职责

在本章中，我们将讨论开发管理以及一个成功的 ETL 系统的管理。我们可以把这一章放在本书的开始，在种种 ETL 系统的职责彻底的讨论之前，但是我们考虑到将内容放在本书的最后，读者可以更好的理解如何高效的管理团队。

本章的前一部分关注于计划和领导问题，后一部分将深入到管理 ETL 系统的细节中。这些观点从《数据仓库生命周期工具箱》中发展而来。

流程检查

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

计划和领导

在某些角度上，数据仓库和 ETL 处理就像其他的软件开发项目。当数据仓库团队建立的时候，通常需要 3 个专家。下面列出了在数据仓库项目初始阶段需要的一般角色，列表中包括了主要的角色和次要的角色（使用圆括号），在小型团队中可以兼任。

- 数据模型师（项目经理）。数据模型师必须接受过维度数据模型建模的专门训练，学习过维度模型的理论。
- ETL 架构师/程序员（DW 架构师）。ETL 程序员以及 ETL 架构师通常是 SQL 和数据库的专家。这个人负责建立 ETL 系统、数据仓库环境的技术框架，并负责设计物理的 ETL 流程。
- 应用专家（业务分析师）。负责搜集并文档化业务、分析和报表需求。这个专家需要设计前端的接口，以及数据仓库中的初始报表。这个职位通常称为商务智能专家。

当启动一个数据仓库项目的时候，通常是由一组专门的人来为整个企业应用的设计基础部分。如果没有一个完整的计划和方法论去构建应用的基础，那么后续的所有工作可能都是无效的。

拥有专门的领导

数据仓库是一个需要专门知识的复杂工程，这些知识并不为大多数 IT 管理者所掌握。最基本的，数据仓库需要一个专门的项目经理，他需要具有使用维度模型原则实施数据仓库的经验。如果你的数据仓库发展的很大，那么每一个组成部分和子部分都需要一个专门的项目经理。一个成熟的数据仓库必须有独立的来负责 ETL、数据建模和商务智能的经理，并且需要一个项目经理在总体上控制数据仓库项目中各个部门，以确保在各个域中实现一致的方案。

通过一个人来管理整个数据仓库项目是值得讨论的，但是我们强烈的建议为每个领域指定专家。不同的领域需要不同的技能，导致了某个人无法覆盖所有的知识。请牢记，个人强不过团队。

团队的力量永远强过个人，但是这不意味着可以通过投票来完成设计！决定设计适合采用专制方式，这样可以维持其一致性。

周密计划，逐步实施

当你从头创建一个数据仓库的时候，最困难的是去想象从现在的单个数据集市扩展成一个规模上超过公司的任何一个现有应用的主要的企业核心系统。但是这个图景通常被遗忘，因为数据仓库通常采用另外的方法实施：在开发下一个数据集市的时候，他们开始完成单个的业务流程或者数据集市，例如人力资源、或者渠道管理。

数据仓库架构师必须采用已知的数据仓库总线架构技术，该技术勾勒出了数据仓库的架构，让所有的结果数据集市按照我们本书中详尽描述的使用规范化的维度和事实按照整合的方式进行工作。

数据仓库总线架构的处理过程中包括了一个数据仓库总线矩阵，需要创建一个所有维度的列表，并将其和数据仓库中的不同的数据集市关联起来。总线矩阵可以帮助架构师在直观上发现哪些数据维度在数据仓库中不同的数据集市之间被共享或者需要规范化。一旦总线矩阵创建，物理的数据集市可以一起被创建。图 10.1 给出了一个简单的数据仓库总线矩阵的例子。

数据集市/维度	DIM_AGREEMENT_TYPE	DIM_CURRENCY	DIM_DATE	DIM_DEPARTMENT	DIM_ABSENCE_REASON	DIM_OFFICE	DIM_EMPLOYEE	DIM_POSITION	DIM_ABSENCE_TYPE	DIM_EMPLOYEE_RATING	DIM_COMPENSATION_TYPE
雇员月度快照		X	X	X		X	X	X			
雇员转换交易	X	X	X	X		X	X	X			
专门交易			X	X	X	X	X	X	X		
补偿月度概要		X	X	X		X	X	X		X	
费用月度概要			X	X		X	X	X		X	
补偿交易		X	X	X		X	X	X			X
费用交易			X	X		X	X	X		X	

图 10.1　数据仓库总线矩阵

正如某些维度在整个的数据仓库中被重复使用一样，在你创建ETL过程的时候某些ETL方法会被反复的重用。例如：你的 ETL 处理通常包含为某个维度生成代理键，那么生成代理键的代码可以被重用来生成数据仓库中的所有的代理键，不同的仅仅是参数。如果你有一

定的软件开发背景，那么你一定听说过：一次书写，多次使用。这个格言意味着尽可能的重用代码。代码重用不仅仅节省了开发时间，也确保了代码间的一致性。

你需要一个高效的重用策略。要建立一个鼓励开发者分享想法和信任彼此的工作环境，下面的一些提示可以帮助你：

- 作为一个小组取得决策的一致：通过例会来讨论技术和功能策略，以一个小组来解决困难。
- 在共享代码的时候分享想法：ETL 开发不是一个竞技项目。共同工作、与团队中其他人分享案例。我们已经花费了大量时间解决了某个困难的问题，通过向其他人解释情况，那么解决的思路就会被别人掌握。在简单的讨论过程中，你会惊奇于产生了多少好的想法。
- 使用资料库。许多 ETL 工具为了重用代码提供了资料库。请确认你的工具中包含了一个资料库，可以利用它在不同的加载方式或者开发者之间进行代码重用和共享。如果你还没有购买专门的 ETL 工具，至少使用一个源码资料库，例如 SourceSafe 或者 PVCS。在一个复杂环境中，你可能需要开发多个单独的资料库，但是作为一个虚拟的资料库进行管理。

一旦你的团队学会了共同工作，所有的代码方法存储在你的资料库中，那么开发的效率的提高是显而易见的。另外共同工作，也帮助开发者对整个项目有了一个整体的了解。要避免将开发者按照主题域来划分。随着经验的积累，每个开发者会成为他或她所开发领域的专家。这对于团队涉足新的领域是非常有益的。拓宽 ETL 开发者的视野提升了 cross-functional panning 并且提高了士气。

应该鼓励你的 ETL 团队尽可能的分享和重用他们的工作成果。确保在资料库中为共享的代码关联正确的元数据。元数据可以标识代码的用途以及介绍使用方式。但是对于跨操作系统和 DBMS 平台代码的重用要有一个正确的期望。

雇用合格的开发者

熟练的开发人员是企业的无价的财富。但是，单纯拥有技术不能表明它是一个专家。这些年，我们曾经面试了很多 ETL 开发员，他们懂很多工具内外的东西，但是却对维度概念，如层级映射表一无所知，开发员必须能够在接受最少培训的情况下迅速的掌握并应用一种新的技术。在我们面试的时候，我们会用很少的时间谈使用的工具的特性，而是把更多的时间用于技术和功能的问题如何解决。我们寻找那些聪明的、有成为合格的 ETL 团队成员的潜质的人，而不是那些只有技术技能的人。

在面试的过程中，专门问一些那些应试人不知道如何回答的问题。观察他/她如何回答。记住，这时候重要的不是答案是否正确，而是处理问题的过程。现实是 ETL 和数据仓库可能会非常的复杂，甚至非常的特殊化。无法像分析原子构成那样分析，否则，科学家就可以提供一个规范了。当你建立你的团队的时候，确保你的开发者有意识主动的追求技术和专业。他必须能够和你以及你的项目一起成长，能够接受新的技术和方法。

和数据库专家一起组成团队

ETL 主管的部分职责是考察你的企业中的所有源系统，根据已有的系统为你的团队选

择合适的技能组合。如果你使用专门的 ETL 工具，雇用专门的数据库专家可能就不再是关键问题。但是，即使是采用最好的工具，在 ETL 开发中也避免不了撸起袖子写 SQL 的情形。

列出 SQL 编码的提示和技术超出了本书的范畴，在市场上有好几种介绍 SQL 的书籍，当然都是基于 DBMS 查询介绍的。单独的工具可能不能完全满足你的 ETL 需求。当你面试那些 ETL 团队成员的候选人时，确认他们有业务系统 DBMS 以及主要 ETL 处理使用的数据库的 SQL 经验。

每种 DBMS 源系统都需要你对其了解，并能够利用产品特殊的 SQL 进行实施。确保你的团队能够操作多种数据库，这样你的 ETL 工具就可以在处理中和本地的 SQL 代码无缝的集成，而不再把它留给应用程序。

不要试图拯救世界

ETL 系统仅是数据仓库项目的一部分，某些想法可能会超出你的控制。注意数据仓库不是，也不会是完美的。你需要面对脏数据，很多时候，你没有办法清理。我们的哲学是最好的数据清理的方法是暴露他们。由于数据难于分析，才会考虑建立数据仓库。要是你的 ETL 处理提交了大量以前从未暴露的数据，尤其是提交到数据仓库的时候。如果你对于保证最终数据的数据质量的请求被忽略，请保持耐心。一旦脏数据被提交，主管将不得不开始对异常进行检查，那样你将会欣喜地见证到数据质量的改变，并且获得你需要的支持。

强制推行标准化

对于一个大型的 ETL 项目，需要尽可能早的建立标准。没有标准，开发人员将写出不一致的 ETL 作业，导致对已有的代码的维护任务变得非常恐怖。ETL 团队必须标准化他们的开发技术，提供一个一致的，可维护的代码环境。下面的列表包含了需要标准化的了大多数 ETL 处理：

- 命名约定。在 ETL 处理中建立和强制推行对对象和代码元素的命名的规范。可以从利用你们现有的命名标准开始，并在其中增加那些 ETL 工具厂商建议的规范。
- 最佳实践。为建立 ETL 过程建立文档，并按照最佳实践的过程实施。在你的过程中尽量的标准化一些东西，例如从一次失败的处理中恢复的最佳方式，或者通常转换开始的位置。本书为你提供了针对策略的完整的建议，包括：
  - 生成代理键。如果你决定使用数据库，ETL 工具或者任何其他的机制来生成代理键，那么在整个 ETL 作业中保持一致。
  - 查找键。你可以使用映射表，物理维度，或者使用准备区技术来将自然键和他们的代理键关联起来。选择一种，并坚持使用。如果你混合使用这些技术，那么维护这些方法将是一场恶梦。
  - 提供缺省值。一些方法和值可以被接受意味着使用默认的缺失值。记住在数据仓库中要对缺失值或者 NULL 值进行认真的处理，因为这些值可以导致报表中空的列或者行，因为一些数据库中在他们的索引中不包括 NULL 值。最好检查进入记录的 NULL 值，在 ETL 过程中使用实际的值，如单字节的 ？来代替。

监控、审计和发布统计信息

ETL 统计对于任何使用数据仓库的人来说都是有效的。如果你的数据仓库有专门的 Web 站点（应该具有），这就需要确保其中包括了对你的 ETL 过程的日统计信息。用户通常希望准确的知道什么时候一张表被装载，或者是否有任何的记录被拒绝。大多数 ETL 工具能够自动的生成加载统计。确保你的工具能够在每天数据加载后自动的发布需要的统计信息。

交叉参考：在第 9 章中给出作为元数据策略组成部分的统计元素列表，他们应该被发布。

维护文档

你的 ETL 处理的 那些文档内容对于数据仓库团队来说是无价的财富，并且已经成为财务或者通常报表在数据仓库建设阶段的必要资料。即便是那些纯逻辑数据映射规范，虽然仅仅只有开发人员才知道从源系统到数据仓库进行了如何的处理。ETL 团队有责任严格的控制并维护数据仓库中每个元数据的文档。一些文档可能以元数据的形式存在，但并不是所有的文档形式都被认为是元数据形式。元数据作为一个复杂的实体已经在很多书中对其内容进行了介绍。无论你怎样去分类，一些文档内容都需要维护和发布。通常，很多工具设计了捕获处理过程的描述信息的字段，但是这些信息往往不存在。这样无可避免的，你需要以 Word，Excel，PPT 的形式提供解释你的 ETL 过程的文档。请使用如 SourceSafe 或者 PVC 等版本控制系统来维护你文档的完整性。

提供和使用元数据

元数据对于 ETL 过程的重用异常重要。事实上所有的 ETL 工具都能够捕获和利用元数据。不要不负责任的让你的团队在创建 ETL 处理的时候不建立元数据。每次当你创建新的 ETL 过程的时候，请牢记如果不能通过元数据获取，那么它就不存在。这在创建一个数据仓库环境的时候尤为重要。如果你不通过元数据公开你的工作，那么团队中的其他人就可能要从头创建那些你已经创建和测试的处理过程。

通常，ETL 工具资料库通常是元数据的资料库，有些 ETL 工具能够从其他的工具，如数据建模或者报表工具中自动的获取元数据并对关联的元素进行影响分析。一些商务智能工具可以利用 ETL 资料库，将元数据和数据仓库的用户接口进行集成。如果 ETL 环境中的元数据被发布，那么它们必须被维护，或者用工具可以将最新的信息发布给他们的用户。

保持简单

如果你相信存在采用更简单的方法去做什么，那么通常这种方法都存在。当你创建 ETL 过程的时候，请时常停下来回头从设计的角度审视你的工作。设计够直接么？设计中是否有一些可以避免的复杂情况呢？你的处理过程越复杂，那么维护的困难性就越大。通常，一个复杂的 ETL 设计几乎不可能再被修改。当业务需求或者源系统改变的时候，你的 ETL 作业必需足够的灵活，以适应变化。我们曾经接触过一个项目，ETL 作业是那么的复杂，没有人确切的知道它做什么，所以没有人能去修改它。费尽周折，最后我们被请来做反向工程，

文档化并将其重新设计为一个改进的、可维护的过程。

彻底优化

对于如何精心的设计 ETL 将源系统转化成数据仓库中的有用的信息，在现实中没有任何的限制。但是作业执行多长时间的限制确实是存在的。ETL 开发人员所面临的挑战是在指定的加载窗口中完成抽取、清洗、转换和加载数据。加载窗口是指每个晚上运行 ETL 过程的时间。通常，数据仓库在加载窗口中是不可用的，因此将加载窗口变得尽量小就成了一种挑战或者压力。

管理项目

ETL 处理是数据仓库的关键过程，直到今天，它还是没有获得应有的重视。在早期的数据仓库中，主要关注的是前端工具，然后随着数据量的增加，维度数据建模成了新的热点。当数据仓库成熟度达到下一阶段时，ETL 将最终成为关注的焦点。

绝大多数设计人员同意在整个数据仓库项目中的 ETL 处理要至少占用 70%的工作量。ETL 团队来建立这些将数以亿计的记录从分散的系统中导入到一个一致的用户信息库的强健的处理过程，对这样团队的成功管理是对技术能力和执行能力的挑战。管理 ETL 团队需要奉献精神以及管理方法。

ETL 主管的职位建立是为了帮助数据仓库项目经理分担 ETL 管理职责。这个职位也给了业务接口人以信心：ETL 团队可以建立一个受控的、高效的环境来向数据仓库加载干净的、一致的数据。本章中包含的任务应该被数据仓库项目中的每个成员仔细阅读，以确保他们理解 ETL 处理并不是数据仓库项目的副产品，而是将整个项目连在一起的粘合剂。对于 ETL 主管，本章的内容提供了保证你的 ETL 项目成功的基本知识。

ETL 团队的责任

从最根本上说，ETL 团队的责任是从源系统中抽取数据，执行数据转换，将转换后的数据加载到目标数据仓库中。进一步，要得到最优的 ETL 结果，ETL 团队的责任包含下列的任务：

- 定义 ETL 范围
- 执行源系统数据的分析
- 定义数据质量策略
- 和业务人员一起工作获取和记录业务规则
- 开发和实现实际的 ETL 代码
- 创建和执行单元以及 QS 测试计划
- 实现产品
- 进行系统维护

要高效的管理你的团队完成上述的任务，我们结合这些任务的概要列出了一个实际的项目计划，并描述了针对每个任务的管理职责。

项目定义

尽管 ETL 处理仅仅是数据仓库整个生命周期中众多组成部分中的一个，但它是整个数据仓库的核心。同时 ETL 处理也是最难管理的一部分。当用户在项目初期阶段开始看到结果数据的时候，你将面对需求变化的冲击。假如没有一个合理的项目执行计划和变更管理策略，冠以 ETL 处理将是不可能的，一个永远没法完成的任务将会将项目推到失败的边缘。当你开始定义你的项目的时候，请牢记如下的准则：

- 为了一个顺畅的处理，ETL 的管理必须和数据仓库生命周期中的其他组件紧密合作。从规律上来说，数据仓库永远没有真正的结束。当一个新的需求产生，建模团队、ETL 团队和报表团队必须一起高效的工作来完成任务、达成目标。本章列出的步骤可以在不断的增加到数据仓库中的不同主题域中重用。正确选择本章列出的方法对于正确管理这些迭代的过程非常关键。
- 在估计工期和定义范围的时候正视现实。不要让数据建模人员或者不了解技术的业务接口人员对 ETL 工作的时间表作出不正确的决定。使用本章中的项目计划作为蓝本，并确认你的业务用户和接口人充分理解在加载数据仓库中包含的那些内容。
- 确保 ETL 团队在数据仓库项目启动会议中的活跃性。这样的会议适合将你的 ETL 团队介绍给关键的业务用户并讨论 ETL 的角色、目标和责任，以及时间表。创建一个鼓励合作的环境，这个会议将帮助与会者理解项目需求，并给你机会调整预期值。

项目计划

什么是计划？计划是一种活动、过程或者安排的方法。它是要完成的任务单。其目标是实现某种观念、想法、项目或者开发。因此，计划是帮助你达成期望的某种方法

--- Earl Prevette

ETL 主管的最终目标是成功的管理 ETL 的处理，并将处理集成到生命周期中的其他阶段。你可能假设管理 ETL 处理和管理其它实施是一样的，但实际上差别相当大。在本节中，我们将介绍那些帮助我们获得成功实现的方法。另外我们会揭示许多你可能会面对的障碍，并提供一些降低这些风险的建议。

开始定义项目计划中的可替换部分的首要条件是，你需要完成一些家务管理职责。这些任务包括决定你 ETL 工具的选择，以及组织你的 ETL 团队。

这两个任务的先后顺序非常的重要。你可能会将选择你的 ETL 工具放在组织团队之前。这样将使你可以招聘那些对你选择的工具熟悉的人。

选择工具

如同在第一章中所介绍的，ETL 主管必须在购买 ETL 产品和手工创建 ETL 处理中做出选择。对于每一种选择都会有很多的理由。但是，从企业数据仓库的成功性和管理接口人员的期望来考虑，我们认为没有时间去手工创建，即使是考虑到数据仓库的迭代属性。无论再小的项目也可以从专门的 ETL 工具的转换可重用性中获益。这些工具的已有的现成功能将

会花掉几个月的时间去设计，更不用提作数据转换处理的实际编码。通过这些工具可以减少开发时间，使其可以作为任何的数据仓库项目的解决方案。

此外，ETL 工具是专门设计来解决那些当前甚至未来的问题。对于购买还是开发的问题，我们听到的所有常见的理由是程序人员懂 SQL，为什么还要学习一种新的工具，因为他们都可以转移数据。当我们听到这个问题的时候两个类似的问题浮现眼前：首先，如果你仅仅知道的工具是锤子，那么你周围的所有的东西都会被认为是钉子，安装螺丝钉将会变得非常困难、费力和随意。第二个是秘书没有时间学习文字处理，因为他们总是忙于打字。这两个问题听起来多么荒唐，但这和不训练你的 SQL 程序员去学习利用专用技术的 ETL 工具来完成任务何其相似。

为了帮助决策，我们建议你记录你的工具选择标准。选择一些诸如吞吐性能、易于修改、厂商支持等 POC 因素，然后针对这些工具完成 POC（包括手工开发的比较），最后选择符合你标准的产品。基于 POC 的结果，你可以决定是开发还是购买产品。如果你决定购买，你需要确认哪种产品符合你的要求。

为项目配置人员

管理 ETL 过程的一个关键因素是建立一个高水平的团队。你的团队成员必须拥有必备的技能来完成交给他们的任务。一个认证训练的团队是你成功地关键。确保你的所有团队成员使用公司文化，并且可以共同工作是非常重要的。

在数据仓库建设到达当前水平之前，所有项目的职责通常是由少数的数据仓库专家完成。这些核心的专家和业务用户交流，记录需求，设计数据模型，加载数据库等等。当数据仓库项目进一步发展后，我们发现每个特殊的任务都需要一组不同的技能，没有一个人可以成为所有问题的专家。

ETL 团队角色和职责

对于 ETL 团队角色的组成是需要考虑的任务。如果你拥有相应的知识，你可以雇佣和培训你内部的员工。否则你需要和招聘人员一起寻找合适的专家来建设你的团队。

下面的列表介绍了我们建立一个最佳的 ETL 团队所需的基本的角色和职责。

为每个角色都雇佣一个人是最理想的。但是，根据项目的实际情况，考虑到条件的限制，更现实的是让人们相互承担一些彼此的职责。记住当你组成一个项目团队的时候，你的主要目标是确保所有的职责都可以被履行。你没有必要为每个角色制定一个专门的人。

- ETL 主管。这个人负责 ETL 团队日常的管理，以及负责和 ETL 相关的数据仓库的运行维护，他对数据抽取、转化和数据仓库中的加载处理，以及测试和质量保证负有责任。他需要为 ETL 环境制定标准和流程，包括命名规范以及最佳开发和设计的训练。
- ETL 架构师。主要的职责包括设计 ETL 环境的架构和底层结构，为 ETL 开发团队设计逻辑数据映射。这个架构师必须对业务需求和源业务系统有很好的理解。ETL 架构师负责为团队解决复杂的技术问题，以及将 ETL 过程迁移到生产环境。
- ETL 开发员。这个人负责建立事件的 ETL 处理。在实际编码以前，他和 ETL 架构师合作解决任何规范中有歧义的地方。开发员负责开发专门的 ETL 方法，并测试他们的可靠性，确保其符合业务需求。通常数据仓库项目中都会安排多个 ETL 开

发员。

- 系统分析师。负责业务需求定义，以及在整个数据仓库生命周期中定义这些需求。他和数据仓库团队以及业务人员合作紧密。
- 数据质量专家。数据仓库的质量包括了内容的质量以及数据仓库中的信息的结构。数据质量专家通常向 ETL 主管汇报工作，有时候也直接向数据仓库项目经理汇报。数据质量专家的主要工作是帮助系统分析师和 ETL 架构师来保证业务规则和数据定义在整个 ETL 处理中是延续的。
- 数据库管理员（DBA）。DBA 的主要职责是将数据库的逻辑设计翻译成物理的结构，并维护物理的数据库。而且，DBA 需要和 ETL 团队紧密地工作，确保新的处理不会和已有的数据冲突。在某些环境中，一旦 ETL 过程迁移到生产环境，就归 DBA 管理。
- 维度主管。他负责定义、创建和发布一个或者多个规范化的维度给整个数据仓库社区。这是一个需要权力集中的职位。规范化的维度需要打上版本标签，同时为所有的事实数据提供者客户所使用。可能在一个企业内部有多个维度主管，原因是可能每个维度可能独立性很强。在任何情况下，一个维度只能有单一的维度主管。
- 事实表提供者。事实表提供者拥有某个专门的事实表。在一个规范化的维度环境中，事实表提供者周期性地接收并更新维度主管那里下发的维度，将事实表中的自然键转换成为维表中的代理键，然后将事实表发布给相应的用户群。

ETL 项目团队人员选择

有一句格言：你的下属决定了你的水平，这句格言揭示了关键业务系统，如数据仓库中 ETL 处理中一个重要的特点。一个聪明的避免项目失败的方法是建议一个专家团队来实施。这一节将对建设你的 ETL 团队的给出一些建议。

和新人一同工作

有总比没有好，在建立你的 ETL 团队的时候你需要把眼光放在公司以外。通常，企业总在寻找最佳的候选人。但是就像数据仓库需要完全可靠的信息给他的用户一样，你需要提供精确的需求给招聘人员。在提供工作描述的时候尽最大可能的细致，这样确保你见到的候选者拥有你希望的技能和工作习惯。让招聘人员了解你的环境的具体信息，尤其是你的开发语言，供应商以及数据库。另外描述你的团队的文化，以及你需要什么样的人。提供的信息越详细就越有可能找到符合你条件的候选者。

从专门的数据仓库公司招聘的人往往拥有数据仓库行业的知识以及所支持的工具的技能。在简历转发给你之前这些人应该是经过初步筛选的，限制需要你评估的简历和人员的数量。其它的工作应该会占用你的更多时间，面试新人的时间需要花的值得。

内部雇用与外部

从内部或者外部资源创建你的团队有很多好处。从内部雇用的好处如下：

- 从企业内部寻找和雇佣成员的最大的好处是这个内部的人员已经对企业内部的结构和 IT 系统非常了解。他们知道每个人负责什么；哪个人可以回答问题；如何按照规矩做事。如果你够幸运，这个人可能已经拥有了你团队角色所需要的技能。如果一个人没有需要的技能，但是有潜力并愿意接受培训，那么他就是一个非常值得考虑的候选人。
- 为有热情的内部的员工提供学习新东西的机会将保持他们的竞争力以及满足他们的需要。

- 内部雇用在经济上考虑非常有利：这比通过招聘人员、花钱安置、退还面试成本、花钱重新安置等经济的多。

如果你从外面雇用，你雇用的人应该拥有你在寻找的技能，以及在不同企业文化中使用这些技能的经验。这些经验远远超出了他的表面价值。经验可以给你的团队增加价值，节省你的时间和金钱。

选择团队成员

一旦你获得了一把简历，那么我们建议你在带他们来之前先和候选人安排一次电话面试。在电话里问一些关键的问题，直接考察他们的沟通技巧，以及对问题的理解程度。

对通过电话面试的候选人应该进行面对面的面世。你的候选人应该接受 ETL 主管、技术开发人员以及功能分析师的提问。让候选人同功能分析师以及技术人员见面将使你能够衡量他们的熟练程度。我们有一些不愉快的经历，团队成员技术很熟练，但是没有甚至不能领会功能要求。他们的无能导致其他的团队成员需要额外花时间来确保他们的工作符合业务要求。

你需要坚信不疑，你的团队的所有潜在成员需要有充分的 ETL 处理设计的知识，充分地掌握所需工具，以及对业务流程的正确理解，领会功能需求。和团队其他成员的协同能力也是关键，在你面试的时候，一定要询问他以前项目中的团队工作方式。

在面试中，对你和招聘人员最基本的要求是不仅要充分了解你们要找的角色，而且要知道问哪些问题。图 10.2 时一张面试调查表，提供了面试过程中的问题。使用这个问卷可以确保你的候选人的知识满足你的角色要求。如何回答这些问题则超过了本书的范畴。

简单调查表

分析
1. 什么是逻辑数据视图？对于ETL团队它意味着什么？
2. 数据仓库项目中搜寻数据阶段的主要目标是什么？
3. 如何确定档案系统？

架构
4. ETL过程的4个基本数据流程步骤是什么？
5. 数据集结区可以接受的数据架构是什么？每个得简单描述。
6. 在ETL过程中，出于安全的考虑，什么时候数据要设置在磁盘上？

抽取
7. 描述从异种数据源中抽取数据的技术。
8. 处理ERP源数据的最佳方法？
9. 描述数据库自然连接与通过ODBC之间的差别。
10. 描述三种缓慢变化维度。

数据质量
11. 四种主要的数据质量检查方法是什么？每种说出一个执行方法。
12. 在ETL的每个集结阶段是否都要对数据质量进行评估？
13. 数据质量检查部分的重要交付物是什么？
14. 在数据仓库中数据质量如何量化？

建立映射
15. 什么是代理键？解释代理键管道如何工作？
16. 为什么在ETL处理中日期要特殊对待？
17. 解释一致化维度中三个基础提交步骤。
18. 给三种基础事实粒度，并描述每一个的ETL 方法。
19. 当维度记录和单个事实表关联时，桥接表如何提交到分类的维度表？
20. 延迟到达的数据如何影响维度表和事实表？他们的处理方法是什么？

元数据
21. 描述ETL 元数据的不同类型，提供每个类型的一个样例。
22. 为捕获操作性元数据提供可行的机制。
23. 描述业务元数据和技术元数据的方法。

优化/操作
24. 陈述数据仓库中表的主要类型，以及他们装载的顺序。
25. ETL 支持模型四个层次的特性都是什么？
26. 你认为ETL 处理性能的瓶颈一般都是什么？
27. 描述如何减少大ETL任务的装载时间？

实时ETL
28. 描述实时ETL的框架。
29. 解释不同的实时方法，以及他们如何应用到不同的业务需求中。
30. 列出实时ETL可能面临的风险，并描述如何克服他们。

图 10.2　面试问卷样例

构建和保持一个成功的ETL 团队

一旦你建立了你的团队，你作为主管的主要职责就开始了。保持一个一流的团队对你来说是一个最大的挑战。对那些 ETL 高水平人员的需求是很大的，招聘人员会毫不犹豫的从你的眼皮底下将他们猎走。我们发现要保留你的项目中主要的 ETL 开发人员和架构师的最佳方式是保持技术挑战性。根据我们的经验，一个无聊的技术专家最有可能离开。安排项目使你的团队成员感兴趣和兴奋是你的职责。

ETL 开发人员完成的任务并不简单。他们的职责是要将无组织的、分离的数据转换为一致的有价值的信息，是一个令人激动，但很多时候令人精疲力竭的工作。不要忽视它们，关注他们的需求，知道是什么让他们奋发激情。我们曾经和一些开发人员一起工作，他们非常喜欢清理数据。他们乐于从垃圾中找到一致可信的数据。而其他人却不能忍受这一点，他们认为既然数据如此重要，那么他们从源头就应该是干净的，这些开发人员不可能胜任这样的挑战：将几乎无法用 SQL 操作的数据从可怕的复杂的模型转化到简单的维度中。另外的一些开发人员却喜欢和时间赛跑，如果一个 ETL 处理应该需要一周开发，他们却疯狂的工

作，在短短几天内将他完成。你作为主管的部分职责是了解你的开发人员的类型，让他们接受挑战。

如果团队成员渴望能够接受更多的责任，授权好了！你的职责是发现每个人的期望，尝试满足他们，另外，你需要提供团队成员需要的培训，让他们成为所从事工作的专家。如果你不能让你的队员进步，那么他们将会离开项目，跑到更具有挑战性的环境中。

和你的下属保持和谐的一种高效的方法是周例会。ETL 环境是变动非常频繁的，超过一周没有得到你的团队的反馈可能会对项目造成危害。这个具有双重目的的会议可以确保项目的成员达到他们的目标，同时你也达到你的目标。给他们授权，鼓励他们做合理的决定。而且，你应该建立这样一个环境，团队成员可以表达他们的意见，交流开发需求等等。他们必须能够信任他们的 ETL 主管并且接受他们矫正的错误。一个被很正确管理的组员应当是一个满意的人。

外包 ETL 开发

在我们写本书的时候，IT 外包是一个热点。尽管实际的数字小于预想的值。在 2003 年，美国在 IT 预算上的投资是$119 亿，仅仅有不超过 5%的 IT 被外包出去。在最近的报告中，一些混淆视听的外包收入被人辨识出来，管理外包项目包括了额外的通讯、到国外的出差，被调整的期望和交付并没有出现在原来的财务报表中。我们不是说外包不是一个坏主意，但是我们需要提醒大家，外包是一个距离数据仓库非常远的概念。

数据仓库必须和数据源 de jure，并且管理的优先性改变，以及最终用户一致。正如我们说过多次的，数据仓库不是一个项目（提供规范和最终交付），而是一个不断发展的过程。数据仓库开发任务是迭代和变化的。事实上，这也是我们喜欢维度模型的一个原因；它是一种弹性十足的架构，能够适应新的要求以及范围的变化。

由于这些原因，我们不看好将很多的数据仓库开发任务外包给异地的团队，不和源数据提供者和最终用户保持沟通。记住数据仓库是一个决策支持系统，对其成功的判断标准只有一条：是否能够有效的支持决策。

ETL 系统的开发体提供了外包的机会，前提是你有特殊可以被完整的定义的转换。

项目计划指南

注意你已经选择了你要使用的工具，并组成了你的 ETL 项目团队，你现在可以专心致力于 ETL 项目计划了。在本节中，我们提供了一个可以供任何 ETL 团队使用的详细的项目计划。

由于数据仓库建设具有迭代的特性，因此项目计划应该可以在数据仓库的任意阶段复用。坚持使用这些指南可以确保整体的规范化，注意不要忽略其中的任何步骤。

每一个步骤将在本章相应的部分作详细介绍。管理 ETL 处理的主要步骤在图 10.3 的项目计划中列出。

任务	任务描述	子任务		角色
1	建立开发环境	1	搭建硬件环境	DBA
		2	安装软件/工具	DBA/ETL架构师
		3	创建测试环境和文档模版	ETL管理员/ETL架构师
2	业务需求分析	1	回顾现存的数据模型文档	ETL架构师/系统分析人员
		2	定义，并将ETL规则文档化	ETL架构师/系统分析人员
		3	分析源系统	ETL架构师/系统分析人员
		4	定义项目可能的范围	ETL管理员
		5	获得用户信息	ETL管理员
3	设计逻辑数据视图	1	回顾数据仓库数据模型	ETL架构师
		2	回顾业务逻辑	ETL架构师
		3	分析源系统	ETL架构师
		4	创建逻辑数据视图文档	ETL架构师
4	数据质量策略	1	定义数据质量规则	ETL管理员/数据质量专员
		2	将数据缺陷记入文档	ETL管理员/数据质量专员
		3	设计缺陷修改	ETL管理员/数据质量专员
		4	确认清洗逻辑	ETL管理员/数据质量专员
		5	将规则纳入逻辑数据视图	ETL管理员/数据质量专员
5	建立物理ETL处理	1	回顾逻辑视图	ETL开发人员
		2	创建简单维度的装载进程	ETL开发人员
		3	开发复杂SCD-2维度进程（历史）	ETL开发人员
		4	开发复杂SCD-2维度进程（增量）	ETL开发人员
		5	开发事实表进程（历史）	ETL开发人员
		6	开发事实表进程（增量）	ETL开发人员
		7	自动化进程	ETL开发人员

6	**测试ETL流程** —单元 —质量保证（QA） —用户认可（UAT）	1	创建测试环境	DBA/ETL架构师
		2	创建测试计划和脚本	系统分析师
		3	装载测试数据（历史的&增量的）	ETL开发者
		4	执行单元测试脚本	系统分析师
		5	验证数据质量控制	系统分析师
		6	验证数据装载	系统分析师
		7	验证业务规则	系统分析师
		8	获得认可	ETL管理者
7	**ETL部署**	1	创建生产支持文档	ETL架构师
		2	创建失败恢复过程文档	ETL架构师
		3	创建生产环境	DBA/ETL架构师
		4	装载历史数据	ETL架构师
		5	为增量处理初始化ETL调度	ETL架构师
8	**数据仓库维护**	1	为已知的问题开发审计报告	ETL架构师
		2	定期查看ETL日志来确保一致有效的装载	ETL架构师

图 10.3　ETL 项目计划

建立开发环境

要执行完整的数据分析和开始开发任何源系统的 ETL，最佳的方式是让 DBA 团队给你建立一个开发环境。使用一个独立的开发环境保证了数据分析和 ETL 的开发不会受到生产交易系统的影响。一旦环境建立，ETL 架构师和 DBA 团队可以一起工作，安装必要的软件，以及执行数据分析和进行开发所需要工具的安装。

在项目的开始阶段，要记录创建开发环境的所需的步骤。从本文中学到的文档标准可以最大化降低你未来犯错的机会以及最小化后续增加系统时带入的风险。

业务需求分析

尽管在数据建模阶段通过分析已经记录了很多的业务规则，但是 ETL 架构师的职责是实现这些规则。通常，ETL 架构师和系统分析师会审阅所有存在的文档，然后和数据建模人员一起讨论遇到的问题。

ETL 架构师和系统分析师必须对源系统和系统内的数据有非常全面的了解，这点非常重要。一定不要压缩完成对这些分析的时间，并且要记住，在没有完成对所有的源系统进行彻底的分析之前，不要开始创建逻辑数据映射。很平常地，根据涉及的范围，ETL 架构师要和系统分析师、数据建模人员、源系统 DBA 开多方会议来讨论源系统的细节。这些讨论的目的是寻找创建逻辑数据映射以及最终编码的业务规则。

图 10.4 是搜集和记录业务规则，以及数据缺失的一个模版样例。这个表格可以分开用

于跟踪数据清理或者 ETL 的细节，甚至两者。我们将这两者整合在了一个模版中，原因是为了一个一致的集成的解决方案，ETL 架构师通常需要同时跟踪业务规则和数据清理转换。这样将这些规则和转换详细地记录就变得很重要，不仅仅单纯是为了编码，更多的原因是这些文档是单元测试、系统测试、QA 测试以及用户验收测试的案例的基础。这些元数据将来还会被用来做最终用户的培训和程序文档。

业务逻辑与数据缺陷						
编号	业务逻辑与数据缺陷	位置（源数据清洗，ETL或两者都有）	源数据清洗	ETL	审计报告	报告发送
1	员工ID必须唯一	ETL	N/A -没有发现不唯一的员工编号	ETL处理会检查不唯一员工编号，如果发现相应的记录就会被作为编号处理。	报告包括PL_ID, FULL_NAME, HIRE_DATE和DEPARTMENT和所有的复制的员工编号记录	职员系统团队
2	当状态为“激活”时，标志没有设置为“Y”	源数据清洗	数据源团队一定要明确这些产品是否激活，对源数据进行适当的调整。	N/A -增加业务逻辑之后，将不会发生	N/A	N/A
3	E-mail地址必须唯一，但是在源系统中存在多个E-mail地址	ETL和源数据清洗	数据源团队必须要清洗多个E-mail地址，并设定业务逻辑组织多个E-mail进入源系统。但这个时候数据源团队没有足够的权限来执行这个业务逻辑。	由于这个时候数据源团队没有足够的权限来执行业务逻辑，必须要建立ETL处理去除重复而不是装载到数据仓库中，并将生成的报告提交给合适的团队。	报告包括多个E-mail地址的EMAIL_ADDRESS, EMPL_ID, FULL_NAME和DEPTARTMENT	系统管理团队（负责分配E-mail地址）
4	部门的关闭标志设为“Y”，但没有结束日期。	源数据清洗	数据源团队一定要明确该部门是否真的取消，并设置了结束日期。要设定业务规则。	N/A—作为添加业务逻辑的结果，不会再发生。	N/A	N/A
5	用户将产品进行分类，在源系统中上报。将这些分类引入到数据仓库中。	ETL	N/A -不需要对源数据清洗	用户给ETL团队提供产品分类列表。在集结表中生成装载到产品维度中，这样比较容易生成产品聚合的报表	N/A	N/A
6	在部门之上的产品编号必须唯一	ETL	建立产品列表，并设定业务逻辑防止不唯一编号进入数据仓库中。但这个时候数据源团队没有足够的权限来执行这个业务逻辑。	由于这个时候数据源团队没有足够的权限来执行业务逻辑，必须要建立ETL处理来明确不唯一的产品编号，防止他们装载到数据仓库中。	报告包括PRODUCT_ID, PRODUCT_DESCRIPTION和DEPARTMENT_NAME	职员系统团队

图 10.4　业务规则和数据检测跟踪表

理论上，重要的内容都要被记载。不幸的是，虽然实际项目中团队刚开始创建文档的时候有很好的命名规范，但是当发生变化后他们却很少回头进行更新，请不要这样做。保持文档实时更新到最新的状态对于一个项目的成功至关重要。不断更新的文档是对将来系统增强或者对下一阶段的工作进行影响分析的基础。而且，当前的文档确保了你把握着仓库的数据继承关系。维护你的文档可能需要时间和努力，但是总比你回过头在 ETL 处理，业务规则或者事实后的数据中寻找发生了什么改变要强的多。无需太多的想象就可以意识到浪费时间、增加成本以及纯粹的挫折都可以通过事先的计划，在改变 ETL 处理的同时修改你的文档来避免。

定义 ETL 项目的范围

定义范围包括决定和用文档记录在 ETL 处理的每个阶段中包括什么内容，以及相关联的主体域、业务规则、转换和清洗策略。另外通常还包括在这个阶段中不包括那些内容。记录每个阶段的范围，并让业务人员查看并签字确认可以有助于你的管理，避免范围的变化。

在定义每个阶段的时候要现实一点。虽然你的用户相信你能实现你的承诺，但他们更喜欢经常在整个阶段中随时改变或者增加新的东西。对范围的改变必须通过谈判以及划分优先级，将低优先级的变更放在后续的阶段。在最终完成的文档中将可能改变范围的变更考虑在内。

在用文档记录业务规则、定义范围之后，请和用户一起浏览整个文档，并获得用户的签字确认。管理范围涉及的技术将在本章的“管理范围”中介绍。

设计逻辑数据映射

为了使设计逻辑数据映射更方便，ETL 架构师必须重新察看数据仓库数据模型和所有的业务规则文档。如果有遗留或者新的问题，需要召开额外的会议来获得答案。如果在业务需求分析阶段搜集的业务规则缺少细节信息，ETL 架构师需要对源系统进行手工的分析。一旦所有的问题都已经解决，那么 ETL 专家就可以着手创建逻辑数据映射文档。

交叉参考：第三章包含了创建逻辑数据映射的额外细节信息。

定义数据质量策略

除了能够友好地进行查询以外，数据仓库可否接受的主要因素是数据是否一致和可信。因此定义数据质量评估标准并定义一种策略是 ETL 处理的重要部分。

ETL 主管和数据质量专家共同负责定义数据质量规则。他们的任务是分析源系统的质量，并记录所有确定的数据缺失值。这个工作不仅仅确保了最大限度的清洁的数据进入数据仓库；更令源系统从数据质量考察中获益。这些分析可以发现源系统应用中的不足，为源系统的系统管理员提供更正的机会，避免了将来的数据缺失。

清洗数据的可选方式通常分成两类：

- 在源系统中清洗
- 在 ETL 中转换数据

在源系统中清洗数据无疑是最希望的和有效的选择。但是不幸的是，一方面由于资源的限制这个方案不可行，另一方面由于交易应用开发周期的复杂性和周期的原因及时性得不到保障。另外，数据清理通常要包含解决数据缺失的更正动作。

在所有的细节和最终期限确定并被认可以后，记录每个数据清洗问题的细节以及相关的清洗方案。创建一个项目计划来跟踪那些在源系统中或者 ETL 处理中清洗的问题，这可以帮助你控制用户对于数据缺失问题的期望。

我们建议和负责源系统数据清洗的管理员一起召开周会。通过这些会议可以回顾和更新项目计划，使计划能够准确的反映进度，并就新的问题进行讨论。这些会议将决定数据清洗的可行性，并对新的清理问题的解决策略提供一个取得一致意见的场所。确保源数据的清洗能够按计划进行，因为你的数据仓库依赖于被清洗的数据。花点时间查询源数据库来确保清洗工作的成功是一个好主意。

尽管数据在源系统中被清洗，但是源系统的所有者却不可能增加业务规则来保证数据不再变脏。所以最好由系统分析师和 ETL 架构师一起决定 ETL 代码是否需要阻止脏数据进入数据仓库。ETL 代码可以根据业务规则选择拒绝或者转换脏数据。

如果 ETL 代码使用了预防策略，无论是通过排除还是转换，ETL 处理最好能够提供审计报告来捕获和生成脏数据报表。这样的报表有助于对源数据的连续清洗，并提供了数据仓库中的干净数据到其源的关联。这些元数据还可以作为审计用途，可以用它来跟踪数据和其

源的差异，从而判断数据拥有者的责任。请确保获得用户签名的业务规则和 ETL 使用的数据清洗逻辑。

创建物理的 ETL 处理

一旦数据分析完成，业务规则和逻辑映射就是最后的，ETL 架构师将和指定的 ETL 开发者一起预演整个逻辑数据映射。这个预演确保了 ETL 开发人员在开发编码之前就能够理解完整的需求。ETL 开发人员的职责是将逻辑数据映射转化为物理的 ETL 方法。无论是写 SQL 脚本还是使用专门的 ETL 工具，这些方法必须经过开发和测试的，在他们 ETL 架构进行迁移之前，结果数据必须得到开发人员的验证。

一旦同时有多个方法同时交给开发人员时，ETL 架构师通常需要准备 ETL 构建顺序文档作为开发指南。如图 10.5 所示，文档中包含了一个需要加载表的列表，创建方法的起点位置，以及在方法实现中需要注意问题的注释。在项目的初期阶段以及对于那些你团队中的新人，此文档非常重要。

表名	ETL任务创建顺序	描述
d_SHIP_TERM_FLAG	1	直接从平面文件中导出
d_DATE	1	直接从平面文件中导出
d_DEPARTMENT	2	从数据库中导出，带有约束
d_PRODUCT_TYPE	3	
d_VENDOR	3	
d_CURRENCY	4	
d_ORDER_TERM_FLAG	4	
d_REGION	4	
d_OFFICE	5	
d_STORE	5	
d_SHIP_TYPE	5	
d_CLIENT	6	
f_CLIENT_ORDER_TRANSACTION	7	
f_CLIENT_MONTHLY_SNAPSHOT	7	

图 10.5　ETL 创建顺序文档

测试 ETL 过程

绝大多数系统的生命周期理论包含了 3 个阶段的测试。在开发 ETL 过程中，对于新的源系统、主题域或者任何主要的版本的开发建议也采用 3 段论方法。在 ETL 项目开发的每个阶段都应该进行下列的三类测试：

- 单元测试。在开发完毕，开始 QA 测试之前进行单元测试。测试在开发环境中由 ETL 开发人员和系统分析师完成。
- 质量保证测试（QA）。当多个独立的小组在独立的生产环境的镜像中开发后通常要

进行这个测试。环境由 DBA 和 QA 团队成员创建和控制。这个环境用来保证所有的 ETL 处理能够按照预想执行，符合所有的业务规则和期限（加载窗口）的要求。利用模拟的生产环境，QA 小组可以确保 ETL 处理在生产环境中工作。

- 用户验收测试（UAT）。这个阶段通常由你的用户小组在 QA 环境中独立的受控的环境中完成。数据库由 DBA 团队成员控制。在较小的公司，出于减少维护费用出发，在 QA 测试完成后，可以将测试环境开放给用户进行用户验收测试。UAT 这个测试阶段非常有用，通过用户直接操作观察数据来确保处理过程按照预期运行。在 UAT 结束后可以获得你的用户的签字确认。一旦签字确认，你就可以准备将其部署到生产环境。

有些项目对于小的版本和修正 Bug，经常忽略用户验收测试阶段，直接从 QA 测试阶段进入生产环境部署。在这些案例中，在代码加入生产环境后，用户不可避免地会发现问题。省略用户验收测试是一种短视的行为，这使你无法在用户发现之前发现问题，这对于生产环境来说太迟了。

在测试新的 ETL 过程的时候，需要确保对已知的数据问题以及源系统的异常情况进行用户测试。这些测试不仅仅能够验证你的努力，输出干净的数据还会使用户满意，令他们乐于使用新的数据仓库。清洗数据的努力会获得一些正面的影响，用户将乐于正面的评价 ETL 和数据仓库项目，并将更多的主题域纳入数据仓库，项目经理开始将相应的数据进行转换，并加载到数据仓库。

开发测试用例

在 ETL 开发开始后，使用业务规则和缺失数据文档，系统分析师和 ETL 架构师共同负责开发单元测试、QA 测试和 UAT 的详细的测试计划。

测试计划应该包括测试各种业务规则场景的案例。将测试结果和预期结果进行校验可以确保 ETL 代码是正常的，以及转换完全按照设计实现。你的测试用例应该故意的将质量很差的数据加载到数据仓库。ETL 过程应该或者拒绝或者经过转换后完成加载，应该生成相应的审计报表。即使是质量差的数据也不是故意的加载，提供测试数据仓库中数据质量的查询语句，以确保数据清洗转换按照预期工作。

很可能，问题将在验证测试案例的过程中被发现。一些问题可能使你的代码产生 Bug，另外的一些可能是由于用户按照新的格式操作数据造成的。在这个阶段，接到用户希望增加新的需求的请求非常正常，这些需要作为增量加入。千万小心：数据不是这里测试的唯一目的。管理初期的 ETL 是一个任务，但是随着增加 Bug 的修正、额外的需求以及不断变化的业务需求，处理可能会变得不可管理。深入的变更管理技术在本章的“管理范围”中进行了详细地介绍。

一个测试案例样例模版如图 10.6。目标是捕获你测试的需求，执行测试的详细步骤，期望的结果；测试的状态：通过或者失败。测试案例样例用于记录你希望记录的详细等级。这个模版适用于测试过程三个阶段中的任何一个。

部署 ETL

下面介绍 ETL 部署。要使得向产品环境的迁移尽量的平滑，需要创建产品支持文档。这些文档应该包括下列的信息：

- 最后的谱系报表
- 运行或者重启增量加载处理的过程

■ 自动加载时间表的细节信息

创建和发布错误恢复过程文档很重要。一旦加载处理失败，用户应该有办法访问坏数据，或者过时的数据。在生产环境交给用户使用之前，必须有一个避免这种情况发生的计划。记录并测试你的错误恢复过程，当错误发生后，你可以快速的恢复数据，并及时地使其能为你的用户使用。

和 DBA 团队一起创建一个健壮的生产环境，加载你的历史数据，使用你的生产调度器开始 ETL 增量加载过程。在生产环境中使用数据测试历史数据和增量数据的加载，确保数据成功地被加载。

测试案例模版							
编号	测试主题	步骤	测试细节	从属	成功或失败	预期结果	注释
1	将在业务系统中新出现的产品成功增量加载到数据仓库中		新产品以及细节存储在业务系统中，运行增量加载。查询数据仓库实例，并验证结果。	测试结果基于成功的增量装载		数据仓库中产品编号89076与源系统相匹配	
		a	将QA实例记入源系统日志				
		b	在源系统中添加编号89076的产品以及细节。				
		c	保存新纪录				
		d	通知ETL经理或DBA执行增量加载				
		e	在查询工具中读取记录，并记入日志				
		f	执行查询Select * from d_Product where Product_Id = 89076			数据仓库中编号89076产品与源系统中一致	
		g	验证查询结果			查询结果与交易系统中一致	
2	增量装载中相同产品编号的记录无法进入到数据仓库中，同时自动生成报告。		多个相同产品以及细节存储在业务系统中，运行增量加载。查询数据仓库实例，并验证结果。	测试结果基于成功的增量装载		数据仓库中产品编号为89076只有一条记录。自动生成报告。意味着源系统中重复的记录被去除。	虽然在源系统中去除了重复记录，但是没有设置业务规则的权限来防止重复记录，所以必须要用ETL程序防止重复记录进入数据仓库，生成报告
		a	将QA实例记入源系统日志				
		b	在源系统中添加编号89076的产品以及细节。（已有相同的一条记录）				
		c	保存新纪录				
		d	通知ETL经理或DBA执行增量加载				
		e	在查询工具中读取记录，并记入日志				
		f	执行查询Select * from d_Product where Product_Id = 89076			返回1条记录	
		g	验证查询结果只有一条记录				
		h	验证是否自动生成报告有相同的产品编号			数据仓库中产品编号为89076只有一条记录。自动生成报告。意味着源系统中重复的记录被去除。	

图 10.6　测试案例模版

维护数据仓库(ETL)

根据你的企业组织的情况，数据仓库项目经理和 DBA 团队通常负责数据仓库正常维护。但是，你是 ETL 过程的所有者，除非安排了其他人，否则它的维护也是你的职责。

当进入生产环境以后，你需要对数据仓库中已知的内容问题进行持续的监控。开发捕获已知问题的审计报告是维护的一部分。这些审计报告应该利用业务规则和数据缺省值文档，这些审计报告需要能够自动的通过 E-mail 发给相关的联系人。

在任何生产 IT 环境中，补丁和升级是不可避免的。这些补丁和升级通常和数据仓库环境关系密切，因为在一个解决方案中集成了如此多独立的工具。根据需要不断的使用补丁和升级，建议你建立系统维护时间表，并在维护时间内执行这些升级操作。所有的补丁和升级的开发必须经过完整的开发周期，包括开发环境中的单元测试，QA 测试和用户验收测试。通过测试的补丁和升级确保了正确的升级，保证了所有的处理能够按照预期运行。

当新版本或者增强版本发布到生产系统的时候，需要让客户努力地适应新的变化。沟通是必要的，沟通可以帮助用户为变化做好准备。你的用户可能会等待一个新的版本或者增强版本。要给他们一个明确的版本的时间表，这可以推进他们的数据仓库经验。

管理范围

在开始项目后，不需要多久你就会意识到定义范围并获得用户签字确认有多么的重要。当你需要应付大量需求变更的时候，项目很容易失控。

对 ETL 规范不去管理，在开发和测试阶段经常会收到增加的需求，任意的变更都会关系到项目的成功与否。问题肯定会被发现，新的想法会浮出水面，所有的都需要被马上实现。根据我们的经验，当数据仓库揭下神秘的面纱后，新的期望和需求会产生，当越来越多的主题域部署后，需求会指数级的增长。在你开始了解这个情况之前，你会被你和你的团队在承受能力之外的工作狂轰乱炸。记得某人提到过的范围蜕变吗？创造一种机制来跟踪和控制这些变化对你的成功至关重要。下一节将提供你需要跟踪变化所用到的文档，以及帮助你执行的推荐流程。

变更的跟踪

作为一个成功的 ETL 主管，需要一个能够跟踪需求改进、Bug 修改或者最初确认范围的修改的处理流程。下面列出了一些已经证实对于捕获和跟踪需求变化非常有效的元素。你可能希望能够跟踪和管理下列的信息，尽管这意味着需要创建一个简单的电子表格。捕获下列的元素可以帮助管理变化，及最小化范围蜕变。

为了这个目标，可以很容易的使用 Microsoft Access 或者其他小型的个人数据库应用来创建一个系统。对于大型的项目，你可以实现一个小的 VB 应用，或者利用已经投资的系统。当然，好的 ETL 工具可以提供这个能力。

- 主题范围。这是需求所对应的数据仓库中数据集市的名字。
- 请求日起。请求发生的日期。
- 变更描述。用于获取这个需求的概要性的描述。
- 优先级。高，中或者低。这是需求协商后的重要性等级。
- 变更类型。无论这个需求是一个新的需求，还是对现有处理的改变
- 状态。状态的值可以包括表示这个需求状态的任何内容。我们曾经使用：新需求、开发人员调研、开发人员开发、需要进一步信息、中止、通过单元测试、通过 QA、通过 UAT，就绪可供生产，等等。这个字段的值在整个需求的周期中变化。
- 提交者。变更需求的提交者的姓名。
- 属主。表明这段时间谁负责这个需求。通常开始于数据仓库或者 ETL 主管，然后

分配到相应的开发人员、测试人员或者在整个需求周期中的其他责任人。

- 问题版本。当发现 Bug 或者提交请求时所使用的版本号。
- 修正后版本。需求实现后，发布到生产环境的版本号。
- 状态。打开或者关闭。只有当状态设为发布到生产环境或者取消后才可以将状态设为关闭。
- 关闭日期。状态改为关闭的时候的日期。
- 功能描述。这个信息描述了用户提出需求的经过。
- 技术性描述。这个信息通常由超级用户或者 ETL 架构师来填写。用于开发人员的编码实现改动或者新需求。

能够从这个列表中产生报表是非常有用的。如果你有资源来将其作为一个 IT 系统，请和你的 ETL 团队讨论他们对这个流程的建议，确保这个流程适合你团队中的每一个人。

但是，对数据仓库生产环境的最小化更改仍然是最佳方案，如果可能的话。请牢记，改动的越多，对其他的处理影响的风险就越大。我们建议同相关的团队成员召开例会讨论每个需求的优先级。让你的客户认识到生产环境的最少变动是多么至关重要。确保他们理解完成每个需求所需的成本和投入。让他们问一下自己：他们的投资回报如何？有多大的好处值得做这样的修改？他们是否能够调整需求呢？

一旦最后确认需要改变，那么你必须讨论改变带来的影响。如果改变影响到其他的 ETL 过程，或者其他域，那么必须进行详尽的影响分析。要做的改动可能会带来 ETL 处理上许多新的改动。要确保增加这些新的改动到你的跟踪系统。

图 10.7 展示了一个变更/改进的申请表格。这个表格中包含了所有必要的信息，供你输入新的请求，进行开发和完成需求。这个表格配合你的需求变更跟踪系统，可以高效的管理需求变更流程。

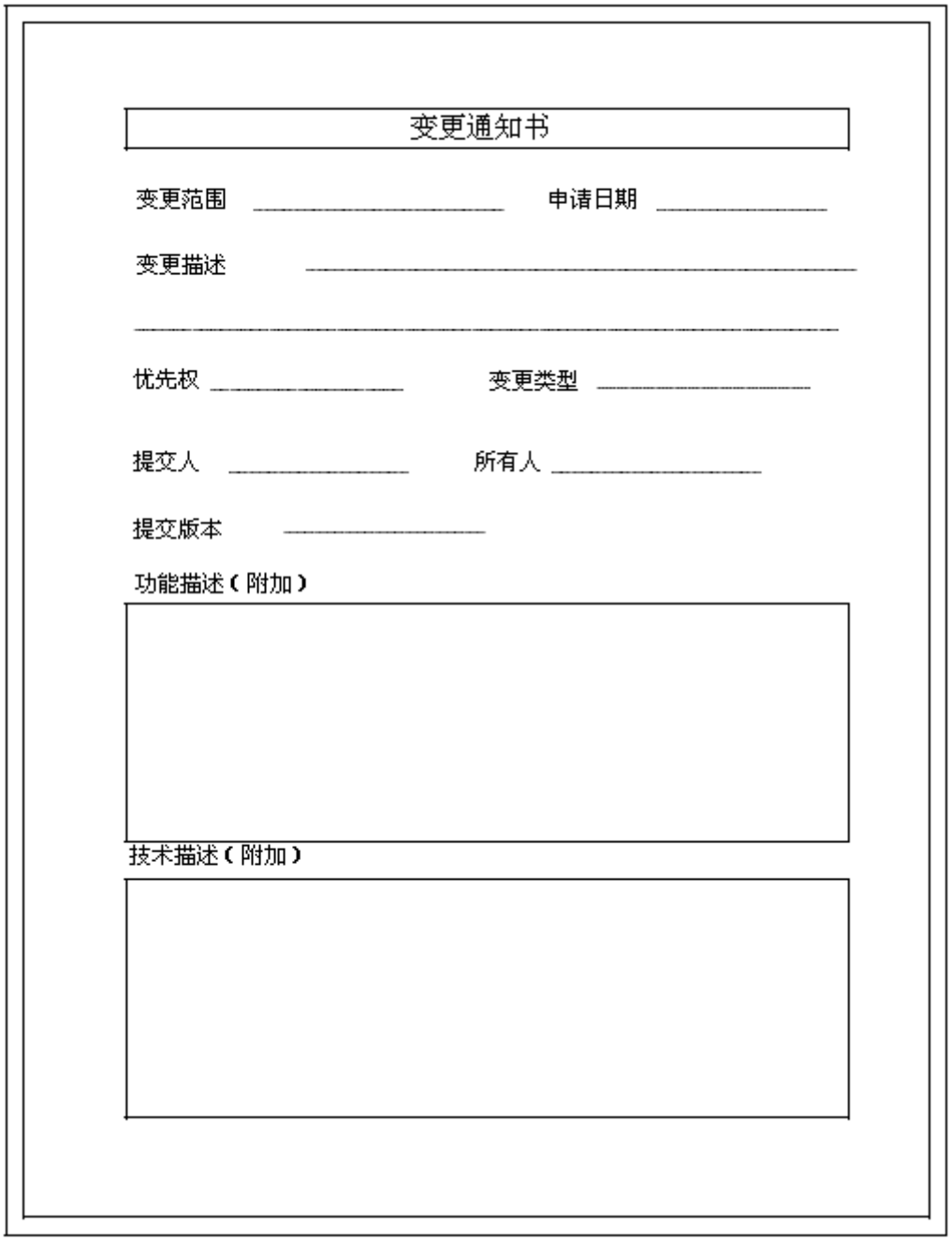

图 10.7　变更/改进申请表

版本发布时间表

通常，许多变更、需求改进、补丁和升级会作为一个单个的版本或体系捆绑在一起。每个版本必须通过所有的开发生命周期。在进入生产环境之前，必须需要单元、QA 和 UAT 测试。之后变更通过测试环节，或者 ETL 架构师进行方法迁移，或者 DBA 团队将代码发布到生产环境。

跟踪你的数据仓库的版本有助于解决生产环境中发现的问题。使用本章开始介绍的跟踪机制控制你整个的版本发布。通常，下列的标准的版本标识技术对数据仓库和 ETL 环境都非常有效。由于很多数据仓库的代码版本都是由 ETL 团队开发和部署的，所有 ETL 主管遵循这样的规则尤其的重要。

版本号有一系列的 3 个由小数点分割的数字组成（##.##.##）。第一组数字表示主版本，

第二组表示小版本号，第三组表示补丁。例如，版本 1.2.1 意味着数据仓库是他的第一个版本，有两个小的版本，应用了补丁 1。

在数据仓库环境中，主要版本发布通常表示有一个包含新的事实、维度、ETL 过程的新的主题域或者新的数据集市。一个小版本表示根本上的 ETL 修改，可能还包括小的数据结构的修改。补丁通常是紧急修订的结果，在生产系统中发现了一个关键错误，需要进行立即的补救。如果补丁和小的更改，或者小的更改和主要的更改同时发生，那么只有最左面的数字会增加，而右侧的数字清零。例如，如果版本 1.2.1 是生产版本，你有两个补丁、一个小的更改，以及一个预定要迁移的主要版本，将这些变更打包将构成一个单一的新的版本。在本例中，版本将是 2.0.0。

在两个间隔时间较长的紧急修订之间安排一次主要版本的绑定是一个好主意。通过安排发布主要版本，可以是月粒度，可以很容易的将小的修订绑定到受控的版本环境，从而减少代码迁移。

我们建议数据仓库版本策略非常的有效，尤其是你的项目使用数据仓库总线架构的时候。在这种情况下，总线矩阵的每个数据集市进入物理数据仓库将是一个主要的版本发布。如果你的数据仓库的版本是 1.0.210，那么你很可能没有使用这个矩阵，而且可能通宵未眠。

总结

本章中，我们最后一次从大量的 ETL 团队任务中退回来，以对整个团队获得完整的概念，有哪些人参与，他们应该考虑什么问题。我们必须牢记本章以及整本书专门讨论涉及企业数据仓库的后端。

我们通过描述 ETL 团队的规划、以及面临的领导能力挑战开篇，然后我们深入这些成员所面临的每一个专门的任务。在书的主要内容中提供了很多案例以及非常细节的信息。