元数据

元数据是个很有趣的话题，几乎所有的在数据仓库应用领域里的工具包括商务智能工具， ETL 工具，数据库和专门的资料库都拥有元数据的解决方案；很多书籍都给出了很好的元数据指导。然而，在数据仓库实现和发展的这些年之后，我们仍然会遇到端到端的元数据的问题。然而，大部分数据仓库还是手工维护那些独立存在于控件中的元数据。本章简单介绍了 ETL 小组作为消费者或厂商应该知道的元数据部分，而不是对元数据知识的夸夸其谈。我们制定了一系列的元数据结构来支持 ETL 工作组的工作。

流程检查

规划与设计：需求/现状 -> *架构* -> *实现* ->发布到生产系统
数据流：*抽取* -> *清洗* -> *规格化* -> *提交*

因为 ETL 系统是数据仓库的核心，它时常承担管理和储存数据仓库大量元数据的职责。可以认为没有比 ETL 系统更好的位置来存储和管理元数据了，这是因为系统环境要完全了解所有数据的细节特性才可以正确的发挥作用。在数据仓库中 ETL 处理程序是元数据最重要的创建者——数据沿袭（the data lineage）。数据沿袭追踪数据从源系统和文件中的精确位置直到最终被装载之前。数据血统包括源数据库系统的数据定义和在数据仓库中最终静止状态。如果你使用一个 ETL 工具，除了数据沿袭之外，属性也被存放在数据仓库中。但是 ETL 环境真的能在数据仓库中捕获和管理元数据？目前还是不可以。

如果你想在数据仓库元数据上探究细节问题，元数据解决方案是一个值得探究的方向。使用元数据模型，资料库，XML 和企业入口按照 Adrienne Tannenbaum(Addison Wesley 2002)要求生成信息。利用元数据，知识库，XML，和通过按照 Adrienne Tannenbaum (Addison Wesley 2002)的要求规划入口产生信息我们现在要开始准确的定义元数据的具体含义了。.

定义元数据

在执行元数据方面落后的一个主要原因是元数据的精确定义是模糊不清的，而且精确定义元数据是一个非常困难的工作。在 1998 年我们已经在《数据仓库生命周期工具箱》中定义过元数据。但是当看到这些词的时候，我们还是会感到吃惊。以下是我们所涉及的：

什么是元数据

元数据是数据仓库世界中令人惊异的话题。鉴于还不是很精确的了解它，不明确它在哪里，我们将会花费更多的时间来了解它，更多的时间来考虑它，如果我们没有什么收获的话会更加得感到羞愧。多年以前，我们很坚决地认为，元数据是有关于数据的数据。但这并不是很有帮助,它并没在我们的脑海中描绘出元数据里面准确的清晰结构。这种失真的观点最近逐渐清晰，而且我们更确定的发现了更多关于后台元数据和前端元数据的信息。后台元数据是与处理相关的，指导抽取清洗装载工作。前端元数据更偏向描述性和使我们的查询工具和报表工具更加稳定。虽然，处理和描述性元数据交叠，但是在我们的头脑中把它们区分开来是很有用处的。

后台帮助 DBA 将数据加载入数据仓库，同时这也是业务用户感兴趣的地方，他们会问

起数据是从什么地方来等问题。前端元数据主要是为终端用户服务的，他的好处不仅可以使我们工具功能运行更加平稳，而且可以作为描述所有数据的业务数据字典。

即使这些定义很有帮助，可还不能指导数据仓库管理者到底应该做什么。但可以把传统的 IT 观点运用元数据上。至少我们应该：

- 制定完善的良好的注释目录
- 判定每一部分究竟有多重要
- 指定专人负责
- 判定一个和谐一致的工作规则
- 判定是否自己构建还是购买
- 专门存储用以备份和恢复
- 对需要他的人们开放
- 保证质量，保持完整和及时更新
- 实时控制

所有这些任务的文档都可以很好地解决问题了。

到现在为止，麻烦只有一个，我们还没有真正指出元数据到底是什么。我们注意到在上面的列表中的最后一项实际上并不是元数据，它是关于元数据的数据。藉由深入挖掘，我们了解我们或许需要元数据的元数据。

源系统元数据

为了更好的理解他，让我们试着给所有的元数据类型制作一张列表。当然，第一次是不会很成功的，但我们会从中学到很多。先让我们看看源系统，它可以是主机系统，非大型机，用户桌面，第三方数据提供者，或在线数据源。我们这里假设在数据源端只做读取数据和将数据抽取到数据分段区域，数据分段区域可以是主机或下游机器。

数据源规范：

- 知识库
- 来源计划
- 拷贝纪录
- 所有权或第三方来源计划
- 存档主机数据的原由格式
- 相关源系统数据表和 DDL
- 电子数据表
- Lotus Notes 数据库
- 表达图形（例如，PowerPoint）
- URL 来源规格说明：
- 来源描述信息：
  - 各来源的所有权描述信息
  - 各来源的业务描述：
  - 最初来源的更新频率
  - 各来源使用的司法局限性
  - 存取方法，读取权限，和各来源存取口令
- 处理信息：
  - 主机或源系统工作计划
  - 使用 COBOL/JCL 或 C 或 Basic 或其他语言来实现抽取

- 如果使用工具的话，那么还包括这种自动化抽取工具的设置
- 特殊抽取的结果，包括抽取时间和工作完成率

数据集结元数据

现在我们列出所有数据集结元数据。我们在主机上通过手工代码 COBOL 或使用自动化工具。或者我们把 FLAT 文件或多或少抽取到一台不同机器的单独数据分段存储区。无论哪种情况，我们都不得不关心元数据的描述

数据需要的信息：

- 数据传输计划和特殊传输结果
- 数据传送区的文件用法，包括持续时间消耗和所有者信息

维度表管理：

- 规范化维度的定义和规范化事实的定义
- 关联源的 Job 规范，剥除域，查找属性
- 降低每一个引入描述属性维度规则的变化速度（例如，重写创建新纪录或创建新域）
- 每一个生产键所分配的当前代理键，也包括在内存中执行映射的查找表。
- 前一天生产维度的副本，作为比较差异的基础

转换和聚合：

- 数据清洗规范
- 数据增加和映射转换（例如，扩展缩写和提供详述）
- 为数据挖掘所准备的数据转换（例如：解释空值和数字的度量 ）
- 目标计划设计，源和目标数据流，和目标数据所有权
- 数据库管理系统的导入脚本
- 聚合定义
- 聚合用法统计，基础数据表使用统计，以及潜在的聚合
- 聚合更改日志

审计，工作日志和文档：

- 数据沿袭和审计纪录（数据确切来源和来源时间）
- 数据转换时间日志
- 数据转换运行时间日志，成功记录摘要，时间戳
- 数据软件版本号
- 抽取过程的业务描述
- 抽取文件，抽取软件和抽取元数据的安全设置
- 数据转换的安全设置(授权口令)
- 数据分段传送区存档文件日志和恢复进程
- 数据分段传送区存档文件日志安全设置

DBMS 元数据

一旦我们最终决定将数据转换导入到数据仓库或数据集市的数据库管理系统，元数据的另一部分开始起作用：

- 数据库管理系统系统表目录
- 分区设置

- 索引
- 磁盘带区规范
- 程序处理提示
- 数据库管理系统级别和安全的权限和授权
- 视图的定义
- 存储过程和 SQL 管理脚本
- 数据库备份，状态备份程序和备份安全性

前端元数据

在前端，我们也扩展了元数据，包括：

- 纵列表分组的业务名称和描述等等
- 模糊查询和报表定义
- 连接规范工具设置
- 灵活打印规范说明（以一种易读的方式重新标记区域）
- 最终用户文档和培训帮助，包括供应商提供和 IT 提供
- 网络安全用户授权
- 网络安全鉴定证书
- 网络安全使用统计，包括试图读取的日志和用户 ID
- 个人使用者描述包括人力资源链接
- 提高影响读取权限的传输流畅性
- 连接到合约人和合作者可以追踪读取权限的影响范围
- 数据原理，数据库表，和视图报告的使用和读取映射
- 资源回收统计
- 收藏的网页（作为所有数据仓库访问的范例）

现在我们可以发现为什么我们不能准确全面地描述出什么是元数据了，它包含了所有的东西，甚至是数据本身。突然之间，数据似乎是最简单的部分。在某种程度上，元数据是数据仓库的 DNA。它定义了所有元素和各元素之间如何协同工作。

所以，你如何捕获和管理这些元数据单元？你不能，至少 ETL 团队是不能完成的。纵观过去的几十年里社团协会，联盟，委员会，企业联合起来解决有关元数据的困惑。但是直到今天，还没有形成一个较为普遍的解决方案。我们发现，作为一名 ETL 团队的成员，你需要一定的元数据做你的工作。它能很方便的聚焦到我们刚才所列出的项目中，并且把那些项目分成以下三类：

1 业务元数据 在业务层面上描述数据的含义

2 技术元数据 描绘数据的技术方面包括数据的属性，例如数据类型，长度，沿袭，数据评估结果等等

3 过程处理元数据 介绍运行 ETL 处理的统计信息，包括度量标准比如纪录导入成功，记录丢弃，处理时间等等

除了这三种元数据之外我们还应该考虑到元数据的另一个方面：标准。标准是 IT 界各企业共同合作和维持的另一项尝试。在第 10 章我们会定义元数据管理员的角色，并给这一角色指定一系列责任。你们的组织在适当的位置上很可能会有很多标准，而你也可以通过本书所介绍的方法采用适合的数据仓库和 ETL 标准。如果你找的是更深一层次的元数据标准研究，我们会在下一部分给你详细的介绍。

当你阅读这一章节的时候，请参照图 9.1 辅助理解。图 9.1 向我们展示了 3 种主要的 ETL

系统元数据，并在正文中会讨论各自部分的元数据表。

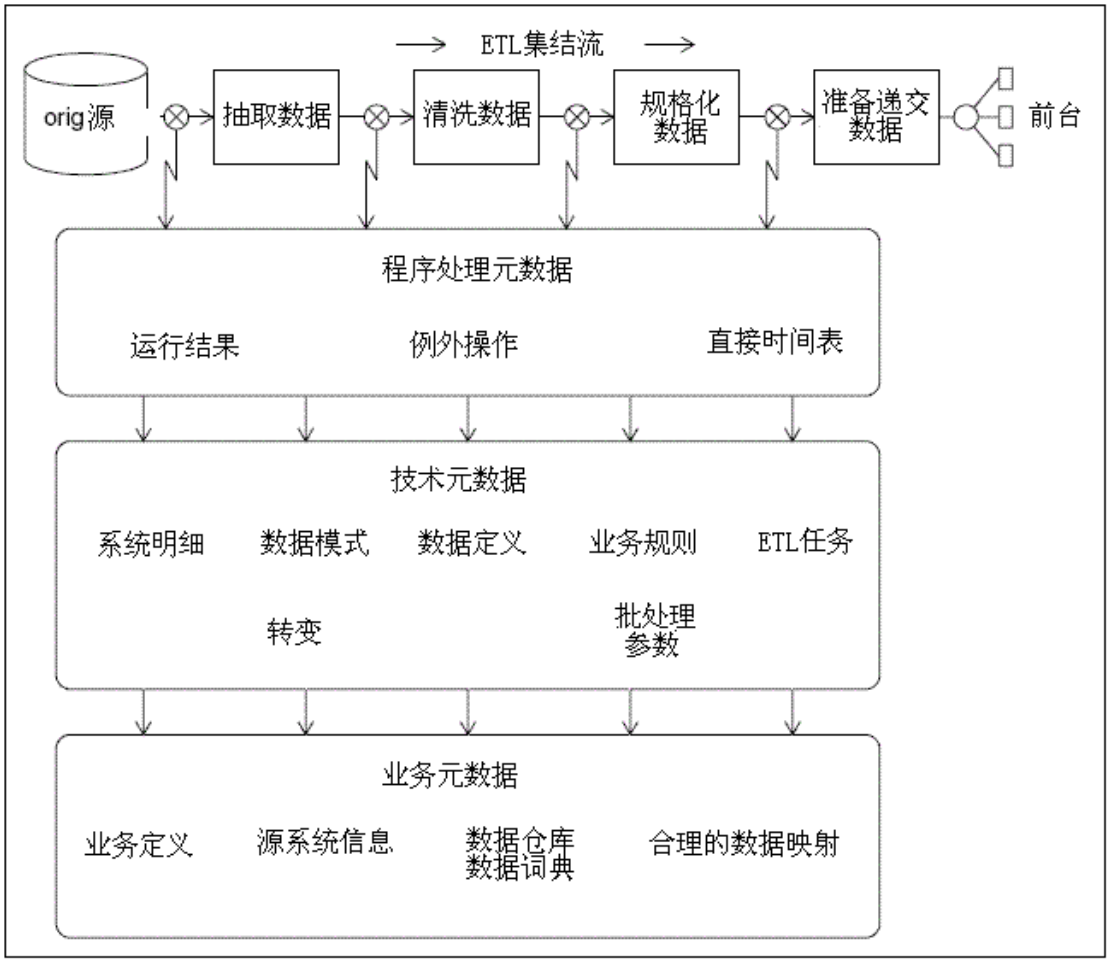

图 9.1　数据仓库后端元数据源

这张清单列出了到目前为止在本书中提到的希望使用和收集元数据的所有地方：

- 每一个数据阶段的起源和处理步骤（第一章）
- 元数据资料库作为第三方 ETL 工具的一个有利条件（第一章）
- 元数据需求构架：源表，清洗和处理（第一章）
- 为终端用户介绍有用的元数据（第二章）
- 抽取转换应用（第二章）
- 规范化元数据(第二章)
- 元数据的 XML 描述（第二章）
- 平面文件中元数据的缺失（第二章）
- 分析元数据的效果（第二章）
- 计划创建描述沿袭的元数据，业务定义，技术定义以及处理过程（第二章）
- 逻辑数据视图（第三章）
- 抽取过程中源数据的捕获计算（第三章）
- 源数据库描述（第三章）
- ETL 工具读取 ERP 系统元数据（第三章）
- 数据评估的结果（第四章）
- 错误事件跟踪事实表（第四章）
- 审计维度（第四章）

- 代理键最高值（第五章）
- 聚合数据（第六章）
- 处理数据搭建 OLAP 架构（第六章）
- 装载到控制文件（第七章）
- 支持恢复处理的元数据（第八章）
- ETL 系统参数（第八章）
- 任务依赖（第八章）
- 任务操作性统计，比如性能和资源的使用（第八章）
- 元数据资料库报告（第八章）
- 数据表净化规则（第八章）

我希望看到如此之长的清单时不要沮丧，本章的其他部分将会为特定元数据追踪信息。

业务元数据

由谁在数据仓库中负责元数据，这经常是个有争议的话题。有人说它是数据仓库业务分析人员的任务，应该在收集需求的阶段被创建。另一些人认为，源系统业务分析员应该创建业务期限，这是因为大多数数据仓库属性是在源系统中建立的。但是，还有一部分人认为创建和维护业务元数据应该是数据建模人员的一部分工作，因为它本身就是逻辑数据模型的一部分。

你无法平息这些争论，但是有一些业务元数据，ETL 团队会受到它的影响，而且还需要维护它来准确反映事实。ETL 团队不用去理会如何生成业务元数据，但如果发生变化，一定需要适当的人员来沟通。从 ETL 的角度来看，业务元数据是一个代理的元数据。代理元数据是通过一个系统获得，可以直接应用于另一个系统而不需要外在处理。一些商务智能工具被设计用来读取 ETL 资料库中的业务逻辑定义，并把它展现给用户。为数据和数据的元数据提供一个一站式服务。

为了让这更有趣些，我们必须要让你明白数据仓库可以理解为一个不同业务定义的集合场所。这些定义是为同一个属性服务的。记住，通过设计，在数据仓库中的数据可来源于多个系统，例如，市场部门定义了一个客户，这个客户有一个已经注册的账户。然而销售部门会认为只有实际发生购买的才可以成为客户。作为一个被推荐的实例，数据仓库管理员应该把所有重叠元素的所有者集中在一起，让他们从企业的角度接受一个单一的概念。我们在本书中描述的这个过程被看作是规范化业务定义、标签和估量标准，这也许说它是来源于一整个新系统或者是来源于众多源系统中的一个。定义的企业规范存放在数据仓库和 ETL 工具中。

ETL 小组作为数据仓库后台的一部分，不应该涉及到创建业务元数据库。然而，你应该理解所造数据的用途，同时当你需要业务定义的时候可以参考一下。

业务定义

数据仓库团队经常要设计查询性能的数据库结构，和优化装载性能的 ETL 处理程序。一旦数据库设计完成，导入数据，小组的工作会集中到数据仓库的商务智能上，开发良好的用户界面，生成漂亮的图形化报表。但是对业务，在数据仓库方法中最重要的因素是对可用

元素的定义。如果用户不能理解数据，对数据产生误解，那么定义的原理对组织来说完全没有意义。业务定义对于数据仓库来说是至关重要的。不仅仅是终端用户需要业务定义，ETL 也需要业务定义给出他们正在操作数据的上下文关系。如果你试图在完全定义业务之前描述数据模型和数据沿袭的话，你将会意识到业务定义对于 ETL 团队来说是多么的重要。如果你急于完成数据沿袭以便 ETL 开发工作尽早开始，你将会被迫向 ETL 团队频繁解释数据模型中的主要数据元素的目的用途。因为他们没有足够的信息对数据开展工作。因此，在 ETL 小组获取数据沿袭之前，开发的重点应该从加快数据沿袭中转移出来，集中整合业务定义之间的关联和数据仓库的数据元。一个典型的业务定义矩阵包括 3 各主要组成部分:

- 物理表和列名称　数据仓库中数据元素的业务解释是基于数据库中真实的表和列名称。如果商务智能工具仅展现业务，而完全隐藏数据结构的物理实例的话，那么物理名称不需要展现给最终用户。然而 ETL 团队只处理物理名称，这就需要把物理名称和数据元素的业务定义适当的联系在一起。
- 业务列名称　数据库所存储的数据元素在技术上是由前缀后缀和下划线组成的。业务组需要这种技术性名称和有实际意义名称之间的翻译。例如，业务名称为 EMP STTS CD，有可能是员工身份代码或者仅仅是员工状态。我们无法在这种模糊不清的情况下工作。记住，业务名称是要被商务智能工具展现出来的。此外，业务名称经常会成为用户报表的行和列标题。
- 业务定义　业务定义是指描述业务属性含义的一两句话。数据仓库中的每个属性一定会有一个业务定义。如果无法用业务定义属性，那么通常意味着这个属性没有分析价值，有可能不需要存放在数据仓库中。如果业务上要求它必须存在于数据仓库中，那么它一定会有业务定义与之相联系。

业务定义矩阵可以简单到是三列电子表格。然而，你应该努力尝试让各种特殊的元数据在技术环境允许的情况下尽可能的集中和共享。事实上，所有的 ETL 工具都支持捕获和存储元数据。ETL 工具应该和数据模型工具、数据库一起获得业务定义，并通过商务智能工具把业务名称和业务定义展现给最终用户。

源系统信息

ETL 团队需要了解在数据仓库中读取每一张表的所有隐藏细节。就像你需要雇用一个人来填补目前 ETL 团队中的空缺，在雇用员工之前，HR 部门会寻找候选人，预先筛选他们，然后把合适的人选介绍给你，安排正式的面试。所有你可以保证这些人是适合这个空缺的职位的。在面试过程中，你在认为候选人适合该职位之前就也可发现一些弱点。当迁移到数据仓库的时候，数据模型就好像那个空缺的职位，数据仓库架构师寻找并预先筛选出数据源，然后数据必须要经过分析来发现它们自身的弱点。制定数据改造计划，使数据能够较为理想的被导入到数据仓库中。一些数据可能会非常完美，其它数据可能就需要做些转换或者因为数据质量的问题而被丢弃。当你分析源系统时，你需要特定的数据仓库，至少你需要以下元数据属性。

- 数据库或文件系统 当涉及到源系统或文件的时候，这个名称经常会用到。这不是一个技术上的服务器或者数据库实例。像有关销售的数据库或资产管理系统这样的名称对于这样的元数据片是很常见的。
- 表规范 ETL 团队一定要了解表的用途，表的大小，主键和预备键，和所有列的清单。
- 排异处理规则 你一定要多了解数据质量的相关内容，仔细考虑 ETL 程序是如何操作数据的。

- 业务定义 非常有用。尽你最大努力来做好这些，那种一两句的定义对于你了解数据的意义是没有任何价值的。
- 业务规则 每个表都有自己的业务规则。对于理解数据和测试异常时，业务规则十分有必要的。每个业务规则都会被测试，每个规则的异常都会记录成文档，被源数据系统或 ETL 程序处理。业务规则包括数据表插入新纪录，更新和删除的记录。如果你很幸运，业务规则会在源数据库管理系统(DBMS)以参照完整性，强制检查和触发的形式存在。

在数据仓库项目中数据分析阶段需要大量时间研究源系统。缺少源系统元数据会导致数据仓库小组的额外研究并且要解决更多的故障。为了缩减成本，所有的源系统元数据，必须在 ETL 程序开发之前提供给 ETL 团队。

数据仓库数据字典

当我们谈到数据字典时，并不是指 DBMS 的目录。数据仓库数据字典是关于所有数据元素和他们业务定义的清单。与源系统业务定义相似，数据仓库数据字典包括物理表和列名称，业务名称和业务定义。对于数据仓库，数据业务元数据电子表格就足够了。很多数据仓库环境依赖于 ETL 资料库来存储数据字典，这是因为商务智能工具需要读取元数据来进行展示工作。

很多商务智能工具与 ETL 元数据库紧密相连的。当你选择工具时，要搞清楚你的 ETL 工具是否具有开放型资料库可以被查询工具读取或至少有一个适配器或代理来完成这个工作。

逻辑数据视图

逻辑数据视图是 ETL 工作组的生命线。第三章讲述了 ETL 规范角度上的逻辑数据视图的信息。从元数据的角度上看，逻辑数据视图是由从源到目的的映射，从逻辑上解释了,数据从源系统中抽取出来，到装载到数据仓库中，数据到底发生了什么。

逻辑数据映射是元数据的重要部分。ETL 工作组首先使用文档作为功能性规范来创建物理 ETL Job。然后与终端用户确认工作。当用户接受测试阶段发生问题时，文档可以提供指导。当数据进行质量保证测试的时候，当 ETL 给质量保证组提供各种视图时，文档也非常有用。最后，ETL 团队将文档提交给数据库管理组，并提供数据转换的信息，在程序发生问题的时候数据库管理员可以及时维护。

交叉参考：参考第三章在逻辑数据映射中创建准确元数据元素，如何创建，维护和使用信息。

技术元数据

技术元数据服务于多种用途，是 ETL 团队所涉及的元数据中最有趣的一个。它包括了到列名称，数据类型，存储和 RAID 矩阵的配置。作为 ETL 团队的一个成员，你不需要太关心类似硬件的配置这些方面。数据仓库中数据元素的物理属性是最重要的。为了得到基本

的元数据，需要了解你的源系统和目标数据仓库的物理数据模型。如果你的源包括平面文件，你需要每一个文件的文件格式。

系统详细目录

ETL 团队必须要十分了解数据仓库中每个系统的技术元数据，才能正确构建物理 ETL 工作。数据的技术定义或许是技术人员被问起有关元数据时首先想到的话题。毕竟，这才是真正的描绘数据的数据，它是数据的容器和框架结构。ETL 团队必须至少了解数据定义的三种环境：

- 源数据库
- 分段处理区域表（例如，抽取，清洗，转化，确认，为传输作准备）
- 数据仓库展现区

最好为每个环境提供一个实体关系图，每个系统至少应有一个清单包含以下元素：

- 表 表或文件的一份详尽清单，这份清单有可能在抽取和装载过程中使用。经常，只提供逻辑数据映射的源系统数据表。有一些关联表没有被列入清单但也是必需的。当源系统有多对多关系的时候，一个设计很好的系统来关联数据表维持这种关系。
- 列 对于每张表，你都会需要一张数据映射所要求列的清单。源系统数据库管理员能够提供你所需要的列的清单。这样会有助于工作。
- 数据类型 数据表每一列都有一个数据类型。在不同数据库系统有些数据类型不尽相同。很幸运，大部分专门的 ETL 工具可以转换相应的数据类型。例如，SQL Server 中的 INTEGER 类型在导入到 Oracle 时可以自动转成 NUMBER 型。要知道数据库管理员可以自定义数据类型作为用户定义数据类型。用户定义数据类型基于数据库核心数据类型，可以在此基础上扩充其定义，包括列长度，是否可以为空，是否可以有特殊字符，比如电话号码。
- 关系型数据库 关系型数据库支持参照完整性。参照完整性和表之间的关联关系能够保证数据的唯一一致的数据导入。数据之间的关联关系是通过表与表之间的主外键连接展示的。

数据模型

在物理模型图表(规格化的或维度的)上的数据模型其实只是一个元数据展示,而且元数据本身没有什么特别。但是，他们可以成为宝贵的财富，因为它可以使你的 ETL 工作组只用瞥一眼就可以了解和确认关联关系。即使列出的逻辑数据视图,你不能指望它是完全彻底的。我们建议，在你办公室的墙上悬挂上标示所有源系统(数据仓库)的物理模型。在图上标示出通过筛选明确的目标。此外，这样做还有一个直接的好处，可以增强作决定的决心，给项目经理们留下积极的印象。

数据定义

每一个潜在的数据存储中的数据定义应该是一致的。数据每一次进入数据库或者进入文件后，数据质量很容易发生变化。如果在各个环境中的数据定义各不相同，那么 ETL 团队

就要做数据转换来避免数据不一致带来的灾难。除了上一节所列举的属性外,下列数据定义元数据的内容必须提供给 ETL 团队.

- 表名称 表和文件的物理名称
- 列名称 表和文件中列的物理名称
- 数据类型 表的数据类型可以分为以下几类：普通类型包括数字型，字符型，日期型，二进制。在绝大部分关系型数据库中还可以允许自定义数据类型。这种类型是基于普通类型大部分由强制的格式规则定义的。数据类型是相互排他的，不能共存于一个列中。
- 域 数值要进入的列被称为域。域通过外键，检查约束或数据库顶端应用来执行。如果将应用加给域，设计团队必须给 ETL 团队提供一张允许值的清单。
- 参照完整性 如果你的数据来源于数据库，你可以通过外键来指出另一张表的主键来保证数据的唯一性。如果要将参照完整性加给应用层，那么设计组一定要提供参照完整性的规则。数据仓库中的参照完整性被认为不是必要的，这是因为所有的数据都是经过 ETL 处理以受约束的形式进入数据库的，在数据库级不需要强制的约束完整性。
- 约束 约束是业务规则的另一种物理执行。数据库约束可以消除空值，增强外键查找等等
- 缺省值 万一实际值是不可获得的，ETL 元数据的缺省值可以分配为字符串型，数字型，日期型或 bit 型。在源系统中，列的缺省值经常是被分配为数据库级别的。在数据仓库中，缺省值的指派发生在 ETL 处理中。数据仓库中的缺省值最好坚持始终如一的一致性。
- 存储过程 存储过程，存储了已经在数据库中写好的 SQL 语句。可以通过存储过程看到你的源数据是如何使用的。每个数据仓库项目都不可避免的要涉及到源系统的分析型存储过程。
- 触发 当数据库系统中的记录要增加，删除或更新时通过触发自动执行 SQL 程序。像存储过程一样，通过触发，也可以知道数据是如何使用的。触发经常通过向加入到表中的数据增加额外的检查来增强外键的约束。当表中的数据发生变化或被删除时，触发还承担审核表的责任。审核表是数据仓库中删除数据的重要审核来源。

业务规则

业务规则可以分为业务或技术元数据。当谈到业务规则的时候我们喜欢把它归为技术，这是因为它对于 ETL 过程来说实在太重要了，而 ETL 过程不折不扣地属于技术范畴。所有的业务规则都要以编码的形式被包含在 ETL 过程中。业务规则可以包括起始域中的任何允许值，缺省值，和计算。在源系统中，业务规则在存储过程，强制约束或数据库触发中被执行。但是业务规则还是最常出现在应用的程序中。以前的那些系统中，特别是大型机环境，不存在实际的源代码，只有应用的编译部分。在这样的情况下，业务规则很难取到，就要经常进行数据分析，或询问原来编写应用的程序员。业务规则的元数据会在功能性或者技术性文档和本地程序语言的源代码或伪代码之间进行改变。

业务规则必须和逻辑数据视图紧密结合在一起。有时，业务规则会在逻辑数据视图中被忽略直到执行完第一次 ETL 处理之后才被人注意，或被用户在 UAT 中被发现。当有新的业务规则时，逻辑数据视图的元数据必须更新来反映新的规则。

ETL 生成的元数据

到目前为止，我们一直关注的是在 ETL 环境之外生成的元数据或者是其他源系统提供给 ETL 团队的元数据。本章的其余部分将会介绍 ETL 生成的元数据。ETL 团队可以通过这种元数据来管理 ETL 程序，而终端用户或数据仓库成员也可以更好的理解数据仓库中的数据。

当 ETL 物理程序被建立之后，一定要生成明确的元数据来捕获每个处理的内部工作。ETL 元数据可以被分为四类：

- ETL 任务元数据　ETL 任务可以被理解为存储所有转换的容器。任务元数据非常有价值，他包含了数据仓库中各元素数据沿袭。每个 ETL Job 任务——从抽取到装载，包括所有的转换——都可以被 ETL 任务元数据捕获。
- 转换元数据　每个任务都由多个转换组成。任务中数据处理的任何形式都是由专门转换工作来执行的。
- 批处理元数据　批处理是一种执行任务集合的技术。批处理应该具有配置执行连续性和并行处理的能力。而且，包含分支批处理。分支处理在数据仓库中非常常见。你可以有一个装载维度的批处理和一个装载事实的批处理。这些批处理可以集中装载数据到明确的数据集市。批处理被定制为阶段执行或者依照触发计划执行。
- 处理元数据　每当执行批处理，就会生成处理元数据。对于描述数据是否被成功装载到数据仓库中，处理元数据十分重要。

如果你不是一个 ETL 新手的话，你大概会注意到元数据种类并不是按照容器顺序列出的。而是以它们出现或者生成的顺序列出的。例如，每个批处理都包含不同的任务，你在批处理任务之前肯定是应建立好各个任务了。ETL 元数据的每一种类型都包含自己明确的属性，这些属性需要被创建，维护和发布。而并发的部分会检查在 ETL 环境中技术型元数据的特性。为了保持本书的一致性，我们谈到数据映射的物理执行时，把它看作任务，而把它的容器看作批处理。这些术语和一些 ETL 工具一致。我们对于术语的指定并不推断任何的特殊技术。如果你的技术不允许用任务调用从源到目的的物理映射,可以用我们自己定义的替代任务取得元数据

如果你不用某种专门的工具来执行你的 ETL 解决方案，你就没有借口去生成主要元数据。工具意味着减轻工作的负担。没有工具，你将不得不手工生成和维护元数据。如果你必须手工创建元数据，可以在 PVCS 或 SourceSafe 这样的版本控制工具中使用电子表格。

ETL 任务元数据

ETL 元数据可以是一个技术含量非常高的话题，那些终端用户往往是对此不甚理解。那些包含程序代码的从源到目的的映射对于业务用户特别是不懂技术的终端用户来说是非常神秘的。然而，这些映射对于理解数据仓库中数据的真实沿袭是非常重要的。当数据的可信性遭到质疑，或需要提供数据的完整需求时，数据仓库小组也要在元数据中寻求答案。

描述从源到目的映射的最佳办法是利用一个专门的 ETL 工具。这些工具可以给用户提供信息报告。要求你可能的 ETL 产品提供商向你明确地展示他们工具的描述能力。

图 9.2 向我们展示了 ETL 任务元数据被创建，存储，和发布的元素。下面的这些元素需要你去追踪，来帮助管理任务和向你的经理和用户发布他们的身份和功能。

任务名称	任务用途	源表/文件	目的表/文件
browser_platform_dim	加载 browser_platform_dim 表	stg_web_log	browser_platform_dim
claim_reason_dim	加载 claim_reason_dim 表	claim_reason	claim_reason_dim
claim_status_dim	加载 claim_status_dim 表	claim_status	claim_status_dim
claims_agent_dim	加载 claims_agent_dim 表	claims_agent	claims_agent_dim
claims_fact	加载 claims_fact 表	claims_fact	claims_fact
clickstream_customer_dim	加载 clickstream_customer_dim 表	weblogs, customer	clickstream_customer_dim
customer_credit_card_dim	加载 customer_credit_card_dim 表	customer_credit_card	customer_credit_card_dim
customer_dim	加载 customer_dim 表	stg_web_log	customer_dim
date_dim	加载 date_dim 表	date_dim.csv	date_dim
delivery_option_dim	加载 delivery_option_dim 表	delivery_option	delivery_option_dim
mktg_stg_tables	加载 mktg_stg_tables 表	campaigns.csv	mktg_stg_tables
page_dim	加载 page_dim 表	stg_web_log	page_dim
page_events_dims	加载 page_events_dims 表	stg_web_log	page_events_dims
page_events_fact	加载 page_events_fact 表	stg_web_log	page_events_fact
partner_promotion_dim	加载 partner_promotion_dim 表	partner, promotions	partner_promotion_dim
payment_status_dim	加载 payment_status_dim 表	payment_status	payment_status_dim
product_dim	加载 product_dim 表	product	product_dim
server_dim	加载 server_dim 表	weblogs, servers.csv	server_dim
stg_web_log	加载 stg_web_log 表到数据仓库	weblogs	stg_web_log
time_dim	加载 time_dim 表	time_dim.csv	time_dim

图 9.2　ETL 任务元数据

- 任务名称 物理 ETL 任务的名称
- 任务用途 最初中心任务的简要描述
- 源表/文件 所有源数据的名称和路径位置
- 目的表/文件 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 丢弃文件的名称和路径位置。那些来存储在装载过程中没有没载入目的地的数据表或文件，我们称之为丢弃文件
- 预处理前置任务 在任务被执行之前需要处理任务或脚本
- 后置任务 在任务处理之后需要执行的任务或脚本

任务

一个任务是一系列转换的集合，这些转换执行物理上的抽取，转换和装载程序。一个任务的元数据是物理的源到目的映射。任务应该根据装载的目标表或文件来命名。如果你的 ETL 处理有很多分段，每个任务必须包含前缀来暗示它的目的。ETL 任务大致上可以被分为三类：

- 抽取 EXT_<table name>暗示这个任务的主要目的是从源系统中抽取数据
- 中间阶段（例如，清洗和规范化） STG_<table name> STG 前缀意味着这个任务并不涉及到源或目的。他只是一个存在于阶段区的中间过程。如果处理不止一次的涉及到阶段区，那么在前缀后面附加上计数器（例如 STG1, STG2, STG3 等等）
- 目标 TRG_<table name> 意味着任务是装载数据到目标数据仓库。我们以前还见过 FAC_<tablename>和 DIM_<table name>的命名方式来暗示目标表是事实表或维表。我们还没有看到这样做的价值，但是这种习惯可以接受。

转换元数据

转换元数据是关于 ETL 处理构造的信息。ETL 开发人员把大部分时间用于创建和复用数据转换。转换是由客户化函数，存储程序，常规程序组成的。这些程序包含了指针，循环，内存变量。这样很难以文档的形式记录或提供元数据。在 ETL 处理过程中的任何对数据的操作就被认为是转换。如果你用 SQL 来写 ETL 的话，你需要辨别你程序中每一个不同的部分，用转换元数据中常用的属性把他们分类标注。

专门的 ETL 工具为数据仓库环境预先定义了通用转换，并把他们打包提供给使用者。预先建立的转换加速了 ETL 的开发，而且还可以在暗中捕获转换元数据。在大部分 ETL 任务中的通用数据转换包括：

- 源数据抽取 这个抽取可以像 SQL 中的 SELECT 语句一样简单，或者像是 FTP, 或是读取 XML DTD 或大型机的副本卷
- 代理键生成器 这些可以简单调用一个数据库序列或者包含复杂的常规程序来管理内存或包含第三方软件。插入到数据仓库中的最后数字，也是需要维护和呈现的元数据
- 查找 主要用来从事实表装载维度中或阶段区中的参照完整性获得代理键。如果你使用未加工的 SQL, 这将包括所有的内外连接和 IN 语句
- 筛选 这条规则决定了哪些行被装载和导入。元数据是业务规则或强制约束用来应用过滤的。在 ETL 处理中随处都可以用到筛选。在 ETL 处理中最好尽早使用筛选来过滤你的数据。
- 路由器 有条件的路由发送，就像 CASE 语句那样
- 联合 通过一致的列的定义合并两个传递途径
- 聚合 当事实表和最低级别的转换不是同一粒度的时候，你需要对元数据进行聚合。有关聚合的元数据包括任何的计算，例如聚合函数，函数本身——计数，汇总，均值，分组等——和聚合函数分组列。分组列声明了聚合粒度。
- 异构连接 当元数据来自于不同系统，他们通常是在任何单个数据库之外连接（除非你使用数据库 link）。在数据库系统环境之外连接不同系统的方法需要在元数据中定义和描述。
- 更新策略 更新策略包含业务规则，业务规则决定了一条记录是否会被增加，更新或删除。它也包含了维度变化策略。
- 目标装载器 目标装载器告诉 ETL 处理哪个数据库的哪张表，哪几列需要被装载。另外，当有散列装载的时候，这个文档被用来载入数据到数据仓库中。

每个转换获得数据，操作数据到某个程度，然后传送数据到任务队列中的下一个转换。描述转换的元数据属性包括：

- 转换名称 单一的任务包含多个转换，把每一个辨别区分清楚对于管理和维护来说非常重要。每个转换必须有唯一的名称，命名要有含义，还要遵从标准命名规则。本章将会描述转换的命名规则。
- 转换的意图 转换的意图必须容易辨别。许多 ETL 工具用不同色彩来标记预先定义的转换。如果用手工代码的方式处理 ETL 程序，那么要确保元数据的内容要包含代码和转换矩阵的信息。
- 输入列 数据元素反馈给转换过程
- 物理计算 操作计算的实际代码
- 逻辑计算 物理计算的文本等价物。在代码含义模糊不清时有用。
- 输出列 数据转换的结果，可用于下一个转换

转换的命名习惯

转换是 ETL 任务的一个构成部分。每种转换在命名格式上都会稍有不同。易于可维护性的原因，在建立 ETL 转换时，最好遵从以下的命名规则：

- 源数据抽取：SRC_<table name>
- 代理键生成器：SEQ_<name of surrogate key column being populated>
- 查找：LKP_<name of table being looked up or referenced>
- 筛选：FIL_<purpose of filter>(例如，FIL_SUPPRESS_BOTS 禁止在网站上点击)
- 聚合：AGG_<purpose of aggregate> (例如，聚合网站月度点击量)
- 异构连接：HJN_<name of first table>_<name of second table>
- 更新策略：UPD_<type of strategy (INS, UPD, DEL,UPS)>_<name of target table>
- 目标装载器：TRG_<name of target table>

批处理元数据

当所有的任务被设计和建立之后，接下来的是这些任务的计划执行。对于那些负责在数据仓库中增加导入任务的小组来说，转载进度是元数据中至关重要的一部分。图 9.3 描绘了对于数据集市来说一个典型的装载是如何进行的。图中装载进度包含以下元数据属性：

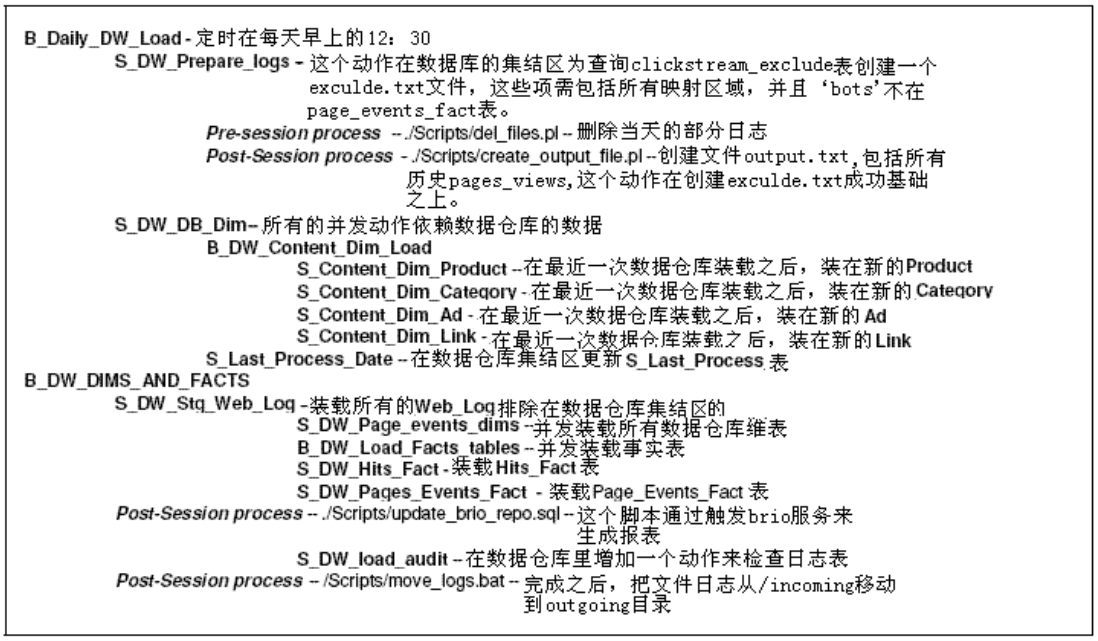

图 9.3　点击流数据集市装载调度

装载进度

- 依赖的批处理 一般一个批处理会嵌套包含好几层的批处理任务，来并行执行多个任务或维持任务与任务之间的完整性。例如，一个装载数据集市维度的批处理必须在执行事实表任务之前装载成功。记住，数据仓库并不完全依赖参照完整性。通过从属的批处理是在数据仓库中实施参照完整性的一个办法。
- 频度 数据仓库中的数据导入频度有：每月，每周，每日或连续装载。这一部分的元数据就是定义批处理的执行频率的。

- 执行进度 如果一个任务是每日执行，那么元数据的属性就会捕获到每次批处理执行的时间。如果是每月执行，那么在元数据中会纪录处理执行的那个具体日期。批处理必须具备在任何预定时间预定计划下执行任务的能力。
- 恢复步骤 当处理失败的时候，需要有一个恢复的动作。恢复有可能是一个很长时间的处理，经常在另一份独立的文件中提出。进行恢复处理要让负责 ETL 批处理的小组了解进程。

数据质量错误事件元数据

在第四章中已经对元数据描述数据质量有了深入的讲述，对三张主要表进行了详细讲述。但是为了保持一致，我们在这里列出了清洗和规范化阶段的元数据索要捕获的数据元素。

首先，筛选表包含：

- ETL 注入阶段 描述所有 ETL 处理中要用到数据质量筛选的阶段
- 处理顺序号 是一种简单的时序安排，它通知 ETL 主处理程序执行筛选的顺序。
- 在同一个 ETL 处理阶段里有相同处理顺序号的数据质量筛选可以并发执行。
- 严重性评分 用来定义错误的严重性，
- 异常反应 异常反应属性会告诉所有 ETL 处理当发生错误时是否让该数据通过，或拒绝数据，或停止整个 ETL 处理过程。
- 筛选分类名称 根据描述给定数据质量筛选分类，例如，完全，验证或拒绝进入
- SQL 声明 记录进行数据检查的 SQL 或 SQL 程序的真实片断。如果这些 SQL 可以应用，将返回那些不符合数据质量规则数据的唯一标示符。所以，它可以用来向事件事实表中插入数据

然后，主要错误事件事实表包括以下几个方面

- 存储数据标示 唯一标示错误纪录
- 错误严重性评分 从 1 到 100 给错误赋值

错误事件事实表用外键关联日期，时间，ETL 批处理，表，和源系统维度。这些维度提供了衡量错误事件事实表的内容。

审计维度包括以下方面：

- 全面数据质量评分
- 完全性评分
- 校验评分
- 超限度评分
- 筛选域的数目
- 最大错误事件
- 清洗和规范化时间戳，包括某个具体 ETL 任务执行的开始时间和结束时间
- 全面 ETL 时间戳，包括完整端对端 ETL 任务的开始时间和结束时间
- ETL 发布版本号
- 其他审计版本号，比如配置版本，目前转换逻辑版本，所依赖的业务逻辑环境

处理执行元数据

事实上数据仓库中所有的处理元数据都是由 ETL 过程生成的。每次一个任务或批处理执行，统计或成功指示器都需要被捕获。导入统计是元数据的一个重要部分。他所收集的信

息包括 ETL 处理执行和实际装载结果。

运行结果

元数据元素可以帮助你理解 ETL 任务中的活动和批处理或评估成功的处理：

- 主题名称 可以是数据集市或者描述为某个特定区域而执行批处理。
- 任务名称 执行程序的名称
- 处理行 从源数据系统中读取或处理的行数统计和百分比统计
- 成功行数 装载到数据仓库中数据的总数和百分比
- 失败行数 被数据仓库拒绝的数据总数和百分比
- 最近错误代码 在数据装载过程中，最近的数据库或者 ETL 异常的错误代码
- 最近错误 最近错误的文本描述
- 读取能力 用来衡量 ETL 处理性能的。用行/秒来描述。当源系统读取发生瓶颈时，记录读取能力。
- 写能力 用来衡量 ETL 处理性能的。用行/秒来描述。当写入目标数据仓库遭遇瓶颈时，记录写能力。
- 开始时间 任务开始时的日期，时间和分秒
- 结束时间 任务结束时的日期，时间和分秒，这个时间是不考虑任务是否成功的时间
- 耗时 与开始时间和结束时间不同，这是性能分析的一个重要指标。在大多数情况下，只记录每秒钟读取和写入多少行是不够的，因为这项指标在很大程度上还取决于你导入数据的总量。
- 源文件名称 ETL 抽取涉及的数据的表或文件的名称，应该不止一张表和文件
- 目标文件名称 ETL 涉及目标的数据的表或文件的名称，应该不止一张表和文件

处理执行元数据在数据存储中保留，以便进行趋势分析。分析元数据可以发现 ETL 处理的瓶颈。可以保证数据仓库性能的可控性。同时也可以衡量数据质量。

异常处理

在执行过程中可能引起的数据纪录异常条件和可以采取的措施：

- 主题名称 可以是数据集市或关于批处理的描述
- 任务名称 执行程序的名称
- 异常条件 异常条件的标准设置
- 严重性
- 采取措施
- 操作员
- 结果

批处理调度

批处理是一系列要执行的 ETL 任务调度的集合。批处理的名称应该可以反映出其所属主题，任务执行的频率和任务中批处理执行方式是并行还是串行。

元数据标准和实践

元数据的一个值得投资的方面是元数据标准。很多组织试图在各种层次上标准化元数据。如果你对标准很感兴趣，比如说命名规则或域标准，你会发现这些由国会图书馆维护的

标准非常有用(www.loc.gov/standards/standard.html)。此外，也提供其他标准组织提供的链接。在他们的网站上，你还可以发现以下链接：

- 元数据编码和传输标准(METS)：有关编码标准的描述，管理，和结构元数据的数字图书馆。
- 美国国家标准协会(ANSI)：美国非官方标准化和一致性评估系统组织。
- 标准化国际组织(ISO)：建立，发展和促进标准化的国际交流。

元数据所涉及的 ETL 处理不仅仅包括数值上的标准和惯例，还包括元数据存储和发布的方法论。连接组织和元数据存储的组织包括：

- Dublin Core DCMI（The Dublin Core Metadata Initiative）是一个发展元数据标准的开放性论坛。DCMI 主持周期小组活动和讨论会在世界范围内促进元数据标准和实践。可以在 www.dublincore.org 上找到有关 DCMI 得更多信息。
- 元数据联合会 Meta Data Coalition (MDC)成立于 1995 年，成员大约包括 50 个厂商和最终用户。 他们致力于研究不同数据仓库产品环境的元数据交换技术。MDC 还在建立和壮大元数据发布的一致性方法做出贡献。2000 年，MDC 与 OMG（Object Management Group）合并。
- 标准数据仓库元模型 标准数据仓库元模型（CWM）是 MDC 和 OMG 合并后的产物。本书有一章专门用来描述这个主题。在 OMG 网站上 www.omg.org/cwm 也可以获得有关 CWM 的详细信息。

建立标准

为了维护你的企业级数据仓库所有 ETL 过程中可管理的任务，数据仓库组必须要建立标准，而 ETL 团队要按此实施。无论你是否会遵从上面那些组织的建议或遵从在本书列出的那些习惯，但从长远的角度上讲，你的组织最好要关注一下下列列出的标准：

- 命名习惯 一般组织都会在目前软件或数据库开发团队中有一套命名习惯。数据仓库也可以遵守这些习惯。事实上，当命名列的时候我们更习惯偏离这些标准，而是经常必须按照组织政策命名。所以，所有的组织政策要在文字上归档，并提供给数据仓库组。
- 体系结构 最佳的实践指导方针应该体现在 ETL 环境元数据中。在许多情况下，高级构架是在 ETL 团队成立之前就设计好的，或是由 ETL 供应商提供的。是否在数据仓库中执行 ETL 引擎或是在一个专门的服务器中？是否要有一个稳定的存储区？目标数据仓库是否要规格化等
- 基础构造 你的解决方案是基于 Windows 或 UNIX 主机系统或 AS/400？像这种组织的一些标准会影响到 ETL 产品或硬件配置的选择。一些 ETL 引擎只能在 UNIX 或大型机上运行，而其他的则可以在 Windows 上运行。你必须在 ETL 环境部件决定购买之前制定基础构造的元数据。一旦建立好 ETL 环境，就要把基础构造元数据归档，并和内部构造支持组一起维护该数据。

命名习惯

数据仓库环境中各个对象的命名习惯应该在 ETL 团队开始编码之前就建立好。表，列，强制约束，索引等命名习惯应该由 DBA 组或数据库管理员提供。而 ETL 团队要遵从这些规则。

如果你觉得现有的命名规则不适合你们 ETL 环境的话，你要把想要替换的命名规范记入文档，并提交给你们的标准委员会。一旦批准的话，标准委员会应该将现有的命名标准和新的 ETL 习惯相融合。

这部分我们提及的那些组织都没有对 ETL 处理提供一套命名的标准。如果你的 ETL 处理只是由一系列的 SQL 组成，而又没有明确提出的命名定义的话，那么在命名上遵从你们原来内部程序和名称标准。

大多数 ETL 工具的数据转换都推荐使用命名规则。不管是用哪种 ETL 工具，或是手工代码，都最好遵从命名习惯。你的工具可能经常变更，不止使用一种工具，但你每次更换工具时不会每次都建立一套新标准。我们在本书中并没有提及在转换上使用厂商们推荐的命名规则。但是，你也可以根据你自己的意图稍稍更改命名规则，以保证环境的连续性。

效果分析

维护 ETL 元数据的一个有利条件是它可以帮助进行效果分析。通过效果分析，你可以列出数据仓库环境中的所有可以改变特征。还可以分析这些变化所带来的影响，这些影响可能是数据仓库的任何组成部分。一个效果分析方案必须能够解答以下问题：

- 哪些 ETL 任务需要这些阶段表？
- 这些源系统表是否被元数据使用？
- 删除这些源系统列是否会影响 ETL 处理？
- 这些维度来自哪个源系统？
- 如果我们将数据类型从 VARCHAR(2000)转换到 CLOB 型，哪个 ETL 任务和数据仓库表需要修改?

使用工具设计 ETL 处理应该可以回答所有这些问题。不使用 ETL 工具，你需要维护电子表格来记录源系统所有表，列，到他们装载到数据仓库中的所有信息。每次更改一个 ETL 任务，就要手工更新电子表格。

总结

本章，我们回顾了数据仓库元数据的一些混乱现象，讲述了数据仓库元数据的重要性，然后将 ETL 元数据分为三类：

1.业务元数据：从业务角度描述数据的含义，从不同组成部分来完整业务的定义，源系统信息，数据仓库数据字典，和逻辑数据视图

2. 技术元数据：描述数据所反映的技术方面，包括数据类型，长度和数据血统，和数据表组成。通过这些信息追踪系统详细清单，数据模型，数据定义，业务规则，ETL 任务定义，详细数据转换和批处理任务定义。

3. 处理执行元数据：描述 ETL 处理结果的统计信息，包含的测评指标包括成功执行行数，丢弃行数，装载时间。我们还在文中提出，在清洗和规范化过程中比较重要的元数据，包括筛选维度表，错误事件事实表，和审计维度表。所有的元数据都是由追踪执行结果表，异常处理表，和直接操作计划表组成的。

北京易事通慧科技有限公司（简称“易事科技”，ETH）是国内领先的专注于商业智能领域的技术服务公司。凭借着多年来在商业智能领域与国内高端客户的持续合作，易事科技在商务智能与数据挖掘咨询服务、数据仓库及商业智能系统实施、分析型客户关系管理、人力资源分析、财务决策支持等多个专业方向积累了居于国内领先的专业经验和技能。

作为Solvento集团旗下的联盟公司，易事科技获得授权为客户和合作伙伴提供MicroStrategy产品，SPSS产品，Pervasive产品和i2产品的销售及技术服务。更多的信息请访问公司的官方网址 http://www.ETHTech.com，或拨打电话+8610 68008008。