致谢

首先我们要感谢阅读了数据仓库工具箱丛书的成千上万的读者，感谢你们对于撰写这本数据仓库 ETL 书籍的大力支持和鼓励。我们确信还会不断地从你们——数据仓库的使用者和开发者——那里学习到更多的东西。

非常感谢 Jim Stagnitto，是他鼓励 Joe 开始此书，并给了他足够的信心完成该项目。Jim 是事实上的第三作者，他在数据质量和实时 ETL 的章节中作出了主要的创造性贡献。

另外还要感谢 Jeff Coster 和 Kim M.Knyal，他们在前/后加载处理和 ETL 项目管理的讨论中做出了特殊的贡献。

我们有一个特别的读者小组，他们仔细阅读了本书手稿的第一版本，并给出了大量有用的建议。一般来说，对于已经“完成”的手稿再做修改是非常令人头疼的事，但是这种深层次的校读对于工具箱系列的书籍来说是一种传统，这次也不例外，并且非常成功。以字母顺序，这些读者包括：Wouleta Ayele, Bob Becker, Jan-Willem, Beldman, Ivan Chong, Maurice Frank, Mark Hodson, Paul Hoffman, Qi Jin, David Lyle, Michael Martin, Joy Mundy, Rostislav Portnoy, Malathi Vellanki, Padmini Ramanujan, Margy Ross, Jack Serra-Lima, 以及 Warren Thornthwaite。

我们还特别感谢 Robin Caserta（Joe 的妻子）和 Julie Kimball（Ralph 的妻子）在整个项目期间给予的支持，还有孩子们 Tori Caserta, Brian Kimball, Sara（Kimball）Smith 以及孙子 Abigail Smith（!），他们对于作者总是处于工作状态给予了充分的理解和耐心。

最后，Wiley 计算机图书小组又一次成为了本书完成的主要原因。感谢你们，Bob Elliott, Kevin Kent 和 Adaobi Obi Tulton。

作者简介

Ralph Kimball 博士，Kimball 集团的创始人，从 1982 年起就成为数据仓库领域的领导者，也是当今最著名的演说家、咨询专家、教师和作家之一。他的书籍包括<<数据仓库工具箱>>（Wiley,1996）、<<数据仓库生命周期工具箱>>（Wiley,1998）、<<Data Webhouse 工具箱>>（Wiley,2000）和<<数据仓库工具箱第二版>>（Wiley,2002）。他自 1995 年以来一直为<<智能企业>>杂志撰写专栏文章，并获得了 1999 年度“读者最爱奖”。

Ralph 在斯坦福大学获得了电子工程学博士学位，论文题目是人-机系统设计。他从 1972 年到 1982 年在 Xerox PARC 和 Xerox System 的开发部门任研究员、系统开发经理和产品市场经理。由于他在 Xerox Star 工作站（第一个视窗、图标和鼠标的商业产品）的系统设计工作，IEEE Human Factors Society 授予他 Alexander C. Williams 奖。从 1982 年开始到 1986 年，Ralph 是 Metaphor 计算机系统公司（第一家数据仓库公司）的应用部副总裁。在 Metaphor，Ralph 发明了“胶囊”工具，这是图形化数据流界面的第一个商业实现，现在已经应用在所有 ETL 工具中。从 1986 年到 1992 年，Ralph 成为 Red Brick 系统公司的创始人和 CEO，该公司提供一种非常快速的关系型数据库技术来实现决策支持。1992 年，Ralph 创办了 Ralph Kimball 协会，在 2004 年这个协会演变成为 Kimball 集团。Kimball 集团是一个高度专业化的数据仓库设计专家团体，这些专家因其在咨询、教育、演讲和著作方面的成就而享有较高的声誉。

Joe Caserta 是 Caserta Concepts, LLC 的创始人和负责人。他是非常有影响力的数据仓

库专家，其专业技能来自于主流数据仓库工具和数据库的行业经验和实践应用。Joe 就读于纽约哥伦比亚大学的数据库应用开发和设计专业。

介绍

抽取-转换-加载（Extract-Transform-Load）系统是数据仓库的基础。一个设计良好的 ETL 系统从源系统抽取数据，执行数据质量和一致性标准，然后规格化数据，从而使分散的源数据可以集中在一起使用，最终再以可以展现的格式提交数据，以便应用开发者可以创建应用系统，也使最终用户可以制定决策。该书围绕这四个步骤进行组织。

ETL 系统既能成就数据仓库也能毁了它。因为虽然创建 ETL 系统是后台工作，对于最终用户并不可见，但是对于实施和维护一个典型的数据仓库系统来说，它所耗费的资源会很容易达到 70%。

ETL 系统能使数据明显地增值，它的工作也绝不是简单的把数据从源系统抽取到数据仓库中。特别是，ETL 系统能够：

- 消除数据错误并纠正缺失数据
- 提供对于数据可信度的文档化衡量
- 为保护数据获取相互作用的数据流程
- 把多个源数据整合到一起
- 将数据进行结构化供最终用户使用

ETL 是个既简单又复杂的题目。几乎所有人都能理解 ETL 系统的基本作用：把源中的数据加载到数据仓库中。另外大多数人也都认为在这个过程中清洗和转换数据是必要的，这就是简单的观点。然而一个无法更改的事实是，紧接着的下一步就要根据数据源、业务规则、现存软件系统以及特定的报表应用系统的不同，将 ETL 系统分拆成成百上千的小的子过程。这带给我们的挑战是，既要耐心地对待这上千个子过程，同时又要保持对整个 ETL 系统主要目标的简单视角。看看本书是如何应对这一挑战的吧！

<<数据仓库 ETL 工具箱>>是创建成功的 ETL 系统的实践性向导。该书并不是所有可行方法的调查和总结！相反，我们只是针对构建维度数据（Dimensional Data）这样的目标而建立了一系列的一致性技巧。维度模型被证明是创建数据仓库最可预计的和最有效节省成本的方法。同时，由于不同数据仓库的维度结构大都类似，因此我们可以重用大量的代码模块和特殊的开发逻辑。

该书是规划、设计、创建和运行数据仓库后台的路线图。我们将传统 ETL 中的抽取、转换和加载扩展为更可操作的步骤：抽取、清洗、规格化和提交。当然我们并没有试图将 ETL 改为 ECCD！

在本书中，你将学习到以下内容：

- 规划&设计你的 ETL 系统
- 从多种可能的架构中选出最合适的
- 对实施过程进行管理
- 管理日常的操作
- 为 ETL 过程建立开发/测试/生产环境
- 理解不同的后台数据结构，包括平面文件、规范化框架、XML 框架和星型连接（维

度）框架

- 分析和抽取源数据
- 创建完整的数据清洗子系统
- 将数据结构化为维度框架，以便更有效提交给最终用户、商务智能工具、数据挖掘工具、OLAP 立方体和分析应用系统
- 使用同一种技术将数据有效地提交到高度集中的或分布的数据仓库
- 调整个 ETL 过程使性能达到最优

以上观点是 ETL 系统中主要的大问题，但是尽可能的，我们还会提供更细层面上的技术细节：

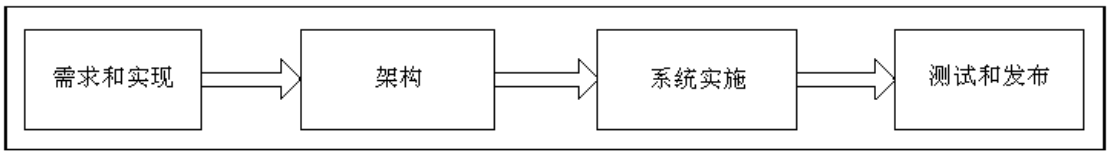
- 针对列属性、结构、有效值和复杂业务规则实施数据清洗系统的关键执行步骤
- 将多个源的异构数据规格化为标准化的维表和事实表
- 创建可复用的 ETL 模块用于处理维表中自然时间变量，例如，三种类型的缓慢变化维（SCD）
- 创建可复用的 ETL 模块用于处理多值维和层次维，这两者都需要相应的桥接表
- 针对海量事实表的加载进行处理
- 优化 ETL 过程以适应加载时窗的要求
- 如何将批处理和面向文件的 ETL 系统转换为连续的流式实时 ETL 系统

 **为方便起见，在提供 SQL 代码时我们选择 Oracle 作为缺省的数据库。然而，产生相同结果的类似代码也可以用于 DB2、Microsoft SQL Server 或者其它主流关系型数据库系统。**

也许作为所有这些特定建议的副产品，希望我们在创建、部署和管理数据仓库 ETL 系统方面的激情能够感染您。

全书概览：两条并存的主线

创建 ETL 系统是一个非同寻常的挑战，因为它是如此不可避免的受现实的约束。ETL 小组必须在业务需求、源数据的格式和缺失、现有的遗留系统、员工的技能以及最终用户持续变化（而合理）的需求当中求生存。如果这些困难还不够，那就再加上预算受限，处理时窗太小，如果 ETL 系统不能向数据仓库及时提交数据则重要业务会慢慢停下来！



图表 1　ETL 的规划和设计主线

当创建 ETL 系统时，头脑里必须有两条并存的主线：规划&设计主线和数据流主线。在最高级别上，它们都很简单，两者在图中都从左至右顺序进行。它们相互作用使整个生命周期变得有意义。在图 1 中显示了规划&设计主线的 4 个步骤，在图 2 中显示了数据流主线的四个步骤。

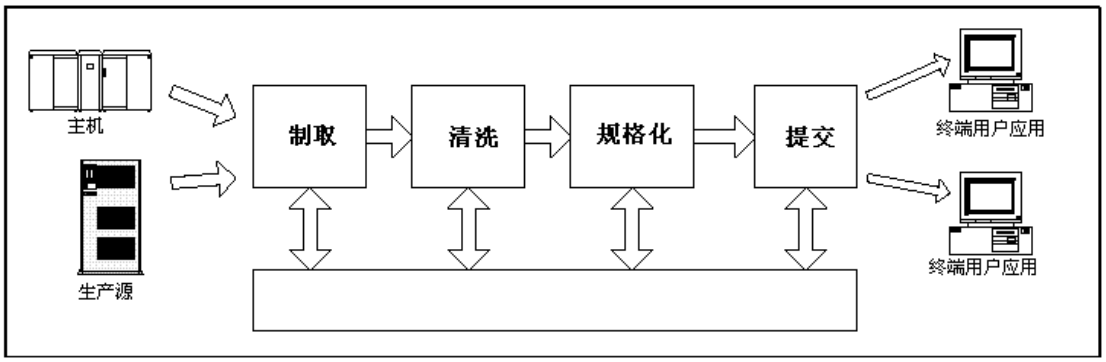
流程检查

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

规划与设计主线

规划与设计主线的第一步是描述所有的需求和现状，包括：

- 业务需求
- 数据评估和数据源现状
- 监察需求
- 安全需求
- 数据集成
- 数据延迟
- 归档和沿袭
- 最终用户提交界面
- 可用的开发技能
- 可用的管理技能
- 已有的许可证



图表 2　ETL 的数据流主线

我们将在第 1 章中详细讲述这些内容，但是我们必须事先弄清楚它们是怎样影响 ETL 系统的本质。在这一步骤中，以及在两条主线的所有步骤中，当我们讨论指定步骤时，都会指出其在这本书的位置。

该主线的第二步是架构，在这里我们必须做出关于创建 ETL 系统创建方法的最主要的决定。这些决定包括：

- 手工编码还是使用 ETL 工具
- 批处理还是流数据处理
- 水平任务依赖还是垂直任务依赖
- 自动调度
- 异常处理
- 质量控制
- 恢复与重启
- 元数据
- 安全

规划与设计主线的第三步是系统实施。我们希望在实施步骤之前已经在前两个步骤上花

费了相当的精力！这一步骤包括：

- 硬件
- 软件
- 编码
- 文档
- 特定质量检查

最后的步骤看起来像是系统管理，但是测试/发布步骤的设计与前面两个设计步骤至少同等重要。测试/发布包括下列设计内容：

- 开发系统
- 测试系统
- 生产系统
- 提交过程
- 升级方案
- 系统快照和回滚过程
- 性能调优

数据流主线

数据流主线可能是大多数读者最熟悉的，因为可以认为这是常规的 ETL（抽取－转换－加载）场景的一个简单的概括。当你阅读下面的列表时，你可以开始去想像规划&设计主线是如何影响下列每一因素的。抽取步骤包括：

- 读取源数据模型
- 连接并访问数据
- 调度源系统，截取通知和后台程序
- 捕获变化数据
- 将抽取的数据集结到磁盘

清洗步骤包括：

- 强制列属性
- 强制结构
- 强制数据和数值规则
- 强制复杂业务规则
- 创建元数据来描述数据质量
- 将清洗后的数据集结到磁盘

紧接着是规格化步骤，包括：

- 业务标志（在维表中）的规格化
- 业务度量和绩效指标（在事实表中）的规格化
- 复制
- 家庭关联信息的规格化（Householding）
- 国际化
- 将规格化的数据集结到磁盘

最后一个步骤，把完美的数据提交给最终用户应用。我们在第 5 章和第 6 章来讨论提交

技巧，就像将在第 1 章中提到的那样，无论如何，在菜做好之后，总得把它端到客户面前！ETL 系统的数据提交包括：

- 加载平面型和雪花型维度
- 生成时间维
- 加载退化维
- 加载子维
- 加载缓慢变化维（包括类型 1、2、3）
- 规格化维和规格化事实
- 处理迟到维和迟到事实
- 加载多值维
- 加载 ragged 层次维
- 加载维中的文本型事实
- 为事实表运行代理健 pipeline
- 加载三种基础事实表粒度
- 加载和更新聚合表
- 将提交数据集结到磁盘

在学习上面这个列表时，你可能会说：“看起来大多数列表中的内容是建模，不是 ETL。这些问题应该属于前台工作！”我们觉得不尽然。我们曾经与 20 多个数据仓库开发小组进行过沟通，超过一半的小组认为 ETL 系统设计应该与目标表的设计是同时的。这些人虽然也认为应该有两种不同的角色：数据仓库架构师和 ETL 系统设计师，但是，这两个角色却经常是由同一人担当！因此这也就解释了为什么本书包含了从数据源到目标维度数据库配置的所有内容。

数据流的四个基本步骤都由运行步骤进行监控，它贯穿了从抽取到提交的全部步骤。运行包括：

- 调度
- 作业执行
- 异常处理
- 恢复和重启
- 质量检查
- 发布
- 支持

理解这两个基本的主线（规划与设计主线和数据流主线）是阅读本书的真正目的。

本书如何组织

为了描述清楚这两条主线，我们把本书分成四个部分：

- 需求、现状和架构
- 数据流
- 实施和操作
- 实时流式 ETL 系统

本书从规划&设计的需求、现状和架构开始，因为我们必须为任何类型的 ETL 系统建立一个逻辑基础。本书的中间部分跟踪了从抽取步骤到提交步骤的整个数据流主线。在第三

部分我们返回到实施和操作问题上。在最后部分，我们会涉及令人激动的实时流式 ETL 系统这一新领域。

第 1 部分：需求、现状和架构

本书的第 1 部分为后续的内容讲解布置了一个沙盘。即使大多数人都非常希望马上开始将数据迁移到数据仓库，我们还是必须回过头来看一下。

第 1 章：环境和需求

数据仓库中的 ETL 部分通常都面临异常复杂的设计挑战。在本章中我们在你牢记一种方法之前告诉你在需要考虑的需求列表上的一些实质性问题。我们还会介绍你必须遵循的主要结构上的决定（不管你是否这样认为）。

本章是定义数据仓库术语表——至少是本书需要的词汇——的最佳位置，这些词汇包括：

- 数据仓库（Data Warehouse）
- 数据集市（Data Mart）
- 操作型数据存储（ODS, Operational Data Store）
- 企业数据仓库（EDW, Enterprise Data Warehouse）
- 集结区(Staging Area)
- 展现区（Presentation area）

我们在描述 ETL 小组负责的任务同时，也描述了数据仓库的任务。我们简短的介绍了四种基础的数据流阶段：抽取、清洗、规格化和提交。最后还尽可能清晰的描述了为什么我们认为维度数据模型是每个数据仓库成功的关键。

第 2 章：ETL 数据结构

每个 ETL 系统都必须将数据集结为各种永久性的或准永久性的格式。当我们提到“集结（Staging）”时，意味着写数据到磁盘，也正是由于这个原因，ETL 系统有时称为集结区（staging area）。你可能注意到我们建议在每一格 ETL 主要步骤（抽取、清洗、规格化和提交）后都有几种形式的集结。我们在本章讨论采用不同集结形式的原因。

然后我们提供了在通常 ETL 系统中所需的重要数据结构的系统性描述，包括：平面文件、XML 数据集、独立的 DBMS 工作表、规范化的实体/关系框架和维度数据模型。为了描述的完整性，我们提到了一些特殊的表，包括用于验证重要数据集出处的合法性审计跟踪表，以及用于跟踪代理键的映射表。我们通常围绕这些类型的表总结出元数据以及命名标准的报告。本章的元数据部分只是一个初步介绍，因为元数据的问题太过重要，我们会在本书中多次提到。

第 2 部分：数据流

本书第 2 部分展示了用于从多种不同的源系统有效抽取、清洗、规格化和提交数据到理想的维度数据仓库中的实际步骤。我们从用于分析源系统的记录系统的选择和策略建议作为

开始。该部分还包括一个重要的章节，即创建 ETL 中的清洗和规格化系统。后两章则把经过清洗和规格化后的数据整合成所需的维度结构，以便提交给最终用户环境。

第3 章：抽取

我们会在本章的开始阶段介绍，数据分析完成之后，设计逻辑数据映射需要什么样的前提条件。我们会建议创建一个逻辑数据图，并告诉用户怎样利用这样的图来避免关键任务描述的含糊不清。逻辑数据映射提供给 ETL 开发者所需的功能特性来建造 ETL 过程。

数据仓库的一个主要任务是把来自整个企业中不同的旧应用系统的数据整合到一个统一的资料库中。本章提供了专门的技术性指导，用于集成整个企业中的异构数据源，包括主机系统、关系型数据库、XML 源、平面文件、Web 日志和 ERP 系统等。我们会讨论在集成这些数据源时遇到的障碍以及提供如何克服它们的建议。我们还会介绍在多个潜在的不一致数据源间规格化数据的概念，这是下一章要展开详述的主题。

第4 章：清洗和规格化

在抽取数据之后，我们建议对它们进行清洗和规格化。清洗的意思是确认和修复数据中的错误和缺失。规格化的意思是解决潜在不一致的数据间的标记（Labeling）冲突，以便它们能够在企业数据仓库中一起使用。

本章提供了一些非同一般的特殊技巧和方法，这些技巧和方法对于清洗和规格化系统的建设非常有帮助。本章的主题集中在数据清洗的目标、技术、元数据和衡量方面上。

具体地说，技术部分主要是数据评估和数据清洗的关键方法，衡量部分主要是一些例子，包括如何实现触发告警的数据质量检查，以及如何为保证数据健康而进行的数据质量管理提供指南。

第5 章：提交维表

本章和第 6 章是本书的关键章节。我们坚信数据仓库的总体目标是，以简单的、可操作的格式将数据提交给最终用户和分析应用系统。维表是业务度量的上下文，同时也是数据的入口点，因为它们是几乎所有数据仓库约束关系的目标，同时为每一行输出提供了有意义的标记。

加载维表的 ETL 过程是非常有挑战性的，因为它必须消除源系统的复杂性，并把数据转换成简单的、可以选择的维度实体。本章一步一步地解释了如何加载数据仓库维表，包括一些最先进的 ETL 技术。本章试图对以下内容能够做尽量清晰的说明：

- 分配代理键
- 加载缓慢变化维（类型 1、2、3）
- 为多值维和复杂层次维生成桥接表
- 对层次维进行扁平化处理，以及对雪花维有选择的进行扁平化处理

我们讨论了一些高级管理和维护问题，包括增量加载维、使用 CRC 代码跟踪维的变化以及迟到数据的处理等。

第6 章：提交事实表

事实表包含了业务度量信息。在大多数数据仓库中，事实表显然比维表要大得多，但是

同时它们更简单一些。在本章中，我们将解释所有事实表的基本结构，包括外键、退化维键和数字型事实本身。我们还会对事实表提供者这个角色进行描述，他负责把事实表提交到最终用户环境。

每一个事实表都应该使用代理键加载，代理键用于将原始事实记录的真实键映射到连接维表所需的正确对应的代理键。

我们描述了事实表的三种基本粒度，它们足以支持所有的数据仓库应用系统。

我们还描述了一些不常见的事实表，包括无事实的事实表，或者目的仅用于登记某个复杂事件的发生，例如汽车事故。

最后，我们讨论聚合表的基本架构，聚合表以物理方式存储汇总数据，正如索引一样，它的目的就是为了单纯的提高性能。

第 3 部分：实施和运行

本书第 3 部分假设读者已经对需求进行过分析，注意到了数据和可用资源的真实情况，并且已经对从抽取到提交的数据流进行了可视化处理。请牢记这些内容，在第 3 部分我们将更详细描述了 ETL 系统的实现和组织运行的主要方法。我们讨论了 ETL 系统中元数据的角色，最后还有 ETL 小组成员的不同职责。

第7 章：开发

第 7 章描述了开发初始数据加载所需的技术，例如针对缓慢变化维重新生成历史数据，以及集成历史离线数据和当前在线交易数据，另外还有历史事实加载。

本章还提供了计算初次加载所需的时间估算技术，从而发现长时间运行的 ETL 过程，然后提出最小化风险的建议方法。

对于数据仓库项目来说，自动 ETL 处理显然是需要的，但是如何做呢？表加载的顺序和依赖关系是加载数据仓库成功的关键因素。本章回顾了 ETL 调度的基本功能，并且提供了执行 ETL 调度的标准和选项。一旦满足了这些基本原则，那么诸如 ETL 的强制参照完整性和元数据运行维护等主题就都可以进行核查。

第8 章：运行

本章我们以下面内容作为开始，包括调度各种 ETL 系统作业的方法，响应告警和异常，以及最后在满足所有依赖关系情况下完成作业运行。

我们会遍历将 ETL 系统迁移到生产环境的所有步骤。就像任何其它关键任务应用系统一样， ETL 系统的生产环境必须得到保障和支撑，因此我们描述了如何建立 ETL 系统保障级别，以确保在调度过程失败时起作用。

为了确定 ETL 系统的执行状况，我们制定了一些关键性能指标，同时我们还揭示了如何监控和捕获统计信息。一旦 ETL 关键性能指标收集完成，我们就拥有了足够的信息来检查 ETL 系统中的组件，以便尽可能地优化整个系统。

第9 章：元数据

ETL 环境经常被认为应该负责存储和管理整个数据仓库的元数据。确实， ETL 系统是存储和管理元数据的最好的位置，因为 ETL 要对数据作合适的处理就必须知道数据个方面的情况。第 9 章定义了三种类型的元数据（业务、技术和流程），并展示了每种类型应用在 ETL 系统中的具体元素。本章提供了生成、发布和使用不同类型元数据的技术，还讨论了数据仓库在该部分可能增强的机会。最后讨论了元数据标准以及最佳实践，并提供了针对 ETL 的建议的命名标准。

第10 章：职责和分工

ETL 过程的技术问题只是 ETL 生命周期的一部分。第 10 章主要讨论为了成功实施所需的生命周期中的管理问题。本章描述了 ETL 小组的职责和责任，然后描述了一个详细的项目规划，可以用于任何数据仓库环境的实施。一旦有了管理 ETL 系统的基础，本章就进入了更为详细的项目管理活动，例如项目任务分配、范围管理和小组建设。这有点非技术的章节将为 ETL 和数据仓库项目管理提供最大的帮助。它描述了一个有效小组所需的角色和技能；提供了可以在数据仓库每一阶段重复使用的丰富的 ETL 项目规划。本章还包括在 ETL 生命周期中管理者需要领导其小组的形式。甚至如果你不是管理者，本章也需要阅读，以便充分理解你的角色如何与 ETL 小组中其它成员一起工作。

第 4 部分：实时流式 ETL 系统

由于实时 ETL 是相对较新的技术，我们更像是在讨论一个还没有完全实现的需求和解决方案。在本章中，我们会分享我们的经验，对实时数据仓库的最新挑战进行分析，并提供应对这些挑战的建议。本章提出了实时 ETL 的关键难点，以及真正实现的详细描述。

第11 章：实时 ETL

在本章中，我们以定义实时需求开始。紧接着，我们回顾目前可选的各种架构并对它们进行评价。我们以一个决策矩阵结束本章，帮助你决定对于特定的数据仓库环境哪一种实时架构是合适的。

第12 章：总结

最后一章总结了本书的贡献，并提供了 ETL 和数据仓库作为一个整体在将来的展望。

谁将阅读本书

任何参与或者将要参与数据仓库建设的人都应该阅读本书。开发者、架构师和管理者将从本书获益，因为它包含提交一个面向维度的数据仓库的详细技术描述，还提供了关于所有后台活动的项目管理视图。

第 1、2 和 10 章提供了 ETL 的功能视图，数据仓库小组每一成员都能很容易地阅读，但其主要是针对业务发起人和项目管理者的。在你阅读这些章节时，希望能提高你的技术水平，最终能成为开发者的手册。本书是加载维度数据仓库所需任务的建议性向导。

总结

本书的目的是让 ETL 系统的建设过程变得可理解。本书展示了 ETL 系统带给数据仓库数据的增值作用。我们希望你能够喜欢本书，并发现它在你的工作中是有帮助的。我们在整本书中尽量保持中立，以便你能选择应用你喜欢的技术。如果本书没有什么作用，我们希望它能鼓励你思考，并给供应商提供一些挑战，使其扩展其产品的功能来满足 ETL 小组的需要，以便使 ETL（和数据仓库）更加成熟。