基于元数据驱动的企业级数据仓库系统①

栾晓宇

(上海烟草集团有限责任公司 经济信息中心, 上海 200082)

摘 要: 通过对元数据建设和维护及其管理系统结构的深入研究, 提出一种基于元数据驱动的企业级数据仓库系统架构. 该系统架构采用关系模型数据结构, 并通过元数据接口和元数据驱动引擎的设计开发将数据仓库中各模块的反馈信息与其元数据存储进行交互, 实现由元数据驱动数据仓库系统的运作机制, 解决了一直以来元数据难以更新维护、数据质量检查效率低的问题, 提升了对企业级数据仓库的管控能力.

关键词:元数据驱动; 数据仓库; ETL; 数据搜索引擎; 数据质量

Enterprise Data Warehouse System Based on Metadata-Driven

LUAN Xiao-Yu

(Shanghai Tobacco Group Co., Ltd., Economic Information Center, Shanghai 200082, China)

Abstract: Based on in-depth study of the construction and maintenance of metadata and its management system structure, this paper presents a metadata driven enterprise data warehouse system architecture. Relational model data structure is used in the system architecture through designing metadata interfaces and metadata driven engine. Feedback information of each module in the data warehouse and metadata storage will interact on each other. This work constructs a metadata-driven data warehouse system operation mechanism, resolving the difficulties to update and maintain metadata and low efficiency of data quality checking. Finally, it enhances the management and control capabilities of enterprise data warehouse.

Key words: metadata-driven; data warehouse; ETL; data search engine; data quality

随着信息技术的快速发展, 企业对数据处理的要求也越来越高, 同时希望对历史数据进行具体的、有针对性的分析和智能化挖掘, 进而从海量数据中发现新客户并能够更好地解决客户新的需求, 寻找新的商机, 使企业在激烈的市场竞争中占据更大的优势. 数据仓库技术的出现, 很好的满足这一技术需求. 在数据仓库结构中, 元数据是必不可少的组成部分, 它贯穿了数据仓库的整个体系, 是连接数据仓库各个模块的桥梁, 是数据仓库的灵魂和基石. 本文通过对数据仓库建设中存在问题的深入分析, 以及对数据仓库维护和管理技术的探讨, 提出并实现了一种基于元数据驱动的企业级数据仓库系统.

1 数据仓库与元数据

数据仓库是一个面向主题的、集成的、相对稳定的、反映历史变化的数据集合[1], 是决策支持系统和联机分析应用数据源的结构化数据环境. 数据仓库的特征在于面向主题、集成性、稳定性和时变性. 其主要功能是将联机事务处理系统(OLTP)经年累月所累积的大量信息, 通过数据仓库理论所特有的信息储存架构, 进行系统的分析整理, 以利于各种分析方法的应用, 并进而支持如决策支持系统(DSS)、主管信息系统(EIS)、商业智能应用(BI)等的创建, 帮助决策者能快速有效的从大量数据中, 分析出有价值的信息, 以利于决策拟定及快速回应外在环境的变动.

① 收稿时间:2014-05-29;收到修改稿时间:2014-07-14

按照传统的定义, 元数据是描述数据的数据[1]. 用于描述要素、数据集或数据集系列的内容、覆盖范围、质量、管理方式、数据的所有者、数据的提供方式等有关的信息. 在数据仓库系统中, 元数据表现出的主要功能有两点: 其一, 元数据描述了数据仓库内数据的结构和建立方法, 它为数据仓库提供了一个目录索引, 可以帮助数据仓库管理员和数据仓库的开发人员非常方便地找到他们所关心的数据, 对此元数据提供了一种强大的搜索功能; 其二, 元数据是数据仓库运行和维护的指挥中心, 可以利用它帮助数据仓库存储和管理数据, 同时也为了解和访问数据提供了便利性.

元数据依据其用途的不同分为两类: 技术元数据和业务元数据. 技术元数据是存储关于数据仓库系统技术细节的数据, 是用于开发和管理数据仓库使用的数据. 业务元数据从业务角度描述了数据仓库中的数据, 它提供了介于使用者和实际系统之间的语义层, 使得不懂计算机技术的业务人员也能够“读懂”数据仓库中的数据.

一套完善的数据仓库解决方案一般来说都是以元数据为核心, 如果没有核心的元数据管理系统, 则不能称之为完善的数据仓库解决方案. 元数据在系统运行、数据问题定位、监控报警等方面都起到核心作用, 利用好元数据可以大幅度提升团队的开发和维护效率, 降低成本并提升质量. 因而可以说元数据是数据仓库的 DNA.

2　系统设计思想与总体架构

2.1 元数据驱动的设计思想

对于一个企业建立的数据仓库而言, 最初建设会因没有数据而苦恼. 随着企业的发展壮大, 数据日益增多, 导致管理人员和开发人员查找数据异常困难. 但在元数据管理系统中, 元数据本身就为数据的查找提供了便利性. 也就是说, 管理好元数据, 也就为管理人员和用户提供了强大的数据搜索功能, 提高了企业搜索引擎的查全率和查准率.

目前元数据的存储仍然缺失一个统一的标准, 使得各部分的元数据都是以特定的单一模式存储, 不利于系统内各部分的交互. 只有采用统一的元数据存储方式, 实现各部分元数据相互关联, 在整个数据仓库体系中形成一个有机整体, 才能为数据仓库的管理和维护提供最大的帮助. 元数据开发的目的是提供给用户和管理人员使用, 否则就失去了开发元数据的意义. 元数据是以特定的方式存储的, 当我们使用的时候, 必须通过与存储相对应的方式进行解析, 方法的统一性保证了存储和解析的相互可逆.

本文设计的基于元数据驱动数据仓库系统主要是利用元数据管理指导数据仓库中各模块的运作. 元数据管理端通过人机交互的方式实现, 提供一个友好的可视化界面实现对元数据的查询和维护, 在可视化界面上定义 ETL 过程中的源、转化规则、目标; 定义应用系统中的业务指标、业务逻辑等; 以及定义数据的查询方式、数据质量的校验规则以及企业级数据搜索引擎[4]所需要的元数据信息. 这些基础信息都以各自特定的形式并作为元数据信息而存放在元数据存储库中, 如图 1 所示.

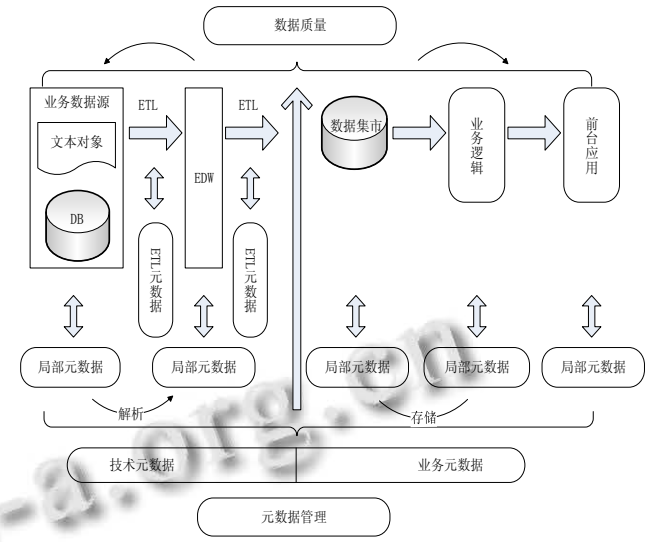

图 1　基于元数据驱动的数据仓库系统原理图

从图 1 中可以看出, 元数据存在于数据仓库的每一个环节, 我们将从数据仓库系统中收集到的元数据信息, 通过元数据驱动引擎, 按已经制定好的元数据存储模型在元数据存储库统一存放元数据信息. 反之, 亦可以通过元数据驱动引擎解析元数据存储库中的元数据, 用于数据仓库系统的管理和维护[1]. 我们可以通过控制元数据来控制数据仓库系统中的每一个环节, 从而控制整个数据仓库的运作.

此外, 数据质量作为校验企业数据有效性的一种方式, 同样也贯穿于整个数据仓库系统. 通过元数据的驱动, 能够准确全面的校验数据的正确性和影响性.

元数据驱动 ETL 过程和业务逻辑的变更, 为数据仓库管理员提供了维护的便捷性.

2.2 基于元数据驱动的数据仓库系统架构

本文实现的利用元数据驱动的数据仓库系统便于管理、维护和决策. 它由五大部分组成: 元数据驱动 ETL 模块、元数据驱动数据质量模块、元数据驱动业务逻辑处理模块、元数据管理模块以及基于元数据的企业级数据仓库搜索引擎[3], 系统架构如图 2 所示.

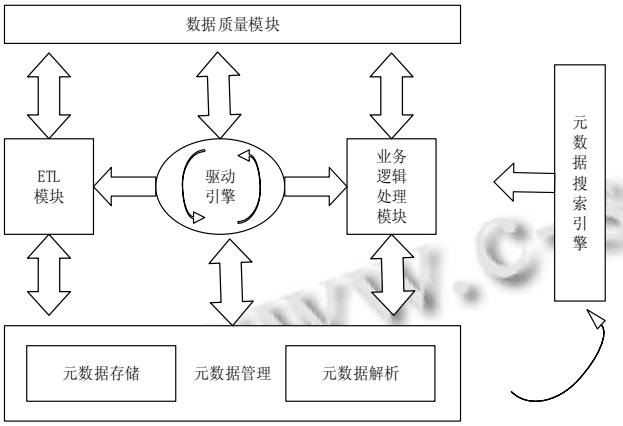

图 2 元数据驱动数据仓库体系结构

其中元数据驱动 ETL 模块的核心思想是将一组 ETL 流程的不一致性或是变化集中到元数据中, 通过固定的逻辑和不同的元数据相结合生成最终的 ETL 目标逻辑. 在更新维护中, 通过修改元数据进而自动实现目标逻辑的复制、扩充和更新. 基于这种思路, 可变部分和固定部分两类元数据负责根据所选择的数据类型, 从元数据存储库中获取 ETL 元数据信息, 并将其转换成通用的 ETL 转换信息, 再通过通用的 ETL 访问接口, 利用相关的驱动程序实现对现有的 ETL 过程更新和维护.

元数据驱动数据质量模块首要作用是对 ETL 过程产生的数据进行校验比对, 保证数据质量. 另外一个重要的功能是作为一个反馈机制, 对 ETL 过程三大环节(数据抽取、数据转换、数据装载)中被数据质量引擎模块所捕获的信息按照已经存放在元数据库中的质量校验规则元数据进行校验. 如若发现数据质量问题, 将终止后续环节的事务调度, 及早发现并防止错误, 杜绝问题数据.

元数据驱动业务逻辑处理模块以元数据形式存储数据仓库中的业务逻辑和数据映射规则, 如后台报表数据与前台业务报表数据的映射关系, 是直接一对一映射还是通过相应的业务逻辑转换处理后再映射等.

元数据管理模块主要负责元数据的存储和解析, 包括 ETL 元数据、数据质量元数据、业务逻辑处理元数据等信息.

3 模块功能与实现

3.1 元数据驱动 ETL

元数据驱动的 ETL 模块主要分三个部分, 如图 3 所示. ETL 元数据用于描述一个完整的 ETL 过程信息规则, 包括抽取规则、转换规则和加载规则, . 工具元数据定义了 ETL 工具执行的方式和调度的周期, . 接口元数据定义通用的 ETL 访问接口, 将 ETL 规则元数据标准化为 ETL 工具元数据, 使得 ETL 工具能够有效识别 ETL 信息规则是整个 ETL 模块的中间枢纽[4].

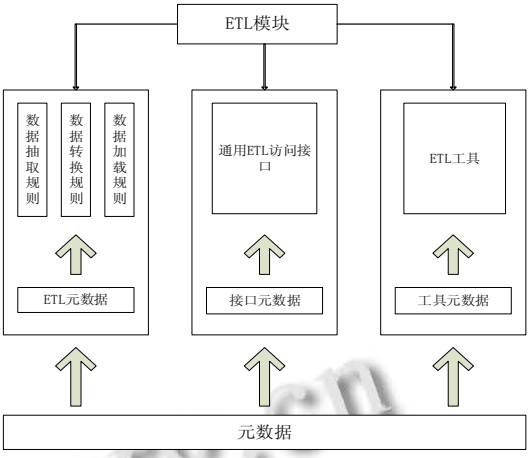

图 3 ETL 模块结构

ETL 元数据: 分为三种, 分别是抽取元数据、转换元数据、加载元数据. 分别指导 ETL 各个环节的运行.

数据抽取是从数据源获取符合需求的数据的过程. 抽取元数据包括源数据库名、用户、密码以及数据抽取的格式和方式等所有描述事务抽取相关的信息. ETL 通过抽取元数据提供的抽取信息, 能够获取业务需求所要的数据, 为数据转换做准备.

数据转换按照数据仓库中设计的数据结构, 对源系统每条记录进行转换. 转换元数据指定清洗条件、映射规则、转换字段、数据类型、聚合方式等信息. ETL 通过转换元数据提供的转换规则, 对抽取的数据进行转换处理, 得到相应的数据结果.

数据加载是将经过清洗、过滤的数据装入数据仓库中指定的主题和细节库中. 加载元数据指定加载方

式, 加载周期等信息. ETL 按照元数据提供的这些信息, 负责处理数据加载环境, 通过通用数据访问接口将得到的数据加载到目标数据库中.

通用 ETL 访问接口: 作为 ETL 逻辑信息与 ETL 工具的中间枢纽, 将外部 ETL 信息标准化为现有 ETL 工具所能识别的标准信息. 接口元数据定义了 ETL 信息的转换规则. 以数据抽取为例, 数据抽取从元数据系统中获取的元数据包括数据源、源所在的路径或所在的数据源库、数据的读取权限、用户名和密码等, 这些信息无法直接被 ETL 工具识别和利用, 需要通过 ETL 接口进行转换, 例如数据源文件或实表名的书写格式、路径的存放规则、用户权限密码的加密方式等规则. 这些解析规则即为通用 ETL 访问接口的元数据, 对这些元数据的解析和存储, 能够保证 ETL 模块正常运作.

ETL 工具: 本系统采用的 ETL 工具是 DataStage. 首先将 ETL 信息转换成 ETL 各个环节的元数据信息, 再通过标准化接口将 ETL 信息导入 ETL 工具中, 最后完成数据的抽取、转换和加载. 通过获取 ETL 工具中的任务文件元数据信息, 与 ETL 访问接口相匹配, 可以直接在 ETL 工具的服务器生成对应的任务文件, 进而调度运行已经存在的 ETL 任务, 实现 ETL 的自动化.

3.2 元数据驱动数据质量

数据质量模块为 ETL 模块服务, 直接影响企业数据的有效性、正确性. 在数据质量模块中最为关键的是数据质量的校验规则. 校验规则都以规则元数据的形式存放在元 数据存储库中, 这些元数据不但可以提供给系统进行自主校验数据质量, 还可以通过开放查询接口的方式, 人工查找某一个确定的指标或报表, 进而利用数据质量校验规则检验数据的正确性. 在数据质量模块中, 元数据驱动的应用, 给用户和开发人员提供了数据质量校验的便利性.

数据质量校验流程如图 4 所示, 利用规则元数据存储数据质量的校验规则, 通过校验规则的解析将校验规则传入数据质量校验引擎, 进而执行数据质量的校验.

数据质量校验规则: 与 ETL 规则元数据类似包括数据的时效性校验(即对最新数据的到达时间校验)、数据的数据量校验和数据的预警校验等. 这些校验规则按元数据的定义标准进行存储.

校验规则解析: 规则元数据是以特定的方式存储的, 使用时必须通过相应的解析规则进行解析才能被执行终端所识别. 解析规则同样以元数据的方式被存储, 以备管理和使用.

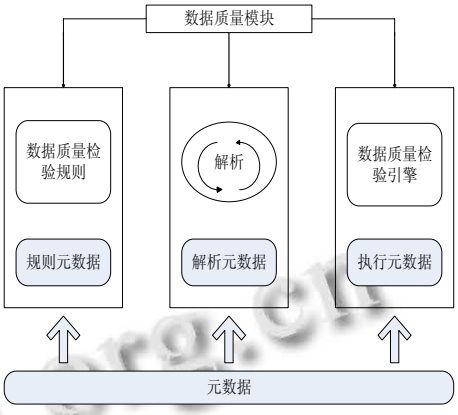

图 4　数据质量模块结构

数据质量校验引擎发出和执行校验命令. 用户和数据库管理员可以利用已经定义的执行元数据, 执行校验指定的数据. 另外, 校验引擎可以与 ETL 调度相结合, 由系统自主校验数据的正确性.

3.3 ETL 与数据质量的交互

ETL 是数据仓库中数据处理最关键的环节, 也是数据质量最难于控制的环节. 如何将 ETL 与数据质量管理相结合, 是解决整个数据仓库中数据质量问题的首要难点[5].

元数据控制 ETL 和数据质量的有机结合, 弥补了现有系统“盲目”的对数据仓库中数据进行校验的缺点. 传统的数据校验只提供一种事后处理机制, 并没有在 ETL 转换数据的过程中实时进行校验, 无法在数据出现问题时就立即停止对数据的转换加载, 从而浪费过多系统资源. 而利用元数据控制 ETL 和数据质量的有机结合, 是将问题数据扼杀在发生点, 避免了问题数据对其他数据的影响.

通过数据质量驱动引擎, 判断最终要加载的目标表数据是否符合数据质量要求. 如果规则校验没有通过, 将停止后续的数据加载, 杜绝错误数据最终入库, 避免用户端显示错误数据. 数据质量模块作为 ETL 调度模块的有力支撑, 为 ETL 的数据质量提供保障, 保证用户获得有效数据.

通过采用 ETL 调度与数据质量相互嵌套的方式, 使数据质量系统可以快速、准确地校验 ETL 过程中的数据准确性. 数据质量校验规则的调度和 ETL JOB 的调度全部由 DataStage 完成. 数据质量校验规则任务和

ETL JOB 都被作为一个事务进行处理, 事务的启用和停止由 DataStage 调度系统完成. ETL JOB 与数据质量校验规则的嵌套执行主要通过 ETL JOB 调度元数据与数据质量元数据相结合的方式实现, 即在每一个 ETL JOB 执行完成后设置节点, 将 ETL JOB 执行完成后的数据信息和状态信息反馈给数据质量系统, 并按照数据质量系统的校验规则进行数据校验, 后续事务则需要得到数据质量系统的校验反馈信息后才可继续运行. 数据质量、ETL 调度和 ETL 模块的结构如图 5 所示.

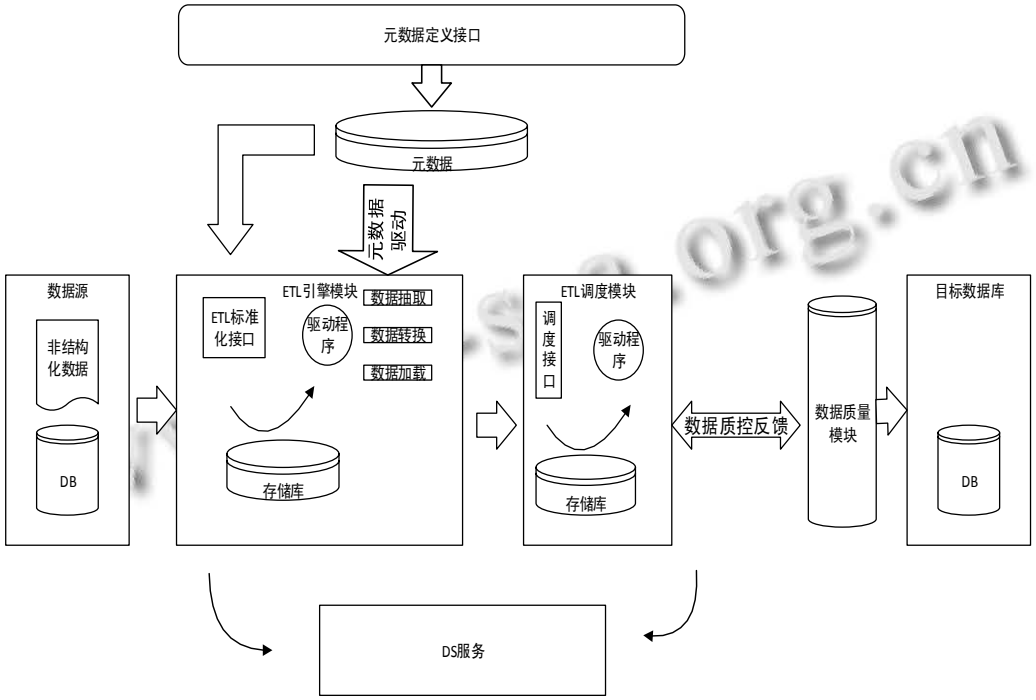

图 5 ETL 模块与数据质量

ETL 元数据与数据质量元数据交互过程中, 元数据可以提供数据仓库系统完整的数据流图. 通过元数据血缘关系分析, 可以快速便捷地找到问题的根源及其影响. 如当发现应用报表中的指标数据错误, 则既要查出错误源头, 还要查出其影响范围. 此外, 还可以通过继承关系分析全面地评估某一主动变更在整个数据仓库中所产生的影响.

3.4 元数据驱动业务逻辑处理

通过元数据驱动管理业务逻辑处理过程, 可以有效解决前台与后台数据不一致的问题. 同时, 由于元数据本身就有导航说明的作用, 可以解决企业业务含义冗余复杂的难题. 随着企业的发展壮大, 其业务层面必将愈加多样, 业务含义及处理逻辑也会极其繁琐和复杂. 系统的复杂性和多样性使得参与人员难以很好地融入并参与系统整体运行管理. 元数据驱动的业务逻辑处理模块应用元数据实现对业务逻辑处理很好的操作管理, 使得用户、管理人员以及非专业人员能够清晰、全面地了解系统. 该模块将业务逻辑处理的元数据信息单独存放于业务元数据存储库中. 业务元数据从业务角度描述数据仓库中的数据, 提供介于使用者和实际系统之间的语义层, 主要包括以下信息: 使用者的业务术语所表达的数据模型、对象名和属性名; 访问数据的原则和数据的来源; 系统所提供的分析方法以及公式和报表的信息.

从整个业务逻辑分析, 针对业务处理的逻辑无疑是最为核心的部分, 即如何将后台数据按照业务处理的规则映射到企业前台. 因此, 将业务处理逻辑按元数据的方式存储, 是管理业务处理逻辑的最好方式. 如图 6 所示.

业务逻辑处理模块制定数据从后台到前台的映射规则, 企业前台用户在读取某一应用的数据时, 首先需要解析相应的业务逻辑处理元数据, 生成可供前台应用程序可以识别的数据处理逻辑, 并最终在前台应用中进行处理以及结果展现.

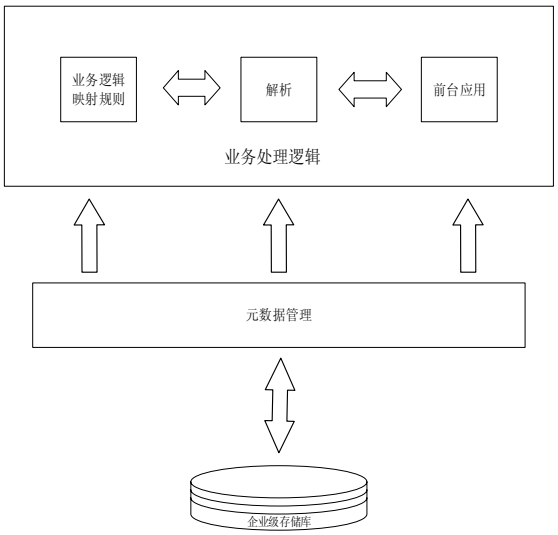

图 6　业务逻辑模块

3.5 元数据管理模块

元数据动态维护管理是现阶段学者面临的共同难题, 也是元数据驱动应用中需要解决的首要问题. 针对这一难点问题, 本系统采用主动元数据库的存储方式, 放置企业元数据. 主动元数据库中的数据随着系统的发展和查询活动而不断地进行交互更新, 知识库有规律的显示出变化以便对现有系统进行正常的更新和维护. 当系统发生变化时, 元数据也必然随之改变[6].

基于上述原因, 本系统采用关系型数据库为基础, 利用交互式的设计方法实现对元数据的管理.

元数据管理模块通过交互管理元数据信息的模式, 将差异信息通过元数据定义接口存储到元数据存储库中, 实现元数据的实时更新, 于此同时根据元数据管理模块实现对相应模块的更新和维护. 对于元数据的管理, 在相对简单的环境系统中, 按照通用的元数据管理标准建立一个集中式的元数据知识库. 而对于比较复杂的环境系统, 应分别建立各部分的元数据管理系统, 形成分布式元数据知识库, 通过建立标准的元数据交换格式, 实现元数据的集成管理.

如图 7 所示, 本系统采用用户管理(可视化的交互界面)和系统管理(系统内元数据管理系统)相结合的元数据管理模式. 系统管理模式面向数据库层面, 由数据库管理系统专业人员完成, 数据用户只有使用权而没有元数据的操作权. 数据应用项目中新生成的数据集元数据由应用系统传递给数据库管理员, 然后由数据库管理员统一管理, 称之为私有元数据. 这种管理模式的缺点在于, 数据在处理过程中形成的动态元数据很难及时记录. 另一种管理方式是面向应用项目的用户管理模式, 允许某些数据用户在数据应用时将元数据的变动信息直接反馈给元数据库, 能够保证元数据的动态更新和新生成数据集元数据的及时捕获及写入元数据文件, 相对私有元数据而言, 这种元数据定义为公有元数据. 用户管理模式比系统管理模式更加灵活, 但对数据用户的权限要进行适当的控制, 避免破坏数据库.

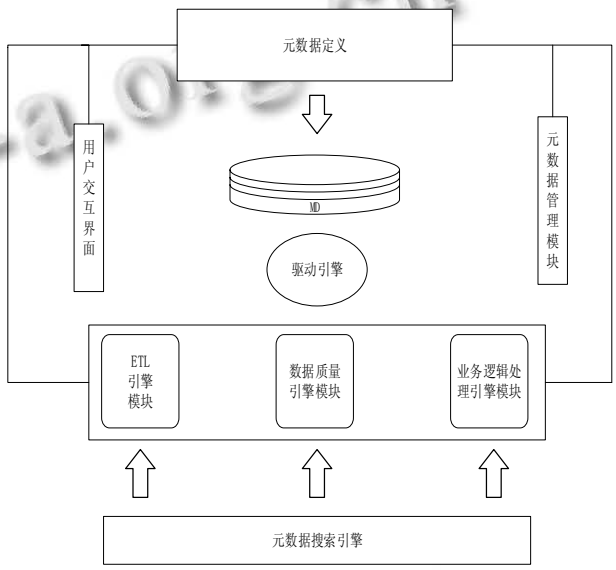

图 7　元数据的交互管理

3.6 基于元数据的企业级数据搜索引擎

企业原有的搜索引擎在数据查询的速度和准确度往往不能满足实际的需求. 相比之下, 应用定位数据仓库系统中数据位置的元数据进行查询, 可以本质的提高搜索的效率和准确性.

元数据的特点是能够发现资源、集成资源, 并对相关的信息资源进行选择、定位和调用, 从而可以追踪资源在使用过程中的变化, 实现信息资源的整合与有效管理. 将元数据与企业级数据搜索引擎相结合可以弥补常规搜索引擎的缺点, 提高企业内部数据搜索的效率和准确度.

在实际应用中, 以元数据为导向, 通过用户搜索语言语义解析器, 对用户输入信息进行解析, 将原始语义搜索转换为元数据的搜索内容与元数据的对应关系, 进而判断数据能否满足用户需要, 并最终为用户实现快速有效的搜索提供支撑.

4 总结

元数据分布在数据仓库体系的每一个角落, 有序的控制着数据仓库中的每一个环节. 将数据仓库中的ETL 过程、数据质量、业务逻辑有机的结合在一起, 不仅给用户对整个数据仓库体系提供了一个有效的管理和维护的方法, 而且对企业也起到了支持和决策的作用. 本文提出的基于元数据驱动的数据仓库系统架构, 通过运用主动元数据库交互的方式解决了一直以来元数据难以更新维护的问题.

下一步的主要工作是对元数据模型的设计进行进一步的完善, 以适应于实际应用中的扩展性需求, 开发一个统一的、可扩展的、可重用的元数据模型服务于企业.

参考文献

1 Inmon WH. Building the Data Warehouse.第 4 版.美国:Wiley 出版,2005.

2 Inmon WH, Strauss D, Neushloss G. DW2.0:下一代数据仓库的构架.北京:机械工业出版社,2010.

3 Marco D.元数据仓库的构架与管理.北京:机械工业出版社,2004.

4 Gonzales ML. IBM® Data Warehousing With IBM Business Intelligence Tools.美国:Wiley 出版,2004.

5 孙安建,王星,等.通用 ETL 工具的研究与实现.计算机应用与软件,2012,29(12):175–178.

6 尤玉林,张宪民.一种可靠的数据仓库中 ETL 策略和构架设计.计算机工程与应用,2005,50(10):172–175.

7 戴超凡,刘青宝.数据仓库中的元数据管理.计算机工程与科学,2003,25(4):54–57.

附录 A 元数据驱动 ETL 及数据质量应用案例

下面是元数据驱动 ETL 及数据质量在上海烟草数据仓库中的应用案例:

本案例主要实现了将上海烟草卷烟商业企业库存日报数据从源文件 JJYX.T_FT_JJYX_SYQYKCRB_WLSJY.TXT 通过其组织机构代码与数据仓库中的维度表 DIM_GJJ.T_DIM_GJJ_ZZJG 中的组织机构代码的映射, 获取对应的组织机构名称, 最终加载到数据仓库中对应的事实表 JJYX.T_FT_JJYX_SYQYKC_WLSJY_R 中.

根据元数据驱动 ETL 的设计原理, 上述数据处理需求被拆分为三个具体的 ETL 流程, 即数据抽取、代码映射和数据加载. 根据元数据驱动系统的转换规则, 将以上三个流程所涉及的元数据标准化到对应的模板中, 表 1 为代码映射的标准化模板.

表 1 ETL 流程_代码映射模板

JOB 名称	JJYX__T_FT_JJYX_SYQYKCRB_WLSJY_02				
JOB ID	ETL 流程	ETL 流程顺序	映射方式	映射字段	映射源字段
JJYX__T_FT_JJYX_SYQYKCRB_WLSJY_02	DMYS	1	leftouterjoin	ZZJGDM	DMYSCQ.ZZJGMC

字段名称	字段类型	字段长度	字段精度	是否主键	是否为空	字段顺序
DQRQ	DATE	4	0	0	N	1
ZZJGDM	INTEGER	4	0	0	N	2
THTXBS	VARCHAR	50	0	0	N	3
SJ	DECIMAL	18	4	0	N	4
JE	DECIMAL	18	4	0	N	5
WFMYSJ	DECIMAL	18	4	0	N	6
WFMYJE	DECIMAL	18	4	0	N	7
ZZJGMC	VARCHAR	100	0	0	N	8
KSRQ	DATE	4	0	0	N	9
JSRQ	DATE	4	0	0	N	10

模板中通过映射字段 ZZJGDM, 获取维度表中标准的组织结构名称, 并将源数据文件中的数据字段和组织机构名称一同输出.

经过 ETL 元数据模板标准化后, 我们能够清晰地了解到该 ETL 链路中每一个流程具体的实现. 以上 ETL 流程模板被存储为 Excel 文件 APP_JJYX_T_FT_JJYX.xls. 然后通过系统解析程序将模板文件导入到元数据接口库中, 解析过程由 Java 程序完成, Java 中的部分模板解析程序如下.

```
File file = new File("D://业务接口配置模型_nsj.xls");
ConfigModelParser parser = new
```

```
ConfigModelParser("test", file);
            ConfigModel configModel = parser.parse();
            DSJobListTemp tempJob = configModel.getTempJob();
            List<Flow> flowDetailList = configModel.getFlowList();
            if (null != flowDetailList && !flowDetailList.isEmpty()) {
                System.out.println("==========Flow Detail==========");
                System.out.print("JOB 名称 ");
                System.out.print("ETL 流程名称 ");
                System.out.print("ETL 流程顺序 ");
                System.out.println("ETL 顺序");
                for (Flow flowDetail : flowDetailList) {
                    System.out.print(flowDetail.getJobId() + "  ");
                    System.out.print(flowDetail.getEtlLcms() + "  ");
                    System.out.print(flowDetail.getEtlLcsx() + "  ");
                    System.out.println(flowDetail.getEtlSx());
                }
            }
```

从标准化的元数据生成具体的 ETL 配置文件主要由元数据驱动程序完成, 并以 ETL 工具软件可以解析的文件格式进行存放. 本系统的驱动程序主要由总程序 ETLPZ.P_ETLPZ_DSJOB_ALL、业务元数据转换程序 ETLPZ.P_ETLPZ 和 ETLPZ.P_ETLPZ_PROCEDURE 及技术元数据(ETL 工具元数据)转换程序 DSJOB.P_DSJOB 组成. 总程序的核心代码如下.

```
IF IN_JOB_BJ=0 THEN
CALL ETLPZ.P_ETLPZ(IN_JOB_ID,IN_MLID,'',V_OUT_MSG,V_OUT_BJ);
ELSEIF IN_JOB_BJ=1 THEN
CALL ETLPZ.P_ETLPZ_PROCEDURE(IN_JOB_ID,IN_MLID,V_OUT_MSG,V_OUT_BJ);
ELSE
SET OUT_BJ=2;
SET OUT_MSG='无法匹配信息';
return;
END IF;
IF V_OUT_BJ<>1 THEN
SET OUT_BJ=-1;
SET OUT_MSG=V_OUT_MSG;
RETURN;
ELSE
call DSJOB.P_DSJOB(IN_JOB_ID,V_OUT_BJ,V_OUT_MSG);
END IF ;
IF V_OUT_BJ<>1 THEN
SET OUT_BJ=-2;
SET OUT_MSG=V_OUT_MSG;
ELSE
SET OUT_BJ=1;
END IF;
```

下列文本为本需求案例最终所生成的 ETL 配置文件的主要信息.

将 ETL 配置文件转换为一个可执行的 ETL JOB 由 ETL 工具完成, 本案例中生成以下三个 ETL JOB:

JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_01;

JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_02;

JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_03;

```
- <DSExport>
  <Header CharacterSet="CP936" ExportingTool="IBM InfoSphere DataStage Export" ToolVersion="8" ServerName="RTSUSE1" ToolInstanceID="dstage1" Date="2014+07+02"
  + <Job Identifier="JJYX__T_FT_JJYX_SYQYKCRB_WLSJY_02" DateModified="2014+07+02" TimeModified="17.40.02">
  + <Record Identifier="ROOT" Type="JobDefn" Readonly="0">
  + <Record Identifier="V0" Type="ContainerView" Readonly="0">
  + <Record Identifier="V0S01" Type="CustomStage" Readonly="0">
  + <Record Identifier="V0S01P001" Type="CustomOutput" Readonly="0">
  + <Record Identifier="V0S01P101" Type="CustomInput" Readonly="0">
  + <Record Identifier="V0S11" Type="CustomStage" Readonly="0">
  + <Record Identifier="V0S11P011"
```

```
Type="CustomOutput" Readonly="0">
      </Job>
  </DSExport>
```

ETL 配置文件主要内容

ETL JOB 的调度与数据质量项目嵌套执行. 通过校验规则, 校验 ETL 过程中每一个流程输入输出数据的有效性. 下列文本实现了 ETL 调度程序与数据质量嵌套执行控制的程序代码.

```
sh ${FileDirectory}/SHELL/SJZX_ETL_RUNJOB.sh "${ETL_DDMC}" "${group}" "${ZYMC}" ${SJZL_STATUS}
if [ $? -ne 0 ]
then
if [ "${SFZDBZ}" = "1" ]
then STATE_JOBFLOW='FAIL'
 break
else STATE_JOBFLOW='OK'
  continue
fi
else
 STATE_JOBFLOW='OK'
fi
```

ETL 调度与数据质量嵌套程序

每一个完整的 ETL 过程执行完成后都有与之对应的详细日志, ETL 调度执行结果和数据质量执行结果如表 2.

表 2 调度执行结果

DDMC	ZYMC	JOB_NAME	JOB_SX	JOBSTART TIME	JOBEND TIME	JOBSTATUS
SJZX_TEST	WLSJ_TEST	JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_01	1	2014-06-23 16:04:23	2014-06-23 16:04:24	RUNNINGOK
SJZX_TEST	WLSJ_TEST	JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_02	2	2014-06-23 16:04:23	2014-06-23 16:04:26	RUNNINGOK
SJZX_TEST	WLSJ_TEST	JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_01	1	2014-06-23 16:09:17	2014-06-23 16:09:19	CHECKFAILED
SJZX_TEST	WLSJ_TEST	JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_02	2	2014-06-23 16:09:17	2014-06-23 16:09:17	NORUNNING

表 3 数据质量校验执行结果

ZX_ID	SL_ID	JG_ID	JG_SM	JCQS
20140607	1766	1	校验通过	1
20140607	23000	1	校验通过	1
20140607	22969	1	校验通过	1
20140607	22968	1	校验通过	1
20140607	22967	1	校验通过	1
20140607	1766	1	校验通过	1
20140607	23000	0	校验不通过	1
20140607	22969	1	校验通过	1
20140607	22968	1	校验通过	1
20140607	22967	1	校验通过	1

从以上 ETL JOB 运行日志结果中可以看出当 JOB: JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_01 的数据质量校验失败后, 其后续 JOB: JJYX_T_FT_JJYX_SYQYKCRB_WLSJY_02 未运行加载(NO RUNNING), 从而减少了不必要的系统资源浪费.