

电子科技大学
UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE



论文题目 基于 Spark 的元数据管理系统
的设计与实现

专业学位类别 工程硕士

学 号 201952090706

作者姓名 马张迪

指导教师 唐雪飞

学 院 信息与软件工程学院

分类号 TP311.5 密级 公开
UDC ^{注 1} 004.41

学 位 论 文

基于 Spark 的元数据管理系统的设计与实现

(题名和副题名)

马张迪

(作者姓名)

指导教师 唐雪飞 副教授
电子科技大学 成 都

(姓名、职称、单位名称)

申请学位级别 硕士 专业学位类别 工程硕士
专业学位领域 软件工程
提交论文日期 2022 年 3 月 23 日 论文答辩日期 2022 年 5 月 17 日
学位授予单位和日期 电子科技大学 2022 年 6 月
答辩委员会主席 _____
评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

Design and Implementation of Metadata Management System Based on Spark

A Master Thesis Submitted to
University of Electronic Science and Technology of China

Discipline **Master of Engineering**

Student ID **201952090706**

Author **Zhangdi Ma**

Supervisor **A/Prof. Xuefei Tang**

School **School of Information & Software Engineering**

摘 要

随着高校数字化建设的深入开展，全国各个高校都在加快建设自己的数据中心，其中的一卡通平台，教学系统和科研管理等系统每天都会产生大量的数据。由于早期各部门数据存储系统和数据定义标准都不尽相同，这给数据的统一管理和使用造成了极大的障碍，建设统一的数据中心并进行数据治理就显得十分必要。目前数据治理在全国各个高校开始兴起，元数据管理作为数据治理的重要部分，参与了数据的整个生命周期的活动。元数据管理是实现高校数据资产的全面聚合和数据深层次共享的核心关键。

本文以传统的元数据管理系统为背景，从高校元数据管理的需求出发，对基于大数据技术的元数据管理系统进行设计和实现，即基于 Spark 的元数据管理系统。并详细阐述了理论基础，系统设计与实现和测试的内容。以下是论文主要的工作：

1. 在元数据分析方面：本系统使用 Spark 的核心组件 SparkSQL 对数据库进行操作，解析 SparkSQL 的逻辑计划，并对其进行拆解得到元数据血缘关系。通过对 SparkSQL on Hive 模块进行修改，解决了 SparkSQL 对字段级别的元数据分析困难的问题，提高了 Spark 对于字段级别元数据的分析能力。

2. 在元数据质量方面：本系统从元数据的填充完备率、一致性、唯一性，有效性和完整性对元数据质量进行检测，排查不规范数据。系统运行这些数据质量规则可产生相应的元数据质量报告，并支持导出报告，让数据分析人员对系统的元数据质量情况有清晰的掌握。

3. 本论文设计并实现了基于 Spark 的元数据管理服务，通过 HDFS 双机热备机制来保证系统存储数据的一致性，通过 YARN 来进行集群节点之间的任务调度，通过 Spark 来处理集群的计算请求，通过 Hive 进行数据仓库管理，使用 HTML 和 Vue.js 开发了 Web 界面进行功能交互，使数据管理员和数据分析员能便捷化地进行元数据管理。

本论文实现了一个基于 Spark 的元数据管理系统，通过对系统进行功能测试分析，系统满足元数据管理的基本功能需求，对于元数据质量可以全程监控。基于 Spark 的元数据管理系统可以为后续数据分析，以及大数据治理活动：数据质量监控，主数据管理，数据资产管理提供深层次的服务，可以进一步加快高校大数据治理的进程。

关键词：数据治理，元数据管理，SparkSQL，血缘分析，元数据质量

ABSTRACT

With the in-depth development of digital construction of university, every university in China is speeding up the construction of its own data centre, in which the one-card platform, teaching system, scientific research management and other systems will generate a large amount of data every day. As the data storage system and data definition standards of each department are different in the early days, this has caused great obstacles to the unified management and use of data, so it is necessary to build a unified data centre and carry out data governance. Data governance is now beginning to emerge in various universities across the country, and metadata management, as an important part of data governance, is involved in the entire lifecycle of data activities. Metadata management is the core key to achieving full aggregation of data assets and deep sharing of data in universities.

This thesis takes the traditional metadata management system as the background. From the demand of metadata management in universities, we design and implement a metadata management system based on big data technology, i. e. a Spark-based metadata management system. This thesis details the theoretical basis, system design and implementation and testing. The following are the main work of the thesis.

1. In terms of metadata analysis: This system uses SparkSQL, the core component of Spark, to operate on the database, parse the logical plan of SparkSQL, and disassemble it to get metadata lineage. By modifying SparkSQL on Hive module, the problem of SparkSQL's difficulty in analyzing field-level metadata is solved, and Spark's ability to analyze field-level metadata is improved.

2. In terms of metadata quality: The system detects metadata quality in terms of filling completeness, consistency, uniqueness, validity and completeness of metadata, and checks irregular data. The system runs these data quality rules to generate corresponding metadata quality reports and supports exporting reports, so that data analysts have a clear overview of the metadata quality of the system.

3. This thesis designs and implements a Spark-based metadata management service, which ensures the consistency of system storage data through HDFS dual hot standby mechanism, task scheduling between cluster nodes through YARN, processing cluster computation requests through Spark, data warehouse management through Hive, and

developing a web interface using HTML and Vue.js for functional The web interface was developed using HTML and Vue.js for functional interaction, enabling data administrators and data analysts to easily manage metadata.

This thesis implements a Spark-based metadata management system. Through functional testing and analysis, the system meets the basic functional requirements of metadata management, and the quality of metadata can be monitored throughout. The Spark-based metadata management system can provide in-depth services for subsequent data analysis and big data governance activities: data quality monitoring, master data management and data asset management, which can further accelerate the process of big data governance in university.

Keywords: Data Governance, Metadata Management, SparkSQL, Lineage Analysis, Metadata Quality

目 录

第一章 绪论	1
1.1 研究工作背景及意义	1
1.1.1 研究背景	1
1.1.2 研究意义	2
1.2 国内外研究现状	3
1.2.1 高校数据治理现状	3
1.2.2 大数据平台中元数据管理现状	4
1.3 本文主要工作与贡献	4
1.4 论文组织结构	5
第二章 理论基础及技术	6
2.1 Hadoop 分布式文件系统	6
2.1.1 HDFS 架构	6
2.1.2 HDFS 的数据一致性	7
2.2 Spark 分布式计算框架	8
2.2.1 Spark 架构	9
2.2.2 Spark 核心组件	10
2.3 元数据管理	11
2.3.1 元数据管理策略	12
2.3.2 元数据模型	12
2.3.3 元数据分析	13
2.4 Hive 数据仓库	13
2.5 Sqoop	14
2.6 YARN	14
2.7 本章小结	15
第三章 基于 Spark 的元数据管理系统的需求分析和总体设计	16
3.1 需求分析	16
3.1.1 功能需求	17
3.1.2 非功能需求	18
3.2 应用场景分析	19
3.3 系统架构设计	20

3.3.1 系统架构	20
3.3.2 功能层次架构	22
3.4 系统功能设计	23
3.4.1 用户登录模块	24
3.4.2 元数据采集模块	25
3.4.3 元数据模型模块	28
3.4.4 元数据基础功能模块	30
3.4.5 元数据分析模块	31
3.4.6 元数据质量检测模块	33
3.5 关键技术设计	37
3.5.1 Spark 对字段血缘分析的设计	37
3.6 数据库设计	39
3.7 本章小结	40
第四章 基于 Spark 的元数据管理系统的详细设计与实现	41
4.1 开发环境	41
4.2 数据源处理	42
4.2.1 数据源选择	42
4.2.2 数据传输	42
4.3 用户登录模块	43
4.4 元数据采集模块	44
4.4.1 元数据的定义	44
4.4.2 元数据采集	47
4.4.3 元数据存储	49
4.5 元数据模型模块	49
4.5.1 基础教学元数据模型	50
4.5.2 数据转换层元数据模型	51
4.5.3 数据仓库层元数据模型	52
4.6 元数据基础功能模块	53
4.6.1 元数据的查询和变更	53
4.6.2 元数据计数	54
4.7 元数据分析模块	54
4.7.1 血缘分析	54
4.7.2 影响分析	56

4.8 元数据质量检测模块	57
4.8.1 元数据质量检测	57
4.8.2 元数据质量报告	57
4.9 关键技术实现	58
4.10 本章小结	59
第五章 系统测试与分析	61
5.1 功能测试	61
5.1.1 用户登录模块测试	61
5.1.2 元数据采集模块测试	61
5.1.3 元数据模型模块测试	62
5.1.4 元数据基础功能及分析功能测试	62
5.1.5 元数据质量模块测试	63
5.2 性能测试	63
5.3 本章小结	64
第六章 总结与展望	65
6.1 总结	65
6.2 展望	66
致 谢	67
参考文献	68

第一章 绪论

1.1 研究工作背景及意义

1.1.1 研究背景

国家教育部在二零一八年颁布的《教育信息化 2.0 行动计划》^[1]中着重介绍了互联网时代基础教育数字化的特点，其总体目标为到二零二二年，高校基本完成“三全两高一大”的发展目标，即学习应用覆盖所有教育、全部适龄学生，建设现代数字化高校体系，提升教育信息化管理水平，提升高校教师和学生信息技术素质，形成基于教育+新一代信息技术的服务平台。这标志着我国教育信息化进入一个全新的时期，也为未来高等教育信息化改革发展指明了方向，同时也对高校信息化工作带来前所未有的挑战。

虽然全国各地的高校争先建设自己的互联网+大数据+教育的平台，校园的信息化发展给高校教学，科研和生活提供了许多便利的服务，但也产生了一些高校管理海量数据的问题。高校各类业务部门较多，应用系统繁杂，随着时间的增长会形成一个巨大的数据资源库，由于没有统一的数据汇聚中心，高校对这些数据的管理会造成人力资源使用上极大的浪费。由于数据标准定义的不统一，各个部门数据不互通甚至还存在数据定义冲突，这使得业务工作人员之间对系统数据的认知出现偏差，造成数据孤岛的问题存在，造成最后的大数据分析预测结果失效，造成各个业务部门领导决策出现错误等问题发生^[2]。

这些数字化系统所遇到的问题关键就是要做好高校的数据治理，目前数据治理正在全国各个高校兴起，数据治理是对校园的各部门信息化系统的数据进行整合、采集，清洗和规范化存储，是实现高校数据资产的全面聚合和数据的深层次共享的核心关键。如何以高校各个部门组织的数据为内核，建立好数据中心，并做好数据治理，在大数据平台上做好元数据管理是关键。元数据管理可以驱动高校数据仓库的建设。但如何针对高校数据定义元数据，做好元数据存储，发挥元数据功能，推进数据质量提升，让高校的数据能真正发挥它的价值，这是目前高校数据治理所面临的一个挑战。

在数字化校园高度发展的当下，各个应用系统也趋向智慧化和便捷化，随时随地都可以使用应用的功能^[3]。这增加了系统中数据交换的工作量，新业务对数据交换的能力要求也越来越高了。而且在特定时期会有大量的请求访问平台数据库，系统的运行会面临巨大并发压力，如何保证系统在高并发请求下的可用性也是一个需要重视的问题。

1.1.2 研究意义

数据的价值在于拥有知道和控制数据的来源和去向的能力，并能让数据为决策分析提供支持。元数据存在于整个数据的生命周期中，并承担着描述数据的责任，有效的元数据管理可以极大地促进高校信息化建设，元数据管理对数据的统一标准和数据关系的描述都有很大的作用，为高校的数据治理任务提供有力支持^[4]。随着大数据技术的不断发展和深入应用，大数据处理已经成为一种必然的趋势，目前我国高校对数据的重视程度越来越高了^[5]。因此，研究在大数据环境下如何构建高校数据管理体系是非常必要。元数据最大的应用价值是驱动数据中心的流程化，规范化，自动化地建设，让数据的处理过程可管，可控和可查。元数据管理可以为高校数据的采集、存储、计算、共享，价值和安全性提供有力保障。所以做好元数据管理是十分必要。

在整个数据中心的建设、运行和维护过程中，元数据是指对数据源信息、数据 ETL 过程、数据库信息、数据建模信息，业务数据的类型和应用场景组织等的描述。元数据管理是为数据管理者提供准确、高效，高质量和易于维护的元数据。在高校数据中心建设中，可以参考教育部发布的元数据规范标准和高校各类业务系统的数据定义，理清元数据所属类型，建立教育教学元数据标准，完善高校数据的解释和定义，形成规范的高校元数据标准，并对所有数据的来源、提取，转换和使用进行跟踪。通过有效的元数据管理，可以生成所有数据的资产概况，更加方便地定义和使用数据，提高数据质量^[6]。

在智慧校园环境下，把大数据技术应用在高校数据中心的元数据管理方面，一方面解决了部门业务数据割裂无法深入合作的问题，另一方面加快了高校系统的智慧化建设。元数据通过对数据采集、交换，分析和处理等步骤的深入描述，规范了数据的管理和服务，为高校师生的智慧化生活提供稳定和可靠的数据服务^[7]。高校通过建立统一的数据中心和完善的元数据管理服务，使大数据应用系统的数据使用更加流程化，规范化和自动化，提高了数据中心的数据管理效率。这一方面加快高校信息化的深入建设，让大数据技术在高校数据价值发掘中发挥更大的作用；另一方面加快了数据治理的进程，基于 Spark 的元数据管理系统^[8]可以给后续的数据标准管理、数据安全治理、数据质量监控，主数据管理和数据资产管理等大数据治理活动提供基础服务。

本系统是在 Hadoop 平台上进行搭建的，采用 Spark 分布式计算框架，数据挖掘人员可以便捷地使用 Spark 核心组件里的机器学习算法对数据进行数据挖掘工作，平台提供从采集，存储到使用的一站式的数据服务，让数据更便捷高效地发挥其自身的价值^[5]。

1.2 国内外研究现状

随着大数据技术和高校信息化建设的深入发展，各个高校已经基本完成自己部门业务系统的建设。然而搭建数据中心并进行统一的数据治理又是当下亟待解决的问题^[9]。对于数据中心的数据治理离不开元数据的管理^[10]。如何利用当下流行的大数据技术对数据中心数据进行有效的元数据管理，这还需要深入探索^[11]。本节将从高校数据治理和大数据平台元数据管理两个方面对国内外研究情况做简要介绍。

1.2.1 高校数据治理现状

由于高校信息化发展的不断深入，高校各部门业务系统已经初具形态，足够覆盖大部分业务需求，加之高校数据积累到了一定的规模，高校系统逐渐从数字化开始向智慧化转型^[12]。为了对高质量数据进行有效地发掘和使用，国内高校开始掀起了数据治理的浪潮^[13]。目前高校数据治理还处在数据集中的起步阶段^[14]。作为数据治理的基础支撑的元数据管理，也伴随数据中心建设而提上了日程。为完善教育信息化标准体系，以保障教育信息化健康有序发展，教育部于 2017 年印发了《基础教育教学资源元数据》系列标准，作为教育行业标准予以发布，并自发布之日起施行。高校也在探索开发“教育教学元数据管理”和“教育数据质量监测”等数据治理子系统，对高校数据中心数据实施有效的数据治理。面对海量的数据，如何对数据进行有效的治理，有许多学者提出自己的意见。在数据治理架构方面，包冬梅和范颖捷等人提出了大学图书馆数据的治理框架^[15]。在数据质量方面，对于数据质量的检测，朱杨勇等人^[16]提出从数据用户的角度出发制定层次化的数据质量框架，采取自动化和信息化相结合的检测流程，规范数据质量检测工作流程，服务于数据质量评估的需求。在元数据管理方面，不同的数据源就像一个个信息孤岛，如何集成到统一的数据中心并在对外提供数据服务，Pegdwendé S 等人^[17]提出一种新型的数据湖架构和元数据管理方案。目前高校普遍面临一种困境：虽然清楚数据很宝贵，但是无法找到如何打破不同部门之间的数据壁垒，释放数据价值，对数据进行有效地管理和使用的解决办法。这些信息系统运行多年积累了大量教学，科研和人事等相关服务的数据，如果这些数据不进行治疗就不能形成有效的数据资产，也就没有办法提供高质量的数据服务^[18]。对于早期部门相对独立造成的数据孤岛的问题，急需要建设统一的数据中心，把数据统一归集到一起，形成关于高校各个部门的数据湖，再配合数据治理对数据中心数据进行元数据管理和数据质量监控，为高校各部门的精细化服务提供数据支撑^[19]。

1.2.2 大数据平台中元数据管理现状

元数据管理解决方案存在于整个数据生命周期中，作为大数据治理中的必不可缺的部分，支持着其他数据资产管理任务^[20]。当前大数据平台上流行使用 Hive 来操作元数据，通过它将 SQL 语句逻辑转化成 MapReduce 任务进行处理^[21]。虽然一定程度上解决了元数据管理的问题，但是 MapReduce 由于执行过程会频繁读写磁盘，导致实时处理效率比较低^[22]。由于 Spark 基于内存的计算特点以及成熟的生态系统，有学者提出过使用 Spark 来对大数据平台数据进行治理^[23]。

在大数据时代下，数据占有重要的地位，有学者提出数据即是服务的概念，对数据中心的数据做好元数据管理可以极大地提高数据生产力和服务效率^[24]。Frank R 等人^[25]提出了在数据湖中基于元数据可扩展分类的元数据管理解决方案。当下国内有一些元数据管理软件，诸如亿信华辰研发的 EsPowerMeta，普元研发的 MetaCube^[26]。这些软件都针对元数据做管理，支持对多种数据源的元数据进行采集，存储和关系分析，帮助各类用户从全方面了解数据，进而发掘出隐藏在海量数据中的数据价值。国外现今研发比较成熟的产品有 Alation^[27]和 IBM 的 InfoSphere Information Governance Catalog，它们提供一系列有关元数据管理的解决方案^[28]。目前在开源社区也有一款叫 Apache Atlas 的元数据管理软件^[29]，虽然它使用起来简洁方便，但是在兼容性上还不够，目前还不能直接使用 Spark 的数据源，还需要依靠中间件才能使用，无法满足数据的一站式服务的要求。Spark 有丰富的生态组件，可以支持多种数据源的操作，SparkSQL 查询可以访问到 DataFrame，对于元数据的分析有天然的优势，可以采用 Spark 对元数据进行有效管理^[30]。

1.3 本文主要工作与贡献

本论文介绍了高校数据治理的背景，基于目前的技术的情况，设计并实现了基于 Spark 的元数据管理系统，使用 SparkSQL 对元数据进行了血缘分析和影响分析，提高了元数据分析的效率，还解决了 SparkSQL 无法对字段血缘进行元数据分析问题，为大数据平台上的元数据管理提供了一种思路。另外还根据 CWM 标准对教学资源进行了元数据模型设计，完成了对于元数据的采集。并且按照数据质量标准，从不同维度对元数据质量规则进行设计，并对元数据进行了数据质量检测，最后产生数据质量报告。

1. 在元数据分析方面：本系统使用 Spark 的核心组件 SparkSQL 对数据库进行操作，解析 SparkSQL 的逻辑计划，并对其进行拆解得到元数据血缘关系。通过对 SparkSQL on Hive 模块进行修改，解决了 SparkSQL 对字段级别的元数据分析困难的问题，提高了 Spark 对于字段级别元数据的分析能力。

2. 在元数据质量方面：本系统从元数据的填充完备率、一致性、唯一性，有效性和完整性对元数据质量进行检测，排查不规范数据。系统运行这些数据质量规则可产生相应的元数据质量报告，并支持导出报告，让数据分析人员对系统的元数据质量情况有清晰的掌握。

3. 本论文设计并实现了基于 Spark 的元数据管理服务，通过 HDFS 双机热备机制来保证系统存储数据的一致性，通过 YARN 来进行集群节点之间的任务调度，通过 Spark 来处理集群的计算请求，通过 Hive 进行数据仓库管理，使用 HTML 和 Vue.js 开发了 Web 界面进行功能交互，使数据管理员和数据分析员能便捷化地进行元数据管理。

本论文实现了一个基于 Spark 的元数据管理系统，通过对系统进行功能测试分析，系统满足元数据管理的基本功能需求，对于元数据质量可以全程监控。基于 Spark 的元数据管理系统可以为后续数据分析，以及大数据治理活动：数据质量监控，主数据管理，数据资产管理提供深层次的服务，可以进一步加快高校大数据治理的进程。

1.4 论文组织结构

本文主要研究了基于 Spark 的元数据管理系统的设计与实现，论文共有六个章节，结构安排如下：

第一章主要介绍了大数据环境下高校元数据管理研究的背景及意义，阐述了当下高校信息化发展的现状以及遇到的数据治理问题，说明了自己主要的工作，并提出全文的框架。

第二章是对基于 Spark 的元数据管理系统的理论基础的介绍，从存储、计算、管理策略，传输和调度五个方面对系统的技术理论做了详细解释。

第三章对基于 Spark 的元数据管理系统的需求分析，软件架构和功能架构做了介绍。并详细阐述了对系统功能的设计，主要有用户登录，元数据采集，元数据模型，元数据基础功能，元数据分析，元数据质量检测六个功能模块，并对关键技术进行设计说明。

第四章对基于 Spark 的元数据管理系统的实现，介绍了开发环境，以及各个功能模块的实现。详细阐述了元模型转换包定义。介绍了使用 SparkSQL 对元数据血缘分析的实现，以及元数据质量检测的实现。

第五章对基于 Spark 的元数据管理系统进行测试，包括功能测试和性能测试。

第六章对基于 Spark 的元数据管理系统的工作内容进行总结和对后续工作的展望。

第二章 理论基础及技术

2.1 Hadoop 分布式文件系统

在 Hadoop 大数据处理框架里, HDFS(Hadoop Distributed File System)是作为数据存储部分, 它有高度的容错性, 由于 HDFS 所拥有的主从结构特性, 其对集群化的数据存储提供了优秀的底层支持, 其还可通过 MapReduce 技术来对分布式并行任务处理提供支持^[31]。HDFS 提供一次写入多次读取的方式, 数据在系统中以块的形式分别储存在集群的各个机器上。

2.1.1 HDFS 架构

HDFS 1.0 采用了主从结构来保存数据, 这种结构主要由四个部分组成, 分别为 HDFS 客户端、名称节点, 数据节点和第二名称节点^[32]。HDFS2.0 又被称作 HDFS HA(High Availability)高可用集群, 它拥有两个名称节点, 一个状态为待命(Standby), 一个状态为活跃(Active), 通过 ZooKeeper 管理, 该高可用集群所拥有的两个名称节点分别称为活跃节点与待命节点, 存储系统可以通过两个名称节点实现对节点状态的同步控制^[33]。这两个节点可通过中间件 ZooKeeper 进行信息同步与互动, 一旦其中活动的名称节点出现宕机, 待命的名称节点可快速进行切换并提供服务。并且数据节点同时将信息报告给两个名称节点, 下图 2-1 是 HDFS HA 的架构图。

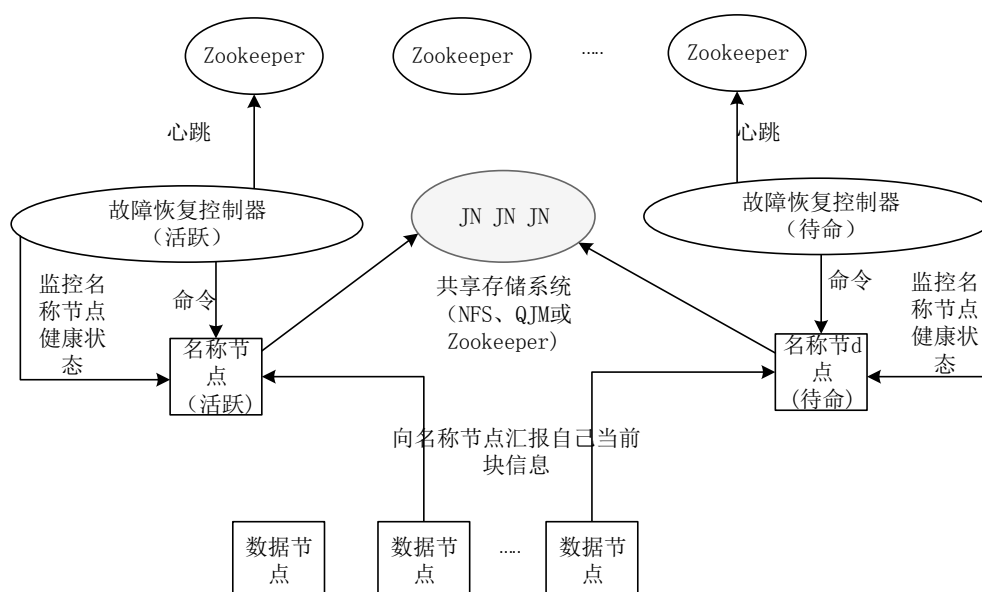


图 2-1 HDFS2.0 架构^[32]

2.1.2 HDFS 的数据一致性

2.1.2.1 Namenode 和 Datanode 的数据一致性

HDFS 的节点以主从架构来区分从属关系，名称节点作为管理节点主要管理系统的命名空间，也就是文件系统的文件目录。名称节点对该树型结构目录进行维护。由于在系统启动时可以重建块信息，所以是不长期存储块的节点信息。文件系统使用数据节点作为基础管理节点，根据该文件系统目录与树型结构进行检索，及时向名称节点传递它们保存的数据信息，名称节点和数据节点之间的数据一致性是通过主从架构来实现，如图 2-2 所示。

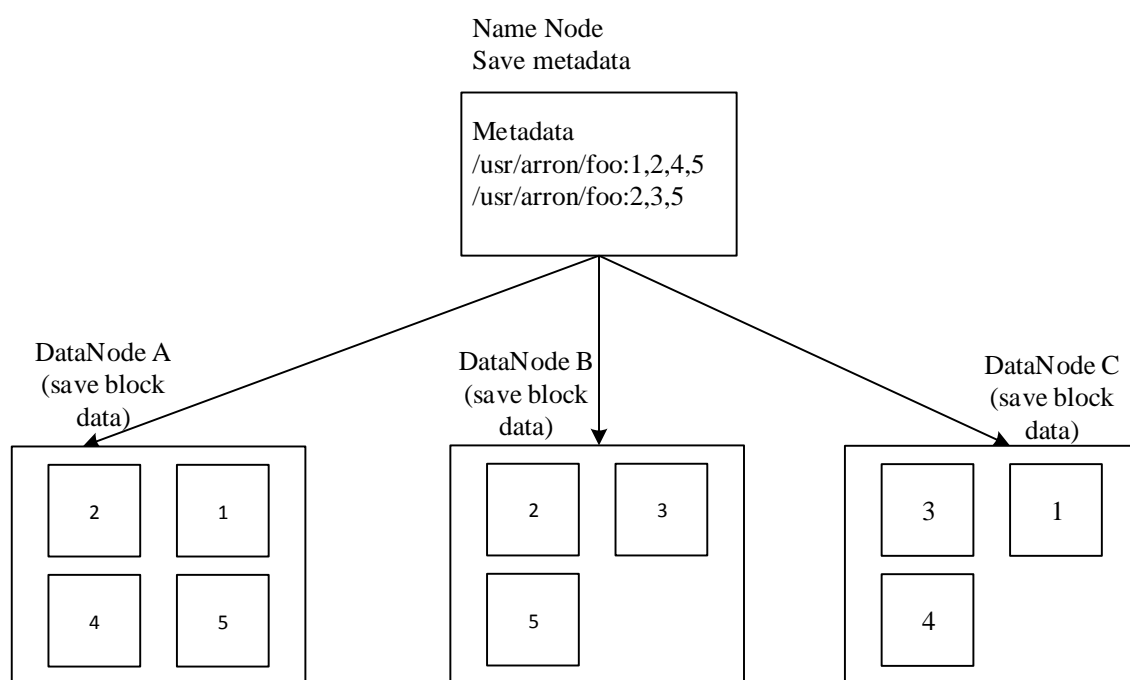


图 2-2 HDFS 节点架构

2.1.2.2 名称节点数据一致性

HDFS2.0 有两个名称节点，分别是 Active 状态和 Standby 状态。HDFS 存储都是通过数据节点来实现，数据节点只要对块信息进行修改，就会通过心跳的方式通知两个名称节点，名称节点就会更新自己存储的元数据信息。

HDFS 为了保证两个名称节点元数据一致性，必须满足两个条件：第一个条件数据同时传输，要确保数据同时传输，数据节点要对两个名称节点同时发送心跳；第二个条件命名空间的文件目录树要一致，这也是为了确保名称节点存储数据的一致性。

当 HDFS 中活跃的名称节点的文件目录发生变更时，活跃的名称节点将通知

所有 JN(JournalNode)此变化,但并不能保证所有的 JN 都会得知修改的消息,有些 JN 由于某些通信原因可能不会收到此变化消息^[34]。此时需要通过条件判断来决定此消息是否可靠,该条件判断也被称为选举法,即多数压倒少数,若多数节点确定收到消息,则此消息是可靠的,反之则是不可靠的。对于可靠的消息,备用名称节点才会执行同步操作,通过 JN 这种方式,才能保证备用名称节点和活跃名称节点之间实现消息同步更新,如下图 2-3 所示。

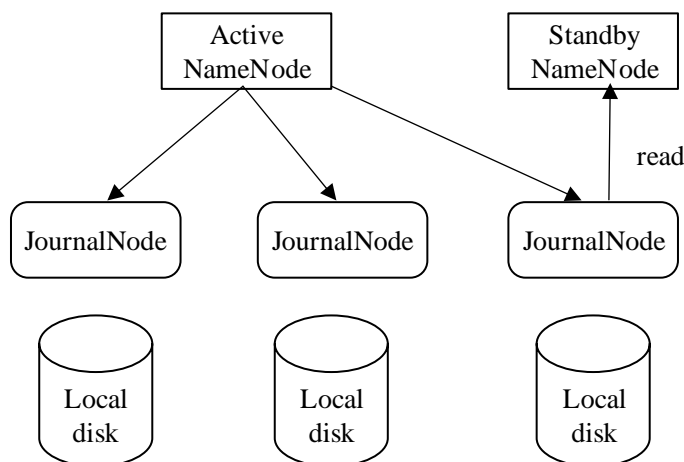


图 2-3 名称节点的同步机制

总的来说,在 HDFS 中数据一致性主要是通过“两个一致”来保证:一个是数据一致,另一个是命名空间一致。前者主要通过有效的数据管理来保证,后者通过 JN 来保证。

2.2 Spark 分布式计算框架

Apache Spark^[35]是基于内存的分布式计算框架,它与 Hadoop 生态系统集成在一起,两者可以完美的契合。Spark 通过和其他的分布式文件系统进行集成来实现其功能,Hadoop 的 HDFS 负责分布式存储,YARN 或 Mesos 负责资源调度,Spark 或 MapReduce 负责大数据计算。Spark 拥有高效的 DAG(有向无环图)执行引擎,可以对流式数据进行处理。在大数据处理需求中,该特性使其处理数据的实时性得到了有效的提升^[35]。Spark 的 RDD(弹性分布式数据库)技术在数据交互查询,工作负载优化和集群的高可扩展性与高容错性上起到了重要作用,Spark 集群不仅提供了复杂查询与计算的优化,还可利用集群特性进行大数据并行计算,提升运算效率。

2.2.1 Spark 架构

Spark 集群架构为了满足分布式计算特性,采用主从架构模型(Master-Worker),由主节点(Master)与从节点(Worker)的组织方式。主节点主要负责集群的控制,负责维护集群的任务有序执行。从节点是工作节点,主要负责集群的计算工作,同时也需要定期向主节点返回集群状态并执行主节点的命令。Spark 集群还提供客户端访问界面,方便用户提交应用任务,该任务有执行器(Driver)负责执行。Spark 的软件架构图 2-4 所示。

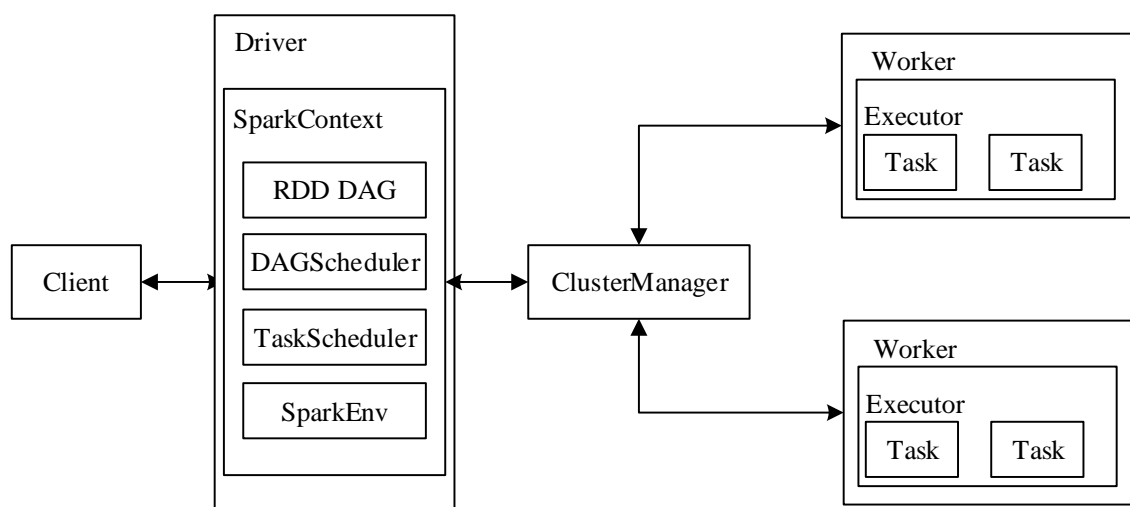


图 2-4 Spark 软件架构^[35]

Spark 架构中的基本组件:

Cluster Manager: 即是集群管理器,在 YARN 模式中,其主要对资源进行管理。在单节点模式中,其作为主节点,对整个集群的运行进行控制管理。

Worker: 负责计算任务的执行,管理线程执行器和 Driver 执行器。

Driver: 驱动程序是运行应用程序的 main()函数的中心点。它也是 Spark Shell(Scala, Python 和 R)的入口。Spark Context 在这里构建。驱动程序组件负责将 Spark 用户代码转换为在集群上实现的实际 Spark 作业,在 Spark 集群的主节点上运行的驱动程序有助于计划作业的执行。

Executor: 负责管理线程池工作。每个应用有自己的线程池执行器,专职数据处理相关任务。

SparkContext: 用于保存生命周期内应用的上下文。

RDD: Spark 的计算核心,弹性分布数据集。

在完成 Spark 集群的环境部署后,需要先启动主节点进程,然后再启动从节点的进程。Spark 应用在执行时需要使用到驱动者和工作者,其中任务的调度和逻辑

的执行由 Driver，复用器主要用来管理计算节点，为并行任务创建执行器。驱动者通过向 Worker 发送消息获得 Worker 节点的状态信息，并根据这些信息将工作负载分配给 Worker。Worker 使用自己的存储文件系统来完成与计算资源的通信。在任务执行时，执行器会把任务及其依赖文件序列化，这样做的目的，是便于进行传输给 Worker 节点的同时，执行器可以处理相应的数据分区任务。

2.2.2 Spark 核心组件

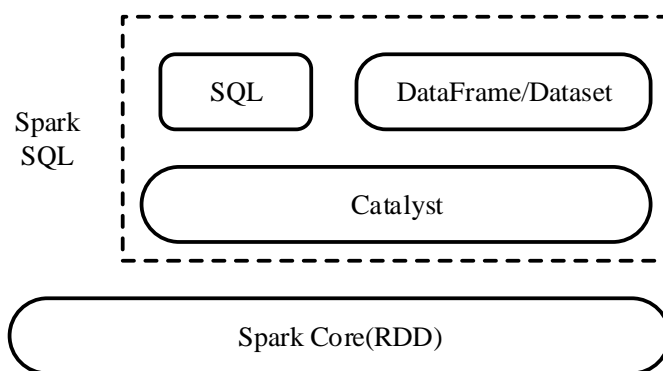
Spark 的基础核心组件被称为 Spark Core，其主要为弹性分布式数据集计算提供了极大的支持^[36]。Spark 核心的主要基础功能还有作业调度与内存交互等，在基础功能之外，其还提供了 RDD 的 API 定义和操作。Spark 生态有一组完善的生态应用链路，在 Spark Core 的基础上，有针对不同工作的处理组件，主要有 Spark SQL、Spark Streaming，MLlib 和 GraphX^[37]。

2.2.2.1 SparkSQL

Apache Hive 为 Spark 的查询操作提供了一种交互的 API^[37]，允许在 Spark 环境中对数据库进行 SQL 操作，该 API 可以把数据库表抽象成一个弹性分布数据集，将 SQL 操作转到 Spark 上面执行。SparkSQL 支持诸如 Parquet，Hive 表和 JSON 等格式数据源^[38]。

SparkSQL 可以运用 Hive 的查询语句，通过 SparkSQL 读取 Hive 的配置信息，采集 Hive 的元数据库的信息。SparkSQL 有了 Hive 数据仓库的元数据信息后就可以得到 Hive 中的所有存储的数据，然后使用 SparkSQL 来操作 Hive 表中的数据。HiveSQL 把要处理的作业转变成 MapReduce 的工作，然后发布到集群上完成，这中间会有很多磁盘读写的操作。SparkSQL 底层使用的还是 RDD，所有的操作都是基于内存的，实时性很高。Spark SQL 语言为 Spark 查询操作提供了丰富的接口和扩展性，其还可以将多种程序语言与 SQL 语句进行融合，便于 Spark 程序的开发。

SparkSQL 下层使用的是 Spark Core，处于 Spark Core 上层的 Catalyst 是一个优化器，主要负责组织 SQL 的语句优化，可以帮助优化查询，提高 SQL 查询效率。在 Catalyst 之上还有两个组件 SQL 和 DataFrame Dataset，支持多种类型的数据源。Spark 应用程序支持 SQL 查询和数据挖掘等功能，如下图 2-5 是 SparkSQL 的架构图。

图 2-5 SparkSQL 架构^[39]

2.2.2.2 Spark Streaming

Spark Streaming 是 Spark 的重要组件，提供了适合不同业务场景的接口，能够有效地实时的数据流进行处理计算。这些接口与 Spark 的核心组件提供的功能基本一致，使得对于它的操作逻辑更加清晰。Spark Streaming 有良好的容错性，吞吐量以及可伸缩性，在应用市场上有很大的需求。Spark Streaming 还可以让程序对实时数据流进行加工处理。

2.2.2.3 MLlib

MLlib 提供了机器学习中需要用到的算法，它可以基于 Spark 实现。它包括了分类，回归和协调过滤等常见算法。MLlib 有对模型进行评价和数据预处理等功能。所有这些方法的设计都是为了提高集群扩展性。

2.2.2.4 GraphX

GraphX 是针对图形计算的组件。它拥有处理弹性分布式图的工作能力，并支持统一图和表的可视化，对于图数据提供很多处理方法。

上述这些 Spark 生态链中的组件都以 JAR 包的形式提供给开发者，这让操作更加便捷。相较于 MapReduce 上的 Hive 和 Pig 等大数据组件不同，Spark 不用进行复杂的部署和管理操作，开发者只需部署好 Spark 环境便可以直接使用这些组件，极大地节省了系统开发和管理成本。把这些组件有机的结合在一起，就可满足同时对大数据进行流式处理、图数据处理，数据分析和交互式查询等功能，形成一站式解决方案。

2.3 元数据管理

本小节主要介绍元数据管理系统的元数据管理策略，元数据模型和元数据分

析的理论基础部分。

元数据是描述数据的数据，从数据源中抽取出来的用于说明其特征、内容的结构化的数据，元数据会对数据的生产到消亡作全程进行描述^[40]。元数据管理主要负责对实际的数据进行元数据采集，建立标准和模型，维护元数据，进行元数据分析，为外部提供访问接口的管理服务。它是对元数据的生成、保存，聚合和使用的全套环节管理，可以支持基于元数据的有关需求和应用。它可以让开发和业务人员快捷地解析数据的上下游关系以及元数据本身的含义，准确定位需要查询的数据，减少数据分析的时间成本，提高数据使用效率。

2.3.1 元数据管理策略

在集群环境中，对于元数据的管理有以下三种部署方式：集中式管理，分布式管理和无元数据管理^[41]。

集中式管理是将元数据的管理任务交给某一个节点，该管理节点负责所有的文件请求，且元数据信息的管理保存操作也由该节点负责，元数据保存在其他节点存储设备中。这种方式应用广泛，有很多集群文件系统使用到了。这种方式的优点如下：能够保证系统的可靠性和可用性；能有效地控制文件传输过程中的错误发生频率。但是它也存在着一些不足之处，主要体现在两个方面：一是扩展性差。集中式管理由于使用单一节点进行管理，其在一定范围内实现可操作的简单与便利性，但若该管理节点出现故障，那对集群将会是致命性的打击，所以其脆弱性是该方式最大的缺点。此外，由于单节点不能有效承受过多操作，所以易造成性能瓶颈。

分布式管理是指将把元数据存放在每一个系统节点上，系统的任意一个节点都可以对数据进行备份。每个节点对元数据的存放有相同的权限。这种管理方式的优点是解决了集中式管理的可用性不高的问题，而且在频繁操作元数据的情况下，性能也不会受到影响。它的缺点是不易于实现，而且对于元数据一致性的实现带来很大的挑战。

无元数据管理方式是借助其他方式对元数据操作，不使用专门的策略对元数据进行统一管理，其中最具代表性的就是 GlusterFS^[42]。

2.3.2 元数据模型

元数据模型又称作元模型^[43]，是对元数据的进一步抽象，是元数据的规范。在元数据管理模型中，CWM(Common Warehouse Metamodel)是一种理论规范，它是基于 OMG(Object Management Group)提出的三个建模标准 UML, XML 和 MOF 设计的，这三个标准是元数据库设体系结构的核心^[44]，实现了对元数据的建模，转换和

读写等操作，为元数据模型的创建和管理提供有力的支持。CWM 主要根据数据仓库和其他关系型数据库之间的元数据交换需求设计的一个规范，它满足在不同的数据结构环境下，实现在不同的数据仓库平台和元数据库之间进行元数据交换^[44]。这套标准主要包括 CWM 元模型、CWM XML，CWM DTD 和 CWM IDL 四个方面内容。

2.3.3 元数据分析

元数据分析是对于目标元数据间关系和元数据属性进行分析，元数据分析主要分为统计分析、特征分析，影响分析与血缘分析等^[43]。通过元数据分析可以帮助数据管理人员识别元数据价值，提升系统数据可信度，为各个部门的数据汇聚提供有力的支持，帮助高校各个业务部门人员理清楚数据关系。

血缘分析在元数据分析最为重要，血缘分析主要是对元数据进行溯源，能有效反映出数据之间的关系，便于定位数据错误，分析数据来源和提升数据质量。从数据转换的角度讲，数据 a 通过 ETL 过程处理得到了数据 b，就可以判定数据 a 与 b 拥有血缘关系，数据 a 是关于数据 b 产生的上游数据，数据 b 是关于数据 a 的应用产生的下游数据。按血缘对象来分表级血缘和字段（列）级血缘。

影响分析是基于当前元数据罗列出下游元数据，影响分析图由分析对象与其子代构成，可从影响分析图得知改变当前元数据对下游元数据所造成的影响。

2.4 Hive 数据仓库

Hive 是基于 Hadoop 构建的数据仓库分析系统，拥有丰富的 SQL 查询方式，把结构化数据转换成数据关系表，通过自带的 Hive SQL 语言执行 MapReduce 程序执行查询功能^[45]。Hive 不适合用于联机(Online)事务处理，也不提供实时查询功能，其主要特点为：可伸缩、可扩展，容错和输入格式的松散耦合^[46]。从元数据存储，数据存储和数据交换三个方面对该数据库进行介绍：

1. 元数据存储

Hive 将元数据存储的关系型数据库管理系统中，可使用三种方式与该数据库系统进行链接：多用户模式，内嵌模式和远程模式。内嵌模式主要用于单元测试中，只有一个会话链接；多用户模式将数据放入 MySQL 数据库中，便于多用户读写；远程模式是通过远程链接进行访问读写数据。

2. 数据存储

由于 Hive 对于数据存储格式没有特定的要求，所以其数据存储非常自由，在创建 Hive 表时，需要根据不同的数据类型描述好 Hive 数据中的列分隔符和行分

隔符，这样方便后续对于数据的存储与解析操作。

3. 数据交换

数据交换是关于 Hive 的数据进行导入导出的操作。因为 Hive 数据库表有外表的存在，不允许直接采用 Hadoop 命令把数据仓库中的数据存储库目录文件获取到本地的备份方式，这可能会造成外表的丢失。如果数据是采用内表的方式存储，则可以直接用 Hadoop 命令获取，Hive 提供支持数据备份和恢复的工具。

2.5 Sqoop

Sqoop 是一款对传统关系型数据库和分布式文件系统之间进行数据交换的工具。比如可以将 MySQL 中的数据导入到分布式数据文件系统中，也可以将分布式文件系统的数据导出到 MySQL 中^[47]。

2.6 YARN

YARN 是一种资源协调框架工具，也称为资源管理器，其主要管理的是 Hadoop 集群的资源，实现集群中作业调度和资源管理功能。为避免数据跨集群移动^[48]，可以在 YARN 之上部署其他框架实现数据的底层存储。YARN 使用统一的资源调度服务，可以有效提升服务的调度效率，实现了集群资源共享和资源弹性收缩。

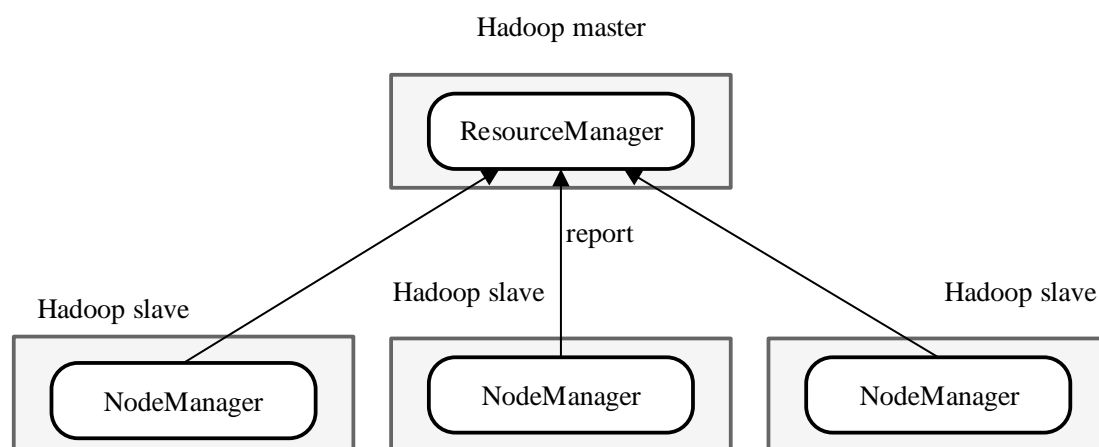


图 2-6 组件关系

YARN 主要由资源管理者(ResourceManager)、节点管理者(NodeManager)，应用主控者(ApplicationMaster)和容器(Container)组件构成^[49]。ResourceManager 和 NodeManager 的关系如上图 2-6 所示。

1. ResourceManager: 作为 YARN 的主进程，它的主要任务决定如何分配集群里的可以使用的资源，处理创建容器的请求，并根据用户特点以及资源情况来决定

如何创建容器。Hadoop 集群一般拥有一个以上的 ResourceManager。

2. NodeManager: 作为 YARN 的从属进程。它主要参与到容器创建, 监听和销毁工作中去。可以服务于其他组件, 处理其他组件的创建容器的请求, 并向 ResourceManager 报告容器的状态。

2.7 本章小结

本章介绍了分布式文件系统 HDFS, Spark 计算框架相关技术以及元数据管理的基础理论知识, 第一节主要介绍了分布式文件存储系统, 第二节主要对 Apache Spark 计算框架进行介绍, 概述了 Spark 的架构以及核心组件。第三节介绍元数据管理的策略, 元数据模型和元数据分析的内容。第四节介绍了 Hive 数据仓库工作原理, 第五节介绍了关系型数据库和数据湖之间的传输工具 Sqoop。第六节介绍 YARN 集群任务调度器的工作原理。本章内容为基于 Spark 的元数据管理系统的设计提供了理论支持。

第三章 基于 Spark 的元数据管理系统的需求分析和总体设计

本章基于上一章对 Spark 集群技术以及元数据管理相关技术的理论研究,首先对高校数据中心的元数据管理系统的需求分析进行介绍,明确本系统的应用场景以及功能需求和非功能需求。然后分别从软件架构和功能层次架构对整个系统进行了介绍。最后对系统的功能设计,关键性技术和数据库设计进行介绍。

3.1 需求分析

随着教育信息化的不断深入,高校也在紧锣密鼓地建设自己的智慧化校园系统并搭建高校统一的数据中心。但是由于各个系统业务繁多和数据的不断累积,导致数据业务人员对数据的使用和管理都变得复杂。如何对数据中心数据进行规范化,流程化和自动化的管理,进行有效的数据治理,发掘高校数据资源的价值是个亟待解决的问题。元数据作为描述数据的数据,存在于数据的整个生命周期,参与了数据从生产到消亡的全过程。元数据管理为高校信息部门人员对数据的管理提供了便捷的服务。

虽然教育部印发了《基于教育教学资源的元数据规范》等元数据规范,这开启了高校数据治理的大门,但是高校数据治理发展缓慢。一方面由于早期各部门规划和技术的局限,各部门管理系统处于割裂的状态,数据标准不相同,数据难以统一归集。另一方面由于传统的元数据管理支持的数据源单一,系统部署繁琐,不能完全解决高校数据中心不规范和数据处理过程费时费力的问题。这些都是高校推进数据治理亟需解决的问题。大批高校信息部门为解决现存的数据孤岛问题,都开始运用大数据技术建设高校统一的数据中心,在当下数据中心建设之际,本论文提出了基于大数据平台上的元数据管理方案。

该方案主要为高校数据中心提供元数据管理服务。在海量数据的情况下,基于 Spark 的元数据管理系统可以帮助系统使用者快速查找数据,明确业务数据所属的功能模块和相关的存储信息。相较于传统的数据管理,即使只知道数据的一个元数据描述,也可以实现快速定位并了解业务数据概况,提高了数据管理的效率。另外,当业务数据出现问题时,可以对问题数据进行元数据分析,理清数据上下游关系,推断问题出现哪个环节,辅助定位问题数据位置。元数据管理主要包括元数据采集、元数据模型、元数据基础功能,元数据分析应用和元数据质量检测。元数据管理提高了数据处理效率,降低了运行和维护成本,通过查询了解记录数据的生成和消亡,实现数据存储和共享,为高校决策分析提供数据支持。元数据管理是高校数据资产

的基础组成部分,在数据仓库系统中,它记录了数据仓库中各类数据的模型标准、数据层次关系,数据处理流程信息,描述各层数据的状态和 ETL 运行过程等。元数据可以帮助数据仓库管理员和数据分析师追溯数据之间的关系,用于指导系统管理员进行数据管理和开发,提高数据中心的效率。下面将从功能性和非功能性两个方面对高校元数据管理的需求分析进行描述,系统按照软件工程标准流程进行设计。

3.1.1 功能需求

基于 Spark 开发的元数据管理平台支持对高校数据中心数据进行统一管理,对现有散、重,乱的数据进行统一梳理,并制定全校的数据标准,对表的名称、表描述、字段格式、存储介质、大小,清洗策略和更新频率等元数据进行规范化管理,保证各系统之间的数据一致性和业务系统的稳定运行。系统主要包含六大功能模块:用户登录模块、元数据采集模块、元数据模型模块、元数基础功能模块,元数据分析模块和元数据质量检测模块。系统主要的使用对象是数据分析人员和数据管理人员。针对以上分析,基于 Spark 的元数据管理系统的用例图如图 3-1 所示。

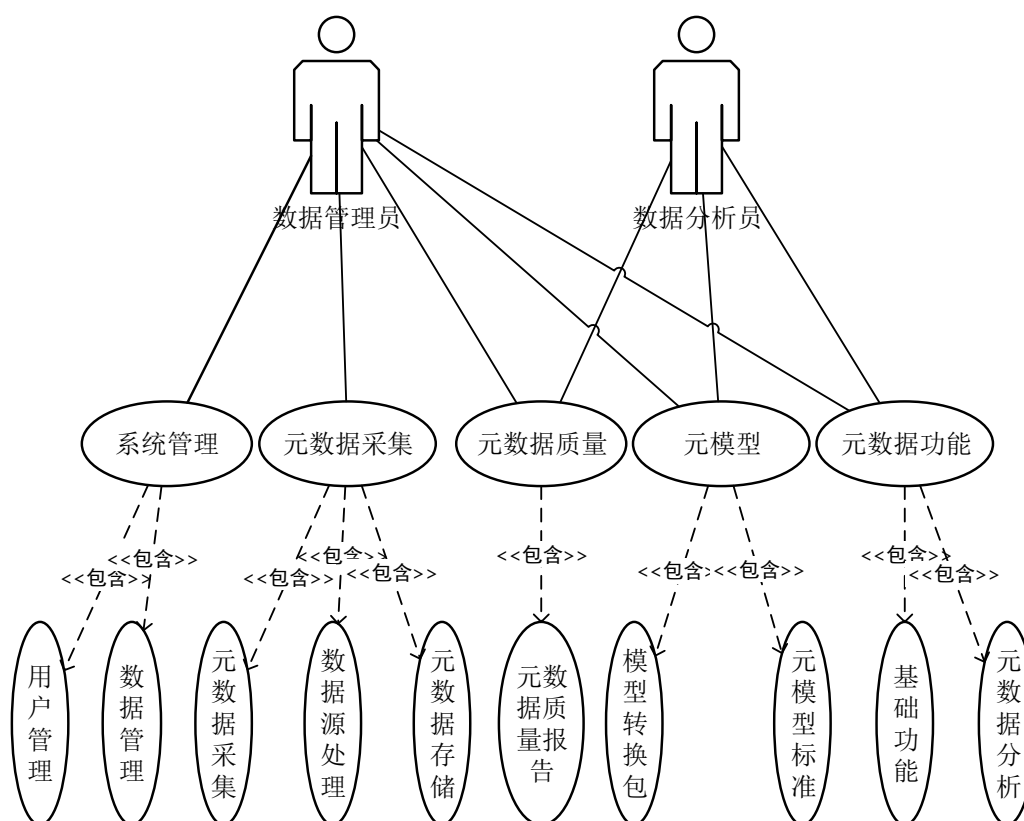


图 3-1 系统用例图

数据管理人员：数据管理人员是数据中心的开发和维护人员，也可以是数据仓库管理员，主要负责对用户登录权限、数据使用权限、数据维护、元数据管理，系统的正常运行和数据正常使用的进行管理。因此数据管理人员要负责系统的管理，元数据的采集，参与元数据模型建立和变更，元数据质量检测等任务。

数据分析人员：数据分析人员是通过数据分析工具对高校数据价值发掘的重要人员，根据数据分析方案对数据进行统计分析，为高校数据挖掘提供基础数据分析。因为数据分析人员主要操作对象是业务数据，元数据管理是为了提高业务工作的效率，所以，数据分析员可以对元数据基础功能、元数据分析，元数据模型和元数据质量模块进行操作。

系统管理：系统管理的使用人员主要是数据管理员。首先是对高校数据仓库人员和数据分析人员进行权限管理，不同的用户拥有不同的使用权限，根据权限使用相对应的功能，数据管理员可以对非法用户进行权限控制。其次是对于数据的管理，这里的数据管理是指对高校数据中心中的数据源进行存储管理，进行数据存储加工，了解高校各个部门数据存储情况，便于高校数据业务人员在合适的时机进行数据分析。

元数据采集：元数据采集的使用人员是数据管理员，采集人员通过选取对应数据源，以及完成数据的抽取，转换，加工操作，让存储在 HDFS 中的数据加载到元数据库中，完成对元数据的初步采集。

元数据模型：元模型的使用者主要是数据管理员和数据分析员，元数据模型负责数据仓库与高校信息化系统之间的元数据交换，涉及对数据进行分类存储，根据不同类型的元数据选择不同的元数据模型进行添加。

元数据质量：元数据质量的使用者有数据管理员和数据分析员，使用者可以根据元数据质量规则对系统中元数据进行质量检查，掌握现存系统中元数据质量概况，再根据需要导出元数据质量报告。

元数据功能：元数据功能的使用者有数据管理员和数据分析员，元数据功能包括元数据的变更和查询功能以及元数据分析功能。使用者在对元数据进行变更或者对元数据权重分析的时候，需要考虑元数据之间的关系，这时候就需要对元数据溯源和追踪，即是对系统中元数据进行血缘分析和影响分析。

3.1.2 非功能需求

实时性：Spark 可以通过内存计算高效地处理数据流，将整个系统划分为多个线程，每个线程负责一个数据访问任务，通过消息传递机制来进行交互。Spark 相较于 MapReduce，它的内存实时运算效率是 MapReduce 的 100 多倍。

稳定性：元数据管理可以提高数据库的数据的使用规范性和稳定性，降低高校信息系统的运行出错率。当有数据新增，变更或者删除时，系统有对应的元数据来确保各系统数据之间的一致性，为数据稳定的服务提供支撑。

可扩展性：若日后有其他业务需求，系统可相应增加服务器实现扩展，也可以根据业务需求增加元数据模型。

可靠性：在数据存储方面 HDFS 有较高的容错率，保证大数据平台即使在访问量激增的情况下也可以正常使用数据。元数据管理可以保证高校数据中心数据使用的正确性和规范性，不会出现高校各部门数据定义不一致的情况。集群之间的资源调度也可以确保即使系统出现节点异常，系统也可以正常运行，不会出现宕机的情况。

3.2 应用场景分析

随着大数据技术的发展，高校的应用系统每天都会产生大量的数据，面对海量的数据，数据库管理人员很容易束手无策。传统的数据管理方法效率较低。在对高校数据中心数据进行管理时，通常会遇到如下几个应用场景需求。

- 1, 在高校数据量成倍增长的情况下，如何对高校数据中心数据进行统一的数据资产目录管理。元数据作为描述数据的数据，是高校数据资产的基础组成部分，在业务场景中，它描述了业务实体数据的各个维度。在数据仓库系统中，它记录了数据仓库中各类数据的模型标准、数据层次关系，描述各层数据的状态和 ETL 运行过程等。元数据可以帮助数据仓库管理员和数据分析师追溯数据之间的关系，用于指导系统管理员进行数据管理和开发，提高数据中心的效率。
- 2, 高校业务人员如何在海量数据中快速找到业务数据信息。基于 Spark 的元数据管理系统可以帮助系统使用人员快速查找数据，明确业务数据所属的功能模块和它的详细信息，相较于传统的元数据管理，即使只知道数据的一个元数据描述，也可以实现快速定位并了解业务数据概况，这提高了数据管理的效率。
- 3, 当业务数据出现问题时，如何对问题数据进行排查定位。元数据管理提供元数据分析的功能，可以通过血缘分析和影响分析理清数据上下游关系，理清问题出现在哪个环节，辅助定位问题数据位置。元数据分析也提供分析数据流向，预估数据变更影响的功能。
- 4, 由于当下高校各部门数据存在数据孤岛的情况，如何消除各部门人员对数据的理解偏差，加强部门间的业务合作也是亟待解决的一个问题。元数据

管理让不懂技术的业务人员不需要理解底层数据逻辑也可以快速明白数据的含义，让不懂业务的数据管理人员也可以轻松地理解数据之间的业务联系，促进了技术与业务之间的合作。

3.3 系统架构设计

本论文设计并研发的基于 Spark 的元数据管理系统，具有数据源层、数据存储层，集群层和 Web 访问层。首先，将各部门的原始数据源导入系统，再通过 ETL 工具把原始数据导入 Hive，使用 Hive 作为数据仓库为上层应用提供数据存储服务。集群层主要负责运算和调度资源。Web 访问层主要负责发送业务处理请求和参数传递。

3.3.1 系统架构

基于 Spark 的元数据管理系统搭建在 Spark 集群上，HDFS 分布式文件系统负责存储原始数据，Sqoop 工具负责数据的抽取，转换和加载。Hive 作为数据仓库，MySQL 负责存储元数据，Spark 负责查询和计算，整个系统从下往上划分为数据源层、数据采集层、数据存储层、计算层、服务层，接口层以及功能展示层。

如图 3-2 是系统架构图。

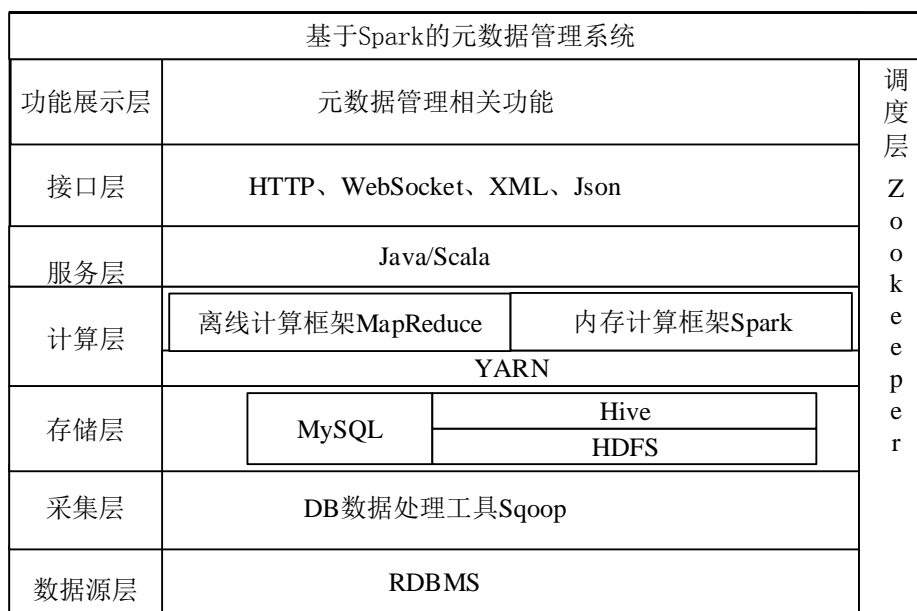


图 3-2 系统架构图

数据源层是需要进行管理的数据源，主要是关于高校各部门系统产生的关系型数据。

采集层主要是使用 Sqoop 工具对学校各部门系统中数据进行采集, 统一归集到大数据平台的数据中心去, 构建数据仓库, 由 Hive 来进行管理。MySQL 主要负责元数据信息的存储管理。

计算层主要是完成数据加工处理相关的工作, 在采集数据时, 需要使用到 MapReduce 执行导入任务, 把数据导入到 Hive 数据仓库。在进行数据分析时需要使用到 SparkSQL, 对数据挖掘需要使用到 SparkMlib 等相关组件。计算层主要负责处理任务请求。

服务层和接口层主要负责处理数据请求以及数据处理结果传输, 为展示层提供接口支撑, 主要有关于传输的协议 HTTP 和 WebSocket 以及关于数据的 XML 和 JSON 协议。

展示层是关于元数据管理系统功能相关的内容, 如下图 3-3 所示, 元数据管理系统在数据源管理模块中可以对数据源进行添加配置, 然后是对元数据模型的设计, 有元模型的设计基础后就可以对元数据进行采集、基础查询变更, 元数据分析和质量检测。

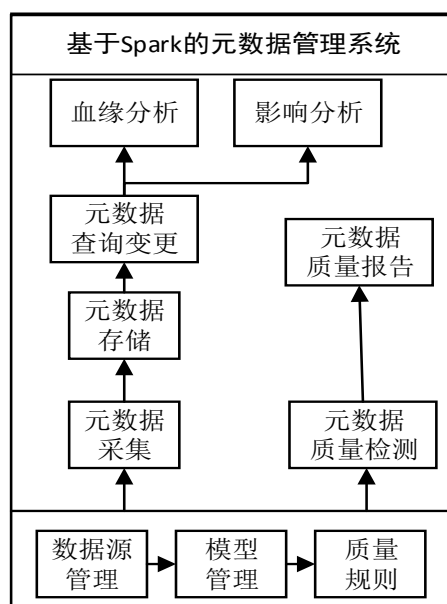


图 3-3 功能展示图

在元数据管理模块中, 为了提高数据处理和分析人员的作业能力, 系统可以为数据管理员提供可视化界面对元数据进行查询和变更的操作, 以及对元数据进行血缘分析和影响分析, 帮助高校数据管理员和数据业务员更好地追溯元数据之间的关系。

3.3.2 功能层次架构

本系统功能划分成四个层次，包括元数据采集，存储，功能和扩展应用层。如图 3-4 所示，展示了每个功能层次之间的关系和它所包含的内容。

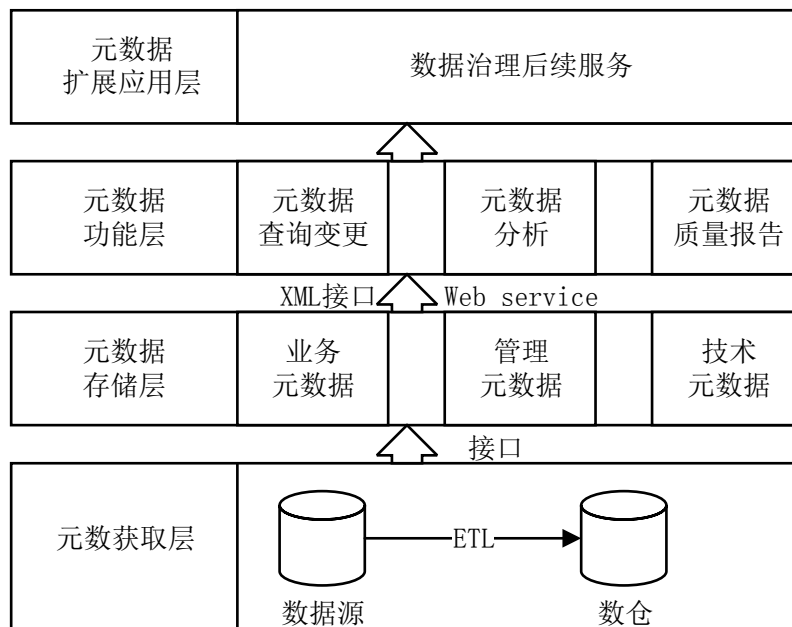


图 3-4 层次架构图

首先根据业务需求，高校的信息系统数据每天都会增加，这些新增的数据需要更新到 HDFS 文件系统中去，可以通过 ETL 工具把数据源导入到 HDFS，再通过数据仓库 Hive 进行管理。所以可以通过配置 ETL 工具任务对系统数据进行定时增量导入。

然后在元数据存储层，根据高校应用系统数据的类型选择对应的元数据模型进行存储管理。元数据存储层将元数据分为三类，其中有业务元数据，它主要描述各个业务的意义，系统的数据字典和业务规则，比如涉及学生基本信息和教学相关信息的数据库的数据字典就是一类业务元数据。业务元数据是在数据使用层面让人们的数据有统一的理解和认识，消除数据偏差，让不清楚数据底层的业务人员快速掌握数据的含义。管理元数据主要是描述数据的权限和归属信息，涉及元数据的安全基础管理。技术元数据是对数据整个获取、清洗、转换，加载和使用的过程进行描述，让数据使用者清楚数据来源和处理过程。技术元数据主要包括数据库的存储信息，表字段的长度和 ETL 过程信息等，比如本系统中的学生表的学号字段精度，类型以及是否是主键的描述。这些对字段的描述都属于技术元数据。

最后，在元数据功能层，元数据进行查询可以帮助高校业务人员快速地掌握数据信息，进行数据检索和数据定位等。元数据分析包括血缘分析和影响分析，帮助

系统使用者追溯数据来源,判断数据的影响力。元数据质量检测可以让系统使用者对数据资产进行全面的评估。元数据扩展应用层涉及到其他的数据治理服务,这里不作阐述。

下图 3-5 是基于 Spark 的元数据管理系统的工作流程设计。

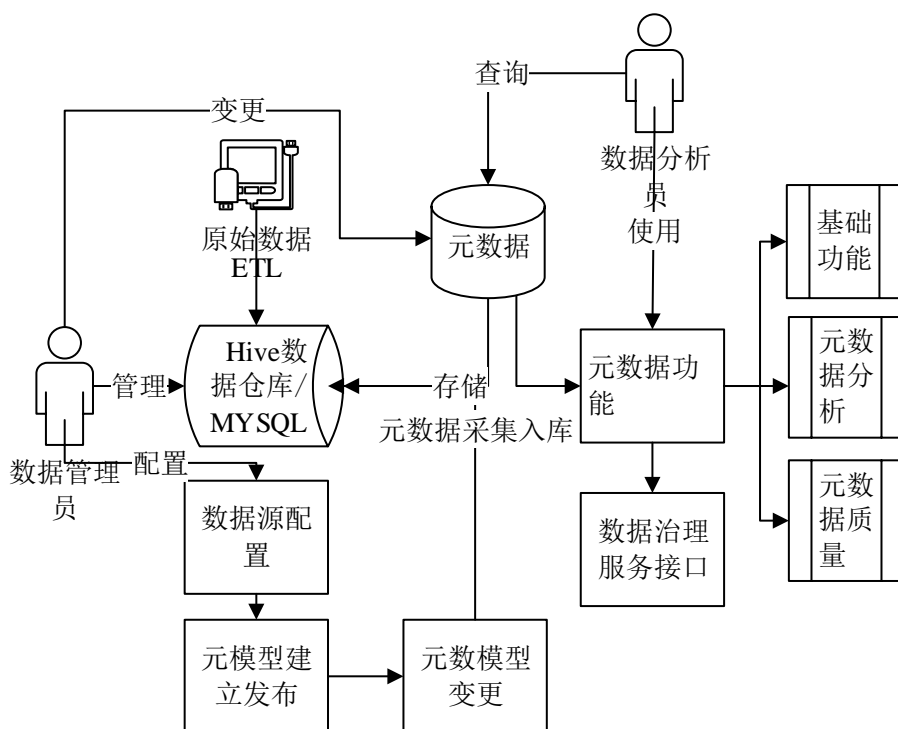


图 3-5 工作流程图

高校信息系统数据经过 ETL 工具进入 Hive 数据仓库,数据管理员会配置数据源,根据高校元数据标准进行元数据模型建立和发布,再对元数据进行采集入库,按照元数据分类来对元数据进行存储管理。数据业务人员针对不同的业务需求,查询元数据,使用元数据分析和元数据质量检测功能解决高校系统业务上的问题。

3.4 系统功能设计

本系统主要的功能是为高校数据的相关工作人员提供元数据采集、元数据模型、元数据分析,元数据质量检测 and 元数据基础服务。在高校系统业务繁多和数据量巨大的情况下,需要使用专门的采集工具对高校各部门数据进行汇聚。数据中心的数据量巨大,需要对数据进行分类管理,以及按照教育元数据标准进行元数据模型定义,便于后续的元数据采集和元数据质量检测。元数据的基本功能是为不懂数据底层存储逻辑的业务人员提供元数据查询和变更操作,业务人员可以通过查询数据的元数据清楚地知道每个数据的含义,归属以及数据间的关联信息,方便定位

目标数据和排查错误数据。元数据管理系统还支持高校数据业务人员对系统元数据进行质量检测,产生数据质量报告,显示各信息部门数据质量健康情况。系统为高校的数据分析员提供元数据分析的功能,比如当系统中出现错误数据时,可以通过元数据血缘分析功能对数据的关系追溯提供全方位支持。另外如果需要对数据进行变更,删除或者权重分析,可以通过元数据影响分析来查看该元数据对下游数据的影响。基于 Spark 的元数据管理为高校数据治理和大数据挖掘奠定了坚实的数据管理基础。

系统的功能结构如下图 3-6 所示。

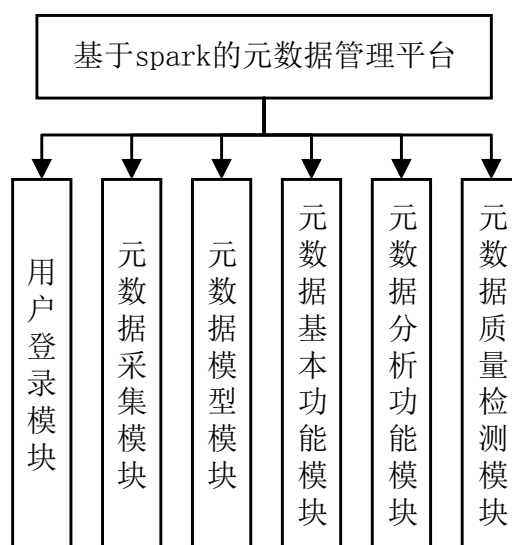


图 3-6 系统功能图

3.4.1 用户登录模块

元数据管理系统因为涉及高校数据中心的数据处理,关系到系统的安全使用,所以要求对系统使用者进行登录权限验证,确保系统数据的安全性。在需求分析中提出系统的使用者只能是高校数据中心的数据管理员和授权的数据业务人员,也就是数据仓库管理员、数据库管理员,数据分析业务人员和数据挖掘人员。前端和后端都会对用户进行登录权限验证确保系统的安全使用,只有登录成功才可进入系统首页使用系统的功能。

用户的登录流程设计如图 3-7 所示。

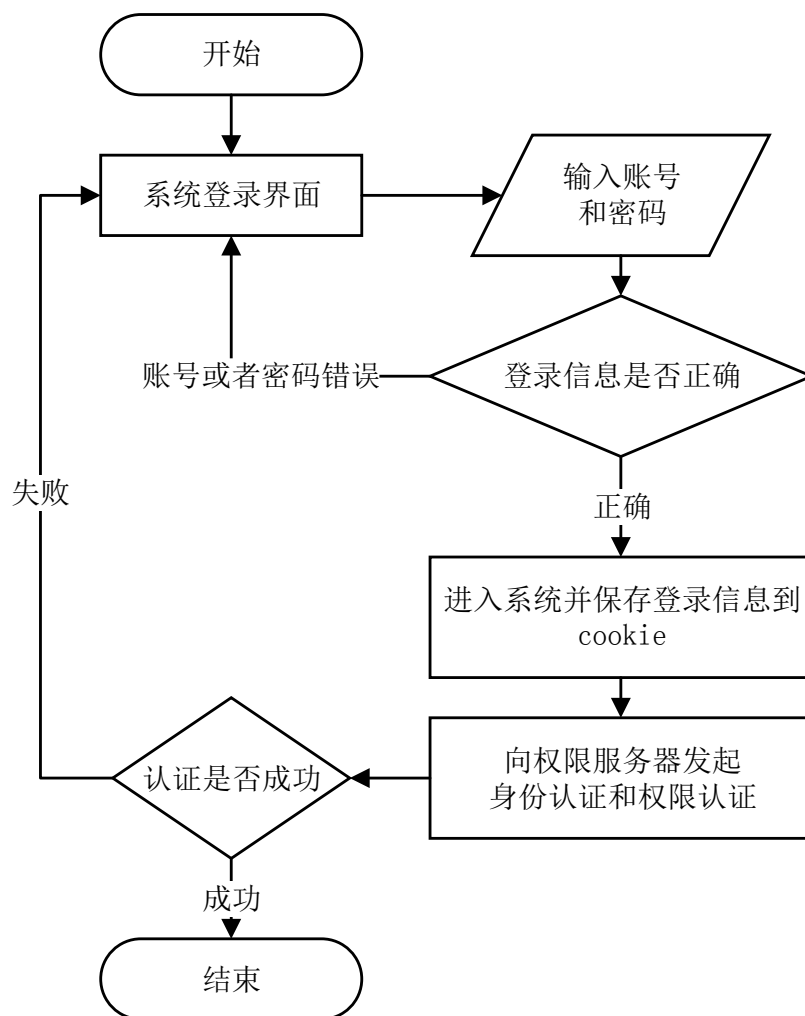


图 3-7 登录流程设计

3.4.2 元数据采集模块

3.4.2.1 数据源处理

在数据源选取时,需要对数据源所在的高校应用系统数据库进行链接配置,配置信息根据数据库类型和链接方式的不同来设计,不同的数据源需要的数据库信息和约束条件各不相同。

如表 3-1 所示为 Oracle 数据库的配置信息及数据约束条件设计,MySQL 数据源数据库配置信息没有展示,在第四章实现部分会有呈现。

表 3-1 数据库配置

数据库类型	链接方式	数据库配置信息	数据约束条件
ORACLE	Native JDBC	主机名	必填项；IP 格式；
		数据库名	必填项； 字符长度不超过 50；
		数据表空间	非必填项； 字符长度不超过 50；
		索引表空间	非必填项； 字符长度不超过 50；
		端口号	必填项； 0~65536；
		用户名	必填项； 字符长度不超过 50；
		密码	必填项； 字符长度不超过 50；
	ODBC	数据库名	必填项； 字符长度不超过 50；
		用户名	必填项； 字符长度不超过 50；
		密码	必填项； 字符长度不超过 50；
	OCI	数据库名	必填项； 字符长度不超过 50；
		数据表空间	非必填项； 字符长度不超过 50；
		索引表空间	非必填项； 字符长度不超过 50；
		用户名	必填项； 字符长度不超过 50；
		密码	必填项； 字符长度不超过 50；
	JNDI	数据库名	必填项；字符长度不超过 50；

在获取数据源后，系统还需要对数据进行 ETL 加工，再导入到数据仓库去，使用 HDFS 存储，生成对应的 Hive 表便于数据仓库管理。ETL 加工是把原始数据源转换成目标数据源的处理过程，整合高校各个应用系统间分散，不规范和不统一的数据，并统一归集到高校数据中心进行存储。

如图 3-8 为数据源导入到 HDFS 的执行流程设计。

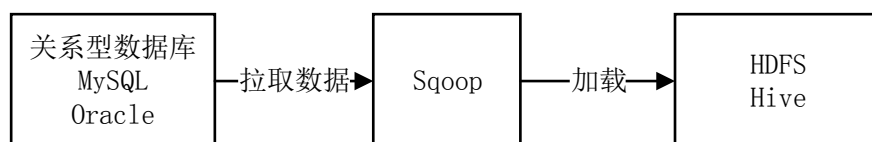


图 3-8 执行流程

3.4.2.2 元数据采集

元数据的采集是指从高校各应用系统数据库中，把数据的元数据信息通过手动或者自动的方式获取出来的过程。元数据采集主要有四个方面：创建数据源、任务配置，元数据获取和元数据入库。它的执行体一般分为采集任务发送端和采集任务处理端，在任务发布端需要选择数据源，制定采集任务和发送采集请求等。在任务处理端需要执行发布的采集任务，并对元数据进行采集，转换和存储等操作。HDFS 和 MySQL 负责实际的数据收集，元数据转换和元数据存储等。元数据采集的管理包括数据源的管理、采集的配置，采集方法的选择和入库审核的操作，如下图 3-9 所示为元数据的采集流程图。

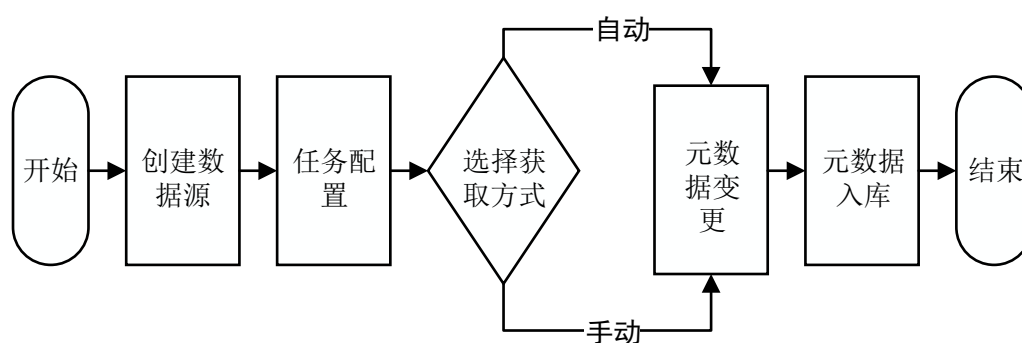


图 3-9 元数据采集流程图

创建数据源：首先选择添加采集的数据源，数据源的选择是元数据采集的首要的任务，然后配置数据库链接参数，数据源选择需要知道数据源的地址、端口、类型，用户名和密码等信息，然后链接上数据源所在的数据库并实施进行下一个采集步骤，数据源的选择可以通过一个接口来管理。

任务配置：根据选择的数据源的类型不同，选择对应的采集方式，并制定采集计划，选择何时按何种要求进行采集。

元数据获取：元数据获取有两种方式，一种是自动获取，另一种是手动获取。对于一些有接口实体的采集任务，可以使用 ETL 工具或者 SQL 脚本来执行采集任务，对于一些不好采集的元数据，只能用手动获取的方式。不管哪一种方式，都支持用 XMI 或者 EXCEL 文件的格式来写入元数据，然后再把这些文件提交到元数

据采集的任务处理端进行处理，如下图 3-10 所示。

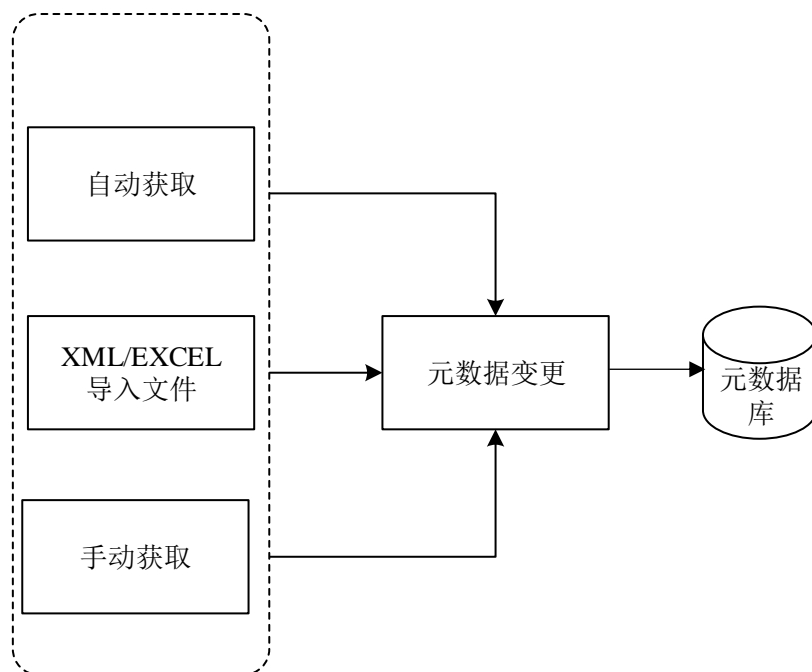


图 3-10 元数据获取

元数据入库：是指在获取到元数据后，元数据以 CWM 关系型模型的结构存储在元数据库中。

3.4.2.3 元数据存储

在高校应用系统的元数据存储中，元数据存储层将元数据分为三类，其中有业务元数据，它主要描述各个业务的意义，系统的数据字典和业务规则，比如涉及学生基本信息和教学相关信息的数据库的数据字典就是一类业务元数据，它有与之对应的业务数据实体。业务元数据是在数据使用层面让人们的数据有统一的理解和认识，消除数据偏差，让不清楚数据底层的业务人员快速掌握数据的含义。管理元数据主要是描述数据的权限和归属信息，涉及元数据的安全基础管理。技术元数据是对数据整个获取、清洗、转换，加载和使用的过程进行描述，让数据使用者清楚数据来源和处理过程。技术元数据主要包括数据库的存储信息，表字段的长度和 ETL 过程信息等，比如本系统中的学生表的学号字段精度，类型以及是否是主键。这些对字段的描述都属于技术元数据。

3.4.3 元数据模型模块

元模型是对元数据的描述，元模型规定了高校数据中心各类元数据的组织形

式和元数据本身的结构，是元数据管理建设的基础。元模型对于元数据可以从元数据规范、元数据关系，内部对象和元数据操作等多方面进行描述。

3.4.3.1 元数据模型建模规范

高校信息系统的数据通常以关系型数据结构存储的，CWM 规范可以把关系型数据的元模型转换成有对应关系的实体关系模型。基于 CWM 规范的元模型到 ER 模型再到实体关系模型的转换过程如图 3-11 所示。

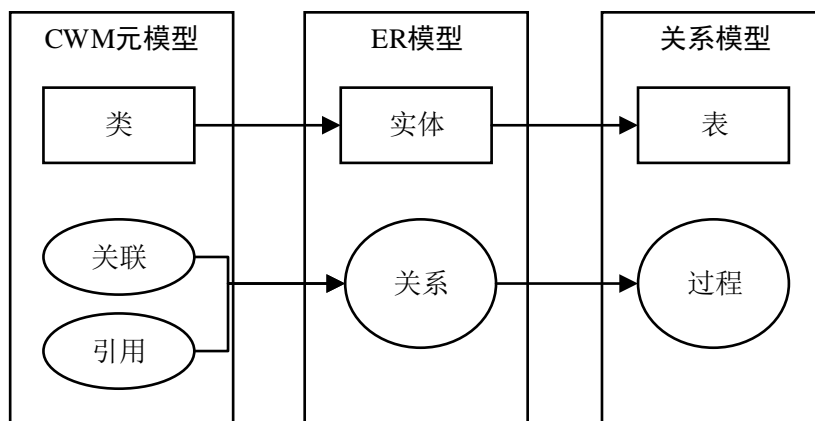


图 3-11 元模型生成图

从基于 CWM 规范的元模型转换成关系模型需要两个步骤。首先，按照面向对象的内容到关系内容进行映射，将 CWM 元数据模型转换为实体相关 ER 模型，其中包含类到实体的映射，关联和引用到关系的映射。然后是按照实例的 ER 实体联系模型的内容到关系模型的映射。CWM 元数据模型到关系模型的转换，包含实体到关系表以及实体关系到过程的映射。

基于 CWM 规范的元模型设计需要满足以下要求：

开放性：使用 CWM 元数据模型作为基础的元数据模型，支持高校各类信息系统进行相互操作数据。

适用性：元模型可在 CWM 元模型基础上进行业务需求扩展，满足对高校不同业务需求的元数据模型新增，适应不同的业务需求，这是基于 CWM 元模型规范的元数据管理系统的特点。

3.4.3.2 CWM 元模型转换包

元模型是对元数据的描述，元模型规定了各类元数据的组织形式和元数据本身的结构，是元数据管理建设的基础。元模型针对元数据库可以从元数据规范、元数据关系，内部对象和元数据操作等多方面进行描述。元模型管理需满足对包，数

据类型和类关系三种要素设计。转换包主要包含每个类关系和数据类型的分组，每个类关系和数据类型只能属于一个包。转换包支持嵌套，允许一个包里嵌套一个包，但是不支持多重嵌套。

在采集了高校各个系统的数据后，把这些元数据进行聚集到一起进行存储，然后根据不同类型的元数据来抽象成元模型的转换包，最后再根据系统的业务需求把元模型划分为三类，一种是基础教学元模型，一种是数据层元模型，另一种是数据转换层元模型。如下表 3-2 展示了基于 Spark 的元数据管理系统每个层的元模型与 CWM 元模型转换包的对应关系。

表 3-2 CWM 元模型和元数据管理系统元模型的对应关系

元数据管理系统元模型	CWM 元模型的包
基础教学元模型	对象模型包，业务信息包，数据类型包，表达式包，键和索引包，类型映射包，软件部署包
数据转换层元模型	数据转换包
数据层元模型	关系模型包，仓库信息包，仓库操作包

从某种角度来说，可以把这个对元数据抽象的过程看成对元模型设计的过程，有关元模型具体实现参考第四章的元数据模型模块的详细设计与实现。

3.4.4 元数据基础功能模块

基础功能模块拥有对元数据查询和变更的能力，通过输入关键信息查找系统中是否存在对应的元数据的数据实体，还拥有对元数据的计数的能力。元数据的查询，变更和计数功能可以以接口的形式向元数据管理系统提供服务。

3.4.4.1 元数据的查询和变更

元数据查询是指可以通过输入特定元数据信息，对表，字段和数据仓库加载数据过程的元数据信息进行模糊查询，并可以根据查询条件返回符合条件的元数据以及对应的实体业务数据。

元数据变更包括元数据修改和元数据版本修改两个部分。元数据修改会对元数据重新审核，审核通过才可入库。版本变更是对不同时间段的同一元数据进行版本管理，记录每一个修改的时间戳。

3.4.4.2 元数据计数

系统使用者可以对系统中不同分类的元数据做统计操作，让数据使用者对各种元数据类型的数目有清晰的了解，元数据的计数可以依照元数据的分类来进行

计算。

3.4.5 元数据分析模块

本节对基于 Spark 的元数据管理系统的元数据分析功能进行概要设计，本系统元数据分析主要包括血缘分析和影响分析。如图 3-12 这两种分析又可分为表级别和字段级别的两分析维度。

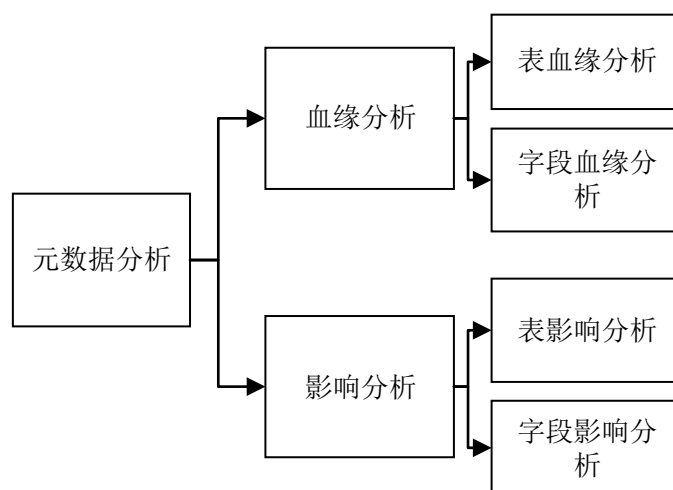


图 3-12 元数据分析维度

3.4.5.1 血缘分析

在高校应用系统中，血缘分析可以帮助数据业务人员定位问题数据的出处，优化数据处理流程，实现对数据的链路追踪。比如当学生选课信息中课程名称出现异常时，可以对选课信息表的课程描述元数据进行血缘分析，向上追溯数据源表的信息，再对相关的数据源表进行问题排查。

系统在对元数据进行血缘分析时，首先需要指定元数据，然后获取该元数据的所有前置元数据，然后为这些前置元数据进行广度优先遍历，获取每个前驱元数据，递归出口是所有元数据都达到数据源采集接口或该元数据没有相应前驱实体。对于不同种类的元数据，涉及的追溯过程不尽相同，比如数据仓库的元数据，会涉及到 ETL 处理流程；对于数据仓库管理元数据，不仅牵涉到 ETL 处理过程，还牵涉到整个仓库汇聚数据的过程。在高校各部门的应用系统中，业务人员希望能从众多的数据源中快速、方便地查找出自己所需要的信息，并将其存入到数据库中。血缘分析提供了这样一个功能，使用户能够清楚知晓元数据不同的处理过程，准确地了解每个流程的作用，需要哪些输入和产生的输出。

本文采用了一个基于 Spark 技术的血缘分析方法：由于 SQL 在 Spark 上执行会形成一个 DataFrame，这个 DataFrame 包含了逻辑计划，这个逻辑计划中包含了

SQL 的抽象语法树。通过对逻辑计划递归解析，便可以得到当前阶段所操作的输入表和输出表之间的关系。将整个关系连通起来，去除掉中间表，就可以得到输入表和输出表之间的血缘关系。血缘分析支持可视化的界面显示元数据之间的血缘关系。

如图 3-13(a)是 SparkSQL 中生成数据表的数据关系图，其中有临时表，把临时表从中剔除再转换形式，就可以得到图 3-13(b)的 tableF 的数据血缘关系。

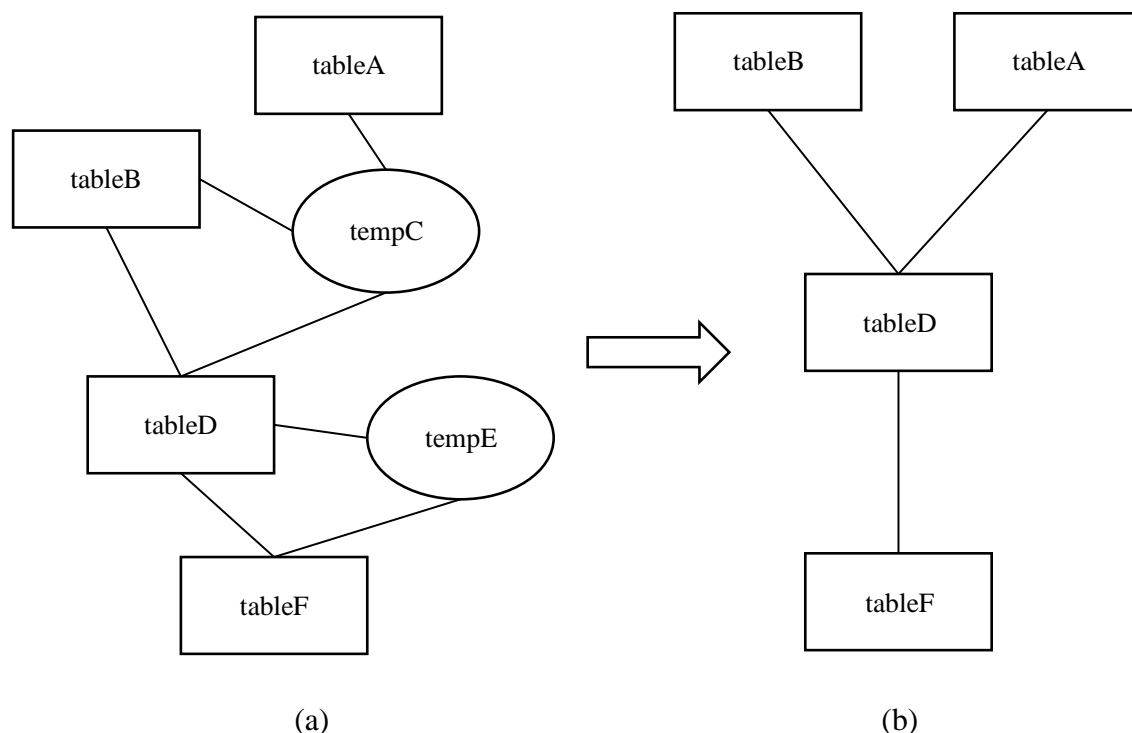


图 3-13 SparkSql 血缘关系解析图

3.4.5.2 影响分析

影响分析是指从一个元数据出发向下寻找依赖该元数据的其他元数据，查找所有依赖这个元数据的其他元数据。当高校业务人员需要修改某个元数据时，影响分析可以判断该元数据的变更会对哪些数据造成影响，帮助数据使用者掌握单个元数据的重要性。该功能主要分析每个元数据影响范围。影响分析能够反映本元数据与向下关联的元数据之间关系，确定该元数据对实体数据的影响范围。

用户在 Spark 平台上进行数据管理操作，会在 SparkSQL 的逻辑计划中生成抽象语法树。对元数据进行影响分析主要涉及对抽象语法树的遍历。

如图 3-14 是影响分析的流程设计。

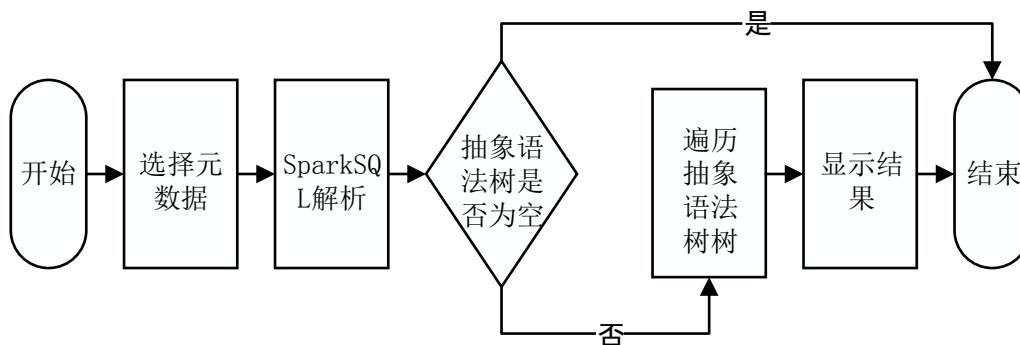


图 3-14 影响分析流程图

3.4.6 元数据质量检测模块

高校各类应用系统较多，业务繁杂，在数据中心建设过程中经常会忽视数据质量的重要性。如果没有采用合适的数据质量管理方法，伴随着问题数据的深入应用，会导致系统数据出现严重的问题，让系统的数据变得不可信，辅助决策失效等情况发生。让数据使用者不再相信系统数据的准确性，这样就失去了高校数据中心建设的意义。元数据质量检测模块具有检查元数据本身质量的功能，元数据质量是指在元数据库中存储的元数据总体健康情况的好坏，比如元数据的有效性，完备性和一致性等。

3.4.6.1 元数据质量规则

元数据质量规则包括元数据的填充完备率、一致性、唯一性，有效性和完整性，这些规则确保元数据可以正常准确地使用。下面是对数据质量规则五个维度的解释说明：

完备率：完备率是指数据的存在性，完备的意思是在系统中存在。比如，学生成绩数据的完备性检测就需要检测学生成绩数据是否符合数据标准，假设学生成绩数据表包含如下属性：学号、姓名、学院、学期、课程编号、课程名称，考核方式和成绩。如某课程有 100 个学生参加考试，需要检测是否有 100 条成绩记录，即是检测记录数是否完备。然后检测属性填充是否完备，即这些成绩中是否每一个属性都被填充，比如有的成绩记录中课程名称或成绩可能为空，空的数据多完备性就不好。

一致性：可以被视为不存在变更，也可以理解为稳定性或连贯性。一致性是时间性的含义，而有效性中对比的是值空间，强调是空间性含义。比如一个班级的学生人数一般应该都是比较稳定的，如果突然学生人数数据减少 20%，基本上可以确认的是发生数据丢失的问题。

唯一性：是指在系统该元数据有且只有一个，主要用于描述一些主键属性。

有效性：是指是否满足具体的业务标准。比如，百分制考试成绩分数不能超过100，身份证号必须符合特定的规律等。

完整性：是指元数据是否有关联元数据，这些数据关系的是否填充完整。比如在 E-R 实体关系模型中，一个实体就可能是一个表，跨表关系的描述就是数据实体与数据实体之间的关系。

基于数据质量规则五个维度的定义，自定义设计了两个元数据质量规则。

1. 自定义元数据记录数一致性规则

元数据记录数的一致性，所表示的是数据实体的不重复计数和记录数的一致性比较。本元规则主要针对合理性检查，将数据中心所表示的数据实体的元数据个数计数。这里主要检查元数据的分类数与期望的分类元数据数是否一致。通过输入和检测数据记录情况，算出元数据记录数一致性比率(CDCR)，见公式(3-1)。

$$CDCR = |\text{参考记录数} - \text{实际记录数}| / (\text{参考记录数} + \text{实际记录数}) * 100\% \quad (3-1)$$

2. 自定义元数据属性完备率规则

本元规则主要指将数据源提供的关键字段的默认百分比进行检测，考查默认记录数的情况，即是元数据属性填充率规则。检测所有记录，求得符合条件记录的默认完备性比率与期望做比较。判断元数据完备率不小于或不大于 X(X, 百分比，整数，小于大于可以选择，满足不同的期望)，当单字段长度完备率小于或大于 X，认为数据有问题。通过输入检测数据记录情况，算出单字段默认完备率(SDCR)，见公式(3-2)。

$$SDCR = \text{数据表中符合默认值的数据条数} / \text{检测的数据条数} * 100\% \quad (3-2)$$

3.4.6.2 元数据质量检测

元数据管理系统具有对元数据库中元数据质量进行检查的功能。元数据质量反映元数据总体数据健康的程度，反映各部门数据的质量好坏。元数据质量检测包括元数据的填充完备率、一致性、唯一性，有效性和完整性方面，这些规则确保元数据可以正常准确地使用。元数据质量检测模块将为这些检查结果制作检查报告，协助高校数据业务人员分析系统数据质量，并以元数据质量报告形式导出指定元数据库的数据质量情况。

如图 3-15 所示为元数据质量检测流程设计：

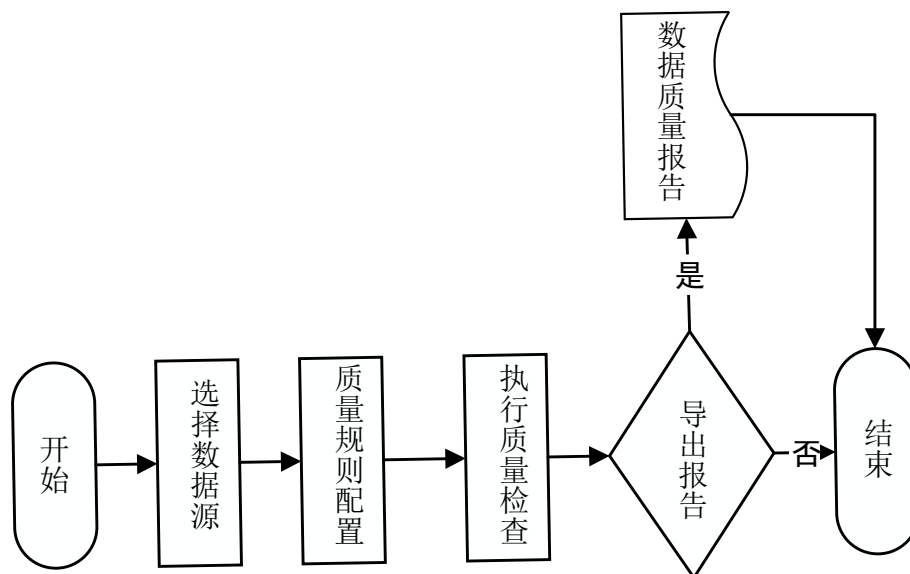


图 3-15 元数据质量检查流程

针对元数据质量规则又提出一些实际检测使用的原则，有一致性检测原则，完整性检测原则和完备性检测原则。

1，一致性检测：元数据的一致性是指在元数据库中元数据与数据仓库中元数据的一致性。首先高校信息系统的数据仓库中抽取元数据，并与元数据库的对应元数据作比较，检查是否有元数据的变更操作，及时发现高校信息系统的元数据变更，确保元数据的更新一致性。元数据一致性检查包括两种方法：自动检查和手动检查。

下图 3-16 是有关一致性检查流程的设计：

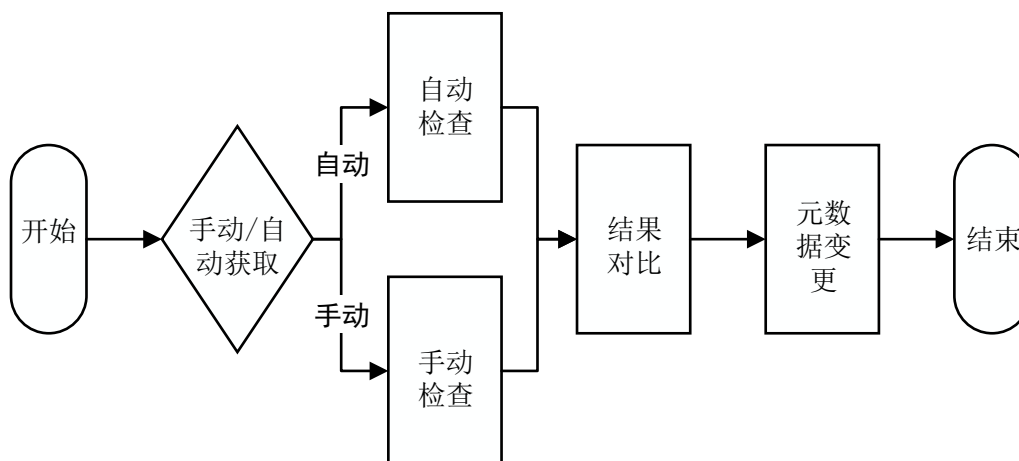


图 3-16 一致性对比流程

手动检查：对于不能自动检查的元数据，需要手动比较高校数据仓库中的元数据和元数据库中的元数据的一致性，检查元数据同步状态是否一致。在手动检查中，

可以比较元数据类型多于自动检查。

自动检查：对于需要检查的元数据，利用接口对高校信息系统中的元数据库进行直接访问，并获取对应的元数据信息，然后相互比较，查看元数据是否保持一致。

一致性检测可以通过元数据的匹配来进行评估。例如，同时存在于高校信息系统和元数据库的元数据，存在于高校信息系统但不存在于元数据库的元数据。高校信息系统中没有但存在于元数据库中的元数据。

当使用自动检测方法时，可以使用元数据名称和主要关键词进行比较。当使用人工验证方法时，可以根据实际比较来判断元数据的一致性。对于元数据属性关系的一致性比较，可以分别对高校信息系统和元数据库中的元数据属性进行比较，找出并统计属性值不一致的元数据比例。

2，完整性检测：在高校信息系统元数据数据库中，系统各子系统中元数据之间的相关性也应保持一致。同时，元数据关联必须一致，比如高校的一个业务部门系统显示引用了另一个业务部门系统的元数据对象，但在另一个系统中找不到。

3，完备性检测：元数据完备性检测是对元数据数据库中实体属性的详细信息进行检测，包括检查元数据填充率、元数据唯一性，元数据完备率和验证元数据属性唯一性。元数据管理系统通过这些元数据关系描述了高校信息系统的数据流，流程依赖性和业务性能的各种内在规律，这些关系决定元数据质量的好坏。元数据关系的正确性直接影响维护人员问题的评估和处理的结果，直接影响开发人员对数据流的分析和评估。因此，元数据管理模块必须做好保证元数据关系正确性的工作。元数据关系的正确性可以对照以下关系进行检查：数据处理关系，层次关系和组织关系。

数据处理关系是数据实体和数据处理过程之间的关系。数据处理关系检查的是从元数据库中找出缺乏应有数据处理关系的数据实体和数据处理过程。例如，找出没有与任何其他数据处理过程的数据实体和找出没有与数据实体建立数据输入输出关系的数据处理过程。

层次关系是在元数据库中对高校信息化系统数据实体进行分级管理所形成的元数据关系，例如将指标按业务主题和业务子主题进行分级管理。层次关系检查是在元数据库中找出存在不合理归属关系的实体，例如找出没有与任何其他业务主体建立联系的指标。

组合关系是高校信息系统数据的整体和部分关系，例如数据库表和字段之间的关系。

3.4.6.3 元数据质量报告

基于元数据质量规则对元数据的检测结果，会产生一个元数据质量报告，元数据质量报告可以显示元数据库中存储的元数据总体数据健康情况的好坏，包括元数据的填充完备率、一致性、唯一性，有效性和完整性，这些规则衍生出一些对于元数据质量的描述术语。

如下表 3-3 是关于数据质量报告的相关术语解释：

表 3-3 数据质量报告相关术语

序号	术语	描述
1	提交表数量	被检测数据源/业务部门提供的所有表数量
2	被测表数量	本次被检测表的数量
3	规则覆盖表数量	配置了检测规则的所有表数量
4	提交字段数量	被检测数据源/业务部门提供的所有字段数量
5	被测字段数量	本次被检测字段的数量
6	规则覆盖字段数量	配置了检测规则的所有字段数量
8	规则覆盖条数	配置了检测规则的所有数据条数
9	规则覆盖率	配置了规则的数据条数在总条数中的占比

数据质量报告内容的设计如下：

- 1，Q 值，为本次数据质量报告的 Q 值计算得分。
- 2，本次检测说明即表示本次检测的一些参数，包括被检测数据源、被检测元数据数量，被检测的字段数量和被检测数据总条数。
- 3，各业务系统元数据的质量得分排名，以柱状图的样式显示，单击柱状图可以显示对应部门和质量得分情况。
- 4，单击报告导出，可以导出对应元数据的质量报告。

3.5 关键技术设计

由于 Spark 的逻辑计划不支持对于字段级别的元数据的解析，这里对 Spark on Hive 的执行模块进行修改，使之支持对字段血缘的分析。

3.5.1 Spark 对字段血缘分析的设计

Spark2.0 以上版本对于 SparkSQL 的支持是通过 Antlr4 进行语法解析，下面介绍对字段的血缘分析的过程。

首先 SQL 通过 Antlr4 会生成抽象语法树，然后通过广度遍历的方式将

UnResolved 的语法信息处理为 Resolved 信息。这样每条经过 Spark 的 SQL 语句，都会预先通过 SparkSQL 解析器执行解析，解析则会添加了 Antlr4 需要的词法以及处理器，然后生成对应的逻辑计划。

在 ParsePlan 的内部会执行 SQL 转化为逻辑计划的操作。为了让逻辑计划能够有字段血缘的识别能力，首先需要对逻辑计划的抽象类进行了修改。为了不影响逻辑计划的其他功能，所以进行了额外 Traits 的添加，这样在解析器中就新添字段级别血缘解析规则，所有影响因素都控制在额外添加的 Traits 内，不会影响 Spark 本身的逻辑计划解析。

逻辑计划其实是针对算子进行操作，并没有对算子里面的涉及成员进行操作，比如 `select a, b, c from table` 语句，这里只会生成一个 `RelationLogicPlan` 和 `ProjectLogicPlan`，其中 `ProjectLogicPlan` 里面对应的 `ProjectList`（即所有的字段列表信息）就是需要关心的字段级别血缘对象，为了不在 `ProjectList` 里面直接对字段列表进行操作，所以可以预先定义字段级别对象，简单地记录在每个字段列表里面，然后放入 `LineageChildren` 中。只要确保 `Analyzer` 在工作时，将 `LineageChildren` 进行复制，并携带到上层节点，在需要操作的节点进行关系的判断，即可保证字段级别的关系正确解析。

`LineageHelper` 中携带的属性包括 `LineageResolved`, `childrenLineageResolved` 用于递归判断所有的子逻辑计划是否已经被 `Rule` 解析过，`markLineageResolved` 用来标注当前逻辑计划解析成功，所有的字段级别对象都会被 `LineageChildren` 容器管理。

对每个环节进行梳理的话，可以分为以下几个流程：

1. 当在 Client 端提交一段 SQL 后，Spark 会通过 Antlr4 进行词法和语法解析来校验 SQL 语法得到抽象语法树，生成未解析的逻辑计划，即 Unresolved logical plan 计划。此时的逻辑计划还不知道字段类型和表数据等信息
2. Spark 可以使用 Analyzer 和 Catalog（元数据信息）对未处理逻辑计划进行规则绑定，生成 Resolved 逻辑计划，此时的逻辑计划也是按照原来的节点原封不动进行绑定，并没有做任何的优化。
3. Spark 中的 Optimizer 在保证数据结果正确的前提下，会对解析后的逻辑计划进行优化。
4. 当得到优化后的逻辑计划后，会通过 Planner 将各种策略应用到逻辑计划中，选择最优的物理计划进行提交执行。

如下图 3-17 是 SparkSQL 具体的底层执行过程：

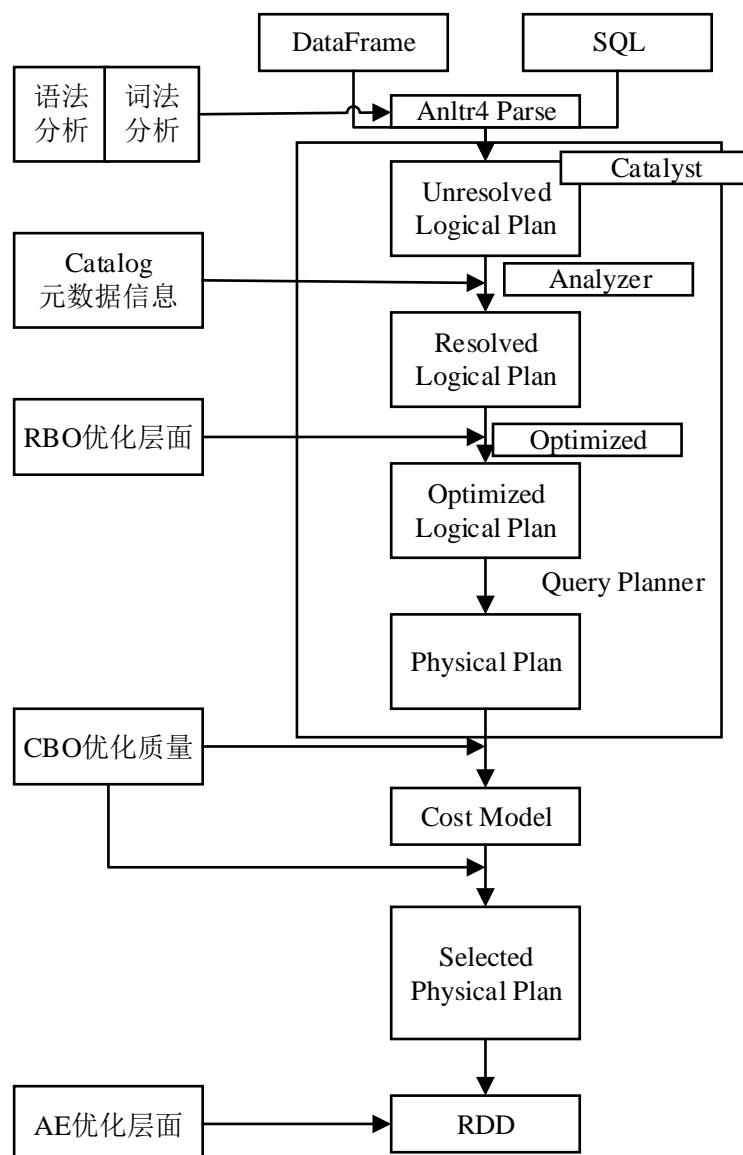


图 3-17 SparkSQL 执行过程

3.6 数据库设计

在高校数据中心中包含各个信息部门的数据，由于涉及的部门业务数据表众多，本系统并没有对所有的数据进行元数据采集。本系统主要对基础教学相关表，一卡通系统相关表，图书馆系统相关表做了采集。基础教学相关表包括学生基础信息表、教师基础信息表、选课信息表、成绩信息表和课程表等；一卡通记录相关表包括消费记录和门禁记录等；图书馆信息相关表包括图书馆借阅信息表。系统中的表有用户信息表，数据源信息表等。下面是部分数据表的设计。

如表 3-4 是系统用户表。表包含对登录用户用户名、密码，用户类型以及创建时间的描述。

表 3-4 系统用户表

序号	字段	字段描述	字段类型	精度	是否必填	主键
1	user_id	用户 id	bigint	20	否	是
2	user_name	用户名	varchar	50	否	否
3	user_pwd	密码	varchar	55	否	否
4	user_role	用户角色	char	2	否	否
5	user_createTime	创建日期	datetime		否	否

如表 3-5 是学生成绩表，对学生各科成绩的记录。

表 3-5 学生成绩表

序号	字段	字段描述	字段类型	精度	是否必填	主外键
1	id	学号	bigint	20	否	主键
2	course_id	课程 id	varchar	50	否	外键
3	score	得分	varchar	100	否	否
4	exam_time	考试时间	datetime		否	否
5	user_createTime	创建日期	datetime		否	否

如表 3-6 是课程表，包含对课程的学时，学分和 ID 等字段的描述。

表 3-6 课程表

序号	字段	字段描述	字段类型	精度	是否为空	主外键
1	course_id	课程 id	bigint	20	否	主键
2	course_name	课程名	varchar	50	否	否
3	teacher_id	教师 id	bigint	20	否	外键
4	course_time	课程学时	varchar	100	否	否
5	course_credit	课程学分	varchar	40	否	否

3.7 本章小结

本章通过对数字化高校数据中心元数据管理的需求进行分析，明确了本系统的功能需求和非功能需求。然后分别从软件架构和功能层次架构对整个系统进行了介绍。然后介绍了系统功能设计：数据源处理模块，元数据采集模块，元数据模型模块，元数据基础功能模块，元数据分析模块，元数据质量检测模块。最后，介绍了字段级别元数据血缘分析的关键性技术的设计和数据库设计。

第四章 基于 Spark 的元数据管理系统的详细设计与实现

本章根据第三章的需求分析和系统的总体设计对系统进行实现，主要对开发环境、数据源的处理、用户登录、元数据采集，元数据模型和元数据基础功能，元数据质量和关键技术的实现进行介绍，元数据的功能具体包括对元数据的查询，变更和计数功能，对元数据进行分析的功能，包括有影响分析和血缘分析。元数据质量检测是根据三章设计的元数据质量规则进行质量检测，并产生元数据质量报告。最后是对关键技术实现描述，包括对于实现字段血缘关系分析所定义的一些关键类的介绍。

4.1 开发环境

基于 Spark 的元数据管理系统在大数据平台上工作，数据存储用到 HDFS，Hive 和 MySQL。对于数据的抽取，转换和加载用的是 Sqoop 工具，计算框架使用的是 Spark，使用 YARN 来进行任务调度，后端使用 Java 语言开发，开发框架有 SpringBoot 和 MyBatis-Plus。前端开发使用的是 HTML5 和 Vue.js。主要的开发技术以及版本如下表 4-1 所示。

表 4-1 开发技术及版本

序号	软件名	版本
1	JDK	1.8
2	CentOS	7.7
3	Hadoop	2.7
4	Spark	2.4
5	Zookeeper	3.4.10
6	Hive	1.2.1
7	MySQL	5.7
8	MyBatis-Plus	3.7
9	Spring	5.x
10	Springboot	2.1.10
11	Vue	3.x
12	Idea	2019.1
13	HTML	5

4.2 数据源处理

4.2.1 数据源选择

本文选取了成都某大学的多个信息部门系统的 2018~2020 年数据作为数据源，由于该校的统一信息系统搭建相对成熟，部门业务完善，数据量大可以真实展现高校信息系统的数据中心建设情况和数据质量情况。使用的数据主要包括基础教学信息数据，包括学生基础信息和成绩数据，还有图书馆系统数据、科研管理数据，人事数据以及一卡通数据等。

如下图 4-1 所示，对于数据源的添加需要选择目标数据源，数据源类型，选择链接方式，再根据数据源类型的不同选择不同的数据库配置方式，关于数据库配置要求见第三章数据源处理。

添加数据源	
* 数据源名称： <input type="text"/> 由数字、汉字、字母、下划线、括号组成，长度不超过20个字符	
数据库类型：	数据库配置
ORACLE	* 主机名： <input type="text"/> IP格式
MYSQL	* 数据库名： <input type="text"/> 字符长度不超过50
SQLSERVER	数据表空间： <input type="text"/> 字符长度不超过50
	索引表空间： <input type="text"/> 字符长度不超过50
链接方式：	* 端口号： <input type="text"/> 0~65535
Native JDBC	* 用户名： <input type="text"/> 字符长度不超过50
ODBC	* 密码： <input type="text"/> 字符长度不超过50
OCI	<input type="button" value="测试连接"/>
JNDI	
<input type="button" value="保存"/> <input type="button" value="取消"/>	

图 4-1 数据源配置

4.2.2 数据传输

在数据中心构建完备的高校系统中，数据传输只需要对 ETL 工具进行任务配置，就可以实现自动化采集高校部门业务数据。在数据中心搭建未完成时，则需要手动通过工具把高校各部门业务数据库中的数据导入到分布式文件系统中去。当

下常用的数据传输工具是 Sqoop，它支持数据在关系型数据库与分布式文件系统之间的相互传输。解决了传统管理系统到大数据平台的数据传输问题，加快高校数据中心的构建。Sqoop 只需要链接到目标数据库，然后接收客户端的导入请求，触发导入的作业，通过 MapReduce 来完成数据传输任务。

Sqoop 从关系型数据库中导入数据到 HDFS 的核心代码如下图 4-2 所示：

```
8 Sqoop import \
9 --connect jdbc:mysql://hadoop1:3306/mysql \
10 --username root \
11 --password 123456 \
12 --table stu_mes \
13 -m 1
```

图 4-2 Sqoop 导入代码

4.3 用户登录模块

用户登录模块是用户进入系统使用元数据功能的门户，为保证系统使用的安全，用户只有通过授权登录才能使用系统的功能和权限，权限认证通过向 Springboot 项目中@Configuration 注解注入 WebMvcConfigurer 来实现拦截器，完成对登录权限的验证。

用户登录界面如图 4-3 所示。



图 4-3 登录界面

登录成功后首页菜单如下图 4-4 所示：



图 4-4 系统首页

首页包含元数据采集模块，元数据基础应用，元数据分析和质量检测。元数据采集包括对数据源的添加，采集任务配置，元数据审核入库，元数据模型转换包。元数据基础应用包括元数据的查询，元数据的变更和元数据的统计计数。元数据分析包括对表级别和字段级别的血缘分析和影响分析。质量检测模块包括元数据质量检测。

4.4 元数据采集模块

4.4.1 元数据的定义

元数据是描述数据的数据，从数据源中抽取出来用于说明数据特征和内容的结构化的数据。如图 4-5 是关于系统信息表的列级元数据，即是字段元数据。

列元数据

	表名	列名	列描述	控件类型	类型
1	FapTable	TableName	表名	文本框	字符串
2	FapTable	TableComment	名称	文本框	字符串
3	FapTable	TableType	类型	下拉框	字符串
4	FapTable	TableCategory	分类	下拉框	字符串
5	FapTable	TableMode	表模式	下拉框	字符串
6	FapTable	SubTable	子表	文本框	字符串
7	FapTable	MainTable	主表	参照	字符串

图 4-5 列级元数据

元数据可以让业务人员快捷地解析数据上下游关系和数据本身的含义，准确

定位需要查询的数据。教育部在 2017 年颁布了基础教育教学资源元数据的标准,规定了基础教学元数据的数据元素,元素描述的顺序和标志符关系的使用等。基于此定义描述可以清楚地描述实体数据的元数据模型。对高校元数据进行组织,分类和查找使用等方面有一个规范性的约束。下表 4-2 是对高校基础教育教学资源的元数据的定义描述,基于此描述可以对业务实体数据进行元模型映射。

表 4-2 基础教育教学元数据标准^[50]

编号	类别名称	解释	约束	大小	数据类型
1	通用, general	该类别描述了学习对象的一些通用信息	M	1	
1.1	标识符, identifier	所描述教学资源的一标识	M	1	
1.1.1	类别, catalog	“1.1.2 表项”所属标识方案或编目方案的名称或指示符。一种命名方案	M	1	字符串* (1000 个字符)
1.1.2	表项, entry	在标识或编目方案中用于标识此学习对象的标识符。一个与名域相关的字符串	M	1	字符串* (1000 个字符)
1.2	标题, title	所描述的教学资源的名称	M	1	多语言字符串* (1000 个字符)
1.2.1	正式标题, proptitle	对教学资源内容的揭示具有关键意义的主要名称	M	1	多语言字符串* (1000 个字符)
1.2.2	其他标题, alternativetitle	正式标题之外的其他名称或替代写法	O	*10	多语言字符串* (1000 个字符)
1.3	评价者, annotator	创建评价的人或组织机构	M	1	多语言字符串* (1000 个字符)
1.3.1	描述, description	评价的内容	M	1	多语言字符串* (1000 个字符)
1.3.2	专题教育资源, special subject	在实施教育教学的过程中,专门研究或讨论的题目	O	*10	教育教学资源元数据服务平台
1.3.3	语种, language	资源知识内容所使用的语言类型	M	*10	
1.4	描述, description	以文本方式对资源内容的简介	M	*10	多语言字符串* (2000 个字符)
1.5	关键字, keyword	用以描述资源主要内容的关键词语	M	*10	字符串

表 4-2 基础教育教学元数据标准 (续)

编号	类别名称	解释	约束	大小	数据类型
2	生存期, lifecycle	该类别描述了资源的历史和当前状态以及那些对资源的发展过程发生作用的实体	M	1	多语言字符串
2.1	版本, version	教学资源的版本状态	O	1	* (50 个字符)
2.2	贡献, contribute	在资源的生存周期中为其发展做出贡献(创建、编辑、发行等)的实体(人或组织)	M	*30	字符串*
2.2.1	贡献者, contributor	对元数据实例做出贡献的实体(人或组织)的标识及相关信息	M	*10	(1000 个字符)
2.2.2	角色, role	贡献的类型。注: 至少应该描述资源的作者	M	1	多语言字符串* (1000 个字符)
2.2.3	日期, date	与资源本身生命周期中的一个事件相关的日期	M	*5	日期时间
3	元-元数据, meta-metadata	该类别描述了元数据实例自身(不是元数据所描述的资源)的信息	M		BERMS2.0
4	技术, technical	该类别描述了资源的技术要求及其相关特征	M		
4.1	格式, format	资源在技术上的数据类型。该数据元素用于确定资源所需的运行软件	M		参照基础教育教学资源元数据平台
4.2	使用环境, requirement	使用资源所需要的技术条件, 如: 硬件、软件、网络等	O		多语言字符串* (1000 个字符) 串* (1000 个字符)
4.3	大小, size	数字化资源的大小, 用十进制数字“0”到“9”表示, 单位是字节(每字节 8 位), 不是兆字节等。该数据元素表明了资源的实际大小, 如果资源经过压缩, 则该数据元素的值是未压缩时的大小	O	1	字符串* (30 个字符)

表 4-2 基础教育教学元数据标准 (续)

编号	类别名称	解释	约束	大小	数据类型
4.4	位置, location	用于表明如何获取资源的字符串。它可能是一个位置(如: URL), 或解析出位置的一种方法(如: URI)。最可取的位置优先列出	O	*10	字符串* (1 000 个字符)
4.5	持续时间, duration	所需要的时间。注: 该数据元素对音频、视频和动画等资源尤为有用	O	1	持续时间
5	教育, educational	该类别描述了资源在基础教育和教学方面的一些关键特征	M	*100	
5.1	学习方式, learningmode	该资源所适用的学习行为, 体现学生在自主性、探究性和合作性方面的基本特征	O	*10	
5.2	资源类型, learningresourcetype	描述该资源的一般范畴、功能、种属或聚类层次, 越主要的类型越先列出	M	*5	字符串* (1 000 个字符)
5.3	适用对象, applicability	该资源所适应的范围	M	*10	
5.3.1	用户类型, audience	该资源的主要使用者, 最重要的优先列出	M	1	字符串* (1 000 个字符)
5.3.2	年级, gradelevel	该资源所适用的学生的年级特征描述	O	*20	字符串* (1 000 个字符)
5.3.3	使用建议, suggestion	针对用户类型对如何使用该资源进行描述	O	1	字符串* (1 000 个字符)
6	分类系统, classificationssystem	所描述的教学资源的所属学科类别及其主要内容	M	1	
6.1	学科名称, curriculumname	资源内容的学科名称	M	1	字符串* (1000 个字符)
6.2	课程标准, curricularstandard	描述该资源与国家课程标准内容框架的对应关系	M	1	字符串* (1000 个字符)
6.3	教材目录, textbookcode	描述该资源与教材内容框架的对应关系	O	*10	字符串* (1000 个字符)

4.4.2 元数据采集

元数据采集是通过选择不同的数据库链接方式连接高校各部门数据源所在的

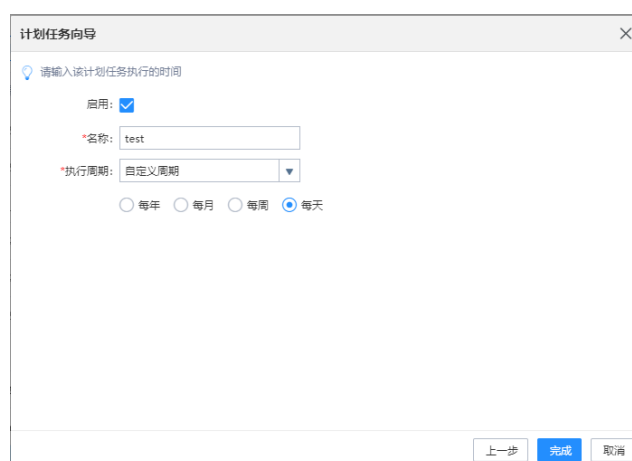
MySQL, Oracle 和 Postgres 等关系型数据库, 然后每个数据源的元数据信息是通过 SQL 查询数据库表来提取的, 这些表在 SQL 中存储了各种元数据信息, 包括系统信息, 数据库的业务元数据和使用数据 ETL 过程的技术元数据。

这里介绍采用 JDBC 方式直连数据库的方式对数据库数据字典的采集, 这一般需要数据库访问地址, 访问的用户名和密码, 一般使用管理员权限对数据库操作, 保证对数据字典数据有访问权限。因为数据库类型较多, 所以这里只列举常用的关系型数据库的链接配置, 如图 4-6 所示为 Oracle 9i 的配置。

```
驱动=oracle.jdbc.driver.OracleDriver  
数据访问URL=jdbc:oracle:thin:@192.168.1.73:1521:myDB  
用户名=user  
密码=password
```

图 4-6 Oracle 数据库配置

另外元数据采集方式还可以运用 ETL 工具进行自动采集, 在用户配置好数据源链接信息后, 可以根据数据源的更新周期进行元数据自动化采集, 包括准时自动解析日志文件、获取元数据、并更新元信息, 确保平台元数据的及时性。对于自动化采集内容如下: 任务名称包括使用物理 ETL 任务名称。任务用途包括对于任务的简洁描述。源表包括所有源数据的名称和路径位置。目的表包括在转换完成之后所有结果数据的名称和目录路径。丢弃文件名称包括丢弃的中间文件名称和中间路径。前置任务包括执行前需要完成的任务。后置任务包括执行后需要完成的任务。如图 4-7 是自动化采集任务表设置, 可以让用户通过配置数据源参数定时执行采集任务, 进行自动化采集。实现直连数据源的元数据采集。



计划任务向导

请输入该计划任务执行的时间

启用: ☒

*名称:

*执行周期:

☐ 每年 ☐ 每月 ☐ 每周 ☒ 每天

上一步 完成 取消

图 4-7 自动化采集任务表

4.4.3 元数据存储

在高校数据库中元数据存储于元数据库中，在存储逻辑上，各类元数据均以对象类的形式存储在 MySQL 数据库中，也就是关系型存储模式，不同的对象与对象之间有关联。按照对象关系进行存储。下图 4-8 系统中元数据的存储模型。

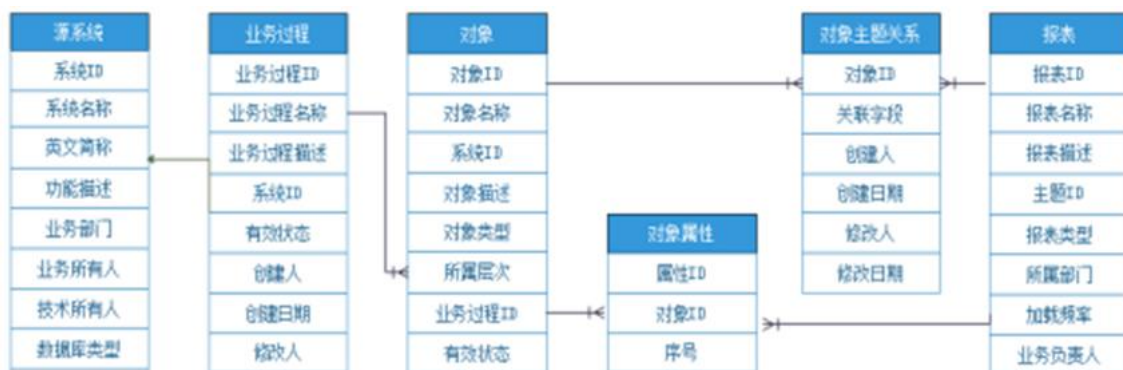


图 4-8 元数据存储关系模型

4.5 元数据模型模块

元数据模型模块涉及对高校元数据模型的基本内容管理，主要包括元数据基本映射关系，元数据描述信息，元数据关联信息，元数据模型变更操作等。预设的元模型不允许变更或删除，只允许新增。元数据模型在发布审核通过后才可以被元数据使用。通过元数据模型的发布和审核可以把元数据模型的设计和使用区分开来，元数据模型只有在发布后才会生效，在审核完成之前，都不会影响到元数据的使用。元模型管理实现如下图 4-9 所示。



图 4-9 元模型管理

对于 CWM 元模型的实现主要在于对转换包定义，下面详细介绍高校基础教学层元数据模型，数据转换层的元模型和数据仓库层的元模型的详细设计与实现。

4.5.1 基础教学元数据模型

基础教学元数据模型包括两种：对象模型包和基础模型包。前者概述了所有元数据元素的结构特征和最基本的类型特征，将基础业务分类包括为上层（采集层、数据层、应用层）服务的各种其他分类。将基础模型分为基本业务服务，类型服务和类型映射服务。

4.5.1.1 对象模型包

CWM 元模型设计了对象模型包的具体内容，并描述元数据类结构，使之成为元数据管理系统中元数据模型包的基础。它根据 CWM 元模型设计的对象模型包，作为 UML 的子部分，只处理和 CWM 元模型有关的内容。所以，元数据管理系统所有的元数据都能够由符合 CWM 元模型的 UML 标准来描述。而对象模型包的主要内容则包括以下四部分：核心元模型、行为元模型，关系元模型和实例元模型。其中核心元模型是所有其他几个包的基础，它们之间的关系如图 4-10 所示。

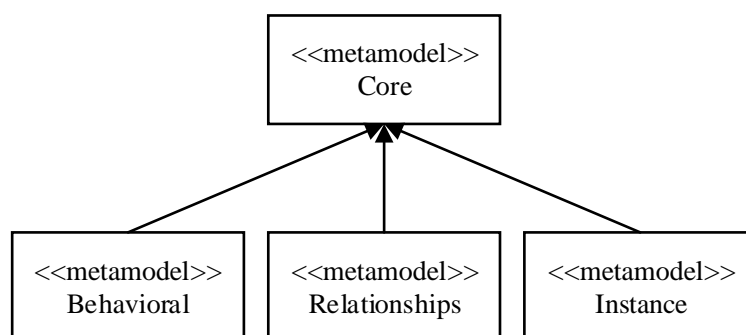


图 4-10 对象模型包

4.5.1.2 基础模型包

基础模型包拥有一个为高级应用提供 CWM 特定服务的包，与对象模型包不相同的是，基础模型包提供的服务更加通用。高级包只需要包含核心服务包相关的内容，不需要包括整个元数据管理系统的核心服务包的所有内容。它只包含其自身的所需要的包。

基础教学相关的核心服务包模型与 CWM 元模型完全一致。核心服务包是 CWM 元模型的基础教学部分的数据类型包，业务信息包和表达式包等。类型映射包和软件导入包在元数据模型管理模块中，相关的基础模型包的设计完全遵循了 CWM 元模型规范。基础模型包可以划分为六个子包，它们分别是 CWM 元模型中的基础教学层包、业务信息包、数据类型包、表达式包、键和索引包，类型映射包和软件部署包，它们之间的关系如下图 4-11 所示。

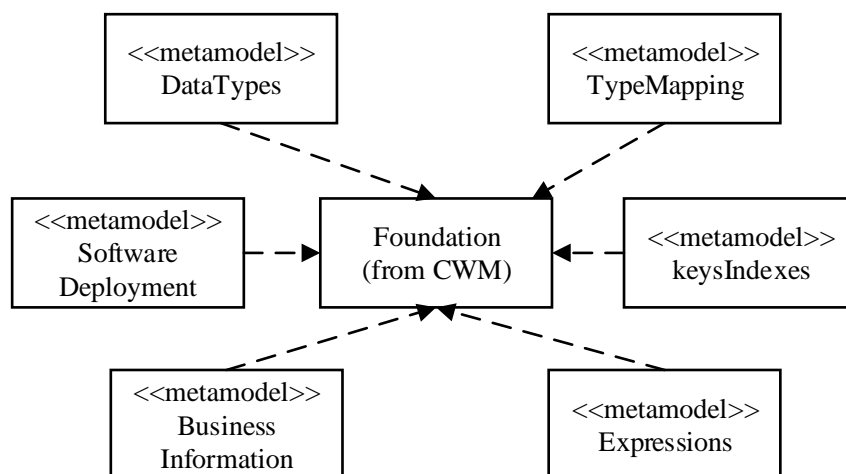


图 4-11 基础模型包

元数据管理系统的基础模型中包含的所有子包都可以单独使用。例如，在传统元数据管理系统中独立开发符合 CWM 标准的数据软件时，需要一个数据类型包，格式包和类型映射包来建立一个支持编程语言的正式元数据模型。关键词索引和业务指示需要扩展包。不需要信息包。关于特定的系统，并不是列出所有基本服务包的子包，而是只列出与系统本身相关的基本服务包。

业务信息包：用于描述提供给元数据管理系统元数据元素的业务相关信息。

数据类型包：定义系统中数据类型的构建包。

表达式包：定义系统中的公式的构建包。

键和索引包：促进访问元数据管理系统中所有数据实例的包。

软件部署包：用于存储元数据管理系统中文档和软件的应用记录，还记录了系统软件、软件配置、子系统类型、部署组件和个别组件等信息。软件配置元数据需要高校对应的部门进行手动输入，用于描述元数据管理系统的各种软件模块的配置和系统的分布式架构等。

类型映射包：在高校各类信息系统对元数据类型进行映射的包。

4.5.2 数据转换层元数据模型

采集层元模型只包含转换包，采集层元数据主要用来描述数据处理过程，这个过程会涉及到处理工具和处理操作。因此，数据处理是元数据管理也是数转换层元数据模型设计的核心，数据处理会涉及到数据源库到数据仓库的数据 ETL 过程，这需要用元数据对其进行定义。

在元数据管理中，数据处理包括对数据的抽取，转换和加载数据的所有部分，并且按照元数据模型规范对其进行抽象解释。通过功能分类分成数据转换相关性，

数据分组和执行和专属转换关系。下面是基于三种功能分类的描述。

数据转换相关性：是指数据在流转过程中的约束条件、转换对象的数据描述和转换步骤等。

数据分组和执行：该元数据包括转换的负载计划及其描述、转换的步骤及其说明、转换的操作、转换之前的约束、每个步骤之前的约束等。

专属转换关系：解释数据源到目标数据库的转换过程和转换规则。该元数据包括转换的映射图、分类符映射图、属性映射图、分类符属性映射图、属性映射图源和属性映射图目的地等相关的元数据。

4.5.3 数据仓库层元数据模型

4.5.3.1 数据仓库层基本元数据

数据仓库元数据：元数据管理系统是基于高校数据中心搭建数据仓库设计的，这样就会得到数据仓库的基本元数据。关系型元数据可以使用关系型包来构建，特别是描述表的元数据、描述视图的元数据，描述模式的元数据和描述存储过程的元数据。描述触发器的元数据，包括描述属性的元数据、描述属性的数据类型的元数据、外部密钥、描述主关键字，公共密钥的元数据和描述索引的元数据等，这些都属于数据仓库的元数据。

4.5.3.2 数据仓库管理元数据

在数据仓库的管理层，有两类元数据，分别是数据仓库过程元数据以及数据仓库操作元数据，将它们结合起来就是数据仓库管理元数据。数据仓库的元数据由数据仓库过程转换包定义，数据仓库操作元数据由操作转换包定义。

1. 仓库过程包

元数据的结构在这个包中被描述，例如数据仓库 ETL 过程。对于过程的元数据描述可以还原到各个数据的处理步骤，并且过程元数据可以与其他数据仓库元数据相联系起来。

2. 仓库操作包

一个仓库业务软件包可以用来构建仓库业务元数据。数据仓库操作元数据包括与数据服务的日常集中处理操作有关的元数据，并分为三个区域：与数据处理过程有关的元数据，与测量有关的元数据和与变更请求有关的元数据。

数据仓库可以记录最近执行的数据处理的细节，并且可以掌握数据处理是否正常执行。系统管理者能够通过该信息掌握元数据管理系统是否及时地应答了信息系统的的功能，并及时进行处理。在此细节中，包括数据处理过程的每一步骤的执

行状况，并且系统管理员可以掌握从元数据管理系统提供的度量与从实际系统提供的数据之间的时间线。

这些指标是对存储的实际大小，预期大小和其他数据单位的测量，可以帮助系统管理员预测元数据管理系统的大小并从数据层面辅助做出决策。

4.6 元数据基础功能模块

基础功能模块拥有对元数据的基本查询和变更的基本操作能力，可根据数据源库、类型等操作元数据，通过输入关键信息查找对应的元数据查询系统是否存在此元数据。还拥有对系统中元数据的统计计数的功能。

4.6.1 元数据的查询和变更

在海量数据的情况下，元数据可以帮助系统使用人员快速查找数据，明确业务数据所属的功能模块和它存储的数据的详细信息，相较于传统的数据管理，即使只知道数据的一个元数据描述，也可以实现快速定位并了解业务数据概况，提高了数据管理的效率。元数据查询是对元数据库中的元数据基本信息进行查询的功能，可以通过该功能查询数据库表、维度表，相关指标和过程的输入输出实体数据的信息，以及其他纳入管理的实体基本信息，查询的信息按处理的层次和业务主题进行划分，查询功能可以返回实体数据及其所属业务模块的信息。如下图 4-12 所示。

图 4-12 元数据查询

元数据的基础功能操作是原子操作，这些原子操作可通过服务封装成接口形式向高校数据中心的元数据管理提供服务。元数据的基础功能都是基于 SQL 的操作，下面将介绍通过 Web 界面发送请求和 Spark 的交互过程以及 SQL 在 Spark 上执行过程。

首先，在实现元数据基础功能过程中，Web 前端可以调用 Restful API，通过 HTTP 请求把 SQL 语句封装并发给 Spark，Spark 收到请求后会对请求做解析以及权限验证，映射和 XML 解析，在验证通过后会按照上述过程对元数据库进行对应的操作。下面是对上述内容的详细解释。

REST 解析：Web 前端会基于 HTTP 协议发送一个 REST 请求，这个请求协议

头是 XML 类型的元数据服务原语。服务端可以对 HTTP 头和 XML 格式元数据服务原语进行解析，分别得到 URL 和数据包。

权限验证：Spark 会根据解析结果，对元数据访问者进行权限验证。

映射转换：在获取到权限后，会根据 URL 的映射获取元数据。

XML 解析：从服务原语中获得 XML 信息。

在完成了 Web 前端和服务端端的交互过后，SQL 在 Spark 上运行的可以分为以下两个流程：

1. 逻辑计划：提交 SparkSQL 进行解析生成逻辑计划，然后进行规则绑定，优化。
2. 物理计划：Spark 将优化后的逻辑计划进行切分生成物理计划，然后进行提交，生成 Job 任务。

4.6.2 元数据计数

系统使用者可以对系统中不同分类的元数据做统计操作，让系统使用者对不同的元数据类型的数目有一个清楚的了解。元数据计数依照元数据分类来统计。

如图 4-13 元数据的统计流程设计如下。

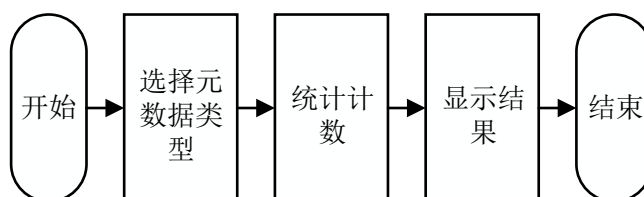


图 4-13 元数据计数流程

4.7 元数据分析模块

4.7.1 血缘分析

血缘分析对于高校信息系统中不同类型的对象实体，其相关的转换流程也有不同类型，比如对于底层数据实体（业务元数据），涉及的是数据 ETL 流程；而对于数据仓库汇总表（管理元数据），可能既涉及数据 ETL 流程，又可能涉及仓库汇总的处理过程；而对于指标（技术元数据），除了以上的处理流程，还涉及指标生成的处理流程。数据源接口实体由部门系统提供，作为元数据管理系统的数据输入，其他的数据实体都经过了一个或多个不同类型的处理过程。在实现血缘分析过程中，对指定的实体，首先会获得该实体的所有前驱实体，然后对这些前驱实体递归地获得各自的前驱实体，结束条件是所有实体到达数据源接口或者是实体没有相

应的前驱实体。血缘分析能够以图形的方式展现所有实体数据表和处理流程。

基于 Spark 的元数据管理系统使用 SparkSQL 对高校数据仓库数据进行增删改的时候，在系统中会形成一个 DataFrame，这个 DataFrame 包含了逻辑计划，这个逻辑计划中包含了 SQL 的抽象语法树。通过对逻辑计划的解析，便可以得到当前阶段所操作的输入表和输出表之间的关系。将整个关系连通起来，去除掉中间表，就可以得到输入表和输出表之间的血缘关系。血缘分析支持以可视化的界面显示元数据之间联系。血缘分析的关键类核心代码定义如下图 4-14 所示：

```
class FlQueryExecutionListener extends QueryExecutionListener with Logging {  
  private val conn= null  
  private val pst= null  
  // 目标表应该只有一张  
  private val targetTable: Map[Long, String] = Map()  
  // source表 可能多个  
  private val sourceTables: Map[Long, String] = Map()  
  // 字段执行过程的关系  
  private val fieldProcess: Map[Long, mutable.Set[Long]] = Map()  
  // 压缩后的血缘关系 只记录source表到 target表  
  private val fieldLineage: Map[String, mutable.Set[String]] = mutable.Map();  
  // SQL类型 考虑 insert select, create as  
  private var processType: String = ""  
  
  override def onSuccess(funcName: String, qe: QueryExecution, durationMs: Long): Unit = withErrorHan  
    // success exec logic plan exec  
    lineageParser(qe)  
}
```

图 4-14 关键类定义



图 4-15 血缘分析

如图 4-15 显示选择对应的表级别元数据，然后元数据管理系统会根据表之间

关系向上回溯,发现与目标元数据有血缘关系的其他元数据。血缘分析除了反映高校数据的来源与加工过程,还描述了数据在处理过程中发生的情况。它不仅可以帮助高校业务人员分析数据表的使用情况还可以追踪业务的关键流程。

4.7.2 影响分析

影响分析可以帮助高校数据业务人员以现在的元数据为起点,快速分析该元数据的下游的元数据信息,进而掌握对此元数据修改可能造成的影响,便于评估元数据变化所带来的风险,让数据管理人员和分析人员高效准确地维护数据资产。这个功能常用于校园信息系统数据源的元数据变更对下游 ETL, ODS 和 DW 等数据仓库应用层的影响分析。

影响分析的实现是运用 SQL 解析器对 SparkSQL 文本进行解析,然后生成抽象语法树 AST(Abstract Syntax Tree),再遍历 AST 得到需要的信息,这样可以避免结果扩大,造成分析结果无法使用的情况。如图 4-16 选择一个元数据,然后对它执行影响分析,会显示向后追踪与这个元数据相关的元数据。

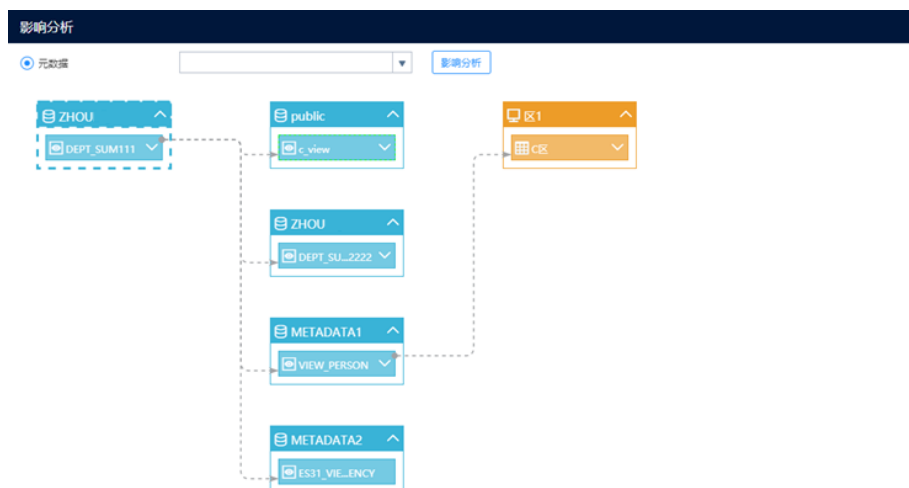


图 4-16 影响分析图

在影响分析中,终点是末端元数据,按照影响关系逐层扩展。影响分析图反映了当前元数据涉及哪些元数据的生成和使用。可以借助此来判断该元数据对象的影响能力,也就是对当前元数据变更会影响到哪些后代元数据。影响分析可以帮助高校业务人员快速地了解数据实体的下游数据信息,快速掌握元数据变更可能造成的影响,以便更有效地评估变更该元数据带来的风险,从而帮助系统使用者高效准确地对数据资产进行管理。

4.8 元数据质量检测模块

4.8.1 元数据质量检测

元数据的质量检测和 Spark 的实时性处理能力相结合,能够极大地提升智慧校园的管理水平,促进大数据技术在高等院校的推广。利用元数据质量的实时检测技术,能够对高校数据中心的数据状态有全面的数据把控。并配合元数据质量检测规则,将高校数据中心的元数据的使用情况以可视化界面方式进行展示,使高等院校的数据管理人员能够对校园各个业务部门的数据运行状态有准确地了解。一旦检测到数据中心内部数据发生异动,可以第一时间通知具体的业务部门,为高校信息化平台的数据维护工作,提供强有力的保障。元数据质量检查可以检测元数据库中存储的元数据总体数据健康情况的好坏,并支持数据质量报告的导出。

4.8.2 元数据质量报告

数据质量报告为后续的数据标准管理,主数据质量监控和主数据管理和数据资产管理等大数据治理活动提供依据,或直接驱动数据治理活动,起着督促数据生产,提高管理数据效率和提升数据质量的作用。元数据质量报告实现如下图 4-17 所示:



图 4-17 元数据质量报告

上述报告 Q 值代表数据的总体好坏程度,满分 100 分。元数据质量报告为元

数据质量的量化标准提供参考依据。检测的元数据数量为 356 个，数据量为 3300 万条左右。得分为 49 分，元数据质量较差，下面柱状图是各数据源的得分情况。

4.9 关键技术实现

基于 Spark 的元数据管理系统为了让 SparkSQL 逻辑计划具备识别字段血缘的能力，首先对逻辑计划的抽象类进行了修改。但为了不影响逻辑计划的其他功能，又增加了额外的 Traits。这样当尝试处理字段血缘时，所有影响因素都控制在额外的 Traits 内，不会影响 Spark 本身正常的逻辑计划的解析。下图 4-18 是 SparkSQL 中对于逻辑计划新增的抽象类和 Traits 的定义描述。

```
//逻辑计划抽象类关系
abstract class LogicalPlan
  extends QueryPlan[LogicalPlan]
  with AnalysisHelper
  with LogicalPlanStats
  with QueryPlanConstraints
  with LineageHelper
  with Logging
//traits LineageHelper 定义类
trait LineageHelper extends QueryPlan[LogicalPlan] with LineageEntity{
  self: LogicalPlan =>
  private var _lineageResolved: Boolean = false
  def childrenLineageResolved: Boolean = children.forall(_.lineageResolved)
  /**判断下游的lineage是否已经被解析（即生成了对应的Col对象）*/
  lazy val lineageResolved: Boolean = childrenLineageResolved
  /**标记对象为已被解析 */
  def markLineageResolved(): Unit = {
    _lineageResolved = true
  } }
```

图 4-18 逻辑计划关键类定义

这里 LineageHelper 中携带的属性包括 _lineageResolved 用于是否被 Rule 解析，childrenLineageResolved 用于递归判断是否所有子逻辑计划都被 Rule 解析，markLineageResolved 用于标记当前逻辑计划解析成功，所有字段对象都将由 LineageChildren 容器接管。

逻辑计划是针对算子进行操作，并没有对算子里面的成员进行操作，诸如 select a, b, c from table 那么这里只会生成一个 RelationLogicPlan 和 ProjectLogicPlan，其中 ProjectLogicPlan 里面对应的 ProjectList（即所有的字段列表）就是需要关心的字段级别血缘对象，为了不在 ProjectList 里面直接对字段列表进行操作，所以可以预先定义字段级别对象，简单的记录在每个字段列表里面，然后放入 LineageChildren 中。只要确保在解析器工作时，将 LineageChildren 进行复制，每

次广度遍历并携带字段到上层节点，在需要操作的节点进行关系的判断，即可保证字段级别的关系正确解析。

对于 Hive 的处理，目前方法是把 Hive 的所有信息都拿走。比如在 Sparksession 中启动 enablehive，默认是重写 Analyzer，所以为了增加对 Hive 数据源的支持，系统在 Analyzer 基类中加入 tailResolutionRules 并在继承的 Analyzer 中重写。图 4-19 是重写 Analyzer 的核心代码。

```
override protected def analyzer: Analyzer = new Analyzer(catalog, conf) {
  override val extendedResolutionRules: Seq[Rule[LogicalPlan]] =
    new ResolveHiveSerdeTable(session) +:
    new FindDataSourceTable(session) +:
    new ResolveSQLOnFile(session) +:
    customResolutionRules
  override val postHocResolutionRules: Seq[Rule[LogicalPlan]] =
    new DetermineTableStats(session) +:
    RelationConversions(conf, catalog) +:
    PreprocessTableCreation(session) +:
    PreprocessTableInsertion(conf) +:
    DataSourceAnalysis(conf) +:
    HiveAnalysis +:
    customPostHocResolutionRules
  override val extendedCheckRules: Seq[LogicalPlan => Unit] =
    PreWriteCheck +:
    PreReadCheck +:
    customCheckRules
  override val tailResolutionRules: Seq[Rule[LogicalPlan]] =
    Seq(new ResolveHiveRelation,
    new ResolveLineage.ResolveExpression) |
}
```

图 4-19 重写 Analyzer 核心代码

为了让字段血缘的修改的生效，系统只需要替换对应版本 Spark on Hive 和 Spark on Catalyst 的 Jar 包即可完成字段血缘关系的支持部署。

4.10 本章小结

本章根据第三章的需求分析和系统的总体设计对系统进行实现，首先对开发环境进行阐述；其次是对数据源的处理和用户登录界面的实现介绍；然后是进入系统后的元数据采集模块，这个模块主要包括对数据源的处理，然后再是具体的采集实现；在采集完成后就是对元模型的实现，元模型的实现主要是对转换包的设计实现，根据不同层次的元数据，使用不同层次的转换包；接着是对元数据功能区的实

现，包括对元数据的查询，变更和计数功能，对元数据进行分析的功能，包括有影响分析和血缘分析；然后是对元数据做质量检测；根据三章设计的元数据质量规则，进行元数据质量检测，并产生元数据质量报告。最后是关于关键技术的实现描述，包括一些关键类的定义。

第五章 系统测试与分析

本章节根据第三章的需求分析和总体设计以及第四章对系统的详细设计与实现,对基于 Spark 的元数据管理系统的进行功能和非功能性的测试。使用的数据是成都某大学的实际数据,包括科研和学工等系统的数据。测试系统的功能是否完善,性能是否满足要求。

5.1 功能测试

功能测试是把已经开发完成的程序发布到测试环境中进行功能测试,保证程序功能的正确性。各个测试环节的具体描述如下:

5.1.1 用户登录模块测试

用户登录作为系统的门户,主管着系统的使用权限,这部分主要针对不同用户进行登录测试,测试详情如表 5-1 所示

表 5-1 登录模块测试用例

用例编号	目的	测试内容步骤	预期结果	测试结果
1	验证用户权限	1, 登录管理员账户。 2, 登录数据分析员账户。	获得对应权限	一致
2	验证密码	输入非法字符, 输入错误密码	登录失败	一致

5.1.2 元数据采集模块测试

元数据采集模块主要负责对数据源的抽取,转换和加载,以及对元数据的采集和存储功能。针对以上功能的测试用例及结果如下表 5-2 所示:

表 5-2 采集模块测试

用例编号	目的	测试内容步骤	预期结果	测试结果
1	测试不同数据源的可用性	1, 选择 Oracle 数据源, 进行使用。2, 选择 MySQL 数据源进行导入。	数据源可正常使用	与预期一致

表 5-2 采集模块测试（续）

用例编号	目的	测试内容步骤	预期结果	测试结果
2	测试 ETL 工具可用性	1, 选择 Oracle 数据源。2, 选择 Sqoop 进行导入数据。3, 检查数据。	在 HDFS 能查询到数据显示	与预期一致
3	测试元数据是否能正常入库	1, 使用 JDBC 链接 Hive。2, 查询元数据库。3, 对比 MySQL 中元数据与 JDBC 中的元数据。	能正常存储	与预期一致
4	测试元数据一致性	1, 采集元数据。2, 查询数据源中元数据描述。3 两者元数据进行对比。	两者相同元数据一致	与预期一致

5.1.3 元数据模型模块测试

元数据模型是针对元数据的进一步抽象，主要包括高校数据仓库元模型库的层次设计，以及转换包和基础教学转换包的设计。针对这些功能设计的测试用例以及测试结果如下表 5-3 所示。

表 5-3 元模型测试

用例编号	目的	测试内容步骤	预期结果	测试结果
1	测试元模型的包是否可用	1, 选择对象包。2, 能否正常使用。	有关元数据可以使用	与预期结果一致
2	测试元模型变更可用性	1, 更换转换包。2 查询元数据。	可以正常使用	与预期结果一致

5.1.4 元数据基础功能及分析功能测试

元数据功能模块主要有，元数据的查询和变更，以及元数据数目统计，元数据分析的功能，由于元数据基础功能和分析功能都是基于底层元数据采集和元数据存储而实现的功能，就统一写的测试计划并进行测试。针对这部分设计的测试用例及测试结果如下表 5-4 所示。

表 5-4 元数据基础功能及分析功能测试

用例编号	目的	测试内容步骤	预期结果	测试结果
1	测试元数据查询功能	1, 输入已知的元数据名称。2, 查询结果。3 查询数据库中对应实例进行对比。	有结果显示	与预期结果一致
2	测试元数据变更功能	1, 查询需要变更的元数据。2, 对元数据修改 3 提交修改。4 查询是否修改成功。	变更成功	与预期结果一致
3	测试统计元数据数目功能	1, 点击统计元数据库中元数据数目。2 查看返回结果。	有结果显示	与预期结果一致
4	测试元数据表级别和字段级别血缘分析准确性	1, 分别选择一个表级别和字段级别的元数据进行血缘回溯。2, 查看血缘显示。3, 手动分析血缘进行对比。	对比结果一致	与预期结果一致
5	测试元数据字段级别影响分析准确性	1, 分别选择一个表级别和字段级别的元数据进行影响分析。2 查看影响那些元数据。4, 手动对影响分析结果进行对比。	对比结果一致	与预期结果一致

5.1.5 元数据质量模块测试

元数据质量模块的是根据设计的元数据质量规则进行对应评价, 达到了解整体元数据的实际质量情况, 让系统使用人员对数据质量有整体把握, 可以放心地使用数据, 对于这部分的模块测试主要包括对规则可用性的测试, 以及元数据质量报告的测试。一致性测试包括两种方法: 自动测试和手动测试。测试计划如下表 5-5 所示。

表 5-5 元数据质量测试计划

用例编号	目的	测试内容步骤	预期结果	测试结果
1	元数据质量规则可用性	1, 选取对应元数据质量规则。2, 执行数据质量检测	规则可用	与预期结果一致
2	元数据质量报告的准确性	1, 执行元数据质量检测。2 得到元数据质量报告。3 对元数据质量报告进行手动对比。	与报告一致	与预期结果一致
3	元数据报告导出可用性	1, 执行元数据质量检测。2, 导出元数据质量报告。	成功导出	与预期结果一致

5.2 性能测试

SparkSQL 和 MySQL 对元数据查询性能测试, 随着系统的数据量增大的时候,

对数据库中数据查询语句消耗的时间对比如下图 5-1 所示。

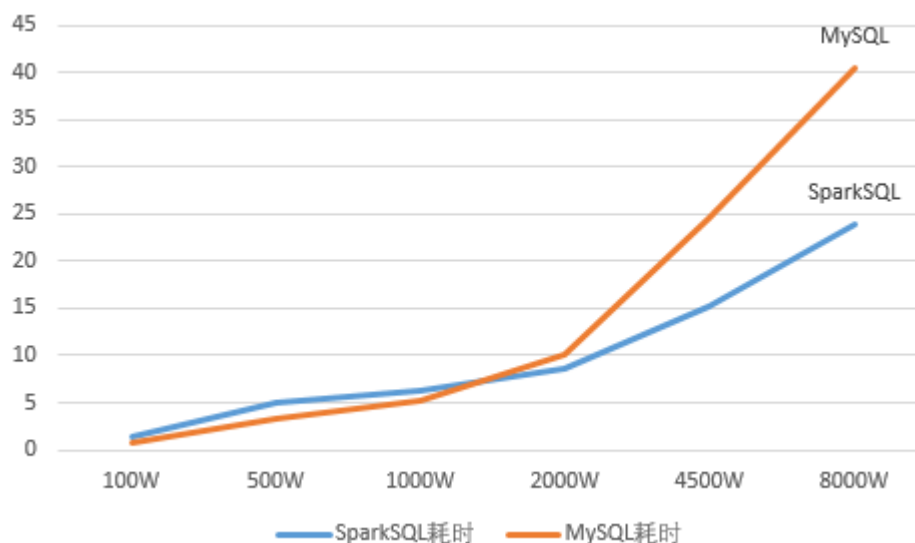


图 5-1 SparkSQL 和 MySQL 查询性能对比图

可以看出随着数据量的不断增大，SparkSQL 的查询优势就显现了出来，所以在查询大量数据的情况下，适合用 SparkSQL 来进行查询。

5.3 本章小结

本章根据上一章的对系统的详细设计与实现，以及基于 Spark 的元数据管理系统的需求分析，对基于 Spark 的元数据管理系统开展了测试。测试元数据管理系统的功能的可用性，各个模块的执行逻辑是否正确，并对性能进行测试。相较于传统元数据管理系统本系统在处理海量数据时执行效率有较大的提升。

第六章 总结与展望

6.1 总结

随着大数据技术的不断发展以及教育信息化的不断深入,高校也在紧锣密鼓地开发自己的智慧化校园,搭建自己的数据中心。由于数据的不断累积,如何对数据中心的数据进行有效的数据治理,挖掘高校数据资源的价值是个亟待解决的问题。元数据作为描述数据的数据,存在于数据的整个生命周期,教育部于印发了《基于教育教学资源的元数据规范》,开启了高校数据治理的大门。然而高校数据治理进程缓慢,一方面由于国内元数据管理发展缓慢,另一方面由于各个部门早期技术和规划的限制,系统处于割裂的状态,数据不能归集到一起,数据标准不统一。传统的元数据管理只能做到管理单独的数据系统,不能同时对多个独立的部门数据进行管理,不能解决高校数据治理的问题。为解决现在数据孤岛的情况,大批高校信息部门都开始运用大数据技术建设统一的数据中心,在当下数据中心建设之际,本论文提出了基于大数据平台上的元数据管理方案,即基于 Spark 的元数据管理系统,对大数据平台上的高校数据进行元数据管理,为发挥高校数据价值提供了数据基础,为上层数据挖掘和数据分析等工作提供充足的保障,为高校系统从数字化向智慧化转型提供有力的支持。

经过对国内外元数据管理的现状的调研,对比了各种大数据工具,制定了本系统的技术路线,系统设计和实现方案。本文主要的工作如下:

1. 在元数据分析方面:使用 Spark 的核心组件 SparkSQL 对数据库进行操作,解析 SparkSQL 的逻辑计划,并对其进行拆解得到元数据血缘关系。通过对 SparkSQL on Hive 模块进行修改,解决了 SparkSQL 对字段级别的元数据分析困难的问题,提高了 Spark 对于字段级别元数据的分析的能力。

2. 在元数据质量方面:从元数据的填充完备率、一致性、唯一性,有效性和完整性对元数据质量进行检测,排除不一致的数据和不规范数据。运行这些数据质量规则可产生相应的元数据质量报告,并支持导出报告,让数据分析人员对系统的元数据质量有详细的了解。

3. 设计并实现了在 Spark 集群的元数据管理服务,通过 HDFS 双机热备机制来保证系统存储数据的一致性,通过 YARN 来进行集群节点之间的任务调度,通过 Spark 来处理集群的计算请求,通过 Hive 进行数据仓库管理,使用 HTML 和 Vue.js 开发了 Web 界面,使得高校数据管理员和数据业务人员能便捷化地进行元数据管理。

6.2 展望

随着教育信息化的进一步发展，高校系统终将迈入智慧化的行列。本系统也有一些需要提升的地方，由于今后将会出现更多种类的数据，本系统目前只支持对结构化的数据进行了元数据管理，以及使用的数据源只能是数据仓库中的数据源，需要外部数据源导入到数据仓库 Hive 中进行存储管理，也就是需要高校的数据中心有很好的建设完整度。另外，对于后续的元数据管理中模型管理，可以通过 Spark 自带的机器学习的算法对高校元数据进行模型训练，在元模型的设计上可得到一种更加智能化的元数据建模方法。

致 谢

三年时光转瞬即逝，一转眼自己就即将毕业，开启另一个新的征程。回望过去三年的研究生岁月，自己在专业知识和眼界都有了很大的提升。

首先我要感谢唐雪飞老师，他为人谦和，德才兼备且又善为人师。唐雪飞老师在我研究生生涯遇到困惑的时候，总能在第一时间给我指出正确的方向，像黑夜里的一盏明灯，指引我去探索，去成长。在我遇到自己感兴趣的方向时，老师鼓励我多去尝试，让我深耕自己喜欢的方向，让我在学习生活中有了很大的进步。本篇论文也是在唐雪飞老师的指导下才顺利完成，唐老师对学生的教导，学生会铭记于心。

其次，还要感谢我的企业导师王东方老师对我论文的宝贵意见，以及在我研究生阶段告诉我学习方法和论文写作指导的实验室的老师，使我在三年能够高效地学习，顺利完成论文的工作内容。

另外，我还要感谢我的室友荣燊，赵建恒，吴岳鸿。在这三年里，我共同讨论专业问题，解决生活里遇到的问题，一起度过了三年难忘的时光，能遇到你们是我的幸运。

最后，感谢我的家人，谢谢你们为我负重前行这么多年，让我能够心无旁骛地完成学业。谢谢你们这么多年以来的鼓励和支持，让我每次都能勇敢地面对学习和生活中的困难。

愿珍惜时间，把握当下，做个对社会有贡献的人。

参考文献

- [1] Wu Minyu, Wu Xiaofei. The era logic of education informatization 2.0 [J]. Education Informatization, 2018: 4-10.
- [2] Chen Fangfang, He Xiaobo. Exploration and effectiveness of university data governance mechanism [J]. China Management Informatization, 2019, 19.
- [3] Jurva R, Matinmikko-Blue M, Niemelä V, et al. Architecture and operational model for smart campus digital infrastructure[J]. Wireless Personal Communications, 2020, 113(3): 1437-1454.
- [4] Mayernik M. metadata[J]. ISKO Encyclopedia of Knowledge Organization, 2020,7.
- [5] 李雨霏, 刘海燕, 闫树. 面向价值实现的数据资产管理体系构建[J]. 大数据, 2020, 6(3): 0-10.
- [6] Qin D, Huang L, Wang Y. Construction of railway metadata management system based on metadata content model and cwm exchange mechanism[C]//2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS). IEEE, 2018: 1-6.
- [7] Tenopir C, Allard S, Sinha P, et al. Data management education from the perspective of science educators[J]. international journal of digital curation, 2016.0-10.
- [8] Al-Badi A, Tarhini A, Khan A I. Exploring big data governance frameworks[J]. Procedia computer science, 2018, 141: 271-277.
- [9] Liu B, Zhou Z, Wu B, et al. Research on the path of smart campus construction facing data governance[C]. 2020 4th International Seminar on Education, Management and Social Sciences (ISEMSS 2020), 2020: 773-777.
- [10] Kusumasari T F, Fauzi R. Design guidelines and process of metadata management based on data management body of knowledge[C]. 2021 7th International Conference on Information Management (ICIM), 2021: 87-91.
- [11] Quinto B: Big data governance and management, Next-Generation Big Data: Springer, 2018: 495-506.
- [12] Liu Z. Practice research on university data center construction from the perspective of data governance[C]//2020 2nd International Conference on Information Technology and Computer Application (ITCA). IEEE, 2020: 489-492.
- [13] Liu Rong. Discussion on data governance under the framework of smart campus in colleges and universities [J]. Computer Knowledge and Technology, 2019, 15(24): 6-7
- [14] 戴炳荣, 闭珊珊, 杨琳, 等. 数据资产标准研究进展与建议[J]. 大数据, 2020, 6(3): 0.

- [15] Dongmei B, Yingjie F, Ming L. Data governance and framework of university libraries[J]. Library and Information Service, 2015, 59(18): 134.
- [16] Cai L, Zhu Y. The challenges of data quality and data quality assessment in the big data era[J]. Data science journal, 2015, 14.
- [17] Sawadogo P, Darmont J. On data lake architectures and metadata management[J]. Journal of Intelligent Information Systems, 2021, 56(1): 97-120.
- [18] Nokkala T, Salmela H, Toivonen J. Data governance in digital platforms[J], 2019.
- [19] Sawadogo P, Darmont J. On data lake architectures and metadata management[J]. Journal of Intelligent Information Systems, 2021, 56(1): 97-120.
- [20] 林焱. 我国政府数据开放的元数据管理研究[D]. 武汉: 武汉大学, 2018.
- [21] Xu H, Wang L, Fan G. Metadata storage and query of hive based on hadoop distributed platform[C]. The International Conference on Cyber Security Intelligence and Analytics, 2019: 1079-1085.
- [22] 吴信东, 嵇圣础. MapReduce 与 Spark 用于大数据分析之比较[J]. 软件学报, 2018, 29(6): 1770-1791.
- [23] 范家宁. 基于 Spark 的多数据源大数据治理平台研究[D]. 中国地质大学 (北京), 2020.
- [24] Dey A, Chinchwadkar G, Fekete A, et al. Metadata-as-a-service[C]//2015 31st IEEE International Conference on Data Engineering Workshops. IEEE, 2015: 6-9.
- [25] Ravat F, Zhao Y. Metadata management for data lakes[C]//European Conference on Advances in Databases and Information Systems. Springer, Cham, 2019: 37-44.
- [26] Yong F. Research on the innovation and development of university education management informationization based on big data environment[C]//International Conference on Application of Intelligent Systems in Multi-modal Information Analytics. Springer, Cham, 2019: 1056-1063.
- [27] Mahanti R. Data governance technology and tools, data governance and data management: Springer, 2021: 145-168.
- [28] Seabolt E, Kandogan E, Roth M. Contextual intelligence for unified data governance[C]. Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, 2018: 1-9.
- [29] Atlas A. Apache Atlas: Data governance and metadata framework for hadoop: From Apache Atlas: atlas.apache.org, 2018.
- [30] Elliott E. Spark SQL and Hive Tables, Introducing. NET for Apache Spark: Springer, 2021: 107-118.
- [31] Zeebaree S R, Shukur H M, Haji L M, et al. Characteristics and analysis of hadoop distributed

- systems[J]. Technology Reports of Kansai University, 2020, 62(4): 1555-1564.
- [32] Wiktorski T: Hadoop architecture, Data-intensive Systems: Springer, 2019: 51-61.
- [33] 金国栋, 卞昊穹, 陈跃国, 等. HDFS 存储和优化技术研究综述[J]. Journal of Software, 2020, 31(1).
- [34] Stamatakis D, Tsikoudis N, Micheli E, et al. A general-purpose architecture for replicated metadata services in distributed file systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(10): 2747-2759.
- [35] Mior M J, Salem K. ReSpark: Automatic caching for iterative applications in apache spark[C]. 2020 IEEE International Conference on Big Data (Big Data), 2020: 331-340.
- [36] Penchikala S. Big data processing with apache spark[M]. Lulu. com, 2018.
- [37] Chellappan S, Ganesan D: Introduction to apache spark and spark core, Practical Apache Spark: Springer, 2018: 79-113.
- [38] Armbrust M, Xin R S, Lian C, et al. Spark sql: Relational data processing in spark[C]//Proceedings of the 2015 ACM SIGMOD international conference on management of data. 2015: 1383-1394.
- [39] AnushaK, Usha Rani K. Performance evaluation of sparksql for batch processing[M]//Emerging Research in Data Engineering Systems and Computer Communications. Springer, Singapore, 2020: 145-153.
- [40] 吴信东, 董丙冰, 堵新政, 等. 数据治理技术[J]. 软件学报, 2019, 30(9): 2830-2856.
- [41] Singh H J, Bawa S. Scalable metadata management techniques for ultra-large distributed storage systems--A systematic review[J]. ACM Computing Surveys (CSUR), 2018, 51(4): 1-37.
- [42] Sim H, Khan A, Vazhkudai S S, et al. An integrated indexing and search service for distributed file systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(10): 2375-2391.
- [43] 甘似禹, 车品觉, 杨天顺, 等. 大数据治理体系[J]. 计算机应用与软件, 2018, 35(6): 7-14.
- [44] Zhang Mingzhi. Metadata management system based on CWM specification design [J]. Computer Knowledge and Technology, 2014 (1X): 254-258.
- [45] 周小娟. 一种轻量级大数据分析系统的实现[J]. 电子设计工程, 2016, 24(8): 40-43.
- [46] Huai Y, Chauhan A, Gates A, et al. Major technical advancements in apache hive[C]//Proceedings of the 2014 ACM SIGMOD international conference on Management of data. 2014: 1235-1246.
- [47] Vohra D. Using apache sqoop[M]//Pro Docker. Apress, Berkeley, CA, 2016: 151-183.
- [48] Vavilapalli V K, Murthy A C, Douglas C, et al. Apache hadoop yarn: Yet another resource

- negotiator[C]. Proceedings of the 4th annual Symposium on Cloud Computing, 2013: 1-16.
- [49] Karanasos K, Suresh A, Douglas C. Advancements in yarn resource manager[J]. Encyclopedia of Big Data Technologies,2019:1.
- [50] 祁涛, 杨非, 曾杰, 等. 基础教育教学资源元数据信息模型 [ol]. <http://www.celtsc.org/info/hybz/335>, 2017:7-12