# 元数据管理

## 元数据的定义

按照传统的定义，元数据（Metadata）是关于数据的数据。在数据仓库系统中，元数据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；元数据是描述数据仓库内数据的结构和建立方法的数据，可将其按用途的不同分为两类：技术元数据（Technical Metadata）和业务元数据（Business Metadata）。

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，它主要包括以下信息：

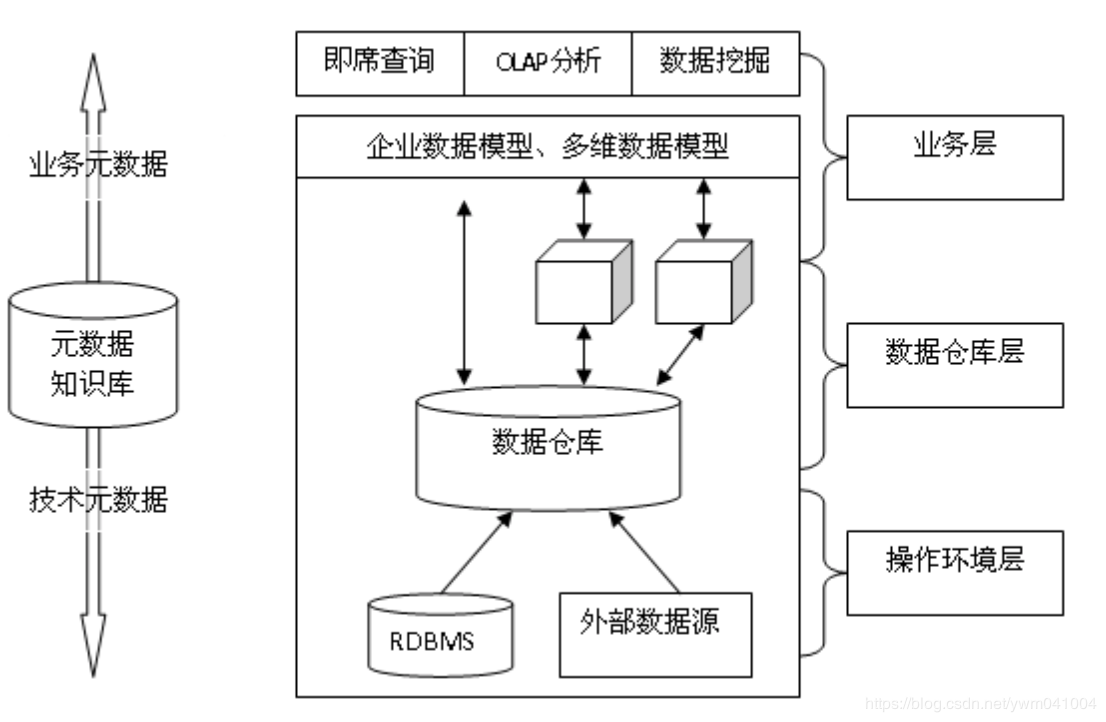
* 数据仓库结构的描述，包括仓库模式、视图、维、层次结构和导出数据的定义，以及数据集市的位置和内容；
* 业务系统、数据仓库和数据集市的体系结构和模式
* 汇总用的算法，包括度量和维定义算法，数据粒度、主题领域、聚集、汇总、预定义的查询与报告；
* 由操作环境到数据仓库环境的映射，包括源数据和它们的内容、数据分割、数据提取、清理、转换规则和数据刷新规则、安全（用户授权和存取控制）。

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。业务元数据主要包括以下信息：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；具体包括以下信息：

* 企业概念模型：这是业务元数据所应提供的重要的信息，它表示企业数据模型的高层信息、整个企业的业务概念和相互关系。以这个企业模型为基础，不懂数据库技术和SQL语句的业务人员对数据仓库中的数据也能做到心中有数。
* 多维数据模型：这是企业概念模型的重要组成部分，它告诉业务分析人员在数据集市当中有哪些维、维的类别、数据立方体以及数据集市中的聚合规则。这里的数据立方体表示某主题领域业务事实表和维表的多维组织形式。
* 业务概念模型和物理数据之间的依赖：以上提到的业务元数据只是表示出了数据的业务视图，这些业务视图与实际的数据仓库或数据库、多维数据库中的表、字段、维、层次等之间的对应关系也应该在元数据知识库中有所体现。

## 元数据的作用

与其说数据仓库是软件开发项目，还不如说是系统集成项目，因为它的主要工作是把所需的数据仓库工具集成在一起，完成数据的抽取、转换和加载，OLAP分析和数据挖掘等。如下图所示，它的典型结构由操作环境层、数据仓库层和业务层等组成。



其中，第一层（操作环境层）是指整个企业内有关业务的OLTP系统和一些外部数据源；第二层是通过把第一层的相关数据抽取到一个中心区而组成的数据仓库层；第三层是为了完成对业务数据的分析而由各种工具组成的业务层。图中左边的部分是元数据管理，它起到了承上启下的作用，具体体现在以下几个方面：

1.元数据是进行数据集成所必需的  
 数据仓库最大的特点就是它的集成性。这一特点不仅体现在它所包含的数据上，还体现在实施数据仓库项目的过程当中。一方面，从各个数据源中抽取的数据要按照一定的模式存入数据仓库中，这些数据源与数据仓库中数据的对应关系及转换规则都要存储在元数据知识库中；另一方面，在数据仓库项目实施过程中，直接建立数据仓库往往费时、费力，因此在实践当中，人们可能会按照统一的数据模型，首先建设数据集市，然后在各个数据集市的基础上再建设数据仓库。不过，当数据集市数量增多时很容易形成“蜘蛛网”现象，而元数据管理是解决“蜘蛛网”的关键。如果在建立数据集市的过程中，注意了元数据管理，在集成到数据仓库中时就会比较顺利；相反，如果在建设数据集市的过程中忽视了元数据管理，那么最后的集成过程就会很困难，甚至不可能实现。

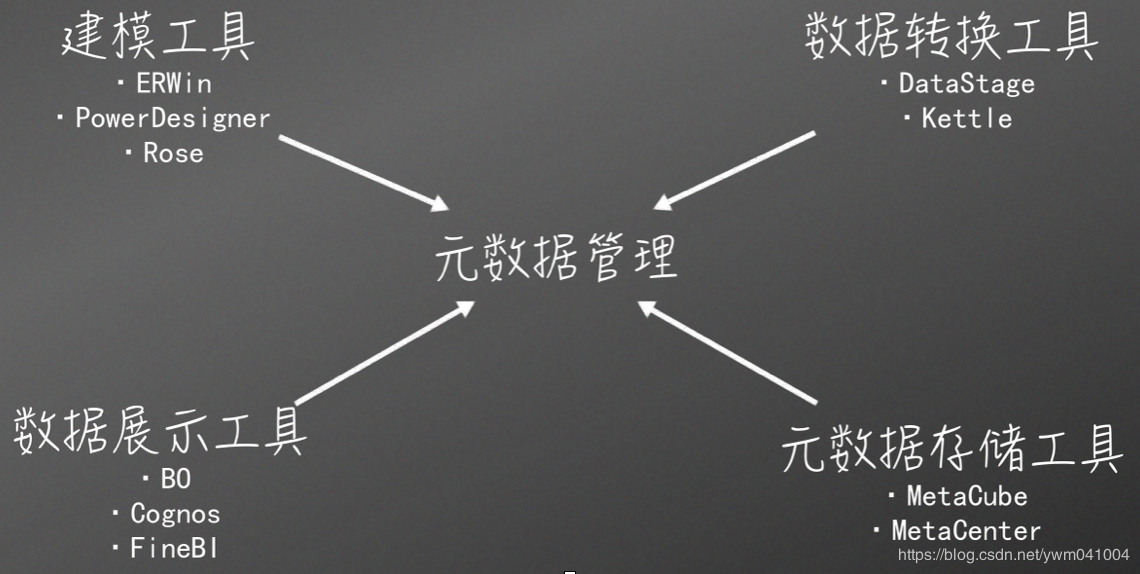
2.元数据定义的语义层可以帮助用户理解数据仓库中的数据  
 最终用户不可能象数据仓库系统管理员或开发人员那样熟悉数据库技术，因此迫切需要有一个“翻译”，能够使他们清晰地理解数据仓库中数据的含意。元数据可以实现业务模型与数据模型之间的映射，因而可以把数据以用户需要的方式“翻译”出来，从而帮助最终用户理解和使用数据。

3.元数据是保证数据质量的关键  
 数据仓库或数据集市建立好以后，使用者在使用的时候，常常会产生对数据的怀疑。这些怀疑往往是由于底层的数据对于用户来说是不“透明”的，使用者很自然地对结果产生怀疑。而借助元数据管理系统，最终的使用者对各个数据的来龙去脉以及数据抽取和转换的规则都会很方便地得到，这样他们自然会对数据具有信心；当然也可便捷地发现数据所存在的质量问题。甚至国外有学者还在元数据模型的基础上引入质量维，从更高的角度上来解决这一问题。

4.元数据可以支持需求变化  
 随着信息技术的发展和企业职能的变化，企业的需求也在不断地改变。如何构造一个随着需求改变而平滑变化的软件系统，是软件工程领域中的一个重要问题。传统的信息系统往往是通过文档来适应需求变化，但是仅仅依靠文档还是远远不够的。成功的元数据管理系统可以把整个业务的工作流、数据流和信息流有效地管理起来，使得系统不依赖特定的开发人员，从而提高系统的可扩展性。

## 元数据管理现状

元数据几乎可以被称为是数据仓库乃至商业智能（BI）系统的“灵魂”，正是由于元数据在整个数据仓库生命周期中有着重要的地位，各个厂商的数据仓库解决方案都提到了关于对元数据的管理。但遗憾的是对于元数据的管理，各个解决方案都没有明确提出一个完整的管理模式；它们提供的仅仅是对特定的局部元数据的管理。当前市场上与元数据有关的主要工具见下图：

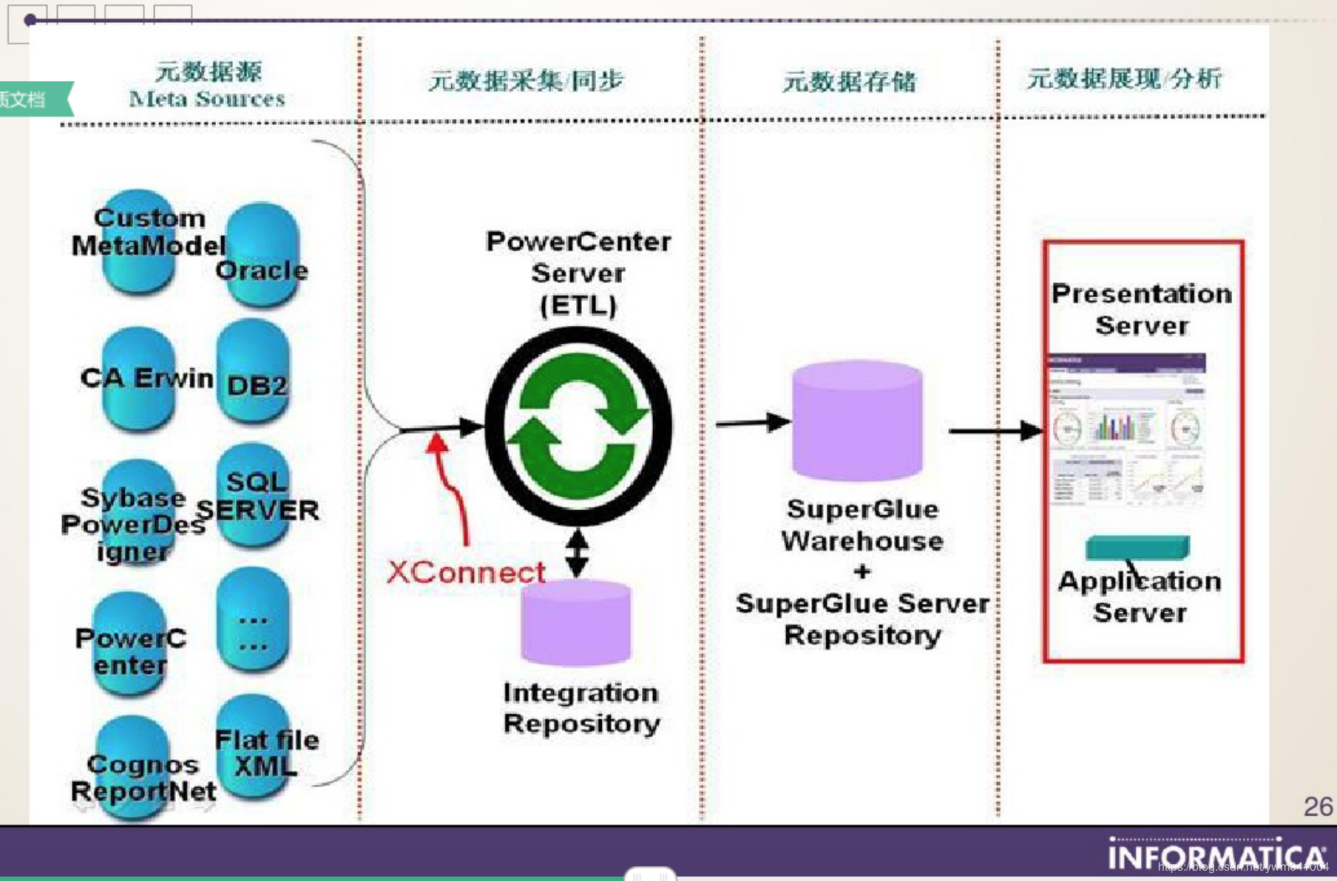


如图所示，与元数据相关的数据仓库工具大致可分为四类：

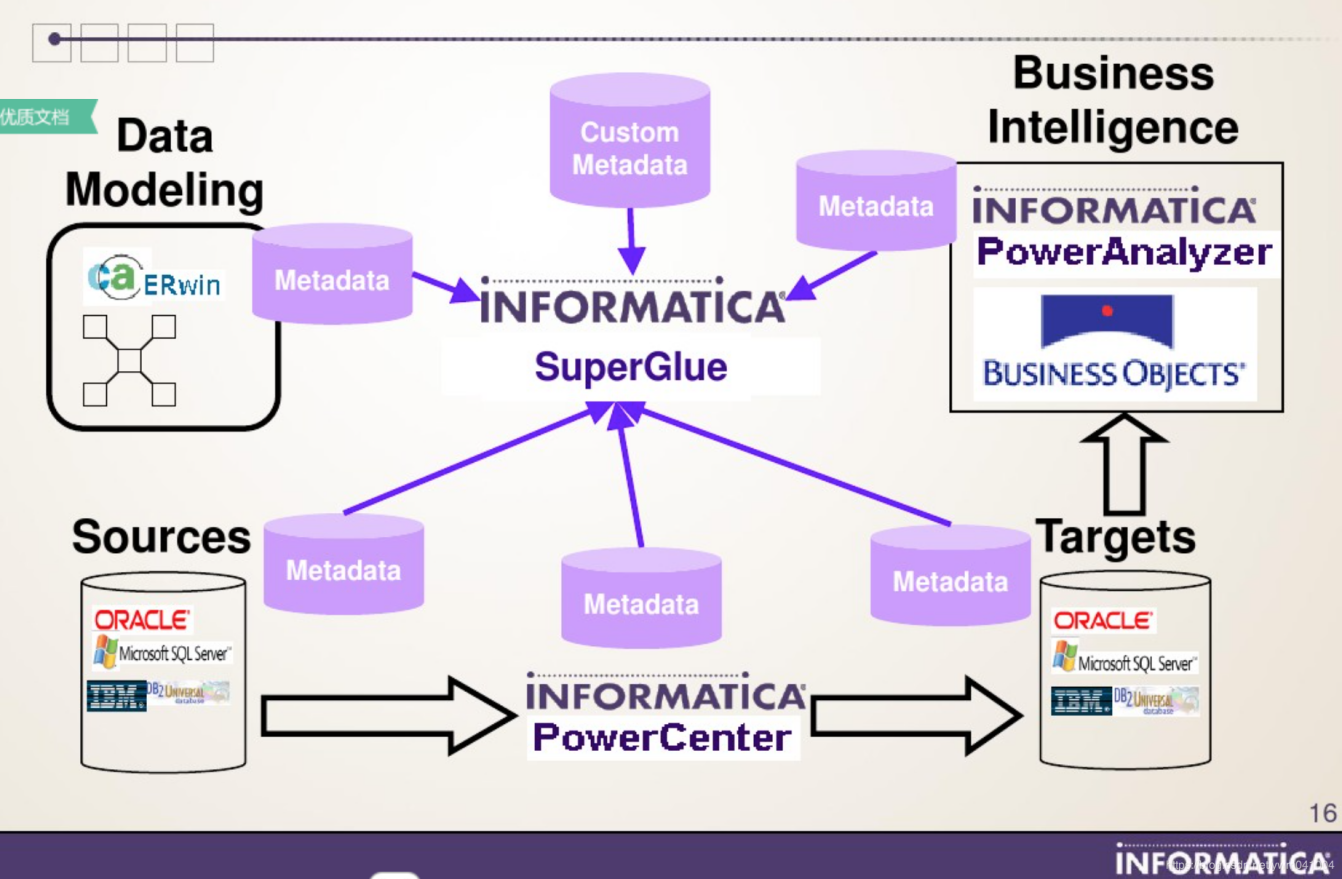
1. 数据抽取工具;  
    把业务系统中的数据抽取、转换、集成到数据仓库中，如Ardent的DataStage、Pentaho的开源ETL产品Kettle、ETI的Extract等。这些工具仅提供了技术元数据，几乎没有提供对业务元数据的支持。
2. 前端展现工具：  
    包括OLAP分析、报表和商业智能工具等，如Cognos的PowerPlay、Business Objects的BO，以及国内厂商帆软的FineBI／FineReport等。它们通过把关系表映射成与业务相关的事实和维来支持多维业务视图，进而对数据仓库中的数据进行多维分析。这些工具都提供了业务元数据与技术元数据相对应的语义层。
3. 建模工具：  
    为非技术人员准备的业务建模工具，这些工具可以提供更高层的与特定业务相关的语义。如CA的ERwin、Sysbase的PowerDesigner以及Rational的Rose等。
4. 元数据存储工具：  
    元数据通常存储在专用的数据库中，该数据库就如同一个“黑盒子”，外部无法知道这些工具所用到和产生的元数据是如何存储的。还有一类被称为元数据知识库（Metadata Repository）的工具，它们独立于其它工具，为元数据提供一个集中的存储空间。这些工具包括微软的Repository，Ardent的MetaStage和Sybase的WCC等。
5. 元数据管理工具：  
    目前国内的元数据管理工具大概有三类。一是像IBM、CA等公司都提供的专门工具，比如IBM收购Ascential得到的MetaStage，CA的DecisionBase都是如此；二是像DAG的MetaCenter，开源产品Pentaho Metadata，它们不依托于某项BI产品，是一种第三方的元数据管理工具；三是像普元、石竹这样的集成商也有自己的元数据管理工具：普元MetaCube、新炬网络元数据管理系统、石竹MetaOne等。  
   专门的元数据管理工具，对自家产品兼容较好，一旦涉及跨系统管理，就不尽如人意了。从国内的实际应用来看，DAG的MetaCenter这一工具使用最多，目前所看到的在电信、金融领域建设的元数据管理项目基本上都是应用了这一产品。  
   我从互联网上搜索了几乎所有的元数据厂家：Pentaho开源的MetaData产品，支持源码下载试用，可以进行集成开发；普元MetaCube下载后，配置麻烦，目前为止还没有调通；其他公司产品均不提供下载试用。

## 元数据管理标准

没有规矩不成方圆。元数据管理之所以困难，一个很重要的原因就是缺乏统一的标准。在这种情况下，各公司的元数据管理解决方案各不相同。近几年，随着元数据联盟MDC（Meta Data Coalition）的开放信息模型OIM（Open Information Model）和OMG组织的公共仓库模型CWM（Common Warehouse Model）标准的逐渐完善，以及MDC和OMG组织的合并，为数据仓库厂商提供了统一的标准，从而为元数据管理铺平了道路。  
从元数据的发展历史不难看出，元数据管理主要有两种方法：



对于相对简单的环境，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。  
 对于比较复杂的环境，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，然后，通过建立标准的元数据交换格式，实现元数据的集成管理。



目前OMG家的CWM（Common Warehouse MetaModel）标准已成为元数据管理界的统一标准：  
 OMG是一个拥有500多会员的国际标准化组织，著名的CORBA标准即出自该组织。公共仓库元模型（Common Warehouse Metamodel）的主要目的是在异构环境下，帮助不同的数据仓库工具、平台和元数据知识库进行元数据交换。2001年3月，OMG颁布了CWM 1.0标准。CWM模型既包括元数据存储，也包括元数据交换，它是基于以下三个工业标准制定的：

UML：它对CWM模型进行建模。  
 MOF（元对象设施）：它是OMG元模型和元数据的存储标准，提供在异构环境下对元数据知识库的访问接口。  
XMI（XML元数据交换）：它可以使元数据以XML文件流的方式进行交换。

## 元数据管理功能

1. 数据地图  
   数据地图展现是以拓扑图的形式对数据系统的各类数据实体、数据处理过程元数据进行分层次的图形化展现，并通过不同层次的图形展现粒度控制，满足开发、运维或者业务上不同应用场景的图形查询和辅助分析需要。
2. 元数据分析  
   血缘分析  
   血缘分析（也称血统分析）是指从某一实体出发，往回追溯其处理过程，直到数据系统的数据源接口。对于不同类型的实体，其涉及的转换过程可能有不同类型，如：对于底层仓库实体，涉及的是ETL处理过程；而对于仓库汇总表，可能既涉及ETL处理过程，又涉及仓库汇总处理过程；而对于指标，则除了上面的处理过程，还涉及指标生成的处理过程。数据源接口实体由源系统提供，作为数据系统的数据输入，其它的数据实体都经过了一个或多个不同类型的处理过程。血缘分析正是提供了这样一种功能，可以让使用者根据需要了解不同的处理过程，每个处理过程具体做什么，需要什么样的输入，又产生什么样的输出。
3. 影响分析  
   影响分析是指从某一实体出发，寻找依赖该实体的处理过程实体或其他实体。如果需要可以采用递归方式寻找所有的依赖过程实体或其他实体。该功能支持当某些实体发生变化或者需要修改时，评估实体影响范围。
4. 实体关联分析  
   实体关联分析是从某一实体关联的其它实体和其参与的处理过程两个角度来查看具体数据的使用情况，形成一张实体和所参与处理过程的网络，从而进一步了解该实体的重要程度。本功能可以用来支撑需求变更影响评估的应用。
5. 实体差异分析  
   实体差异分析是对元数据的不同实体进行检查，用图形和表格的形式展现它们之间的差异，包括名字、属性及数据血缘和对系统其他部分影响的差异等,在数据系统中存在许多类似的实体。这些实体（如数据表）可能只有名字上或者是在属性中存在微小的差异，甚至有部分属性名字都相同，但处于不同的应用中。由于各种原因，这些微小的差异直接影响了数据统计结果，数据系统需要清楚了解这些差异。本功能有助于进一步统一统计口径，评估近似实体的差异
6. 指标一致性分析  
   指标一致性分析是指用图形化的方式来分析比较两个指标的数据流图是否一致，从而了解指标计算过程是否一致。该功能是指标血缘分析的一种具体应用。指标一致性分析可以帮助用户清楚地了解到将要比较的两个指标在经营分析数据流图中各阶段所涉及的数据对象和转换关系是否一致，帮助用户更好地了解指标的来龙去脉，清楚理解分布在不同部门且名称相同的指标之间的差异，从而提高用户对指标值的信任。
7. 辅助应用优化  
   元数据对数据系统的数据、数据加工过程以及数据间的关系提供了准确的描述，利用血缘分析、影响分析和实体关联分析等元数据分析功能，可以识别与系统应用相关的技术资源，结合应用生命周期管理过程，辅助进行数据系统的应用优化.
8. 辅助安全管理  
   企业数据平台所存储的数据和提供的各类分析应用，涉及到公司经营方面的各类敏感信息。因此在数据系统建设过程中，须采用全面的安全管理机制和措施来保障系统的数据安全。  
   数据系统安全管理模块负责数据系统的数据敏感度、客户隐私信息和各环节审计日志记录管理，对数据系统的数据访问和功能使用进行有效监控。为实现数据系统对敏感数据和客户隐私信息的访问控制，进一步实现权限细化，安全管理模块应以元数据为依据，由元数据管理模块提供敏感数据定义和客户隐私信息定义，辅助安全管理模块完成相关安全管控操作。
9. 基于元数据的开发管理  
   数据系统项目开发的主要环节包括：需求分析、设计、开发、测试和上线。开发管理应用可以提供相应的功能，对以上各环节的工作流程、相关资源、规则约束、输入输出信息等提供管理和支持。

# 元数据架构



# Atlas

官方网站：<http://atlas.apache.org/#/>

## 介绍

Atlas是Hadoop的数据治理和元数据框架

## 总览

Atlas是一组可扩展和可扩展的核心基础数据治理服务，使企业能够有效，高效地满足Hadoop中的合规性要求，并允许与整个企业数据生态系统集成。

Apache Atlas为组织提供了开放的元数据管理和治理功能，以建立其数据资产的目录，对这些资产进行分类和治理，并为数据科学家，分析师和数据治理团队提供围绕这些数据资产的协作功能。

## 特性

### 元数据类型和实例

* 各种Hadoop和非Hadoop元数据的预定义类型
* 能够定义要管理的元数据的新类型
* 类型可以具有原始属性，复杂属性，对象引用；可以继承其他类型
* 类型的实例（称为实体）捕获元数据对象详细信息及其关系
* REST API与类型和实例配合使用，可以更轻松地集成

### 分类

* 动态创建分类的能力-如PII，EXPIRES\_ON，DATA\_QUALITY，SENSITIVE
* 分类可以包含属性-例如EXPIRES\_ON分类中的expiry\_date属性
* 实体可以与多种分类相关联，从而使发现和安全实施更加容易
* 通过沿袭传播分类-自动确保分类遵循数据经过各种处理的过程

### 血统

* 直观的UI，可查看数据沿各种流程移动的沿袭
* REST API访问和更新沿袭

### 搜索发现

* 直观的用户界面，可按类型，分类，属性值或自由文本搜索实体
* 丰富的REST API，可通过复杂条件进行搜索
* 类似于SQL的查询语言以搜索实体-域特定语言（DSL）

### 安全和数据屏蔽

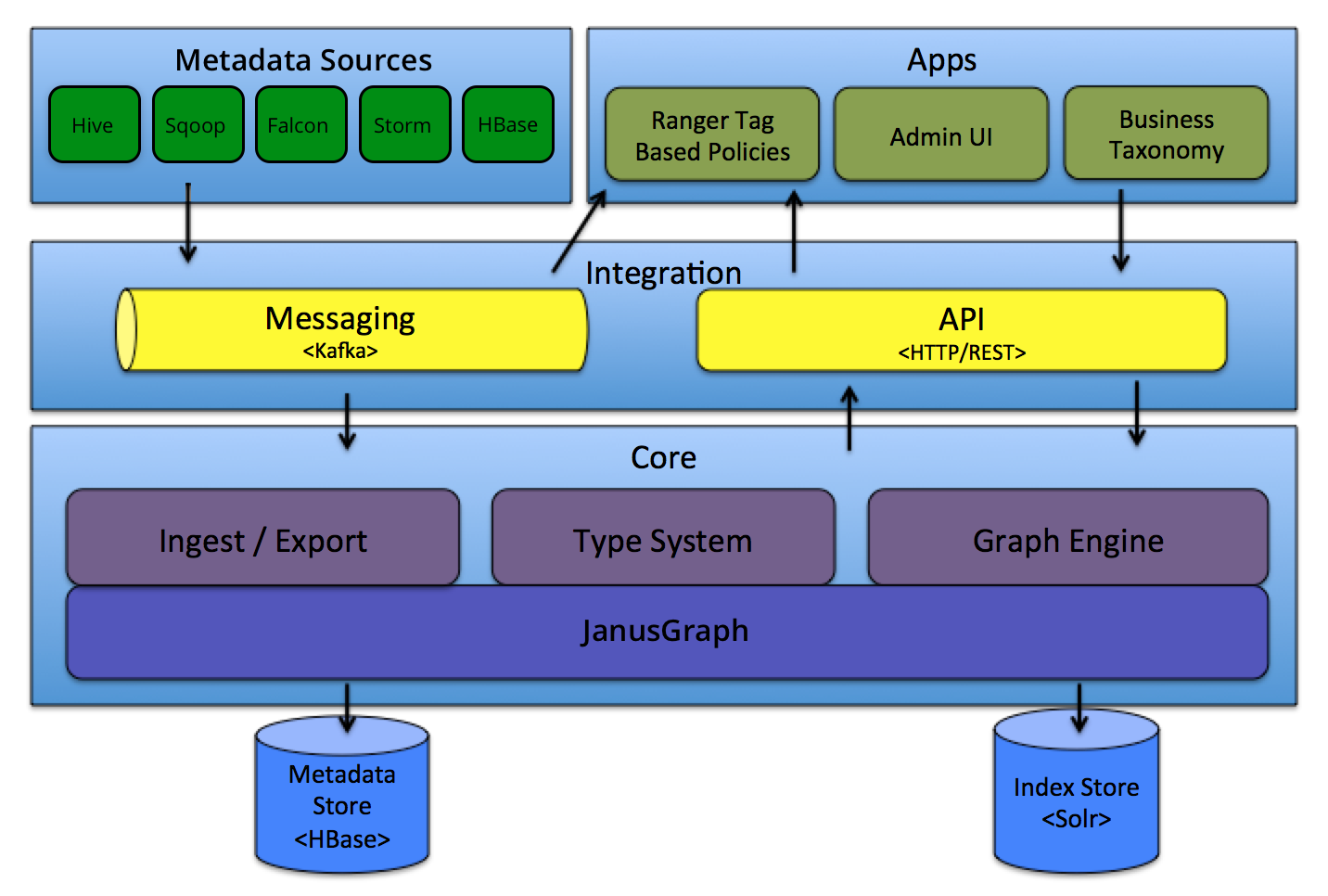
* 用于元数据访问的细粒度安全性，使您能够控制对实体实例的访问以及诸如添加/更新/删除分类的操作
* 与Apache Ranger的集成可基于与Apache Atlas中的实体相关联的分类对数据访问进行授权/数据屏蔽。例如：

谁可以访问分类为PII，敏感的数据

客户服务用户只能看到分类为NATIONAL\_ID的列的最后4位数字

# Atlas架构

## 架构概述



Atlas的组件可以分为以下主要类别：

### 核心

Atlas核心包括以下组件：

**类型系统**：Atlas允许用户为他们要管理的元数据对象定义模型。该模型由称为“类型”的定义组成。称为“实体”的“类型”的实例表示所管理的实际元数据对象。类型系统是允许用户定义和管理类型和实体的组件。开箱即用的Atlas管理的所有元数据对象（例如，像Hive表）都使用类型进行建模并表示为实体。为了在Atlas中存储新类型的元数据，需要了解类型系统组件的概念。

要注意的一个关键点是，Atlas中建模的一般性质允许数据管理员和集成商定义技术元数据和业务元数据。也可以使用Atlas的功能在两者之间定义丰富的关系。

**Graph Engine**：在内部，Atlas保留使用Graph模型管理的元数据对象。这种方法提供了极大的灵活性，并可以有效处理元数据对象之间的丰富关系。图引擎组件负责在Atlas类型系统的类型和实体以及基础图持久性模型之间进行转换。除了管理图形对象外，图形引擎还为元数据对象创建适当的索引，以便可以有效地搜索它们。Atlas使用JanusGraph存储元数据对象。

**提取/导出**：“提取”组件允许将元数据添加到Atlas。同样，导出组件将Atlas检测到的元数据更改公开为事件。消费者可以使用这些更改事件来实时响应元数据更改。

### 集成

用户可以使用两种方法在Atlas中管理元数据：

**API**：Atlas的所有功能都通过REST API向最终用户公开，该API允许创建，更新和删除类​​型和实体。它也是查询和发现Atlas管理的类型和实体的主要机制。

**消息传递**：除了API之外，用户还可以选择使用基于Kafka的消息传递接口与Atlas集成。这对于将元数据对象传达给Atlas以及从Atlas消耗元数据更改事件（用于构建应用程序）都是很有用的。如果希望使用与Atlas的松散耦合集成，以实现更好的可伸缩性，可靠性等，则消息传递接口特别有用。Atlas使用Apache Kafka作为通知服务器，用于挂钩和元数据通知事件的下游使用者之间的通信。这些事件由钩子和Atlas编写，涉及不同的Kafka主题。

### 元数据源

Atlas支持与许多现成的元数据源集成。将来还将添加更多集成。当前，Atlas支持从以下来源提取和管理元数据：

* Hbase
* Hive
* sqoop
* Storm
* Kafka

集成意味着两件事：Atlas原生定义了一些元数据模型来表示这些组件的对象。Atlas提供了一些组件，可以从这些组件（实时或在某些情况下为批处理模式）中提取元数据对象。

### 应用领域

Atlas管理的元数据被各种应用程序使用，以满足许多治理用例。

**Atlas Admin UI**：此组件是一个基于Web的应用程序，允许数据管理员和科学家发现和注释元数据。这里最重要的是搜索界面和类似SQL的查询语言，可用于查询Atlas管理的元数据类型和对象。管理员界面使用Atlas的REST API来构建其功能。

**基于标签的策略**：[Apache Ranger](http://ranger.apache.org/)是针对Hadoop生态系统的高级安全管理解决方案，与各种Hadoop组件广泛集成。通过与Atlas集成，Ranger使安全管理员可以定义元数据驱动的安全策略以进行有效的管理。Ranger是Atlas通知的元数据更改事件的使用者。

## 类型系统

### 总览

Atlas允许用户为他们要管理的元数据对象定义模型。该模型由称为“类型”的定义组成。称为“实体”的“类型”的实例表示所管理的实际元数据对象。类型系统是允许用户定义和管理类型和实体的组件。开箱即用的Atlas管理的所有元数据对象（例如，像Hive表）都使用类型进行建模并表示为实体。为了在Atlas中存储新类型的元数据，需要了解类型系统组件的概念。

### 种类

### 实体

### 属性

### 系统特定类型及其意义