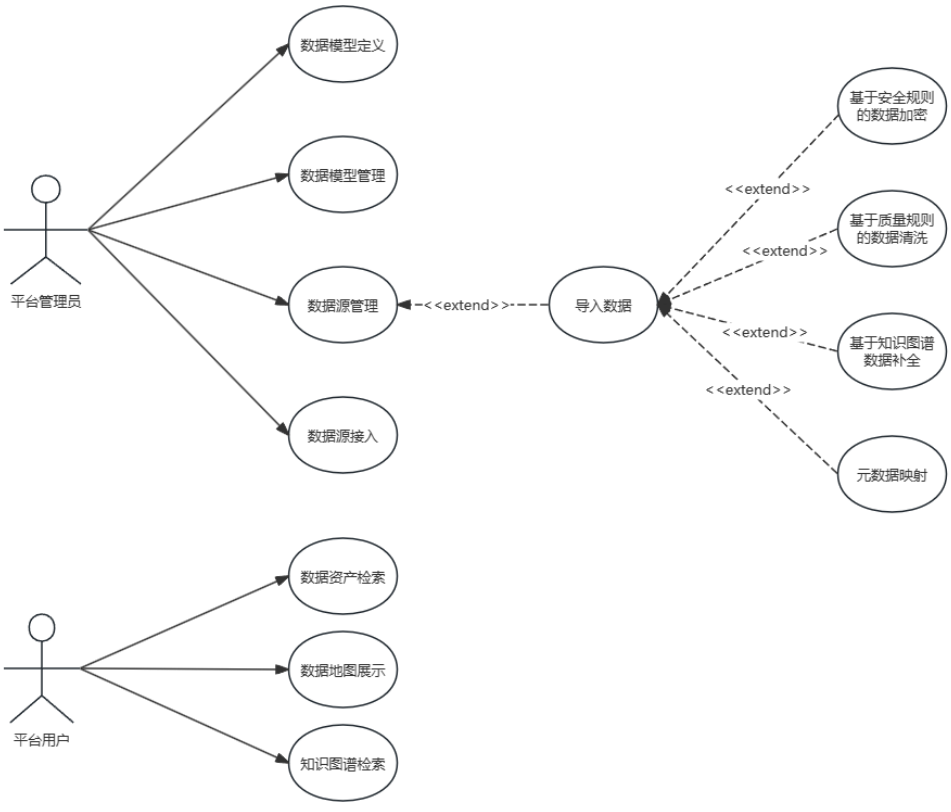

图 2-2 基于知识图谱的数据治理平台用例图

系统的功能性需求主要为数据模型定义、数据模型管理、数据源管理、数据源接入、数据资产检索、数据地图展示、知识图谱检索和知识图谱导出。用户在软件界面上能够根据业务类型或数据源结构来自定义数据模型，包括对标签描述、质量规则以及安全规则的定义，用户还能够管理已有的数据模型，用户还能够进行数据源管理，配置不同的数据源来导入数据等。对于用例图中的各个用例，在表 2-1 中进行详细说明。

表 2-1 基于知识图谱的数据治理平台用例说明

用例名称	用例说明
数据模型定义	平台管理员能够根据业务类型或数据源结构来自定义数据模型，包括对标签描述、质量规则以及安全规则的定义 。
数据模型管理	平台管理员能够查看已有的数据模型详细信息，包括配置时定义内容以及绑定的数据源信息，并且能够管理已有的数据模型，包括修改模型、删除模型、导出经过治理的数据。
数据源接入	平台管理员通过数据源模型的注册信息，将多模态的数据源接入平台中。对于已注册的数据源用户可以选择与数据模型绑定，系统会完成元数据提取、映射的工作，根据数据模型定义的规则对数据进行清洗、加密以及基于知识图谱的的数据补全工作
数据源管理	平台管理员可以查看已注册数据源详细信息，包括数据源URL、映射到数据模型信息、元数据信息等，还可以管理已注册数据源，修改信息、删除数据源等
数据资产检索	数据资产界面以树状节点展示平台中所有数据资产，平台用户能够对数据资产进行检索同时也能对选择数据源信息进行导出
数据地图展示	平台用户可以通过数据地图界面查看多模态数据模型之间的关系
知识图谱检索	平台用户可以通过知识图谱来进行详细数据的检索和
知识图谱导出	平台用户可以导出查找到的知识图谱文件

在构建基于知识图谱的数据治理平台时，关键的非功能性需求包括性能、可用性、可靠性和安全性。性能方面，平台应快速响应用户的查询和报告请求，能够处

理大规模数据集并支持高并发用户操作。在可用性方面，系统应确保至少 99.9%的在线时间，并提供直观易用的用户界面以便于所有用户类型的操作。可靠性需求涉及系统在故障发生时的快速恢复能力和定期的数据备份，以最小化数据丢失或损坏的风险。安全性需求强调数据在存储和传输过程中的保护，确保所有敏感数据都通过加密和安全措施得到妥善管理，防止未授权访问。这些非功能性需求对于确保平台的整体效能和用户体验至关重要。

2.3 系统架构设计

在系统的架构设计中使用三层架构（3-tier）设计模式，如图 2-3 所示。

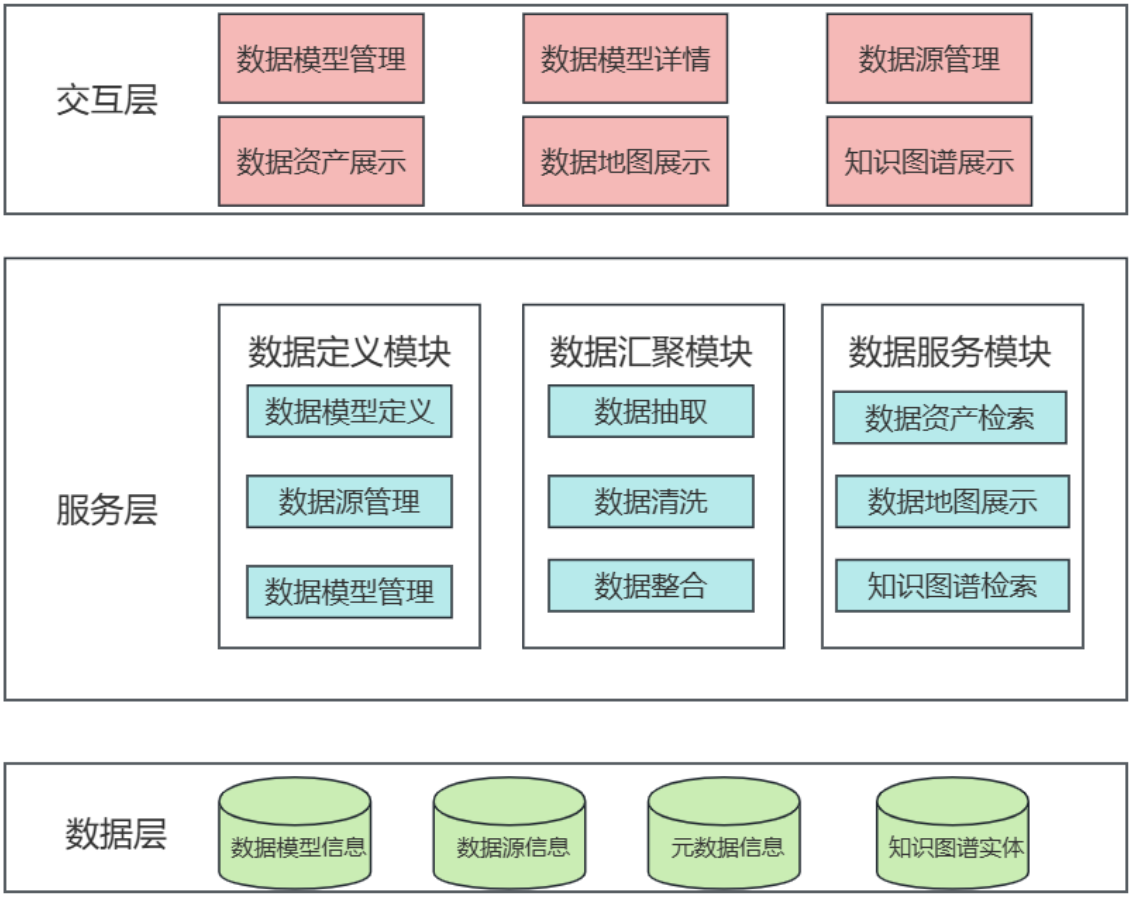

图 2-3 基于知识图谱的数据治理平台架构图

数据层包含数据源信息、数据模型信息、元数据信息、知识图谱等业务数据，包含的具体内容在表 2-2 中详细说明。

表 2-2 数据层内容

数据名称	数据内容
知识图谱实体	包括实体类型的定义和实例，如数据源、数据库表、字段、业务实体、概念等，每个实体可能关联多个属性和关系。
数据模型信息	包括数据模型的基本信息，如数据模型的名称、类型、所属业务域等，也包括数据模型所含字段信息、安全规则、质量规则、关联数据源信息等。
数据源信息	包括数据源的基本信息，如数据源的名称、类型、标签描述等。
元数据	关于数据本身的信息，如数据源描述、字段格式、数据创建和修改时间、数据作者、数据权限、标签等。

服务层主要用于完成业务逻辑，数据定义模块中数据模型定义负责根据业务需求来新增数据模型，数据模型管理负责对已有的数据模型进行管理，包括增删改查功能以及选择绑定数据源，数据源管理负责对已有的数据源进行管理；数据汇聚模块中数据抽取负责对接入的数据源进行元数据抽取、映射工作，数据清洗根据数据模型中定义的安全规则和质量规则对数据进行加密和清洗工作，数据整合负责构建数据地图和知识图谱以及基于知识图谱数据补全工作；数据服务模块包括数据资产检索导出，让用户可以通过数据资产目录使用标签和描述快速找到数据并按需处理导出，数据地图展示可以让用户清晰看见不同数据模型间的关联，知识图谱检索可以通过对属性值及对应值的搜索来展现对应节点及相关节点信息。

交互层与用户进行交互，包括数据模型管理、数据模型详情查看、数据源管理、数据资产展示、数据地图展示以及知识图谱展示等可视化交互界面，满足用户对界面展示的要求，并且保持系统的界面风格简约一致。

2.4 本章小结

本章主要分析了系统所涉及的现有业务场景和系统实现后对于业务场景的改进，然后从应用业务场景出发，提出了对系统的功能性和非功能性需求。在功能性需求方面，本章以功能列表的方式做出了说明；在非功能性需求方面，系统注重于平台的可用性、可靠性和安全性。基于系统应用场景和需求的分析，本章提出了系统的整体框架和技术栈。本章的需求分析和整体框架对系统的实现有重要的指导意义。

第三章 系统设计及实现

本章将会阐述系统的方案设计和实现方式。系统设计实现包括数据模型模块、数据汇聚模块数据服务模块。

3.1 数据治理流程

本节将介绍基于知识图谱的数据治理平台的总体流程。流程中的每一步将在本章后续小节中详细展开算法原理与实现方式。如图3-1所示，数据治理一共分为三个模块，分别为数据模型模块、数据汇聚模块和数据服务模块。数据治理首先从数据模型定义开始，用户根据业务需求自定义数据模型信息并存入数据库，接着用户进行数据源配置并将数据源与选择的数据模型进行绑定，选择完数据源信息和对应的数据模型后平台可以开始进行数据源接入，将数据源的数据导入平台中，在导入时数据汇聚模块会进行数据源元数据抽取映射、基于安全规则和质量规则的数据清洗以及进行基于知识图谱的数据补全同时构建知识图谱，经过数据汇聚模块的到清洗后的数据存入对应的数据库中，最后数据服务模块提供数据资产检索、数据地图检索以及知识图谱功能。

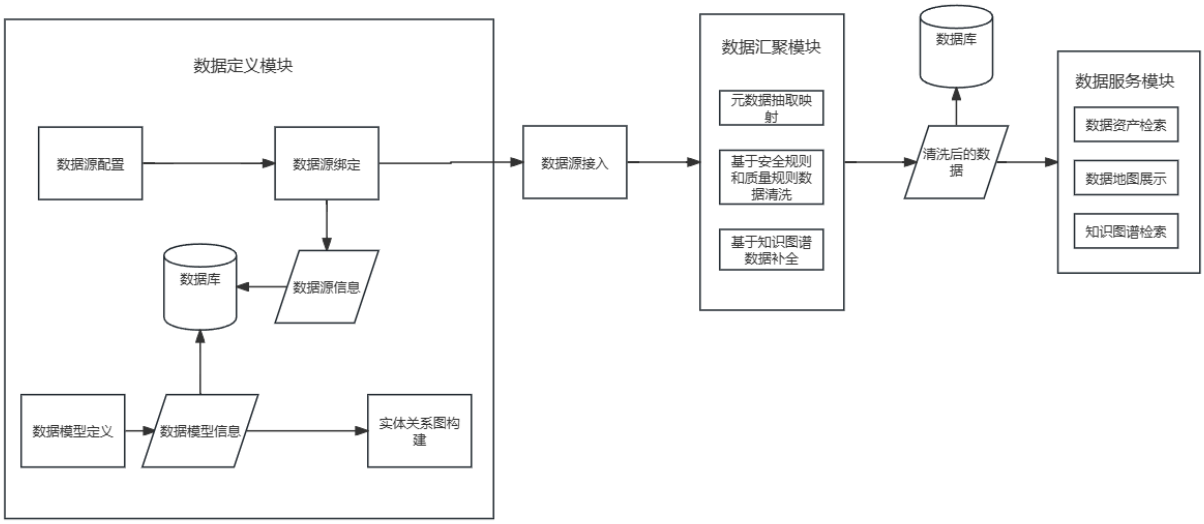

图 3-1 数据治理流程图

3.2 数据定义模块

数据定义模块为整个平台的基石，它定义了如何在系统中表示和存储数据，数据模型为数据存储建立一致的数据标准和定义，为后续数据汇聚和数据服务模块提供必要的结构。数据定义模块主要包括数据模型构建、数据源配置、实体关系图构建和知识图谱构建，本节将介绍这四部分的具体实现 。

3.2.1 数据模型构建

在系统设计中，数据模型扮演者至关重要的角色，它是一种抽象和表示数据及其关系的工具，用于对系统中的数据进行建模。通过数据模型能够从抽象的层次描述系统的静态特征、动态行为以及约束条件，这位系统的设计和开发提供了强大的支持。

数据模型的优势体现在多个方面。首先，通过定义数据模型能够提供标准化的数据结构，规范系统所使用的数据格式和组织方式，从而提高系统的可维护性和扩展性。其次，数据模型能够很好支持数据治理，通过定义数据标准、质量规则等，可以实现对数据的有效管理和控制，保障数据的一致性和可靠性。同时数据模型能够降低系统中不同模块之间的耦合度，提高系统的可扩展性和灵活性，最后数据模型可以提供强大的检索支持，便于快速定位和查询数据。

如图 3-2 展示数据模型设计的类图。

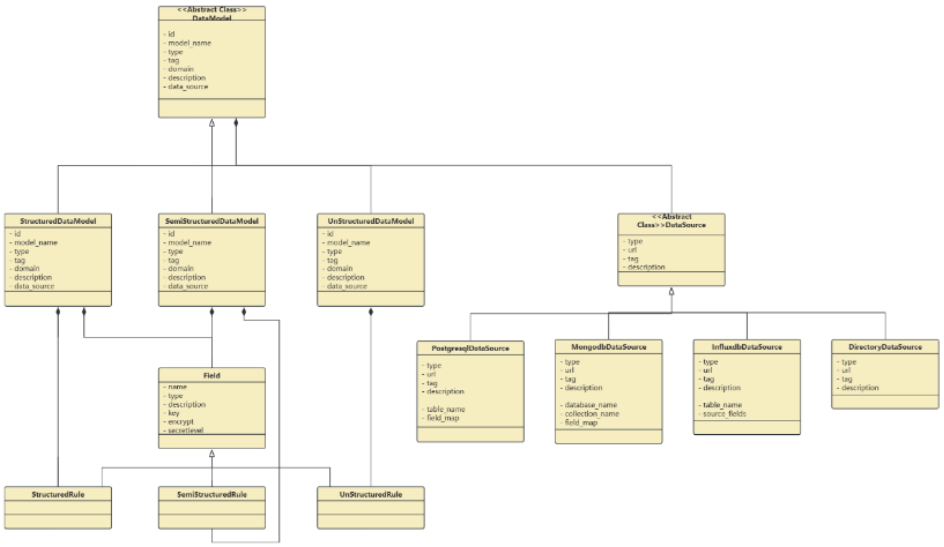

图 3-2 数据模型类图

在这个体系结构中，DataModel 作为一个抽象类，为不同类型的数据模型提供了基础属性，表 3-1 阐述了 DataModel 类设计，包括其拥有的属性和详细描述。它通过组合关系与 DataSource 类型连接，表示数据模型依赖数据源。StructureDataModel、SemiStructureDataModel 和 UnStructuredDataModel 分别继承自 DataModel 并且包含对应类型的 QualityRule，表示不同类型的数据模型具有不同的质量规则。DataSource 类及其子类提供数据存储的具体实现。

表 3-1 DataModel 类设计

属性名	属性说明
id	数据模型的唯一标识
modelname	数据模型的名称
type	数据模型的类型（结构化数据、半结构化数据、非结构化数据，时序数据）
tag	数据模型的标签
domain	数据模型所属的业务域
description	数据模型的描述
tag	数据模型的标签
datasource	数据模型使用的数据源

DataModel 作为父类，包括 id、modelname、type、tag、domain、description 以及 datasource 这些属性，为不同类型的数据模型提供基础属性。作为 DataModel 的子类，StructureDataModel 和 SemiStructureDataModel 除了继承基础属性，还包含了字段的集合 fields 和质量规则 rules，表 3-2 阐述了 Field 类设计，包括其拥有的属性和详细描述，表 3-3 阐述了 QualityRule 类中预设的质量规则。

表 3-2 Field 类设计

属性名	属性说明
name	字段名称
type	字段类型
description	字段描述
key	是否为 key 值
encrypt	是否加密
secretlevel	加密级别

Field 类描述了数据模型中的字段信息，其中 key 属性用于标识该字段是否作为关键字段，encrypt 用于标识该字段是否需要加密存储来确保数据的安全性和隐私保护，加密算法实现伪代码如算法 3-1 所示。secretlevel 用于标识该字段的加密级别，

根据不同的安全要求采取不同的访问控制措施，只有用户权限高于该字段的加密级别才允许访问。这些属性和特征为后续的数据治理打下坚实基础。

Algorithm 3-1 AES Encryption Algorithm

Input: A plain text string *input*

Output: Encrypted text string in Base64 format

```
Initialize static SecretKey secretKey;
Create a KeyGenerator instance for AES;
keyGenerator ← getInstance("AES");
Initialize keyGenerator with 128-bit key size;
keyGenerator.init(128);
Generate the secretKey using keyGenerator.generateKey();
Get a Cipher instance for AES;
cipher ← getInstance("AES");
Initialize the cipher in ENCRYPT_MODE with secretKey;
init(ENCRYPT_MODE, secretKey);
Encrypt the input string into bytes;
encryptedBytes ← doFinal(getBytes(input, StandardCharsets.UTF_8));
Encode the encrypted bytes into Base64 string;
return encodeToString(encryptedBytes);
```

本文使用 AES 加密算法对给定的明文字符串进行加密。首先，它通过密钥生成器生成一个随机的 128 位密钥，然后使用该密钥初始化 AES 加密器，将输入的字符串转换为字节数组，并利用初始化的加密器对其进行加密，最后将加密后的字节数组编码为 Base64 格式的字符串，并将其作为加密结果返回存储，这种加密方式保证了需要加密数据的安全性和隐私保护。

表 3-3 质量规则设计

质量规则	规则说明
非空规则	字段不允许为空值
比较规则	两个字段之间比较关系
范围限制规则	限制字段范围

质量规则规定了相应字段的限制，在数据源导入的过程中系统会根据预设的质量规则对相应的字段进行限制和清洗，保证数据的质量和完整性。这些质量规则包括非空规则、比较规则、范围限制规则等，通过自动化的数据清洗过程，系统能够有效地识别和修复数据中的问题，提高了数据的可信度和可用性。

3.2.2 数据源配置

为实现面向结构化数据、半结构化数据和非结构化数据等数据的全面采集与管理，实现多种数据源的配置和抽取。本节设计了一个基于 3.2.1 所述数据模型的多数据源配置和抽取方案，分别利用 PostgreSQL、MongoDB、InfluxDB 存储提取的结构化数据、半结构化数据以及时序数据，对于文件类型的数据将记录数据源地址到 MongoDB。在提取数据时会映射到对应的的数据模型，并将数据存储到数据模型对应的数据库，具体数据源类设计如图 3-3 所示。

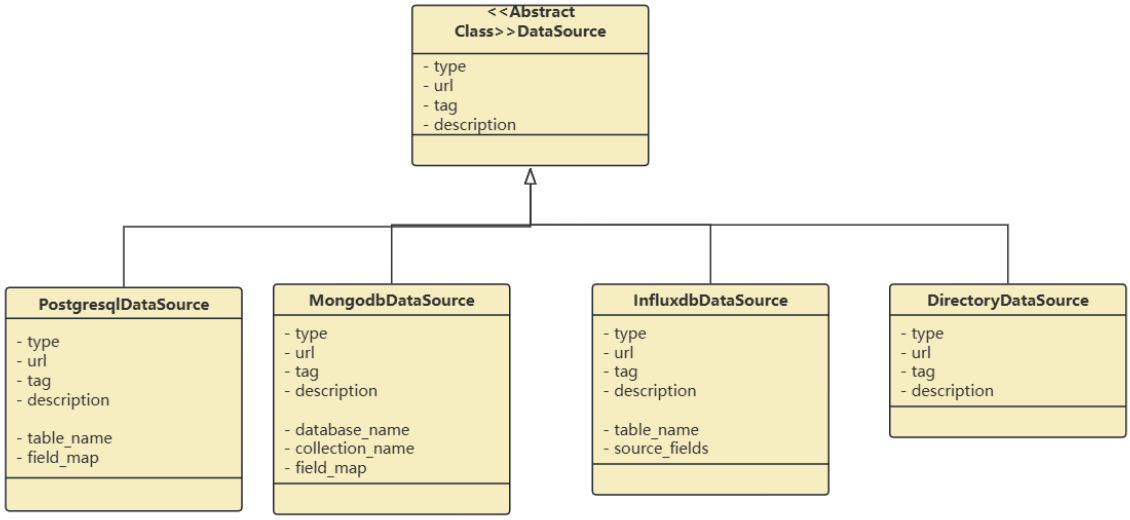

图 3-3 数据源类设计

DataSource 作为一个抽象类，为不同类型的数据源提供基础属性，表 3-4 具体阐述了 DataSource 类的设计，包括各个属性的具体信息。

表 3-4 DataSource 类设计

属性名	属性说明
type	数据源类型
url	数据源的 URL
tag	数据源的标签
description	数据源的描述

由于不同类型的数据库结构各异，作为 DataSource 的子类，各自具有除了基础属性之外的独特属性，以适应其特定的数据存储需求。

PostgreSQLDataSource 继承了 DataSource 的基础属性，包括类型、URL、标签和描述。它的独特属性包括表名称（table_name），用于指定存储数据的表名，以及字

段映射表（field_map），用于定义数据库字段与数据模型字段的映射关系。选择 PostgreSQL 作为结构化数据库的存储方案，主要因为其强大的 SQL 支持和复杂查询处理能力。PostgreSQL 在处理事务时具备高可靠性和数据完整性，适合需要严格数据结构和关系的场景。

MongodbDataSource 继承了 DataSource 的基础属性，包括类型、URL、标签和描述。其独特属性包括数据库名称（database_name），用于指定存储数据的数据库名称，以及集合名称（collection_name），定义数据存储的集合名称，还包括字段映射表（field_map），用于定义数据库字段与数据模型字段的映射关系。选择 MongoDB 作为半结构化数据的存储方案，主要是因为其支持灵活的数据模式和动态架构，可以高效处理多变和复杂的数据结构。

InfluxdbDataSource 继承了 DataSource 的基础属性，包括类型、URL、标签和描述。其独特属性包括表名称（table_name），用于定义数据存储的表名，以及源字段（source_fields），指定时序数据中的源字段信息。选择 InfluxDB 作为时序数据的存储方案，主要因为其在时间序列数据的写入和查询方面具有优化优势。

DirectoryDataSource 继承了 DataSource 的基础属性，包括类型、URL、标签和描述。其独特属性为数据源地址（source_path），用于记录文件数据源的存储地址。选择将文件类型数据的源地址记录在 MongoDB 中，是因为 MongoDB 的灵活性和可扩展性能够高效管理和追踪文件数据，适用于文档管理、媒体存储和归档系统等应用场景，确保文件数据的可访问性和管理的高效性。

3.2.3 实体关系图构建

实体关系图是一种图形化的数据模型，常用于初步设计和展示知识图谱中的实体、实体之间的关系以及实体的属性。这种图形化表示方法不仅帮助设计者清晰地理解和构建数据结构，而且促进了知识图谱中复杂关系的逻辑表达。通过精确定义实体间的多样化关系，实体关系图为知识图谱的逻辑设计和实现提供了坚实的基础，确保知识图谱的数据存储和查询操作既高效又有序。

本节分析面向数据治理的知识图谱结构，并结合实际情况中大规模数据存储情况，构建实体关系图如图 3-4 所示。

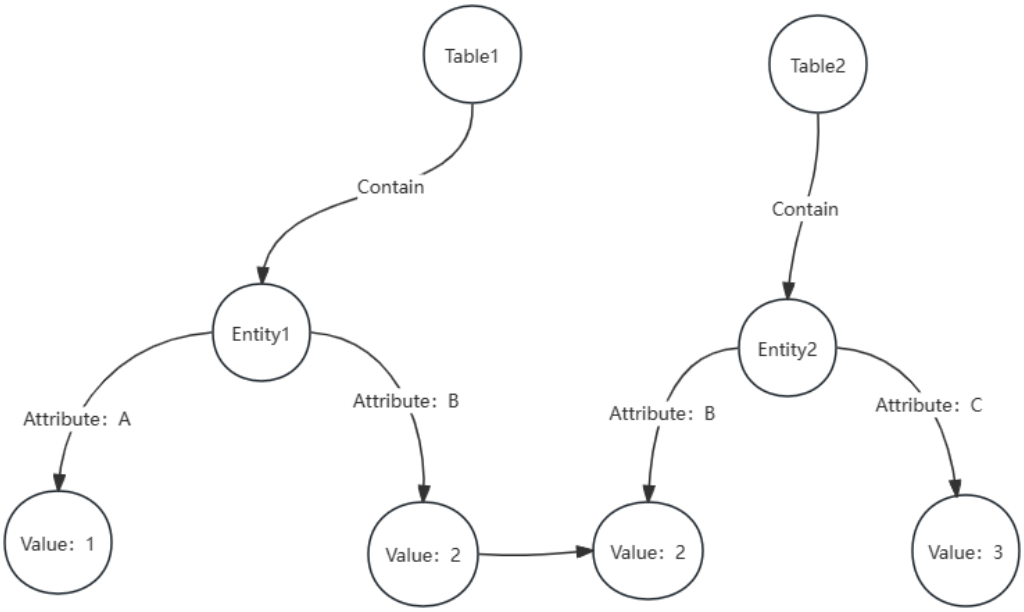

图 3-4 实体关系图

根据 3.2.1 对数据模型的构建，本文提出的实体关系涵盖了表（table）、实体（entity）、属性（attribute）、值（value）关键要素，表 3-5 详细介绍了各节点描述、属性以及关系。

表 3-5 实体关系图节点结构

节点名	描述	属性	关系
Table	代表数据模型，作为知识图谱核心节点	name：表名称，对应数据模型的名称 key_attribute:关键属性列表，用于标识表中的主键字段	包含：连接到多个 Entity 节点
Entity	代表数据记录，每条记录对应一个实体节点	id：实体唯一标识符 data：实体的所有属性及其值，以键值对形式存储	属于：连接到 Table 节点，表示实体属于哪个表 具有属性：连接到多个 Attribute 节点，表示实体的属性值
Attribute	代表属性数据，每个属性节点表示一个具体的属性值	name：属性名称 value：属性值 rule：数据模型中定义的字段质量规则和安全规则	属于：连接到 Entity 节点，表示该属性属于哪个实体 共享属性：连接到多个 Entity 节点，表示多个实体共享该属性

该实体关系图明确了数据模型、实体和属性之间的关系及其在图谱中的表现形式，通过这种结构，可以有效组织和管理从不同数据导入的数据，形成一个可扩展、高效的知识图谱，为数据分析和业务决策提供坚实的基础。

3.2.4 知识图谱构建

根据 3.2.3 实体关系图的构建，知识图谱构建的知识源来自数据模型对应数据库存储的数据。

构建知识图谱的过程包括实体识别、属性提取和共享属性识别三个主要步骤。首先，在实体识别阶段，系统将遍历每个数据模型以提取相应的数据表。根据数据模型的名称，为每个数据表创建一个名为“Table”的节点。然后，系统将遍历这些数据表的每一行，为每行数据创建一个“Entity”节点，并将其与相应的“Table”节点相连。接下来，在属性提取阶段，系统进一步遍历每一列数据，为每列创建一个“Attribute”节点，并将其连接到对应的“Entity”节点。最后，在共享属性识别阶段，系统会检查并识别是否有不同“Entity”节点具有相同的“Attribute”节点。如果发现有相同的属性，则将这些实体通过共享的“Attribute”节点连接起来，从而实现属性的共享。

3.3 数据汇聚模块

数据汇聚模块是整个系统的核心功能，为数据服务模块奠定了数据基础，数据汇聚模块通过 ETL 工具和 API 采集等方式将分散的数据进行采集处理，获取数据治理的原始数据。数据汇聚模块主要包括元数据抽取、数据清洗和数据补全工作，其中元数据抽取映射中包含元数据匹配融合算法，数据清洗和数据补全工作基于定义的质量规则、安全规则完成。因此本文将阐述元数据抽取算法实现、元数据匹配融合算法的实现、基于知识图谱的数据清洗实现方法。

3.3.1 元数据抽取

由于不同类型数据源的存储结构存在差异，元数据抽取方法也需要相应地进行调整。结构化数据通常存储在关系数据库中，半结构化数据如 XML、JSON 和 CSV 具有一定的可识别格式但不完全符合传统的数据库模型，而非结构化数据如文本文

件、图片、视频等则通常缺乏明确的结构，因此元数据抽取主要聚焦于识别文件的基本属性。

对于结构化数据，由于其一般存储于关系型数据库中，元数据抽取通常可以通过执行简单的 SQL 查询来实现。这些查询能够有效地获取数据库的结构信息，包括表名、列名和数据类型等。

对于常见的半结构化数据类型，本文采用如下处理方法：使用 XML DOM 解析器来处理 XML 数据并提取出 XML 的标签结构，利用 JSON 解析库来解析 JSON 数据并提取出 JSON 键，以及采用 Apache Commons CSV 库来读取 CSV 文件并提取出 CSV 的列头。

对于非结构化数据，如文本文件和图片，本文采用的策略是提取文件的基本属性，例如文件名、创建日期、修改日期和文件大小，这些属性作为文件的元数据。

最后，对于抽取出的元数据，需要进行进一步的处理，将其与数据源所绑定的数据模型元数据进行映射，以确保数据的一致性和完整性。这一过程不仅增强了数据的可用性，也为后续的数据管理和分析提供了坚实的基础。

3.3.2 元数据匹配融合算法

元数据匹配融合在数据治理中主要用于增强数据的一致性、可访问性和整体价值，在本系统中主要应用于数据源元数据提取映射工作和知识图谱模糊查询。

传统基于字符串匹配的方法进行元数据匹配容易受到拼写差异、同义词和多义词的影响，基于结构化信息的方法在数据结构复杂多样时效果有限，因此需要一种能够识别词语含义的方法来支持元数据匹配，语义表征可以支持这种需求。

语义表征是指将语言中的意义（语义）以一种可理解和处理的形式进行标识的过程，在自然语言处理（NLP）和人工智能（AI）领域，语义表征是实现语言理解和生成的基础，通过将词、短语、句子或更大文本的一一映射到一种结构化的形式，通常为向量或符号结构，使计算机能够进行处理和理解。其目的使捕捉语义信息，包括词义、上下文、句法结构等。通过语义表征技术可以识别和表示数据中的深层次语义关系，超越简单的字符串匹配。利用语义表征技术，可以更准确地匹配和融合语义相似的元数据，提高整体算法的效果。

在本系统实际应用时元数据一般为单独的一个词语，不会出现较多上下文的情况，所以选择语义表征方法中的词级别表示作为元数据匹配融合算法中的基础。

词级别语义表征方法一般使用词嵌入（Word Embedding）方法，词嵌入是将词汇映射到低维向量空间的技术，旨在捕捉词汇之间的语义关系，它的目标是使相像词语在向量空间中的距离较近，反映它们在语义上的相似性。其中最常用的为Word2Vec，GloVe 和 FastText 技术，他们有各自的特点和使用场景：

（1） Word2Vec

模型类型包括 CBOW 和 Skip-Gram 两种模型，CBOW 通过上下文预测目标此，Skip—Gram 通过目标此预测上下文，它的嵌入方式为将词汇映射到低维向量空间，使得此役相似的词在向量空间中距离较近，该方法虽然能够捕捉词汇之间的语义关系，生成的向量可以有效表示词汇的语义信息，但是对训练语料中未出现的词汇无法生成有效表示。

（2） GloVe

模型类型为基于词共现矩阵的词嵌入模型，它利用全局词汇统计信息，将词汇映射到低维向量空间，嵌入方式为通过构建共现矩阵和利用 SVD 等技术生成词向量。它能够综合考虑整个语料库的词共现信息，生成的词向量具有全局一致性，但是对训练语料中出现频率较低的词汇表示能力有限。

（3） FastText

模型类型为基于字符 n-gram 的词嵌入模型，通过将词表示为字符 n-gram 的组合，生成词向量，它的嵌入方式不仅会词汇生成向量，还为词内部的 n-gram 生成向量，从而结合词汇的内部信息。通过字符 n-gram，能够为未在训练语料中出现的词汇生成有效表示，并且能够捕捉词汇的内部信息，考虑词的内部结构信息，如前缀、后缀等，提高了表示的细粒度，但是模型复杂性高，实现和训练过程更耗时复杂。

根据以上三种算法构成的词向量再使用余弦相似度计算两个词语的相似度，计算公式如公式 3-1 所示：

$$similarity = \cos(\theta) = \frac{A*B}{||A||*||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}} \tag{3-1}$$

其中 A 和 B 分别代表两个词语对应的词向量，通过计算两个向量的余弦值来表示两个向量的相似度。

基于上述语义表征方法将元数据映射为词向量后可以通过式3-1计算两个元数据的相似度，当相似度高于某个阈值时可以将两个元数据在语义上有较高的近似性，这两个元数据可以进行匹配融合。算法实现伪代码如3-2所示。

Algorithm 3-2 Metadata Matching adn Fusion Algorithm

Input: Word1, Word2
Output: Similarity
Load FastText model *and* Word2Vec model
Set weights: $w_{\text{FastText}} = 0.7$, $w_{\text{Word2Vec}} = 0.3$
function GET_WEIGHTED_AVERAGE_VECTOR($word, FastText, Word2Vec, weights$)
    if word **in** FastText_model **then**
        Add $weights[0]$ × FastText_vector to list
    if word **in** Word2Vec_model **then**
        Add $weights[1]$ × Word2Vec_vector to list
    return sum of vector list
end function
vector1 ← `get_weighted_average_vector`(word1,FastText,Word2Vec,weights)
vector2 ← `get_weighted_average_vector`(word2,FastText,Word2Vec,weights)
similarity ← `cosine_similarity`(vector1, vector2)
return similarity

3.3.3 基于知识图谱的数据清洗

数据清洗环节主要包括基于质量规则的数据清洗、基于安全规则的数据清洗以及数据补全处理。

在构建知识图谱时系统会根据数据模型中定义的质量规则对不符合质量规则数据及逆行删除处理，对符合安全规则的数据进行加密存储处理。

根据3.2所构建的数据模型，对于不同数据源导入相同数据模型的情况，在实际应用中可能出现映射到统一数据模型的数据可能会出现部分记录不完整的情况。例如，一部分数据可能包含部分字段，另一部分数据包含一部分字段。为了增强数据的完整性，可以通过数据模型中的相同的 key 值，在构建知识图谱时将这些数据合并存储到同一个节点中，并在需要时可以导出完整的数据。

根据 3.2.3 的描述，每个 Entity 节点拥有 key_attribute 作为表中的主键标识字段并且在表中具有唯一性，存储在知识图谱时每个 Entity 节点都有 id 作为在知识图谱中的唯一标识符。

该算法首先识别数据模型中哪些数据是关键属性（key）值，在构建 Entity 节点时，算法会识别出哪些拥有 key_attribute 但 id 不同的 Entity 节点，然后利用 Neo4j 内部的 APOC 库中的 mergeNodes 函数，将这些重复的 Entity 节点合并为一个,对于两个实体中可能出现的相同属性的值不相同的情况，mergeNodes 会将两个值都保留。这个算法不仅保证了 Entity 节点的 key_attribute 在知识图谱中的唯一性，同时也维护了数据的完整性。

3.4 数据服务模块

在 3.2 和 3.3 中，分别已经完成了数据定义模块以及数据汇聚模块，在本节中，将基于 3.2 和 3.3 完成的成果实现数据服务模块，主要实现的功能有：基于知识图谱的数据检索、数据地图与数据资产构建。

3.4.1 基于知识图谱的检索

在实际应用中，异构数据通常在数据库中以不同的方式存储，例如，关系型数据库使用表格形式，而非关系型数据库可能采用文档、键值对或图结构。这种多样化的数据结构使得查询变得复杂和繁琐，开发者需要掌握多种查询语言和技术，才能有效地访问和处理不同的数据源。因此，缺乏统一查询接口会大大增加数据处理的复杂性和时间成本。

为了解决这一问题，本文通过 3.2 节中定义统一的数据模型，将不同数据源的数据映射到该模型中，形成实体-属性-值的结构，进而构建知识图谱。知识图谱使数据查询过程得以简化和抽象化，能够统一查询来自不同数据源的数据。知识图谱不仅包含实体及其关系的语义信息，还能够理解用户查询的实际意图，并展示实体之间的多层次关联，帮助用户发现潜在的相关信息。

通过检索特定属性及其对应的值，知识图谱能够展示相关的节点及其连接的节点，从而实现高效的数据查询和展示。这种方式不仅减少了在多种数据库系统中使

用不同查询语言的需求，还简化了数据访问过程，加快了决策制定和业务洞察的获取速度。

在实际检索过程中，用户可能不会精确输入所需的属性信息。为了解决这个问题，本文首先通过 3.3 节中实现的词嵌入算法对输入的属性值进行向量化，接着将得到的词向量与 Milvus 数据库中存储的向量进行比较，找出相似度高于 0.9 的字段名，这些字段被认为是用户可能想要查询的字段。然后，系统会自动生成相应的 Cypher 查询语句，向 Neo4j 数据库发起查询，并展现查询结果。这一流程不仅提高了查询的准确性，也优化了用户的检索体验。

3.4.2 数据地图与数据资产构建

为清晰简洁明了展现平台拥有的多种数据模型间的关系以及数据模型和数据源之间的关系，本节完成了构建数据地图与数据资产展示。

数据地图通过 3.2 构建的数据模型，遍历每个模型和模型的字段，对每个模型和字段创建一个节点，将模型节点和字段节点通过链接关联起来，然后使用 Echart 插件进行数据地图绘制，通过该数据地图用户能了解给数据模型之间的关系。

数据资产通过 3.2 构建的数据模型和导入的数据源，遍历每个模型和模型绑定的数据源，利用 Echart 插件构建属性图来展示业务域、数据模型与数据源之间的层次关系。

3.4 本章小结

本章深入探讨了系统的设计与实施，涉及数据定义、数据汇聚和数据服务三大模块。数据定义模块作为平台的基础支撑，详细介绍了数据模型的构建，包括结构化、半结构化和非结构化数据模型的定义和实现、实体关系图构建以及知识图谱构建方法；数据汇聚模块关注于如何从分散的数据源中抽取、清洗和整合数据本章详细讨论了元数据抽取的策略，包括结构化、半结构化和非结构化数据的处理方法，以及如何利用知识图谱技术进行数据清洗和补全。数据服务模块展示了如何基于前两模块的进行高效的数据服务，包括基于知识图谱的数据检索、构建数据地图和数据资产展示。

第四章 原型展示及测试验证

本章利用前文所述的系统框架与核心方法，建立了一个原型系统，为基于知识图谱的数据治理平台提供了可视化与交互界面。

4.1 系统实现及部署

系统实现架构图如图 4-1 所示，本系统由可视化交互界面、后端与数据库构成。

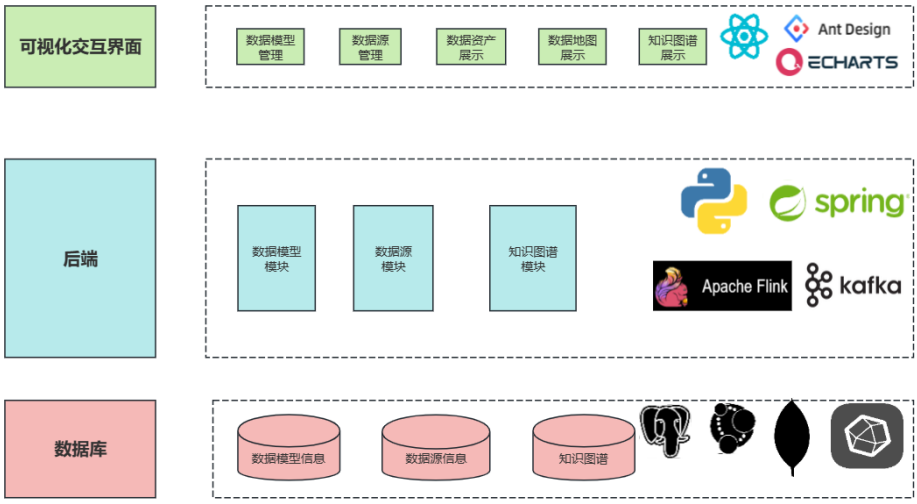

图 4-1 系统实现架构图

可视化交互界面使用 React 框架进行开发，并且借助 Ant Design React 和 ECHARTS 组件库，完成了数据、组件的可视化。可视化交互界面主要包括数据模型管理、数据源管理、数据资产展示、数据地图展示和知识图谱展示等界面。

后端主要分为三个模块，数据模型模块、数据源模块以及知识图谱模块。主要使用 SpringBoot 框架进行开发。数据源模块实现添加数据模型、删除数据模型、查看数据源详情以及数据源绑定功能；数据源模块实现添加数据源、删除数据源、数据质量管理，还使用 Flink 和 Kafka 实现实时数据采集；知识图谱模块主要通过 Python 实现自动化导入数据构建知识图谱及查询功能。

数据库主要使用关系型数据库 PostgreSQL 存储数据源中的结构化数据，非关系型数据库 MongoDB 存储数据源中半结构化数据、数据模型具体信息以及数据源的具

体信息，时序数据库 InfluxDB 存储数据源中的时序化数据，图数据 Neo4j 存储、管理知识图谱，Milvus 存储元数据映射的向量数据。

在后续系统界面展示中将以 Olist 商店下订单的巴西电子商务公共数据集为用例，对原型系统进行测试，保证其可靠性和可用性。

4.2 系统界面展示

本节将根据实际使用场景展示软件操作界面，主要包括数据模型管理、数据源管理、知识图谱查询、数据资产展示、数据地图展示界面。

4.2.1 数据模型管理界面

如图 4-2 所示，该图展示了数据模型管理界面，可以看到当前存在数据模型，还包括添加数据模型、查看数据模型详情、绑定数据源以及删除数据模型功能。

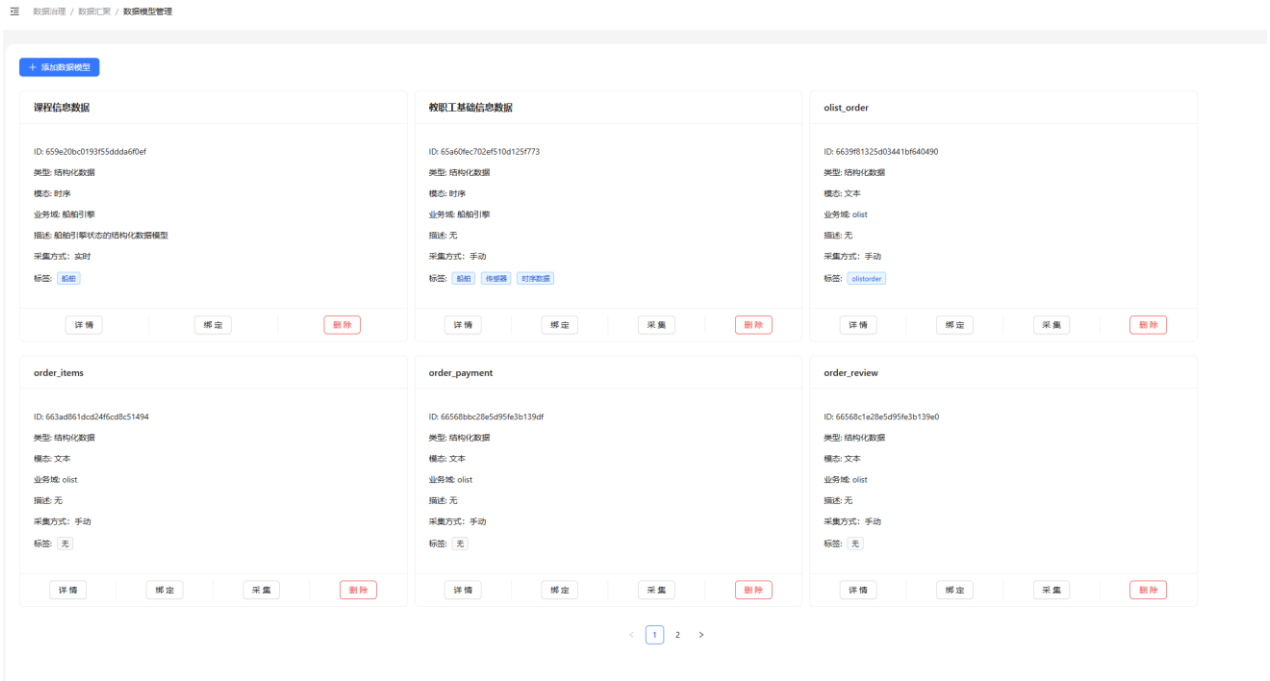

图 4-2 数据模型管理界面

管理员点击“添加数据模型”可以通过添加数据模型功能添加新的数据模型，如图 4-3 所示。

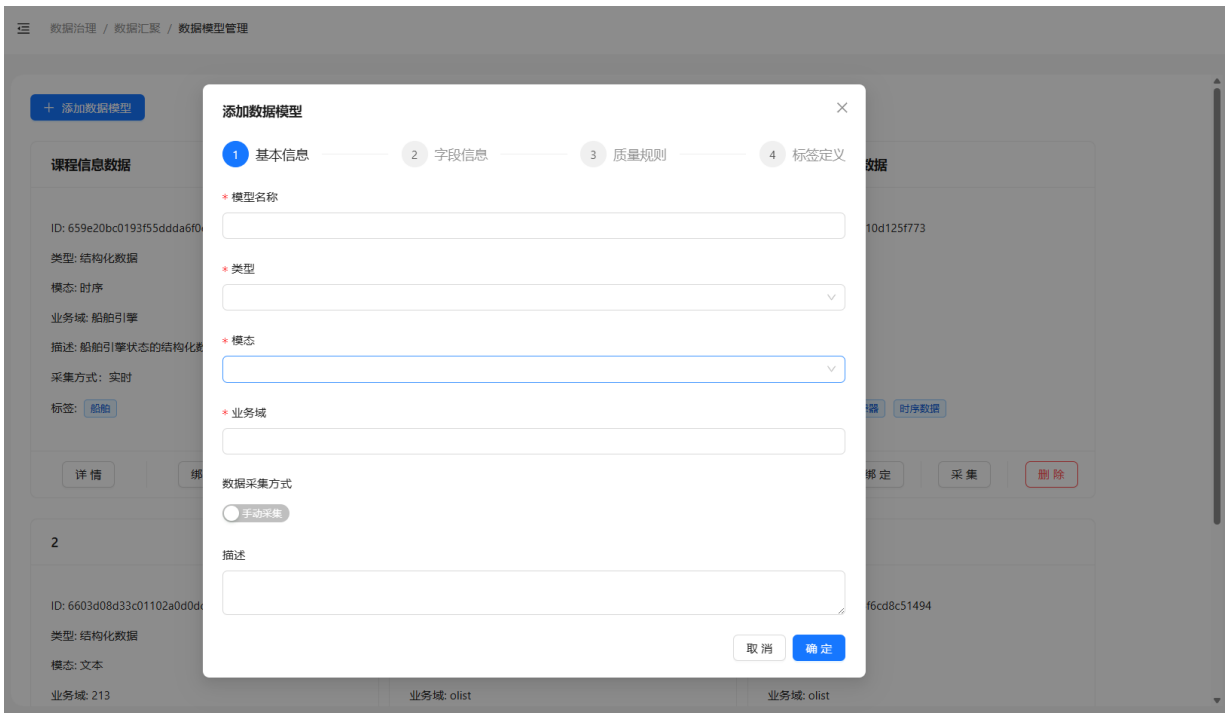

图 4-3 添加数据模型页面

定义信息主要包括添加基本信息和字段信息、选择质量规则、添加对该数据模型的标签定义。

基本信息包括数据模型的名称数据模型的类型、标签、所属的业务域、基本描述以及标签，除了填写这些信息还有一个按钮选择该数据模型是否为实时采集，若为实时采集，系统将通过 Flink 和 Kafka 监控数据库的变更日志，实时捕获数据变更，确保了数据的实时性和准确性；

字段信息包括字段名称、字段类型、字段描述、是否为 key 值、是否加密以及加密级别，当添加字段时，系统会通过第三章中元数据匹配融合部分实现的语义表征算法将该字段名进行词嵌入算法并将生成的词向量存入 Milvus 数据库中。

质量规则包括 3.2 中定义的非空规则、比较规则、范围限制规则。用户可以选择对哪些字段进行质量控制，后续在导入该字段的数据时系统会自动通过这些质量规则对数据进行质量控制删除不合规数据。

如图 4-4 所示，数据详情页面显示在添加数据模型时添加的信息，包括模型基础信息、包含的字段信息以及模型规则。除此之外，因为数据源会与数据模型进行绑定，所以该页面还显示了绑定数据源的基本信息和查看数据源字段与数据模型字段的映射信息，页面最下方显示导入进来数据源映射到数据模型字段后存储的数据信息，点击“全量导出”可以导出经过数据清洗和知识图谱补全的数据。

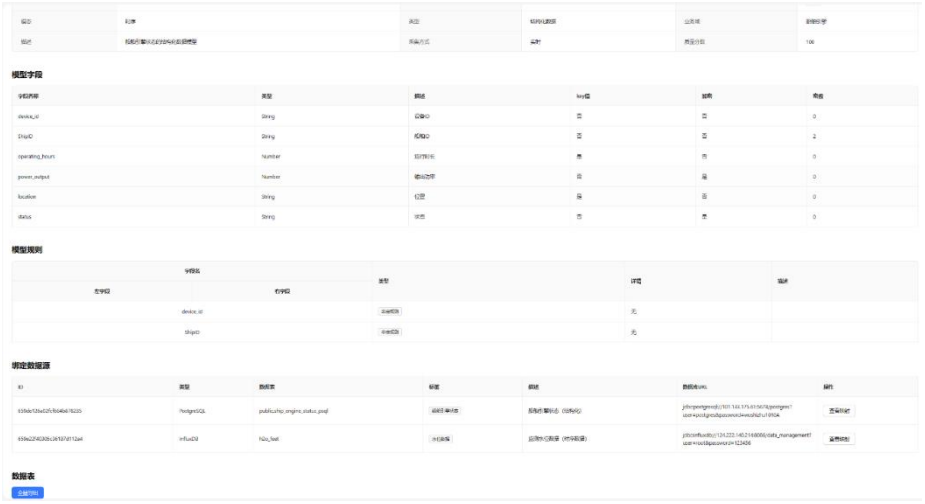

图 4-4 数据模型详情页面

管理员点击“绑定”按钮可以进入绑定数据源页面，如图 4-5 所示为数据源与数据模型绑定的界面。

图 4-5 数据源绑定数据模型界面

对于一个数据模型，可以绑定多个数据源，在选择数据模型绑定数据源时会进行字段映射工作，通过第 3 章实现的元数据匹配融合算法，在选择字段映射时系统会对数据源中存在的元数据与数据模型所定义的字段进行比较，会推荐选择最相近的字段自动填充，提高数据源绑定效率，同时管理员也可以修改要进行映射的字段。

字段映射选择完毕后会进行数据源数据的抽取工作，根据数据源所映射数据模型的安全规则，系统会对具备密级的数据进行加密，对为 key 值的字段进行标记，根据数据模型的质量规则，系统会对数据进行清洗以及判断质量分数添加至该数据模型的信息中，并且系统会通过第 3 章实现的知识图谱构建功能进行知识图谱构建、数据补全以及存储工作。

4.2.3 数据源管理界面

数据源管理界面是数据汇聚的一个重要界面，如图 4-6 所示，改图展示了数据源管理界面，可以看到当前已经注册的的数据源的基本信息，还包括查看数据源中元数据详情以及删除数据源功能。

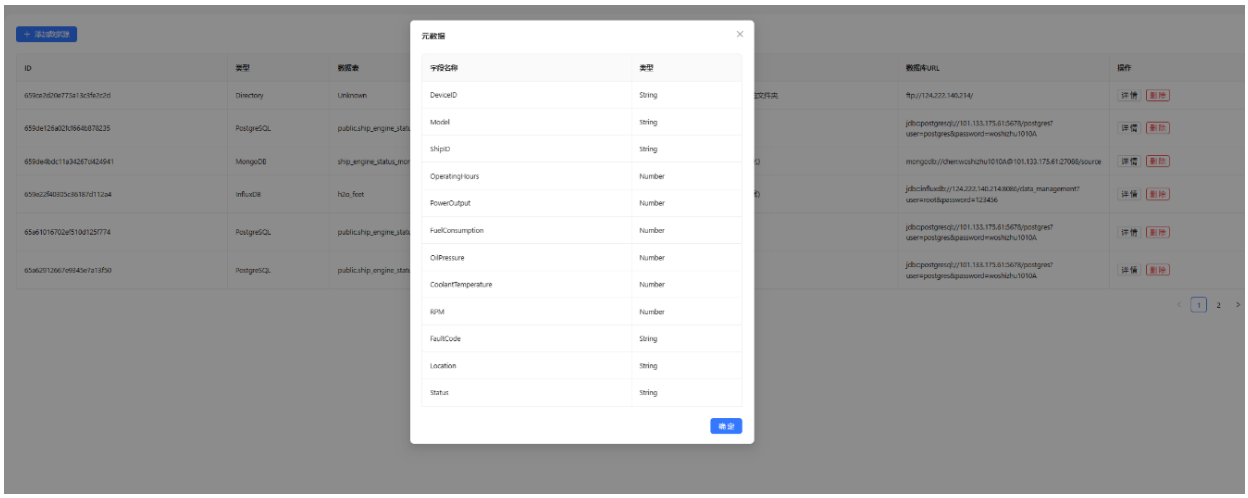

图 4-6 数据源管理界面

管理员通过点击“添加数据源”注册新的数据源信息，如图 4-7 所示为数据源添加界面。

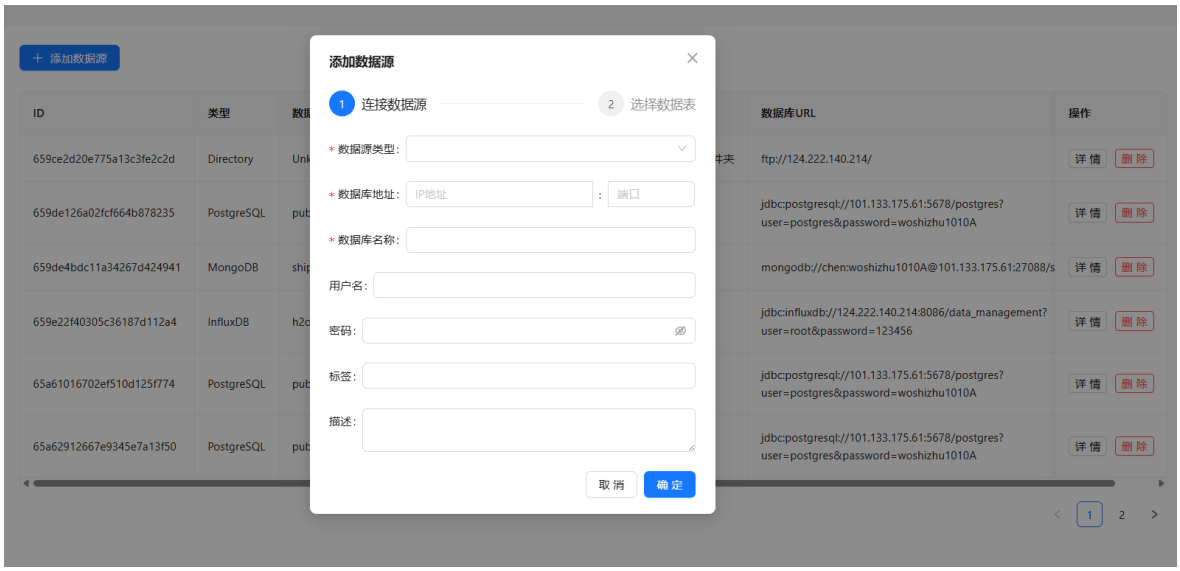

图 4-7 添加数据源界面

包括数据源类型、地址、数据库名称以及数据源用户名密码等，在连接上数据源后选择数据表，将多模态的数据源接入平台中，对已注册的数据源，平台会完成元数据提取的工作，后续可以在进行与数据模型绑定的工作。

4.2.4 知识图谱界面

根据 Olist 巴西公共电子商务数据集集构建知识图谱，共计 30306 个节点，56605 条关系，如图 4-8 所示。

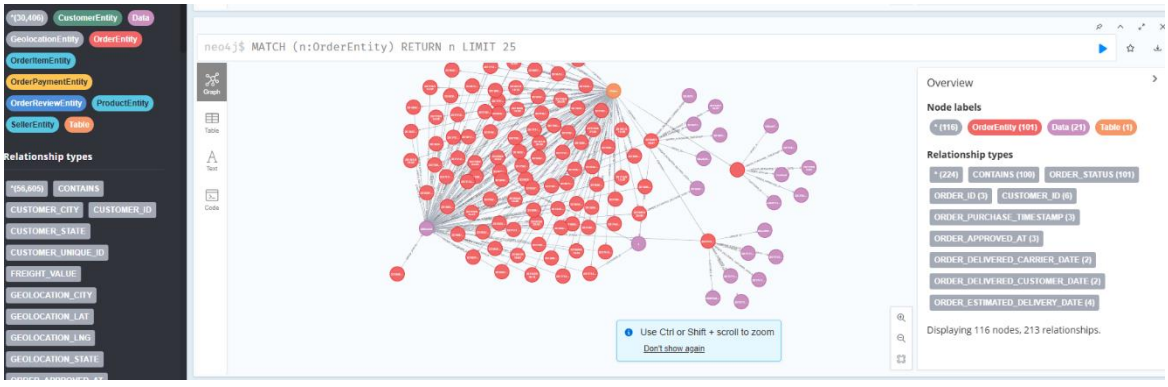

图 4-8 构建知识图谱展示

知识图谱界面基于 3.2 和 3.4 节知识图谱构建和基于知识图谱检索完成，前端通过 React 的 Neo4j 库与 Neo4j 数据库建立连接并将知识图谱展示在前端，知识图谱检索界面如图 4-9 所示，该界面主要实现基于知识图谱的数据检索以及展示功能。

该界面能够展示已经构建完成的面向数据治理的知识图谱，所以用户还能够对想要检索的信息进行检索，检索时输入想要检索的字段名和对应的值，系统会根据第 3.3 节实现的元数据匹配融合算法，将查询属性值转化成向量与向量数据库中存储的元数据进行比较，选择相似度超过 0.9 的属性值作为查询的属性值，系统会自动 Cypher 查询语句传给 neo4j 数据库并将查询到的结果展示到前端。

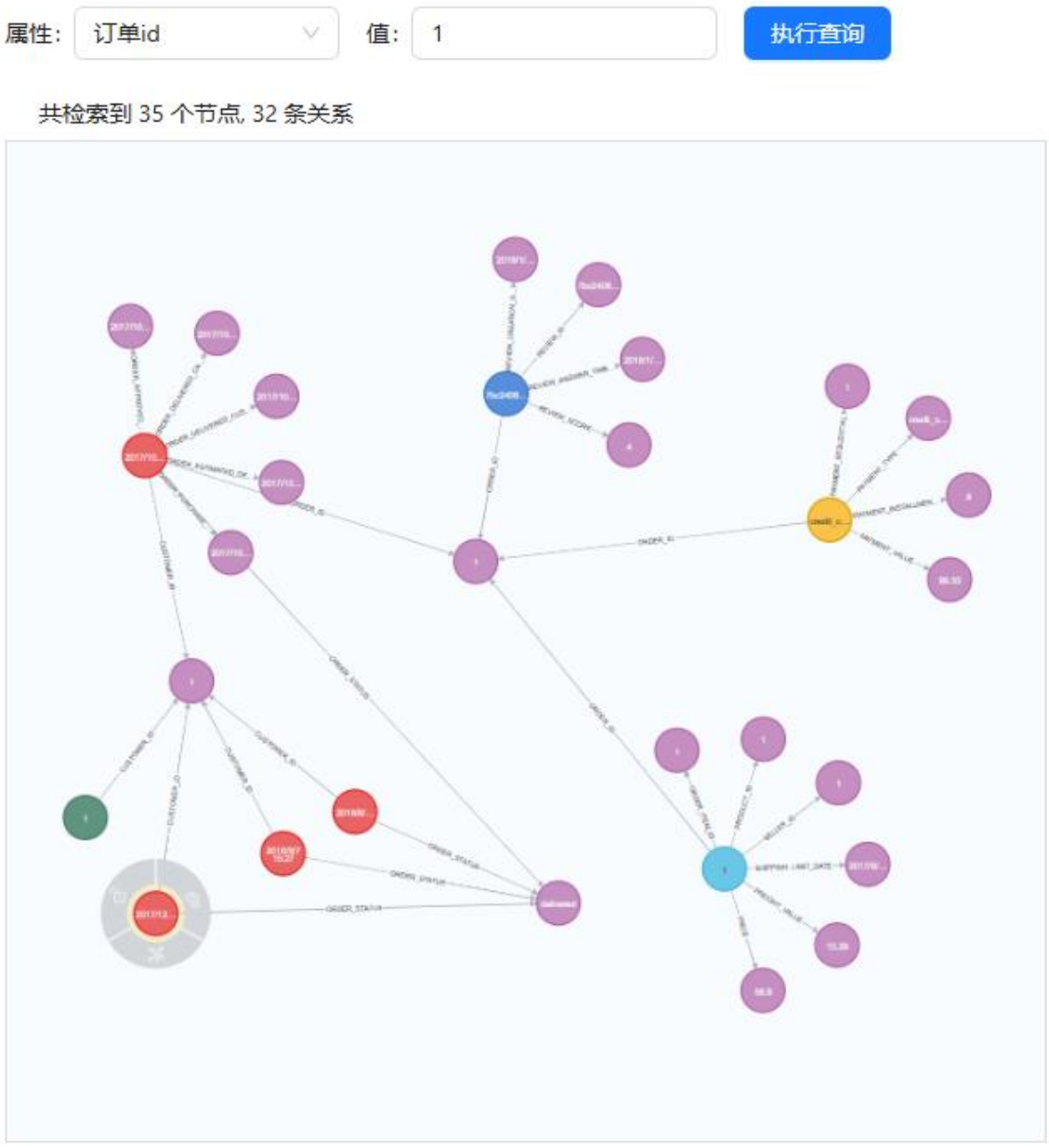

图 4-9 知识图谱检索界面

如图 4-9 展示了搜索“订单 id”为 1 的节点，可以看到最终展示的搜索结果为 order_id 为 1 的节点，最中心为 order_id 为 1 的 Attribute 节点，红色为 OrderEntity 节点、深蓝色为 OrderReviewEntity 节点、黄色为 OrderPaymentEntity 节点、浅蓝色为 OrderItemEntity 节点，通过 order_id 为 1 的 OrderEntity 节点，可以连接到所属 customer_id 同为 1 的 CuatomerEntity 节点，通过该节点又可以找到该 Customer 所有的所有 OrderEntity。

4.2.4 数据资产界面

数据资产界面是数据服务的一个重要界面，如图 4-10 示。 根据 3.4 节实现的数据资产构建方法，在前端通过获取数据模型和绑定的数据源信息，用 Echarts 渲染可视化数据资产信息。

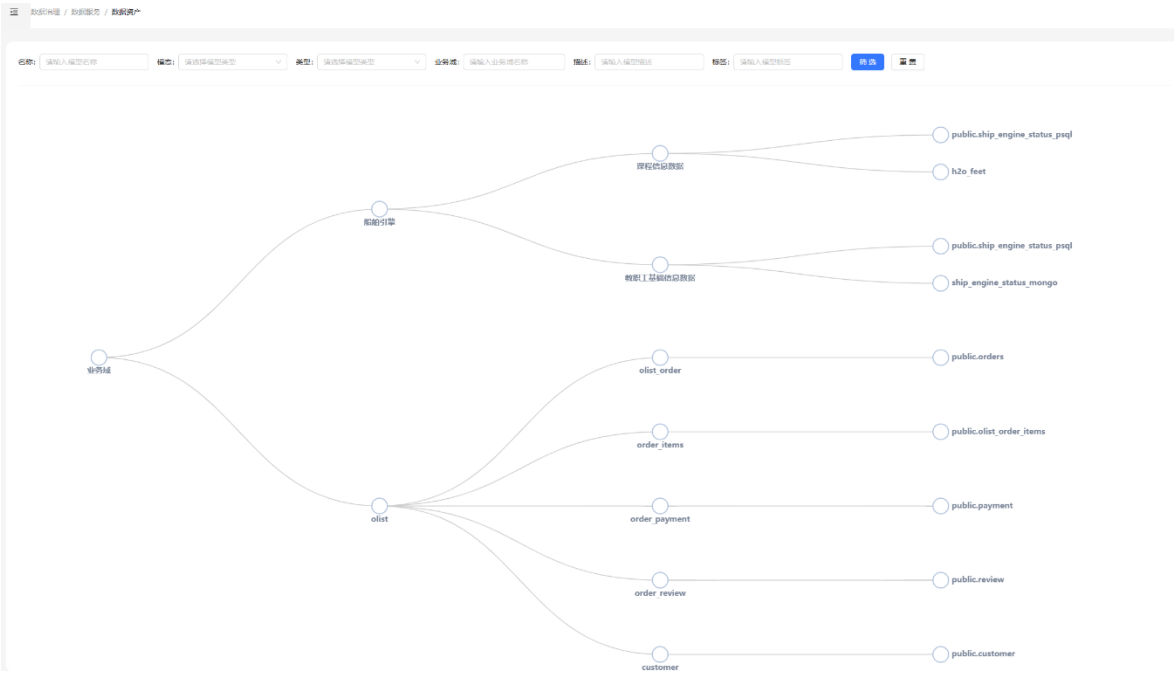

图 4-10 数据资产界面

该界面通过一种直观的树状节点布局呈现了数据资产。此界面使用户能够清晰地查看各个业务域中的数据模型以及与之关联的数据源信息。通过这个界面，用户不仅可以一目了然地掌握公司的数据资产概况，还可以对特定的数据源进行数据导出操作同时在导出时也会根据数据模型中的定义对数据进行清洗和脱敏操作。此外，

界面顶部的筛选功能允许用户按照特定的类型或业务域筛选和查看数据，这大大便利了用户在需要时找到特定范围的资产类型。

4.2.5 数据地图界面

数据地图界面如图 4-11 所示， 根据 3.4 实现的数据地图构建方法，在前端获取所有数据模型字段信息以及关联关系，通过 Echarts 渲染可视化数据地图信息。

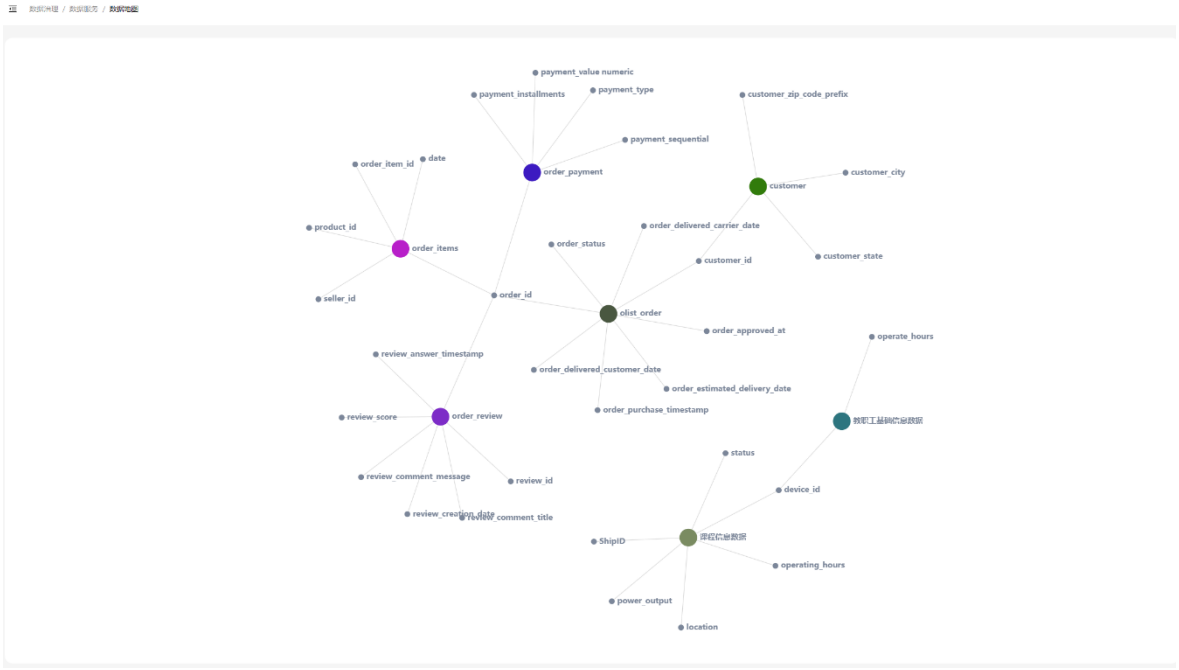

图 4-11 数据地图界面

该界面以数据模型名称为中心节点，以数据模型中字段名称为边节点，不同中心节点通过相同的边节点产生联系，实现多模型的数据关联。通过数据地图，用户不仅可以看到数据模型之间的直接关系，还可以识别出跨模型的复杂数据依赖和交互。这有助于发现潜在的数据整合机会，优化数据管理策略。

可以从图中看到的四个数据模型 olist_order、order_items、order_payment、order_review 以 order_id 节点产生关联，customer 与 olist_order 通过 customer_id 产生关联。

4.3 实验验证

本节将通过两个实验验证本文所实现的元数据匹配融合算法和基于知识图谱的数据补全算法的可行性。第一个实验采用 SimLex 数据集作为实验数据，第二个实验采用 Olist 中“Customer”数据集作为实验数据。

4.3.1 元数据匹配融合

为测试不同模型实际应用效果，本文使用 SimLex999 数据集进行测试，该数据集用于评估词汇相似性的模型，包含 999 对英文单词，目标是提供一个更严格和一致的基准以评估各种词向量模型，本文将使用上述三种词向量模型对数据集中的词语进行相似度评分，并通过与每对单词的人工标注相似度评分计算 Spearman 系数，Spearman 相关系数是一种非参数统计量，用于衡量两个变量之间的关联强度，它适用于分类数据和连续数据，其值范围从-1 到 1.当两个变量的取值完全相关使，Spearman 相关系数的值为 1，完全无关则为-1。通过 Spearman 系数大小可以判断该模型是否准确。测试结果如表 4-1 所示：

表 4-1 模型准确度测试结果

模型名称	Spearman 系数
Word2Vec	0.441965511
GloVe	0.408285489644521
FastText	0.4499649189464246

通过表格可以看到 FastText 具有最高的 Spearman 系数，这意味着 FastText 在捕捉单词相似性时与人工评分的相关性最强，在该数据集上效果最好，能够较好地捕捉单词间的语义关系，同时 Word2Vec 的 Spearman 系数略低于 FastText，说明两者在此数据集上的表现效果相近，尽管 GloVe 的 Spearman 系数最低但是也与其余两个相差不多。

实验数据说明这三个模型在该数据集上表现都不错，属于中等相关性，但是为了进一步提高准确性，优化元数据匹配融合算法，本文选择结合这三种模型的词嵌入技术，充分利用它们各自的优点，提高元数据匹配和融合的效果。主要方法为组合词向量并应用融合词向量来进行词语间相似度计算，具体实现如下：

（1） 词向量拼接

将同一词汇在不同模型中生成的向量拼接在一起，形成一个更高维的向量，这种方法结合了多种模型的语义信息，丰富了词汇的语义表示。

（2） 词向量平均

对同一词汇在不同模型中生成的向量取平均值，这种方法融合不同模型的信息，降低维度增加的复杂度。

（3） 基于加权的融合

为同一词汇在不同模型中生成的词向量赋予不同权重，生成加权平均向量，这种方法可以根据模型的优劣和任务的需求调整权重，提高融合效果。

根据以上不同方法重新构建算法并进行测试后的结果如表 4-2 所示：

表 4-2 改进后模型准确度测试结果

改进方法	Spearman 系数
词向量拼接	0.42602175485392013
词向量平均	0.4155065758294917
加权融合（FastText:0.2,Word2Vec:0.4,GloVe:0.4）	0.4147633109027252
加权融合（FastText:0.3,Word2Vec:0.4,GloVe:0.3）	0.4297325753807274
加权融合（FastText:0.4,Word2Vec:0.4,GloVe:0.2）	0.4501021050164013
加权融合（FastText:0.5,Word2Vec:0.3,GloVe:0.2）	0.45223026844907713
加权融合（FastText:0.6,Word2Vec:0.3,GloVe:0.1）	0.45479026548450656
加权融合（FastText:0.6,Word2Vec:0.4,GloVe:0）	0.47673976094019303
加权融合（FastText:0.7,Word2Vec:0.3,GloVe:0）	0.49757835929506635
加权融合（FastText:0.8,Word2Vec:0.2,GloVe:）	0.46651377961165246

通过表格中的数据可以看出来词向量拼接方法比单独 Glove 要好，但是比单独的 Word2Vec 和 FastText 差，这可能是因为向量维度过高而导致计算复杂度增加，且不同的模型特征可能相互干扰导致结果还不如单个模型；词向量平均法也不如单个的 Word2Vec 和 FastText 方法，可能是因为各模型特征不同，简单平均会丢失某些重要信息。

在进行加权融合的多种策略中，在前两个策略中，Word2Vec 权重不变，提高 FastText 权重可以发现 Spearman 系数增加，在第三个和第四个策略中 GloVe 权重不变，提高 FastText 权重可以发现 Spearman 系数增加，并且随着 FastText 的权重逐渐增加至 0.7 时，Spearman 系数达到最高。以上结果显示 FastText 在词向量相似性任务中表现尤为突出，GloVe 的贡献较小，甚至在权重设为 0 时也获得了较好结果，所以最后本文选择 Fasttext 与 Word2Vec 权重比为 7：3 的组合来完成词嵌入（Word Embedding）工作。

4.3.2 基于知识图谱的数据补全

为验证所提出算法的有效性，本文采用 Olist 中的“Customer”数据集测试，该数据集以”customerid”作为关键属性，并包含了其他属性。在数据集中，存在多个具有相同”customerid”但信息不全的记录，测试时首先不使用该补全算法，，通过对比图 4-12 和图 4-13 中展示的知识图谱，可以清晰看到数据不全前后的差异，数据补全之前，对于 customer_id 为 1 的 Customer 实体节点，存在三个不同的实例，每个实例都连接着一些不完整的 Attribute 节点。而在应用了数据补全算法之后生成的知识图谱中，对于 customer_id 为 1 的 Customer 实体节点只剩下一个实例，并且这个实例拥有完整的数据信息。这一结果证明了该算法能够有效地实现数据的补全，确保了知识图谱中实体的完整性和一致性。

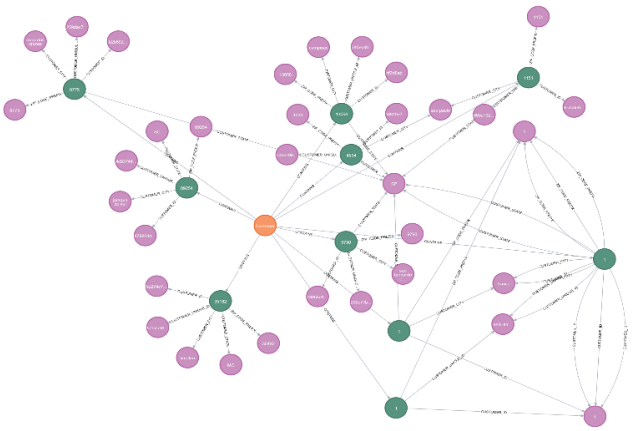

图 4-12 数据补全前知识图谱

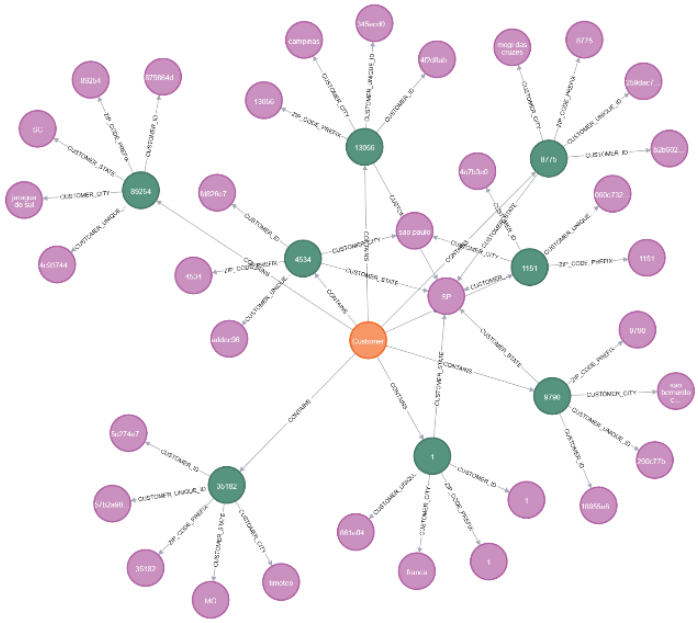

图 4-13 数据补全后知识图谱

4.4 本章小结

本章主要展示了基于知识图谱的数据治理平台的系统部署架构和原型。我们验证了数据补全处理及元数据匹配融合算法，并对实验结果进行了深入讨论。该系统支持多数据源配置和抽取，利用知识图谱支持数据源映射和元数据融合，确保元数据的一致性。此外，系统通过知识图谱识别和关联不同数据源的相似实体，有效解决了数据异构性问题，并构建了统一视图以提高数据整合的效率和准确性。通过编目和可视化展示数据资产，系统极大地方便了用户操作数据资产。总体来说本系统有良好的可靠性和可用性。

第五章 结论

5.1 主要结论

本文分析了企业数据管理中存在的数据孤岛、数据质量不一、元数据不匹配等问题，为了解决此问题，提出了一种基于知识图谱的数据治理方法，并讨论了该平台的设计与实现，目的在于在知识图谱的基础上实现一个具有通用意义的数据治理平台，帮助企业更加高效地管理其数据资产，提高数据的可访问性和质量。

对于本文所做工作，总结如下：

（1）实现多种数据源的配置和抽取，完成面向多种数据类型数据的数据汇聚。构建数据模型、对多种数据源进行配置和抽取，在数据模型的基础上实现数据源对数据模型的映射，保证元数据统一，开发元数据匹配融合算法

（2）实现面向数据治理的知识图谱构建。基于已构建的数据模型和抽取的数据源完成知识图谱的构建，知识图谱包括数据源中映射到数据模型的实体还包括丰富的实体关系。

（3）实现基于知识图谱的数据治理。基于已构建的知识图谱和元数据匹配融合算法完成数据补全工作以及数据检索功能，实现基于知识图谱的数据治理。

（4）实现可视化的软件界面，包括数据模型管理、数据源管理、数据资产、数据地图和知识图谱界面，提供用户友好的交互界面，用以对整个数据治理流程进行操作

5.2 研究展望

基于本课题已有的研究成果，对未来的研究进行如下展望：

（1）本文在数据模型的质量规则设置中提供了三个较为常见的质量规则，但是在实际的应用、使用场景中，可能需要针对特定领域数据的质量规则，未来需要对质量规则进行丰富并可以实现用户对质量规则进行自定义功能。

（2）本文对于数据质量管理实现在数据源汇聚部分，在进行元数据映射后根据数据模型设置的质量规则进行质量管理，在后续的应用过程中可能还会出现质量问题，所以未来需要实现持久化质量管理

（3）在元数据匹配融合算法中只用了较为简单的将三个 WordBedding 模型进行加权融合来实现词嵌入，未来可以进一步优化构建神经网络对模型进行训练，进一步提高元数据匹配融合算法的准确率。

参 考 文 献

[1] Xiao YH, Knowledge Graph Concept and Technology[M]. Beijing:Electronic Industry Press, 2020.

[2] 张天成, 田雪, 孙相会, 于明鹤, 孙艳红, 于戈. 知识图谱嵌入技术研究综述. 软件学报, 2023, 34 (1): 277–311.

[3] 郭昌雨.知识抽取方法的研究与实现[D].电子科技大学硕士论文,2023.

[4] Ling Xiao, Weld D.S.Fine-grained entity recognition[C] //Proc of the 26th Conf on Association for the Advancement of Artificial Intelligence. Menlo Park, CA: AAAI,2012: 94-100.

[5] GUO J, LI Z, YU Z, et al. Extraction and relation prediction of domain ontology concept instance, attribute and attribute value[J]. Journal of Nanjing University: Natural Science, 2015.

[6] 李冬梅，张扬，李东远，林丹琼.实体关系抽取方法研究综述[J].计算机研究与发展,2020,第 57 卷(7): 1424-1448

[7] CARLSON A, BETTERIDGE J, WANG R C, et al. Coupled semi-supervised

[8] 冯轩闻，袁新瑞，孙霞，高厦.结合强化学习和 DenseNet 的远程监督关系抽取模型[J].计算机应用与软件,2024,第 41 卷(2): 138-144，208.

[9] CHEN L, FENG Y, ZHAO D. Extracting relations from the web via weakly supervised learning[J]. Journal of Computer Research and Development, 2013, 50: 1825-1835.

[10] BANKO M, CAFARELLA M J, SODERLAND S, et al. Open information extraction from the web [C]//Proceedings of the 20th International Joint Conference on Artifical Intelligence. Morgan Kauf mann Publishers Inc., 2007: 2670–2676.

[11] 邢双双, 刘名威, 彭鑫. 基于软件知识图谱的代码语义标签自动生成方法. 软件学报, 2022, 33(1 1): 4027–4045.

[12] Seitner J, Bizer C, Eckert K, Faralli S, Meusel R, Paulheim H, Ponzetto S. A large database of hype rnymy relations extracted from the Web. In: Proc. of the 10th Int'l Conf. on Language Resources a nd Evaluation Conf. 2016.

[13] 王利亚, 邱航, and 陈若雅. "基于元数据可追溯性的健康医疗大数据治理方法及可视化呈现." 中国卫生信息管理杂志 16.6(2019):6.

[14] 严承希,房小可.开放世界视角:面向多源词表的知识融合框架 MtFFO 研究[J].中国图书馆学报, 2017,43(4):114~129

[15] Peterson, Richard E .A Cross Section Study of the Demand for Money: The United States, 1960-62 [J].Journal of Finance, 1974, 29(1):73.DOI:10.2307/2978215.

[16] 高伟. 数据资产管理 盘活大数据时代的隐形财富 = Data Asset Management How to Activate Hidden Wealth of the Big Data Era. 机械工业出版社, 2016. Print. 大数据科学丛书.

[17] 纪春华,石浩瀚,王艳磊.基于 Oracle 数据库整合技术的研究[J].信息技术与信息化,2018(11):95-101.

[18] 阿里巴巴.阿里云通用数据中台白皮书[EB/OL].[2022-10-10].

[19] 朱红甫.打造企业数据中台推进企业智慧运营[J].通信企业管理,2018(2): 32-33.

[20] Campos, Jaime, et al. "A big data analytical architecture for the Asset Management." Procedia CIRP64 (2017): 369-374.

[21] 熊拥军，白瀚祯，张廷成.基于数据中台的图书馆数据资产管理架构[J].图书馆学研究,2023,(8): 36-47.

致 谢

本科四年的生涯转眼间就要落下帷幕，在本篇学士学位论文将要完成之际，我要向所有在这四年间对我有过教导、关心和支持的所有人表示由衷的感谢和真挚的祝福。

首先我要衷心感谢我的导师兼班主任蔡鸿明老师在我大学四年学业和生活上的帮助和指导，在考研期间我曾一度迷茫，蔡老师为我拨开迷雾指明方向，在本篇论文撰写时也耐心指导为我答疑解惑，解决问题，同时我也要感谢实验室王钰霄、曾宇欣、葛煜、钱昱辰、张瑞轩等学长在本课题研究以及论文撰写过程中对我给出的建议。

感谢我的父母在我四年大学生活中对我生活和学习上的关心鼓励。

感谢我的室友李睿翔、王维谆、胡煜大学四年在生活上的包容与帮助。

感谢周宇呈同学两年多的陪伴与关怀。

CONSTRUCTION AND IMPLEMENTATION OF A DATA GOVERNANCE PLATFORM BASED ON KNOWLEDGE GRAPH

In today's business environment, data has evolved from a simple asset into the core of corporate competitiveness. In this era driven by big data and artificial intelligence, a company's success largely depends on how it collects, manages, and utilizes data. Effective data governance not only enhances the efficiency of data usage but is also crucial for improving operational efficiency, enhancing the quality of decision-making, ensuring data security, and maintaining compliance.

However, with the explosive growth in data volumes and the diversification of data types, enterprises face unprecedented challenges in data management. First, the issue of data silos prevents enterprises from fully understanding and utilizing their data assets. Secondly, inconsistencies in data quality can lead to inaccuracies in decision-making, increasing operational costs. Lastly, mismatches in metadata can hinder the integration and utilization of data. These issues severely impede the effective use of data, increase operational costs, and diminish decision accuracy.

In this context, strengthening data governance is particularly critical. Data governance is a comprehensive management process aimed at ensuring the quality, availability, integrity, security, and compliance of data assets throughout their lifecycle. By establishing a series of policies, procedures, and controls to manage the acquisition, storage, distribution, and usage of data, effective data governance can systematically enhance data accuracy and value, address data silo issues, and promote the integration and sharing of data, thereby significantly improving enterprise decision-making efficiency and accuracy.

Among the numerous tools and methodologies for data governance, knowledge graphs have emerged as a powerful tool to tackle these challenges due to their unique advantages. The core strength of knowledge graphs lies in their ability to clearly define the relationships between entities and the connections between entities and their attributes, thus enabling more precise data analysis and reasoning. Through this approach, knowledge graphs help businesses uncover patterns hidden in data, gain insights into potential business

opportunities or risks, and achieve more accurate predictions and decision support. Moreover, this structured expression of data makes the retrieval of complex information more intuitive and efficient. The application of knowledge graphs also fosters data standardization and metadata management within data governance, which is crucial for maintaining data consistency. By systematically managing and utilizing knowledge graphs, enterprises can effectively enhance the overall quality and value of their data.

Firstly, this article analyzes the importance of data governance and the challenges faced, with a special emphasis on the role of knowledge graphs in enhancing data management efficiency. By discussing issues such as data silos, inconsistencies in data quality, and metadata mismatches, this chapter highlights not only the advantages of knowledge graphs in the semantic representation of data but also their novel approaches to data integration and quality control. Furthermore, this chapter proposes the design concept of a data governance platform based on knowledge graphs, aimed at enhancing data accessibility and decision support capabilities, ultimately driving continuous innovation and growth for enterprises. This provides a theoretical foundation and research direction for subsequent chapters regarding specific implementation methods and system frameworks.

Then, this article primarily analyzes the existing business scenarios involved in the system and the improvements to the business scenarios after system implementation. Starting from the application business scenarios, it proposes both functional and non-functional requirements for the system. In terms of functional requirements, this chapter explains them in the form of a feature list; in terms of non-functional requirements, the system focuses on the usability, reliability, and security of the platform. Based on the analysis of system application scenarios and requirements, it proposes the overall framework and technology stack of the system.

Next, the article delves into the design and implementation of the system, involving three major modules: data modeling, data aggregation, and data services. The data model module, as the foundational support of the platform, provides a detailed introduction to the construction of data models, including the definition and implementation of structured, semi-structured, and unstructured data models; the data aggregation module focuses on how to extract, clean, and integrate data from dispersed data sources. This chapter discusses in detail the strategies for metadata extraction, including the treatment of structured, semi-structured,

and unstructured data, as well as how to use knowledge graph technology for data cleaning and completion. The data services module shows how to provide efficient data services based on the previous two modules, including data retrieval based on knowledge graphs, construction of data maps, and data asset displays.

Finally, the article mainly showcases the system deployment architecture and system prototype display in the data governance platform system based on knowledge graphs and verifies the data completion processing based on knowledge graphs and the metadata matching fusion algorithm through experimental validation, discussing the experimental results to demonstrate the system's usability and reliability.

In summary, the work done in this article includes:

(1) Implementing the configuration and extraction of multiple data sources, completing data aggregation for multiple types of data. Building data models, configuring multiple data sources, implementing the mapping of data sources to data models, ensuring metadata uniformity, and developing metadata matching fusion algorithms.

(2) Implementing knowledge graph construction for data governance. Completing the construction of knowledge graphs based on the built data models and extracted data sources, including entities mapped from data sources to data models and a rich network of entity relationships.

(3) Implementing data governance based on knowledge graphs. Based on the constructed knowledge graphs and metadata matching fusion algorithms, completing data completion tasks and data retrieval functions, implementing data governance based on knowledge graphs.

(4) Implementing a visual software interface, including data model management, data source management, data assets, data maps, and knowledge graph interfaces, providing a user-friendly interactive interface for the entire data governance process.