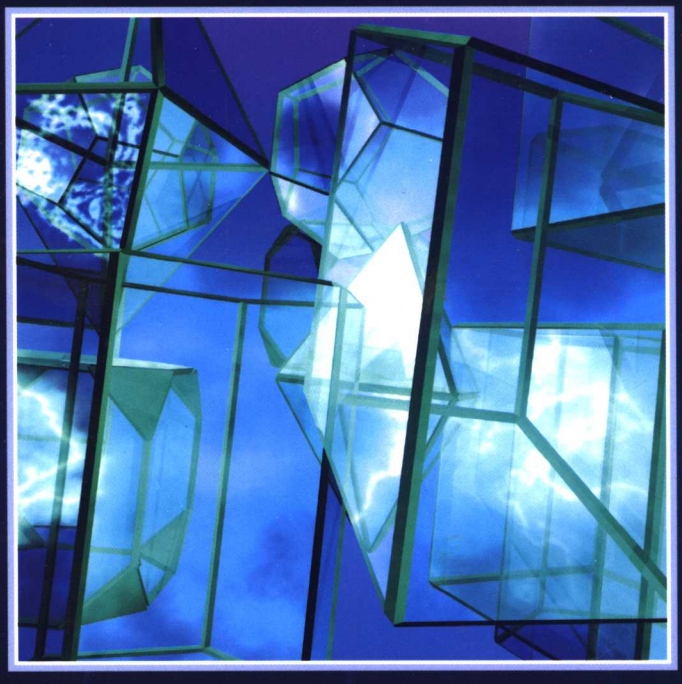
数 据 库 技 术 丛 书

公共仓库元模型
开发指南

Common Warehouse Metamodel Developer's Guide



(美) John Poole Dan Chang Douglas Tolbert David Mellor 著

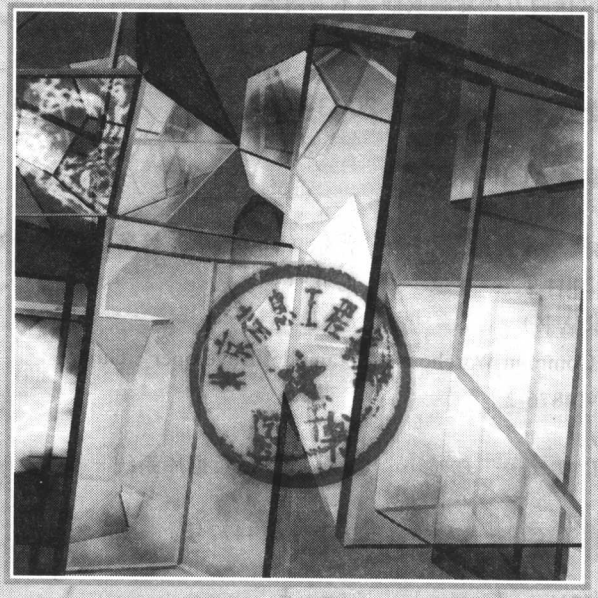
彭蓉 刘进 译 何克清 审校

机械工业出版社
China Machine Press

数据库技术丛书

公共仓库元模型开发指南

Common Warehouse Metamodel Developer's Guide



(美) John Poole Dan Chang Douglas Tolbert David Mellor 著

彭蓉 刘进 译 何克清 审校

机械工业出版社
China Machine Press

本书介绍公共仓库元模型（CWM）的开发方法，从CWM导论入手，由浅入深地阐述了CWM的体系结构及基本技术、使用CWM建模元数据、数据仓库管理模型、维模型、CWM元仓库模型开发集成体系结构以及元数据存储的实现等方面的内容。

本书的作者都是曾经参与制定CWM的专家，从事数据仓库的开发、设计、分析与系统集成的专业技术人员都会从本书受益。

John Poole, et al: Common Warehouse Metamodel Developer's Guide (ISBN: 0-471-20243-6).

Authorized translation from the English language edition published by John Wiley & Sons, Inc.

本书版权登记号：图字：01-2003-6964

图书在版编目（CIP）数据

公共仓库元模型开发指南/（美）普尔（Poole, J.）等著；彭蓉，刘进译. －北京：机械工业出版社，2004.9

（数据库技术丛书）

书名原文：Common Warehouse Metamodel Developer's Guide

ISBN 7-111-14876-2

Ⅰ.公… Ⅱ.① 普… ② 彭… [illegible]数据库系统－指南 Ⅳ.TP311.13-62

中国版本图书馆CIP数据核字（2004）第074477号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：王镇元

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2004年9月第1版第1次印刷

787mm × 1092mm 1/16 · 32.5印张

印数：0 001 - 4 000册

定价：69.00元

对本书的赞誉

“通过深入介绍使用了新的元数据设计模式的数据仓库和业务智能工具链中的模型驱动一体化，本书开辟了一片新天地。通过大量示例不仅介绍了使用UML和MOF定义平台无关模型的方法，同时以大量实例阐述了使用XMI和JMI实现XML和基于Java的元数据管理的方法。正试图解决工具、数据和应用集成复杂性这一难题的软件设计师、CTO、系统集成者和制造商们都能够从这本书中了解OMG模型驱动体系结构的强大功能。”

Sridhar Iyengar

IBM OMG体系结构组杰出工程师

“第一本关于CWM的书《公共仓库元模型：数据仓库集成标准导论》[⊖]已经在很大程度上补充了CWM规范。紧接着的这本书在实现领域进行了更加深入地研究，这对于任何标准的成功都是至关重要的。这本开发指南实现了从“书面标准”到工具集成时实际使用的标准的重要转变。

本书不仅写得很好，而且还密切关注了与标准相关的应用。我向那些想在自身产品策略中把标准转换为现实的人们重点推荐这本书。”

Christian H. Bremeau

Meta Integration Technology，Inc.（MITI）总裁兼CEO

“本书为数据仓库和业务分析领域中集成系统的强大的新方法提供了实践性很强的指南。通过调节这个新标准，使其适用于对应用、工具和实例元数据的建模和交互，作者展示了如何通过更高层次的抽象来表示公共业务和领域概念的方法来解决复杂的、现实世界中的集成问题。

在当今的分布式异构环境中，基于模型的开发方法具有大大简化开发者在构建集成解决方案时所面临的日益增多的复杂问题的潜力，而CWM则是此方法中最主要的一个成功案例。”

Chuck Mosher

Sun Microsystems市场开发工程部高级工程师

“本书不仅说明了如何用CWM描述复杂数据仓库系统，还说明了如何用CWM促进互操作和集成。对于开发平台无关领域模型和为集成和信息交互而实现领域模型感兴趣的人来说，本书是一本极好的指导书籍。”

Ravi Dirckze

Unisys公司JMI 1.0规范制定者和高级软件工程师

⊖ 这本书已由机械工业出版社引进出版。——编者注

译 者 序

随着计算机技术的迅猛发展，计算机软件的规模、数量以及复杂性与日俱增。对大部分组织而言，为决策者提供高质量的业务智能或决策支持需要使用来源于各种不同计算机软件的信息。但这些软件产品往往是不同厂商在不同时期生产的，因而存在着元数据不一致问题，也就为集成这些软件产品的信息带来了极大的困难。因此，将信息供应链各环节中所涉及到的元数据表示标准化迫在眉睫。

为此，早在1993年，电子信息组织（Electronics Information Group, EIG）就发布了计算机辅助软件工程数据交换格式（CASE Data Interchange Format, CDIF），从而为计算机辅助软件工程工具提供了元数据的交换标准，但没有得到业界的普遍认可。1995年10月，一些IT界的精英组成了元数据联盟（Meta Data Coalition, MDC），并于1996年4月发布了元数据交换规范MDIS（Meta Data Interchange Specification）V1.0。MDIS定义了一种与特定产品无关的元数据交换机制，支持MDIS的工具之间能自由地进行元数据交换。与此同时，微软公司也联合了其他一些合作者开发了开放信息模型（Open Information Model, OIM），并于1996年10月，形成了OIM草案。其中采用了UML作为OIM的规范语言，并且紧紧依赖绝大多数由微软和其合作者开发的存储技术。为了确保OIM成为一项公开的标准，1998年11月，微软加入MDC并向其提交了OIM。随后，MDC于1999年7月发布了OIM V1.0。

1998年9月，对象管理组织（Object Management Group, OMG）发布了征求意见稿，要求以其现有元数据和存储体系结构技术标准（即UML、MOF、XMI）为基础，制定公共仓库元数据交换规范。1999年初，一些OMG成员组织（如IBM、Unisys和Oracle等）决定合作编写一个提案。因此，来自IBM、Unisys、NCR、Hyperion、Oracle、UBS AG、Genesis和Dimension EDI等软件公司的设计师在IBM硅谷实验室的Daniel T.Chang博士的领导下开始着手开发这个基于元模型的解决方案，最终形成了著名的OMG的CWM规范。1999年9月，CWM规范的初始版本被提交到OMG；11月，几个合作提交CWM的成员组织举办了一个CWM技术的展示会。

为了解决出现两种元数据集成规范而无法实现统一元数据交换标准的问题，业界对二者进行了长期深入的比较与分析。2000年9月，考虑到业界对CWM建设的支持，MDC成员投票赞成终止他们在OIM上的工作，从而使业界最终拥有了被厂商广泛支持的、单一的和开放的元数据集成标准CWM。

CWM一问世就受到计算机界的广泛重视，因为它指明了数据仓库及业务智能领域发展的方向。因此，许多厂商已经开始使用CWM来开发能够互操作的软件工具。由于CWM能够与OMG推出的系列标准（如MOF、UML、XMI）等无缝连接，并且能够灵活地应用于MDA的模型编译（PIM→PSM）之中，因此，CWM势必成为今后软件工程的主流技术之一。我国软件界对CWM也相当关注。许多研究人员和技术人员已经开始学习和研究CWM。还有许多人想学习CWM，但苦于找不到合适的书籍。由于CWM的复杂性，仅通过CWM的标准文献来学习和使用它确实

不是一件轻松的事。以往国内外也曾发表过一些介绍或评述CWM的著作或论文，但是与CWM的丰富内容相比，这些介绍远不能满足读者需要系统了解CWM的需求。

值得高兴的是，CWM规范的主要设计者J. Poole、D. Chang、D. Tolbert、D. Mellor亲自撰写了这本详细阐述如何开发CWM解决方案的著作，对具有代表性的四个纵向模型的CWM解决方案进行了结构完整、条理清晰、细致全面的介绍，并列举了大量实例，图文并茂、语言生动，是读者掌握和使用CWM的良师益友。

本书是一本出色的有关如何开发支持CWM技术和元数据集成方案的开发指南，书中除了介绍CWM的基本原理和技术之外，还介绍了如何使用元数据交换模式简化对支持CWM的工具的开发、如何使用CWM建模技术开发数据仓库和业务分析领域的四个纵向模型以及如何选择与实现整个元数据集成体系结构。本书采用了从需求分析到详细设计，再到代码实现的介绍方法，充分展示了CWM解决方案的技术细节。因此，我极力将它推荐给需要在自己的软件产品中实现CWM解决方案，或者需要在自己的公共数据仓库、信息工厂和信息供应链的构建和改进中使用支持CWM的工具的软件从业人员；对于那些系统设计师、数据库设计师、数据仓库设计师、软件工程师、软件分析师和系统集成人员以及准备实施CWM解决方案的高级技术管理人员和规划人员来说，本书也大有帮助。

鉴于CWM本身的重要意义，译者在翻译这本著作时采取了特别认真和严谨的态度，力求既能准确地反映英文的原意，又能符合中文习惯。对于著作中的专业术语，译者经过反复探讨，使其最终能够以一致的面貌呈现给读者。在翻译中遇到的许多疑难问题，译者是通过进一步研究CWM以及有关的学术和技术问题加以解决的。因此，这本著作的翻译不仅是文字方面的工作，还包含着译者在技术上的研究。我们希望这些研究最终能够通过较准确的翻译，使读者受益。同时诚恳地希望广大读者对可能存在的疏漏和错误之处给予批评和指正。

译 者

2004年4月8日

于武汉大学软件工程国家重点实验室

前　言

对于实现不同软件产品和应用程序之间的无缝集成和互操作来说，元数据被广泛认为是一个最重要的因素。为了在软组件之间进行有效地互操作，这些组件必须能够容易地共享数据。共享数据需要对数据是如何构造的（它的组织和数据类型）和数据的含义（或语义）进行通用的定义。由于数据一般是由元数据定义的，因此，拥有元数据的一个通用定义是在数据层实现集成的先决条件。我们需要的是描述或表达元数据的一种通用语言，以及在软组件之间交换元数据的一致格式或接口。如果元数据的描述语言和交换机制能被标准化，并为软件开发商一致认可，那么就消除了实现真正互操作系统的主要障碍。

公共仓库元模型（Common Warehouse Metamodel, CWM）是对象管理组织（Object Management Group, OMG）制定的一个互操作标准。它为数据仓库和业务分析领域中使用的元数据定义了一种通用语言和交换机制。CWM不仅提供了极受欢迎的描述数据仓库与业务分析元数据的公共元模型，而且还提供了基于XML的交换工具。这个特殊领域的领导者和分析家很早就认识到，这样一个公共元模型和可扩展标记语言（XML）交换格式的标准化能够极大地提高任何复杂数据仓库或信息供应链的长期的投资回报（Return on Investment, ROI）。CWM使得制造商能够真正构建可以互操作的数据库、工具和应用。客户可以从市场上选择最好的产品而不用局限于某一个厂家的产品，不用担心他们的投资因为不同工具之间缺乏互操作能力而被浪费。在数据仓库和业务分析领域，CWM已经建立了自己作为元数据交换标准的地位，并且已经融入到许多制造商的产品套件之中。

从技术的角度看，CWM扩展了OMG建立的元建模体系结构，使其包含数据仓库和业务分析的领域概念。CWM支持用模型驱动的方法进行元数据交换，其中表示共享数据的形式模型是依据CWM元模型（实质上是实现数据仓库集成的一个对象技术方法）规范构造的。这些模型以XML文档的形式进行存储和交换。元数据的定义独立于任何与特定产品相关的考虑或格式。它能够作为一个信息商品永久地存储在产品之外，而且很容易被其他产品作为信息结构的一般定义而使用。

采用CWM定义的基本领域概念和关系而建立的数据仓库和业务分析工具，能够理解大量描述特殊元数据实例的模型。工具、产品和应用程序可以在元数据层集成，因为它们都用一种通用语言外化它们的元数据，而不需要具备彼此专有的信息结构和接口的知识。尽管CWM主要关注数据仓库和业务分析领域，但它的基本组件和方法很容易扩展到其他领域的主题。

本书的任务

本书的任务是为有如下需求的软件从业人员提供一个全面而实用的指南：需要在自己的软件产品中实现CWM解决方案，或者需要在自己的公司数据仓库、信息工厂和信息供应链的构建和改进中使用支持CWM的工具。

本书是开发支持CWM技术和元数据集成方案的开发指南，介绍了一种全新的方法。本书继承Ralph Kimball原创著作（*The Data Warehouse Toolkit*, Kimball, 1996）的精髓，通过以下四个具有高度代表性的纵向数据仓库模型探讨了如何实现CWM的一般问题，这四个模型均获益于基于标准的元数据集成方法：

- 数据仓库实现和装载
- 维建模
- 支持Web的数据仓库
- 元数据存储库

这四个纵向模型中的每一个模型都有一个完整的CWM元数据集成解决方案，读者可以通过这些方案的开发和实现进行学习。元数据存储库将其他三个模型结合在一起，形成了完整的集成元数据解决方案的体系结构。

本书的另一个重要方面在于，它为CWM元数据集成提供了一个完整的基于模式的处理方法。这一部分代表了能够提供真正可互操作的元数据体系结构的下一阶段，通过为CWM模型交换规定与领域相关的语义环境这一手段来增强CWM的功能、可表达性和灵活性。本书充分探讨了元数据交换模式的理论和实践，同时还将这些模式应用于创建每个纵向模型的CWM解决方案。

本书是《公共仓库元模型：数据仓库集成标准导论》的姐妹篇。《*Common Warehouse Metamodel: An Introduction*》（简称《导论》）是CWM的一个简单的初级读本，提供了对CWM标准的完整描述。它介绍了CWM的价值主张和经济原理，CWM的基本技术和体系结构，使用CWM建模基本的数据仓库和业务分析元数据，实现和扩展CWM，以及CWM在定义大规模元数据集成解决方案时所起的作用。本书通过提供从最初的需求分析到详细设计，再到代码实现的开发完整CWM解决方案的技术细节，极大地丰富了《导论》的内容。

资深的软件开发人员能够仅以本书为基础成功地实现一个CWM解决方案，而一般的读者应在阅读完《导论》后，才能对CWM基本原理和技术有一个全面了解。本书总结的许多CWM基本概念在《导论》中进行了详细地论述。因此，我们将《导论》强烈推荐给那些希望在根据本书进行实现之前，首先需要全面了解CWM领域的人作为一个起点。

本书的组织结构

本书为开发CWM解决方案提供了一种合理而直接的方法。下面列出的是本书的章节。已经熟悉了CWM概念的读者可以跳过前两章从第3章“使用CWM建模元数据”开始阅读。已经了解CWM在元数据解决方案建模所采取的方法的读者也可以考虑跳过第3章，从第4章“元数据交换模式”开始阅读。第3章介绍了构成本书其他章节基础的具体的新内容。

第一部分“概述”介绍了本书概念性内容。

- 第1章“CWM导论：基于模型的信息供应链的集成”通过描述基于标准的元数据集成如何实现对信息供应链的长期投资回报（ROI）的最大化，概述了开发基于CWM的元数据集成解决方案的经济理由。
- 第2章“CWM体系结构综述”概述了CWM的体系结构及其基本技术。总的来说，第1章和第2章基本上总结了《导论》中第1章到第4章的大部分详细内容。

- 第3章“使用CWM建模元数据”详细说明了如何将CWM作为一种建模语言来使用，用与制造商和产品无关的方式描述元数据，并在很大程度上形式化了《导论》中第5章开发建模示例中所使用到的技术。
- 第4章“元数据交换模式”介绍了在形式化元数据模型时使用基于模式的构建技术这一新内容。基于模式的方法增强了互操作性，并大大简化了支持CWM的工具的构造。这一主题在《导论》中的第8章提到过，但那里没有给出完整的处理方法。

第二部分“纵向模型介绍”开发了综合数据仓库和业务分析纵向模型，使用CWM建模和实现了这些模型的各个方面，包括从逻辑设计到物理设计，到底层实现资源建模的各个方面。

- 第5章“数据仓库管理模型”描述了如何使用CWM设计和实现一个通的用数据仓库纵向模型。这一章介绍了数据仓库模型的整个维的组织，开发了描述源到目标的映射和转换的元数据，作为整个数据仓库装载过程的一部分。该章还开发了描述调度、控制和跟踪数据仓库装载过程的过程模型。最后，开发了一些扩展核心数据仓库模型的例子，为处理与工具相关的元数据提供附加的特化的元数据结构。该章的重点是使用内嵌的CWM扩展机制和扩展CWM元模型本身。
- 第6章“维模型”介绍了利用CWM设计和实现一个大型维纵向模型。该章描述了CWM在模型中使用的所有的基本多维结构中建模时的应用，包括维度、级别、层次结构、立方体和度量指标。该章设计了一种物理关系数据库，作为关系星型模式中维模型的一个底层实现结构的模型，包括表、列、键、外键和连接关系等。该章描述了通过开发一个连接逻辑维模型和底层关系资源模型的CWM映射模型对部署映射进行建模。该章的重点在于从逻辑概念到它们底层的物理实现的说明和解析。随后，介绍了另一种逻辑－物理映射的开发，其中，同一个逻辑维模型被部署到另一个底层资源模型之上。这一部分的重点是如何在保持逻辑模型不变的情况下，修改物理模型以进行不同的优化。最后，通过核心维模型的扩展对此维模型进行了扩展，以提供附加的、特化的元数据结构来处理与特定工具相关的元数据。这一部分的重点是使用内嵌的CWM扩展机制和扩展CWM元模型自身。
- 第7章“支持Web的数据仓库模型”开发了附加的维概念和可操作概念，从而使得数据仓库能够在分布式的、高度合作的、基于网络计算的环境中使用。它说明了如何使用CWM为以下情况建模：将数据仓库用于Web服务器的后台，以及提供新的维结构和其他支持点击流分析的元数据结构。和第5章、第6章一样，该章也列举了一些示例，它们展示了扩展核心CWM模型以更充分地涵盖Web所需的附加领域语义。
- 第8章“CWM元仓库”以前面三个面向领域的场景为基础，通过示范如何使用CWM来定义整个建模和元数据管理环境，统一了这三个方案。通过这个特定案例，充分地展示了源于OMG标准建模体系结构和基础技术的CWM高级设施。该章特别为本书后续面向实现的章节提供了基础，同时，在相关网站中还提供了CWM实现软件。

第三部分“实现和部署”深入阐述了元数据存储的实现，其中元数据存储是作为纵向建模那些章节的一部分来开发的。这一部分还提供了完整CWM实现的一个具体示例。

- 第9章“集成体系结构”以《导论》第6章介绍的内容为基础，为详细设计的组合场景定义了一个具体的元数据管理策略。然后将这个策略精炼为技术和集成体系结构模型，这些模

型可以指导CWM实现的后续开发。

- 第10章“接口表示”描述了根据某种形式化映射标准，将CWM元类转换成以某种目标实现语言编写的接口的过程。在本案例中，映射标准采用Java元数据接口，一种OMG的MOF到Java编程语言的映射。使用本书提供的软件工具，读者可以根据书中介绍的步骤，生成表示CWM元对象的Java接口。
- 第11章“实现开发”概述了整个元数据集成体系结构的开发过程，其开发过程以第8章“元数据交换模式”中大量构建的元数据存储、第9章“集成体系结构”中介绍的各种集成体系结构模式和第10章“接口表示”中定义的CWM接口为中心。这一章的目的是告诉读者一个有效的CWM实现需要些什么。该章还涵盖了许多主题，包括适配器的构造、将CWM扩展到Web服务以及大量自动元数据集成服务的开发等。
- 第12章“总结”在最后总结了本书所有的重要结论，提出了对CWM标准和相关技术未来发展的预测和未来研究的方向。

本书适合的读者

《导论》为掌握了一定技术知识的读者提供了一个CWM概念性介绍，而这本书面向的是高级技术人员，主要包括负责开发或集成和部署那些支持CWM软件产品和工具的开发人员。本书中所提到的“开发人员”指的是具备丰富软件设计和实现背景的人，包括系统、数据库和数据仓库设计师、软件工程师、分析师、建模者和系统集成人员。本书对于那些需要足够详细的技术信息来有效地计划和预算CWM解决方案的高级技术管理人员和规划人员来说，也有巨大的价值。

当然，那些只希望了解少量CWM技术问题和只需对CWM有更多概念性理解的读者应该阅读《公共仓库元模型：数据仓库集成标准导论（2002）》一书。该书适用于那些只希望快速获取一系列CWM基本知识而不深入研究如何构建一个CWM解决方案的核心和关键问题的读者。另一方面，本书以《导论》中提到的许多关键概念为基础，我们强烈推荐，正在学习CWM的那些严谨的学生应该首先阅读《导论》，或至少将概述作为参考。

网站提供的内容

本书的合作网站是www.wiley.com/compbooks/poole。该网站提供了本书所有章节中涉及的模型、源代码、元数据交换模式目录、标准模式模板和其他所开发的内容，并提供了这些产品的下载、安装和使用说明。网站还详细说明了编译和执行CWM解决方案源代码所需要的第三方软件，同时还提供了系统需求、有效期以及各种制造商网站的相关链接。

该网站还提供了本书和所提供的代码的勘误表、软件的升级版本以及bug补丁，其中bug补丁是每个升级软件所发布的内容的一部分。

展望

很多人（甚至是具有丰富经验的建模师和设计师）都发现OMG的建模技术（如OMF、UML、XMI和CWM）很难理解。但是必须牢记，这些技术的目标是解决长期以来一直存在的难题。正如那些用于解决十分复杂的系统集成问题的工具一样，这些技术需要在提供健壮、有效和广泛

适用的解决方案的同时，做到尽可能简单。在必须有效地应用类似于CWM的技术的情况下，任何人都应该深信，为透彻理解这种复杂标准所花费的时间和努力将会使无缝集成系统的终端用户获得更好的实现和长期的效益。

虽然现在使用CWM及其相关技术有时会很难，但随着实现经验的积累，这些标准会变得越来越容易使用。目前，正在研究一些形式化技术来简化基于CWM解决方案的开发（如，元数据交换模式的开发），这些形式化技术的发展终将使CWM得到更为广泛地认可。当然，大量用于设计、实现、部署和管理模型驱动体系结构的自动化工具的出现，将会在很大程度上减缓当前的实现压力。为了实现这一目标需要大量坚持不懈的艰苦努力，但可以得到的收益也是相当可观的。在构建真正的互操作智能系统时，将会证明本书是很有价值的。现在，万事俱备，你可以开始学习了。

希望大家都能愉快地建模!

目　录

第三部分 实现和部署

第一部分 概 述

第1章 CWM导论：基于模型的信息供应链的集成

公共仓库元模型（Common Warehouse Metamodel, CWM）是一个开放的业界标准，用于在数据仓库及业务分析领域，为元数据定义公共元模型和基于XML（eXtensible Markup Language，可扩展标记语言）的交换格式。CWM是数据仓库和业务分析领域中的一种通用领域模型，拥有到XML的一系列映射标准。CWM不仅提供了极受欢迎的公共语言来描述元数据，而且还提供了基于XML的元数据交换工具。CWM已经在数据仓库和业务分析领域中建立起自己的地位，并成为众多下一代数据仓库和业务分析产品、工具的一部分。

本章讲述了支持CWM的数据仓库和分析产品、工具最终所能实现的经济效益。我们会从以下方面进行分析：当元数据不能真正集成时，数据仓库家族所面临的经济影响；为什么CWM方法是实现集成的最好尝试；以及CWM是如何集成各种不同厂家的软件产品和应用程序的。由于任何有用的技术都有其局限性，所以本章还界定了CWM的适用范围，并描述了CWM无意解决的数据仓库集成的有关方面。

1.1 信息供应链的集成

典型的数据仓库和业务分析环境通常都是根据信息供应链（Information Supply Chain, ISC）（Kimball, 1996）或信息经济（information economy）（Thomsen, 1997）来描述的。这些比喻反映了在该环境中信息流动的实际情形：它从源头（即原始数据的提供者）流出，通过一系列的精炼过程，最终产生对企业决策者具有很大战略价值的信息产品。图1-1展示了一个典型ISC。

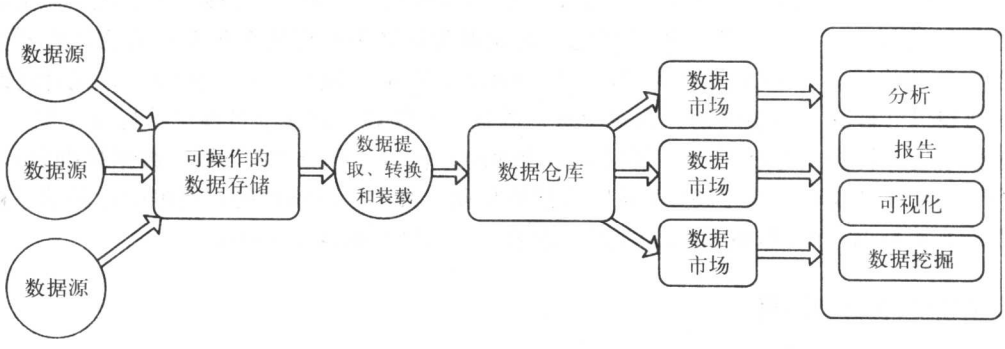


图1-1 信息供应链

1.1.1 信息供应链中的组件

在许多方面，ISC都与耐用品的制造和销售过程非常相似。在一个耐用品的供应链上，原材料（木材、钢铁矿石、石油等）通过各种制造工序，最后加工成为有用的消费产品。这些产品被储存在仓库中，直到有了客户订单之后，将产品运出仓库，直接交给客户或者交给一些中间商。在ISC中，与制造-销售过程相似，在供应链的起始端获得的*原始数据*，经过一系列提炼、变换，直到最后被交给最终用户（企业决策者）。

ISC中的第一个加工步骤通常就是协调各种事物数据，使它们用一种更统一的方式表达。这一步骤可以有很多不同的名字，包括*数据提取、转换和装载*（data extraction，transformation，and loading，ETL）；*数据规范化*（data normalization）；*构造数据仓库*（data warehouse building）；*数据整理和协调*（data cleansing and reconciliation）等等。不管使用什么术语，这一步都包括从不同的事物系统中获得数据，将这些数据转换成某个通用的格式，然后将这些被转换的数据存储在一个专用的数据库中。这个专用数据库，使得这些数据能够作为战略信息更好的为决策者服务，而不是仅仅作为一些个别业务事件的详细记录。有趣的是，这个提供战略信息的专用数据库（提炼和组装信息产品）通常称为*数据仓库*（data warehouse），这在某种程度上和我们前面提到的制造-销售方式是一致的。

也许数据仓库最显著的一个特点就是，它将被提炼的数据以有利于进一步分析的方式组织起来，而这对于一个业务的持久发展是至关重要的。数据仓库在本质上往往是*有维的*（dimensional）；也就是说，它们根据业务的不同*维*（dimension）以统一的方式来组织数据，这些维可能包括：账户、产品、地区、销售单位、商店等等。这些业务的维充当了定义数据的查找关键字。数据仓库往往也是按照时间来组织数据的。例如，一个业务分析员可能会用数据仓库对一个特定产品线在不同财政时期不同地区的销售情况进行比较。数据仓库中的数据可能是从各种完全不同的事务系统中获得的，但是数据仓库的有维组织结构能够将这些事务数据的差异抽象出来。数据仓库的最终目的，是要通过将原始的业务数据转换为*策略性的业务数据*来促进业务分析。

先进的分析和报告工具可以直接脱离数据仓库工作或是隶属于部门的数据市场。这些工具显著增加了那些从数据仓库中获得的信息的价值。尽管数据仓库已经建立了对信息的有维观点，但分析和报告工具还是能够提供一些特有的功能，如对按维组织的数据的处理、对特定操作的处理或可视化的处理等。例如，一个先进的金融分析和建模工具包能对基本维信息进行复杂的统计分析。同时它还可能利用OLAP（Online Analytical Processing，在线分析处理）服务器或数据挖掘工具的能力。先进的报告和可视化工具能够为最终用户提供多种有效的方法查看分析结果，以增加其价值。这些方法包括不同类型的表格、图形、颜色代码、报警框，或是终端用户可以直接操纵（旋转、中轴转动、调整形状、调整大小等等）的多维可视化结构或图象。

数据市场、先进的分析工具（包括基于软件的分析包、OLAP服务器、数据挖掘包）、报告工具和可视化工具集中代表了ISC中最后的精炼步骤。其中，战略性的、有维的业务信息被有效地转换为*业务知识*（business knowledge）、*洞察*（insight）或*远见*（vision）。

1.1.2 集成ISC的经济问题

所有ISC最重要的一个特点就是，它有一个定义良好（并且具有高度目的性）的数据流，从

起始源，经过一系列转换，最终到达某个目的地（通常为分析和报告工具的终端用户）。ISC中的每一个精炼步骤都是通过使用一个或多个与该步骤特定目标相关的软件产品来实现的（见图1-1）。为了有效地实现一个ISC，这套工具必须能够完全参与到数据交换的过程中。每个工具都必须对要处理的数据的本质有所了解：它来自何处；它的不同域意味着什么；需要对这个数据进行什么转换；转换的结果存储在哪里；什么目标过程需要这些结果等等。

元数据是理解数据含义和如何使用数据的关键。所有实现ISC各阶段的不同的软件工具和产品，都要依赖元数据来描述它们需要处理和转换的数据。要使一个特定的产品能够正确处理它的数据，该产品必须对数据的结构和语义有一个全面的理解。这个理解通常是由元数据提供的。对构成一个ISC的产品和工具来说，元数据是其内部处理逻辑的一个关键输入。这个产品的内部逻辑在决定如何处理数据时，将元数据作为基础。为了使一系列给定的软件产品能够有效地参与ISC，并在数据层上进行互操作，就必须对描述那个数据的元数据有一个共同的理解。也就是说，构成ISC的每一个软件产品和工具能够在数据层进行有效集成的前提是，它在元数据层就必须被有效集成。

然而，元数据层的集成是相当困难的，因为绝大多数的业务产品存储元数据所使用的格式千差万别。通过一个特定产品提供的某些接口，可以访问它的元数据。但元数据易于访问并不表示它可以被完全理解。元数据的格式和语义，以及访问它的接口，在产品之间很少是统一的，而且它们通常都更偏重于每个产品的有效操作，而不是与其他产品的集成。

这种差异很大程度上是产品开发历史的产物。通常，软件产品和工具的设计和安装都是相对孤立地。例如，某个厂商生产的OLAP服务器，是根据内部的元数据定义而设计的，这个元数据定义对于该服务器也许是十分理想的，但是对于那些需要与该服务器协同工作的各种报告工具就不一定是这样了。还有一种情况，就是厂商可能因为怕失去市场份额，而不愿意使他们的元数据标准化。直到当公司合并或兼并其他公司时，他们才发现，不管这些产品线多么相似，它们也无法合并，必须支持多个不同的产品线和大量不同的元数据结构。

实际上，拥有不同元数据的工具之间是通过建立复杂的元数据桥（meta data bridge）来实现集成的。元数据桥是一种能够将一个产品的元数据转换成另一个产品所要求的格式的软件。这样的桥需要具备与其集成的每个产品的元数据结构和接口的详细知识。图1-2显示了几个ISC组件通过若干元数据桥相互连接的情况。空心箭头表示整个数据流，每个带阴影的箭头则表示一种不同的元数据桥和与它关联的元数据流。需要注意的是，每个桥只适用于它所连接的这对被集成的工具。而且，这些桥往往是双向的（也就是说，它能够理解双向的元数据映射）。例如，将数据从一个事物系统移动到一个数据仓库，就需要一个能够将事物元数据映射到多维元数据的桥（如，将事物表定义映射到关系数据库模式）。如果要向数据仓库用户提供回滚到事物的能力，则需要一个可以将多维元数据映射到事物元数据的桥。每个映射不一定是另一个映射的逆过程。通常，当从一种格式的元数据转换到另一种格式的元数据时，都会丢失一定数量的信息，而桥多多少少也要对此负一定的责任。

桥的构造，无论是由产品厂商、第三方顾问、或是由ISC和数据仓库的实现者执行，都是一项非常艰巨和昂贵的工程。桥必须具备所有者的元数据模型和接口的详细知识。关于不同模型间如何相互映射的知识也要融入其中。此外，构成一个特定桥的处理逻辑不一定能够在其他桥