# 元数据管理文献综述

【摘要】 随着移动终端、物联网和传感器技术的飞速发展，数据的产生速度和数量，呈现爆炸式增长。传统的数据处理方式，很难适应当前的现状，于是元数据，以及元数据管理逐渐走进人们的视野[1]。

【关键词】 大数据；元数据管理；

1. 引言

随着可移动终端，5G技术，物联网，传感器等新技术的蓬勃发展，人类社会所产生的数据正在呈现出几何式增长，人类正逐渐走进大数据时代，面对这些海量数据，传统的信息处理方式很难处理，如何进行收集、储存、检索、分析、应用这些海量数据，是摆在全人类面前的一个亟待解决的问题[2]。1989年，Gartner Group的Howard Dresner首次提出“商业智能”（Business Intelligence）这一术语。商业智能通常被理解为企业中现有的数据转化为知识､帮助企业做出明智的业务经营决策的工具，主要目标是将企业所掌握的的信息转换成竞争优势，提高企业决策能力､决策效率､决策准确性[3]。“大数据”一词流行于2009年，起初成熟于互联网行业，因为互联网行业的数据增长迅速，每两年翻一番，而目世界上90%以上的数据是最近几年才产生的。全球互联网公司都意识到“大数据”时代的到来，数据对于企业来说有着重要的意义和价值。如果不能进行很好的分析和提炼，也很难发现大数据背后的价值。要想更好的处理这些数据，首先需要了解这些数据的特点，并针对这些特点提出对应的解决方案。

在现在的IT行业，各个企业对大数据这一概念都有着不同的解读。但对于大数据的特征，却形成了共识，大数据特征首先由维克托迈尔-舍恩伯格和肯尼斯克耶在编写的《大数据时代》中提出，大数据的4V特征：规模性（Volume）、高速性（Velocity）、多样性（Variety）、价值性（Value）[4]，这一点被大家普遍接受。

Volume是指大数据巨大的数据量与数据完整性[5]。十几年前，由于存储方式､科技手段和分析成本等的限制，使得当时许多数据都无法得到记录和保存。即使是可以保存的信号，也大多采用模拟信号保存，当其转变为数字信号的时候，由于信号的采样和转换，都不可避免存在数据的遗漏与丢失。那么现在，大数据的出现，使得信号得以以最原始的状态保存下来，数据量的大小已不是最重要的，数据的完整性才是最重要的。

Variety意味着要在海量､不同数据来源，以及不同存储介质的数据间发现彼此的关联。在大数据时代，各种设备连成一个整体，需要一个平台在这个整体中担任既是信息的收集者也是信息的传播者的角色，推动了数据量的爆炸式增长。这就必然促使我们要在各种各样的数据中发现数据信息之间的相互关联，把看似无用的信息转变为有效的信息，从而做出正确的判断。

Velocity可以理解为数据的产生的速度快，如果不能快速的处理这些数据那么这些数据的价值就会丢失，这也要求我们在设计平台的时候要考虑满足实时性需求。目前，对于数据智能化和实时性的要求越来越高，比如开车时会查看智能导航仪查询最短路线，吃饭时会了解其他用户对这家餐厅的评价，见到可口的食物会拍照发微博等诸如此类的人与人､人与机器之间的信息交流互动，这些都不可避免带来数据交换。而数据交换的关键是降低延迟，以近乎实时的方式呈献给用户。

大数据特征里最关键的一点，就是Value。这也是我们进行数据分析的理论基础。Value的意思是指大数据的价值密度低。如果不经过处理，人眼很难从这些数据中发现有用的信息。大数据时代数据的价值就像沙子淘金，虽然数据量很大，但里面真正的价值很少。现在的任务就是将这些海量数据，利用云计算、分布式等技术，提取出有价值的信息，将信息转化为知识，发现规律，最终用知识促成正确的决策和行动。

1. 元数据的发展

元数据的发展和计算机技术的发展一脉相承，同时也吸收了信息管理学的一部分研究成功。使之成为一个较为复杂的学科[6]。

20世纪60年代，早期的⽂件存储在磁带上，因此文件的访问是顺序的，访问成本与⽂件⼤⼩成正⽐。使⽤简单的索引可以加快文件的访问速度。然⽽，随着索引的增⻓，它们也变得难以管理。由于这个原因，在 20世纪60年代初期， 应⽤树结构的想法成为⼀种潜在的解决⽅案。在 20世纪60年代后期，因为使⽤B-树和B+树，许多商业供应商创建了速度更快且不连续的⽂件系统[10]。对于所有⽂件系统，⾸先确定记录的格式，记录的字段名称及其数据类型，使存储的数据结构化，本质上是⽂件中的一种元数据[7]。

20世纪70年代，可以称为元数据现象开始的⼗年。随着数据库管理系统 (DBMS) 的出现，元数据的使⽤急剧增加。例如，在关系 DBMS 中，元数据被⼴泛⽤于定义数据。这些元数据包括关系名称、属性名称、 键和域信息。元数据集合⽤于定义模式和⼦模式。 1976 年，随着实体关系 (ER) 数据模型的引⼊以及后来其他语义数据模型的出现，更⾼级别的元数据被使⽤。由于语义数据模型（如 ER）没有⽀持的 DBMS，因此部署了⼀种转换机制来将更⾼级别（如 ER）的元数据转换为关系元数据。

20世纪80年代，随着数据库管理系统在存储和检索业务数据⽅⾯的成功，⼈们开始努⼒研究⾮业务数据类型。这些数据类型来⾃不同的应⽤领域，例如计算机辅助设计/计算机辅助制造 (CAD/ CAM)、计算机辅助软件⼯程 (CASE)、地理信息系统 (GIS)、⽂档存储和检索、科学和医学。此时，数据的概念被称为资产的术语所取代。资产是⼀种有⽤的项⽬，它是应⽤程序开发过程的产品或副产品。 资产可以是有形的，如数据（和以前⼀样）、设计和软件代码，也可以是⽆形的，如知识和⽅法。有形资产可以采取框架或完整应⽤程序等⼤粒度组件的形式；它也可以是细粒度的组件，如⼦程序、类或封装的组件。

20世纪90年代，元数据出现了三种独⽴的研究范式，它们负责推动元数据技术前进。1. 代码可重⽤性中的元数据；2. 资产存储库中的元数据；3. 数据仓库中的元数据。由于数据仓库概念的出现，元数据在20世纪90年代发挥了重要作⽤。⾦博尔等⼈将数据仓库上下⽂中的元数据定义为“数据仓库环境中所有不是实际数据本⾝的信息”。数据仓库上下⽂中的元数据基本上有两种：后台元数据和前台元数据. 后台元数据与流程相关并指导提取、 清理和加载流程。具体包括与源数据相关的数据，例如源模式、归档⼤型机数据的旧格式、源的所有权描述、⾃动提取⼯具设置等；数据暂存元数据，例如数据清理规范、缓慢变化的维度策略等；数据转换⽇志；和DBMS系统表内容。前台元数据更具描述性，有助于查询⼯具和报告编写者顺利运⾏。⽰例包括加⼊规范、⽹络安全⽤⼾权限配置⽂件、使⽤和访问映射、⽹络安全使⽤统计等。资产存储库的需求在1990年代后期变得⾄关重要，因为软件项⽬越来越关注集成和重⽤代码、 类、组件、模式、框架和应⽤程序。在实体⾏业中⼤量使⽤数据仓库技术也推动了这种需求与⽹络世界⼀起使⽤⼤量运营数据开发决策⽀持。

21世纪以来，按时间顺序说明元数据概念的演变。⾃ 1960 年代以来，元数据的概念有了很⼤的发展。从简单的⽂件名、字段名和字段类型开始，1970年代的元数据将数据定义描述为由各种数据模型建模。 在 80 年代，随着⾯向对象编程的出现，元数据开始包括类定义和类层次结构（聚合和泛化）。1980 年代引⼊了⼀种称为分⾯分类的特殊分类⽅案，⽤于对资产进⾏分类。然⽽，在2000年初，处理元数据的主要努⼒是认识到需要创建元数据标准。与软件开发不同的是，该标准由以下事实推动，数据仓库项⽬需要异构⼯具和数据环境。例如，在数据仓库世界中，不同⼚商开发的数据质量⼯具、数据建模⼯具、ETL⼯具和终端⽤⼾⼯具，完全不同的规范。使⽤元数据标准的集成允许它们相互通信。数据仓库中的数据也可以是各种类型和格式。标准化⼯作也将有助于数据集成[11]。

1. 元数据的使用

数据增长速度的加快激发了人们从元数据中获得潜在商业价值的新兴趣。因为各业务数据存在各种数据结构，不同业务数据之间差别很大，既带来机遇，也带来挑战。元数据管理框架应运而生，元数据管理框架提供了一个组织框架来协调存储在各种系统中的离散数据集。它还提供了描述信息的组织共识，通常分为业务、运营和技术数据。元数据管理提供了一个组织框架来协调存储在各种系统中的离散数据集。它还提供了描述信息的组织共识，通常分为业务、运营和技术数据。

公司实施元数据管理以筛选旧数据并开发分类法以根据其业务价值对数据进行分类。其中的一个组件是用作元数据存储库（也称为数据字典）的目录或中央数据库。除了对数据进行分类之外，元数据管理策略还用于改进数据分析、制定数据治理策略并建立符合法规的审计跟踪。元数据管理的核心是使人们能够使用基于Web等用户界面识别特定数据片段的属性。该属性可能是文件名、作者、客户ID号等。因此，请求文档的人能够查看和理解数据的不同属性、数据所在的企业系统以及创建这些属性的原因。

在第20 IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings 会议上，Scott A. Brandt发表了《Efficient Metadata Management in Large Distributed Storage Systems》，在这篇论文中，Scott A. Brandt提出了一种新的元数据管理框架，惰性混合（LH）元数据管理[12]，这种框架旨在提供一个高性能和可扩展的元数据管理解决方案。这也为后来的元数据管理框架，提供了方向和功能框架。随后，元数据管理框架发展迅速，截至 2020 年 11 月，Alation、ASG、Alex Solutions、Collibra、Erwin、IBM、Informatica、Oracle、SAP 和 SmartLogic 在 IT 分析公司 Gartner的元数据管理解决方案魔力象限中被列为领先的元数据管理平台供应商。

虽然应该收集什么样的数据，没有绝对的标准，但是对大数据开发平台来说，常见的元数据信息应包括：数据的表结构信息，数据的空间存储，读写权限和其他各类统计数据，数据的血缘关系，数据的业务数据[8]。收集元数据信息是为了数据质量管理，虽然这些数据在原数据源中也存在，但是业务系统并不适合从原数据源系统中直接读取这些数据信息，这样会产生大量的重复功能 ，业务系统应该和这些功能解耦，使业务系统更加贴近于业务。所以，收集表结构信息，不光是简单的信息汇总，更重要的是从平台管理和业务需求的角度出发来考虑，如何整理和归纳数据，方便系统集成，实现最终的业务价值。数据的空间存储，读写权限和其他各类统计数据应该至少包括：数据占取的底层存储空间，历史记录，审计数据。虽然具体的底层数据管理组件自身也记录着这些信息，但是这些功能并不能直接满足业务的需求，需要通过专门的元数据管理平台统一进行采集，加工和管理。数据的血缘关系信息，指的是数据之间的上下游来源去向关系，利用这些关系，建立起生产这些数据的人物之间的依赖关系，进而辅助调度系统的工作调度，或者用来判断一个错误和失败的任务可能对哪些下游数据造成影响以及对于失败的任务进行恢复。数据的业务属性信息，通常与底层系统自身的运行逻辑无关，多半就需要通过其他手段从外部获取了，这些数据与业务系统强相关。对于这些数据的管理和分析，是元数据管理平台的最重要的功能。

1. 元数据管理平台发展现状

目前市场上的元数据管理系统分为开源和商业。由于现在的互联网企业普遍对元数据管理重视性不高、大多数据仓库体量小以及各行业、企业数据现状复杂不同，严重影响着元数据管理平台的发展，目前元数据管理平台目前还只是在一些大的企业得到应用，对于一些小企业，因为业务简单，元数据管理平台的需求并不那么急切，元数据管理平台的发展还处于起步阶段，但是随着企业业务数据的不断积累，元数据管理平台一定是各个企业未来发展的方向。

* 1. 开源元数据管理平台Apache Atlas

Atlas的架构方案相当典型，基本上这类系统大致都会由元数据的收集，存储和查询展示三部分核心组件组成。此外，还会有一个管理后台对整体元数据的采集流程以及元数据格式定义和服务的部署等各项内容进行配置管理[13]。

对应到Atlas的实现上，Atlas通过各种hook、bridge插件来采集几种数据源的元数据信息，通过一套自定义的Type 体系来定义元数据信息的格式，通过搜索引擎对元数据进行全文索引和条件检索，除了自带的UI控制台以外，Atlas还可以通过Rest API的形式对外提供服务。

Atlas的整体设计核心在于数据血缘关系的采集以及数据表维度的基本信息和业务属性信息的管理。为了达到这一目标，Atlas设计了一套通用的Type体系来描述这些信息。主要的Type基础类型包括DataSet和Process，前者用来描述各种数据源本身，后者用来描述一个数据处理的流程，比如一个ETL任务等。

Atlas现有的Bridge实现，从数据源的角度来看，主要覆盖了Hive，HBase，HDFS和Kafka，此外还有适配于Sqoop, Storm和Falcon的Bridge，不过这三者更多的是从Process的角度入手，最后落地的数据源还是上述四种数据源。具体Bridge的实现多半是通过上述底层存储，计算引擎各自流程中的Hook机制来实现的，比如Hive SQL的Post Execute Hook，HBase的Coprocessor等，这些Hook机制可以监控对应数据源的数据变化和数据源结构的变化，而采集到的数据则通过Kafka消息队列传输给Atlas Server或者其它订阅者进行消费，以此来保持数据源和元数据系统的数据的一致性。在业务信息管理方面，Atlas通过用户自定义Type 属性信息的方式，让用户可以实现数据的业务信息填写或者对数据打标签等操作，便于后续对数据进行定向过滤检索。

最后，Atlas可以和Ranger配套使用，允许Ranger通过Atlas中用户自定义的数据标签的形式来对数据进行动态授权管理工作，相对于基于路径或者表名、文件名的形式进行静态授权的方式，这种基于标签的方式，有时候可以更加灵活的处理一些特定场景下的权限管理工作。

总体而言，Atlas的实现，从结构原理的角度来说，还算是比较合理的，但从现阶段来看，Atlas的具体实现还比较粗糙，很多功能也是处于可用但并不完善的状态。此外Atlas在数据审计环节做的工作还不够，与数据业务流程的集成应用方面也很有限。Atlas项目本身很长时间也都处于孵化状态，截至2012-12-03，github的心跳图一直都很活跃，但是目前功能还不够完善，因此还需要大家一起努力。对于一些自研项目或者是一个很好的开始，可以基于Apache Atlas针对自己的业务进行二次开发。

* 1. 闭源元数据管理平台Navigator

另外一个比较常见的元数据管理平台解决方案是Cloudera CDH发行版中主推的Navigator，相比Atlas而言，Navigator的整体实现更加成熟一些，更像一个完整的解决方案，不过，Navigator是一个商用版本的元数据管理平台的解决方案。

Navigator的产品定位是数据管理，本质上也是通过管理元数据来管理数据，但周边工具和配套设施相比于Atlas相对完善，和Cloudera Manager管理后台的产品集成也做得很好。相比Atlas来说，Navigator的整体组件架构也更加复杂一些[14]。

因为Navigator定位为一个完善的数据管理，所以对数据的审计管理方面的工作也做的更好，除了采集和管理Hive、Impala等表格的血缘信息，Navigator也可以配置采集包括HDFS的读写操作记录，Yarn、Spark、Pig等作业的运行统计数据在内的信息。Navigator同时还为用户提供了各种统计分析视图和查询管理工具来分析这些数据。

从底层实现来看，Navigator同样通过Hook机制或者Plugin插件的形式从各种底层系统的运行过程中获取相关信息。但与Atlas不同的是，Navigator的元数据采集传输处理流程并没有把这些变动信息写入到Kafka消息队列中，而是主要通过这些插件写入到相关服务所在的本地Log文件中，然后由Cloudera Manager在每台服务节点上部署的Agent来读取，过滤，分析处理并传输这些信息给审计服务器。此外Navigator还通过独立的元数据服务器来收集和分析一些非Log来源的元数据信息，并统一对外提供元数据的配置管理服务。用户还可以通过配置Policy策略，让元数据服务器自动基于用户定义的规则，替用户完成数据的Tag标签标注工作，进而提升数据自动化管理的能力。

总体而言，Navigator和Cloudera Manger的产品集成工作做得想当完善，如果你使用CDH发行版全家福套件来管理你的集群的话，使用Navigator应该是一个不错的选择。不过，如果是自主管理的集群或者自建的大数据开发平台，需要很大的个性化空间，深度集成定制的Navigator就很难应用于你，但无论如何，对于自主开发的元数据管理系统而言，Navigator的整体设计思想也是值得借鉴的[15]。

1. 元数据管理平台展望

随着公司的业务不断扩展，数据源不断积累。每个业务数据源之间又都是相互独立的，很多工作都要重新做起，在计算机技术领域，现象就是重复的造轮子。而元数据管理平台可以把这些重复的开发工作复用起来，上层业务可以直接基于元数据管理平台已有的功能去实现，这样达成目标的路径更短更高效，对于业务可以做到敏捷开发，对于需求可以做到快速响应。所以元数据管理平台在未来一定可以得到广大企业和组织的应用，如果可以实现一个可以普遍接入的元数据管理平台，在未来一定可以拥有广大的市场。

引用

1. Labrinidis, A., & Jagadish, H. V. (2012). Challenges and opportunities with big data. Proceedings of the VLDB Endowment, 5(12), 2032-2033.
2. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. Mobile networks and applications, 19(2), 171-209.
3. McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., & Barton, D. (2012). Big data: the management revolution. Harvard business review, 90(10), 60-68.
4. George, G., Haas, M. R., & Pentland, A. (2014). Big data and management. Academy of management Journal, 57(2), 321-326.
5. 邬贺铨. (2013). 大数据时代的机遇与挑战. 唯实: 现代管理, (5), 33-34.
6. Duval, E., Hodgins, W., Sutton, S., & Weibel, S. L. (2002). Metadata principles and practicalities. \*D-lib Magazine\*, \*8\*(4), 1-10.
7. Baca, M. (Ed.). (2016). \*Introduction to metadata\*. Getty Publications.
8. Greenberg, J. (2005). Understanding metadata and metadata schemes. \*Cataloging & classification quarterly\*, \*40\*(3-4), 17-36.
9. Jones, M. B., Berkley, C., Bojilova, J., & Schildhauer, M. (2001). Managing scientific metadata. \*IEEE Internet Computing\*, \*5\*(5), 59-68.
10. Baldonado, M., Chang, C. C. K., Gravano, L., & Paepcke, A. (1997). The Stanford digital library metadata architecture. \*International journal on digital libraries\*, \*1\*(2), 108-121.
11. Sen, A. (2004). Metadata management: past, present and future. \*Decision Support Systems\*, \*37\*(1), 151-173.
12. Brandt, S. A., Miller, E. L., Long, D. D., & Xue, L. (2003, April). Efficient metadata management in large distributed storage systems. In \*20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings.\* (pp. 290-298). IEEE.
13. Apache. (2021, December 31). Apache Atlas Documentation. The apache atlas website.
14. Cloudera. (2021, December 31). Cloudera Navigator Data Management Overview. The Cloudera website.
15. 彩色蚂蚁(2018)大数据平台-元数据管理系统解析 Retrieved from https://blog.csdn.net/colorant/article/details/79549785