结论

为一个数据仓库设计并构建一个 ETL 系统是一种需要保持上述观点的演习，这是一种典型的具有复杂的后台却需要在开始就进行全面计划的任务。这很容易从某个源转换数据并立即放入那些可以查询的表中。然而，用户并不期望得到这些原始的结果，因为这样的做法没有扩展性，也不可管理的。

深化 ETL 的定义

我们回顾一下第一章中介绍的必须考虑的条件。这些包括业务需求，兼容性需求，数据评估结果，如安全性、数据一致性、数据潜伏期、压缩和血缘跟踪，以及用户使用的工具。你们也必须精通你们现有的技术和你们现有的遗留许可。是的，这是一个超越限制的问题。

如果你们时时将这些牢记在心，你们必须做一个很大的抉择：是买一个综合的 ETL 工具还是你们自己写脚本和程序来做？对任何一种选择来说我们都已经努力将这本书不要变得太厚，但是我们认为范围越大，项目周期越长，厂商提供的 ETL 工具就更适合。你们的工作是准备数据，不是做一个软件开发经理。

从我们的观点来说，这本书的实际价值是抽取、转换和加载这三个标准步骤的结构。这本书介绍了一系列为每个步骤设计的特定技术。这不是一本包含所有可能应用来创建一个 ETL 系统的书！我们已经将传统的 ETL 步骤分解成四个步骤：抽取，清洗，规范化，和提交。这本书中的这四个步骤交付的内容是特别区分的，包括：

- 抽取：为 ETL 系统选择特定的原始数据源并结合逻辑数据映射和数据评估结果成为一个计划的方法。通常从源系统开始，我们也建议某些转换在这里处理而不是在更加传统的清洗步骤去做。
- 清洗：为一个错误事件事实表，一个审计维度和一系列的数据质量筛子设计框架。我们说明如何使这些内容有效的整合到 ETL 系统中。
- 规范化：精确的定义一致的维度和一致的事实（通过充分说明维度管理者的责任和义务以及维度和事实的公共策略）。规范化是现在在业界被称为主数据管理的基础。
- 提交：为所有范围的维度模型来细化结构化指定部分，包括缓慢变化维度，主事实表类型，以及为多值维度和层次结构所需的桥连接表。我们展示了如何设计所有的维度框架种类，同时我们提供为每个这些情形的特殊代理键的管理方法。

每一个步骤的内容提供了 ETL 元数据的基础。通过提供实际数据的状态来降低了处理 ETL 元数据的许多疑惑和困难，在清洗步骤中的审计维度直接验证这个观点。由于维度总是用来描述上下文的度量，我们发现在 ETL 系统中的提交部分，交付一个表就是另一种上下文。以这种思想，我们仔细的将所有的数据附加各种审计维度，就像用户用他们熟悉的工具一样。

在第 7 章—开发，我们介绍了许多你们需要的特殊的转换步骤和工具来建立 ETL 系统。如果你们自己写代码，我们提供的例子代码也是直接相关的。如果你们已经购买了 ETL 工具包，大多数的这些步骤和工具都可以在图形化的界面中表达你们的 ETL 数据流程。在第 7 章的第二部分，我们给出了 DBMS 特有技术的一些指南来提高性能，如高速的块加载，强制参考完整性，并行处理的好处，计算维度聚合以及性能问题纠错。

在第 8 章—操作，我们开始全面介绍在 ETL 环境中的作业调度，记住每个环境都有其独特的瓶颈。我们于是提出特定的控制文件来帮助你们进行每天的 ETL 系统管理。这些包

括一个数据集市版本文件，一个 ETL 性能跟踪文件，以及一个使用的度量列表。我们对第 8 章的结论是推荐安全的和压缩的架构。

在第 11 章，我们开始介绍设计一个实时的数据仓库系统。实时对你们目前的 ETL 来说是太快了。但是更进一步的，迁移到实时模式几乎总是需要从批处理 ETL 到流程 ETL 的跳跃。当做这种跳跃时，可能你们的 ETL 系统的每一步和你们的用户工具将需要重新设计。显然，这是不能轻视的一个步骤。然而，几乎所有的批处理 ETL 中的重要步骤都必须在流程 ETL 设计中被用到。你们仍然需要抽取、清洗、规范化和提交。因为这些原因，我们在前 10 章的课程开发中作为实时设计的基础。

数据仓库和 ETL 的未来

IT 事实上只有两个相辅相成任务：获得数据和输出数据。获得的数据,也就是事务处理。过去 30 年来,很多组织已经投资了万亿美元以上的花费，来建立日剧增长的交易处理系统，这些系统是以操作性为目标来获取数据。但数据不能单向流动：在某些点上,我们必须消化数据，并从中获得价值。在商业界，有一个深刻的文化思维：只要我们能够看到所有的数据,我们就可以更有效地管理企业。这一文化思维如此根深蒂固，我们理所当然地承认. 然而这是数据仓库的任务,并且这就是为什么数据仓库在我们所有的组织中是一个永久的实体，即使它有些改变。从这个角度来说，最终,在输出数据的总投资将能与获得数据相匹敌，这似乎是合理的。

在过去五年里,许多重要的观点已经成为数据仓库的动力：

- 数据仓库的蜜月期已经结束. 企业对技术已经失去耐心,他们坚持说从数据仓库可以获得有用的商务结果. 这个主题的名字，至少就现在来说，就称为商务智能（BI）。BI 是由最终用户趋动的，并且 BI 的供应商都控制用户看到的最终显示结果界面。
- 数据仓库已经成为鲜明的操作业务.传统的数据仓库和业务型报表之间距离已经消失.这样，以操作型为中心引起了数据仓库两个大的需求。第一，数据仓库必须能处理企业中的原子交易。如果你要看某个订单是否已经发货，你不能看汇总数据. 数据仓库中的每个主题区域在单独的交易水平上必须有好的途径去获取原子数据。第二,企业中的许多业务观点必须能够实现实时操作，当然,在本书中，我们已经深入地谈到了对实时的定义和对实时的技术支持的挑战。
- 企业希望对他们的业务状况有一个全方位的了解（360 度的视图），而需要全方位了解的信息中最重要的是客户方面的信息。每一个关于客户的信息在企业的数据仓库中是期望能够得到的，而终端用户需要的是一个包含所有相关客户信息的统一视图。这对于数据仓库的数据清洗和规范化是一个巨大的负担，特别是如果没有考虑到操作系统中所有客户视图的合理化。虽然客户是趋使全方位需求的最重要的因素，但是产品、人口、供应商在其它的环境中也具有相同的挑战。
- 最后,数据的爆炸性增长有增无减。在数据获取(尤其 RFIDs)和数据存储技术的进步正在使我们的数据仓库陷入不利境况，引起对建立可用于分析的每个数据粒度的期望。

因此，这些主题如何改变原来的 ETL 任务？

我们认为,最突出的现实问题就是开发和运行 ETL 系统的复杂性。正如我们所说的，这是一个约束过多的问题。再一次阅读第一章的需求列表，当数据的数据量、软件、硬件处理数量的迅速增长，编写你自己的系统是越来越不可行的。将来的系统是允许你集合高水平的建立逻辑块。

当今 ETL 系统的演变

其他技术领域也已经经历了类似的阶段，这个阶段复杂的开端只是简单地集中在工具集成的水平更加复杂，一个芯片的数百万集成电路设计和数百万行的代码的软件发展就是这个演变的例子。如果我们要保持输入数据量的不断增加，ETL 处理的发展不可避免地必须经历相同的过程。

这意味着 ETL 设计者必须渐渐地以系统集成、系统监控和系统建立块的集成为向导，而不是编程。简单来说，是因为没有足够的时间去做低水平的编程。

分析原子数据的主题更加明确，并将会加速发展。市场细化已经细分到个别的用户水平，并且市场分析将要实现能够查询到基于非常复杂特性的组合和连续行为的独立客户集。第六章中，当我们把事实文本放置在客户维度的时间序列位置上，我们看到分析系列行为的挑战。再一次，我们重复我们的基本信念------必须重视 ETL 系统的主键分析模式，比如为了使终端用户的应用程序可用的系列行为分析。ETL 系统就像是一个精美的餐厅中的厨房：ETL 系统必须在食物端出厨房之前安排整理好“盘子”。

系列行为分析也会对查询分发系统造成更多的压力。RFID 标签就像是旅程中通过的门。每一个门就是数据收集的设备，它记录着 RFID 标签的通道。只有当不同的数据库中每一个门可能被集合成一个统一的数据视图时，才可能存在系列行为分析。这样，“旅程中”单独的 RFID 标签和整组的标签就可以被分析。显然易见这是一个合并和规范化的挑战。最近的疯牛病恐慌是这个问题的一个很好的例子。每只牛上的 RFID 标已经标上，但是，没有人能够分析某只特定的有问题的牛从哪里来或者在哪，因为不同的 RFID 产生的数据库不能轻易获得或者被整合。

最后,应该回到这本书贯穿全文的主题，实际上也是作者主旨。数据仓库的价值已经能够响应企业的商业需求。在最后的分析， ETL 系统设计最重要的的特征是以商业为中心，以正确的导向来最有效的分发数据仓库的任务：输出数据为目标，从而来保持数据仓库的高水平的系统技能。