/THEORY/IN/PRACTICE

视图更新与关系数据库理论

解决视图更新问题

View Updating & Relational Theory

[美] C. J. Date 著
田远帆 译

O'REILLY®
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

视图更新与关系数据库理论

[美] C.J.Date 著

田远帆 译

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

视图更新与关系数据库理论 / (美) 达特
(C. J. Date) 著 ; 田远帆 译. -- 北京 : 人民邮电出
版社, 2017.2
ISBN 978-7-115-43546-0

Ⅰ. ①视… Ⅱ. ①达… ②田… Ⅲ. ①关系数据库系
统 Ⅳ. ①TP311.138

中国版本图书馆CIP数据核字(2016)第272589号

版权声明

◆ 著 [美] C.J. Date
译 田远帆
责任编辑 陈冀康
执行编辑 胡俊英
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：800×1000 1/16
印张：15.75
字数：292 千字 2017 年 2 月第 1 版
印数：1 – 2 000 册 2017 年 2 月北京第 1 次印刷

著作权合同登记号 图字：01-2013-8796 号

定价：59.00 元

读者服务热线：(010) 81055410 印装质量热线：(010) 81055316
反盗版热线：(010) 81055315

内容提要

本书是数据库专家 C.J. Date 的又一部力作，详细介绍了数据库架构的设计和实现，同时讨论了视图更新及关系数据库理论。

全书内容分为 15 章，用示例和理论相结合的方式，介绍了关系数据库的相关理论，还对视图作了深入探究。书中不仅介绍了一些基础的数据库操作，同时重点结合示例给出了视图相关的操作和讲解。通过阅读本书，读者能够在实际工作和学习中，改进自己的数据库设计，更好地实现数据库相关任务。

本书是经典的数据库参考指南，非常适合数据库架构师、数据库设计人员以及相关领域的研究人员参考阅读。

献辞

内涵的外延
Edgar F. Codd 发明了一种概念
就是我们知道的视图
现在的视图和基础关系变量
可相互转换
让我们高唱一曲
关于视图的布鲁斯

—Anon.:《Where Bugs Go》

“奥蒙德的公爵昨天审视了他的部队的全景，命令把所有的枣红色马和灰色马都换成黑马。”

——目前所知的最早关于视图更新的案例（1693），引自《Oxford English Dictionary》中的《A Brief Historical Relation of State Affairs 1678–1714》，作者 Narcissus Luttrell (1857)

浅尝辄止是如此危险；
诗泉之水若不深饮，弗如不尝：
浅浅一酌让人头脑发热，
尽兴狂饮却让人重新清醒。

——Alexander Pope《An Essay on Criticism》（1711）

—— ◆◆◆◆◆ ——

献给我最爱的妻子 Lindy、女儿 Sarah 和 Jennie

作者简介

C.J.Date 是一位专攻关系型数据库技术的独立作家、演说家、研究员和顾问。他最著名的著作《数据库系统导论》（第 8 版，Addison-Wesley，2004）销量已经突破 850000 册，并被全球数百所高校作为教材使用。他同时还撰写过许多其他数据库管理相关书籍，包括：

- Addison-Wesley 出版：《Databases, Types, and the Relational Model: The Third Manifesto》（第 3 版，与 Hugh Darwen 合著，2006）
- Trafford 出版：《Logic and Databases: The Roots of Relational Theory》（2007）
- Apress 出版：《The Relational Database Dictionary, Extended Edition》（2008）
- Trafford 出版：《Database Explorations: Essays on The Third Manifesto and Related Topics》，（与 Hugh Darwen 合著，2010）
- Ventus 出版：《Go Faster! The TransRelational™ Approach to DBMS Implementation》（2002，2011）
- O'Reilly 出版：《SQL and Relational Theory: How to Write Accurate SQL Code》（第 2 版，2012）
- O'Reilly 出版：《Database Design and Relational Theory: Normal Forms and All That Jazz》（2012）

Date 先生于 2004 年正式进入计算机行业名人堂。他以能够用简明易懂的方式解释复杂的技术问题而著称，造诣之高，无出其右者。

前言

本书是这个系列的第 3 本书，它的两位“前辈”是：

- 《SQL and Relational Theory: How to Write Accurate SQL Code》（第 2 版）
- 《Database Design and Relational Theory: Normal Forms and All That Jazz》

以上两本书于 2012 年由 O'Reilly 出版发行。第 1 本书的目标读者是所有种类数据库的从业人员，书中解释了关系理论的基本原理，以及以此为基础如何将 SQL 当作一个关系型数据库使用（在那本书中我使用的一条准则是“关系化地使用 SQL”）。第 2 本书则针对性强一些，它瞄准的读者群是那些对数据库设计感兴趣的行业专家，书中解释了关系型数据库设计的一些理论，并展示了为什么这些理论如此重要。而第 3 本书，也就是本书，则针对性更强。本书专注于对一个非常关键的问题的研究，而这个问题则涉及关系型数据库应该如何运行（与目前大部分商业 SQL 数据库系统的运行表现恰恰相反）这一核心内容。这个问题就是“更新理论”。这个理论正如本书的题目显示的那样，适用于视图的更新——不论是在一般情况下，还是特定情况下。同时，本理论也适用于“基础数据”的更新。

注意：尽管我的理论包含上面的最后一句话，但我还是决定在本书的题目中更加强调对视图的更新，因为据我观察，一般数据库从业者相信他们自己能够理解对于基础数据为对象的更新是如何运作的，而一旦对象变成视图，他们的典型反应就是极度怀疑无论使用什么办法，对视图的更新能否真的实现。其实我一直很奇怪，关于视图更新的讨论居然曾经是并且依旧是一个有争议性的话题，当然这也是最初让我决定撰写这本书的一个重要原因。

顺带说一句，关于前两本书，我要对本书中大量引用它们（特别是第 1 本）而表示歉意。目前，本书中大部分引用的其他出版物都会给出全名，例如下面的例子。

David McGoveran：“Accessing and Updating Views and Relations in a Relational Database” 美国，专利号 7263512（2007 年 8 月 28 日）

不过从现在开始，当我提到之前出版的著作时会使用简称，即《SQL and Relational Theory》和《Database Design and Relational Theory》。

作者注：我说过我会用全名引用其他的著作，不过其实也没有多少能引用的。虽然大量关于视图更新的论文、著作或者其他文章在最近 30 年内层出不穷，但它们中大多数，包括 David McGoveran 发表的某些例外的特定著作，所主张的论调和本书所体现的论点大相径庭（请参看前言后面部分关于这一点的讨论）。因此大多数时候，我都觉得不适合引用这些著作，除了偶尔因需要提及一下。如果你有兴趣详细地了解一下这些“其他的论调”，那么可以在《An Introduction to Database Systems》（第 8 版，Addison-Wesley，2004）的第 10 章中找到一个关于这些书籍名称的简短列表。

在这里我需要强调的是本书假定你们完全了解《SQL and Relational Theory》所讨论的内容。举例来说，我认为你们理应知道什么是关系，什么是属性，什么是数组，所以，我不会因为未对这些基础名词做提前解释而道歉。本书的目标读者是数据库相关专业人士，而专业人士应该对我之前的书所包含的这些基础知识很熟悉才对。为了使本书内容更加独立，我会在第 2 章“技术背景”中对前面的书的相关部分做一个简短的回顾。同时我也会在第 3 章“视图概念：近距离观察”中给出一个关于什么是视图以及它们应该如何运作的详细综述。

本书的目标读者

我的目标读者群是数据库相关专业人士，进一步说也可以是所有对关系模型、关系型技术或者关系型系统感兴趣的人。如前所述，对《SQL and Relational Theory》熟悉的话会对阅读本书有很大帮助，不过我相信本书无论对关系理论整体而言，还是对视图更新具体来讲，都会有新鲜的见解。而且，我认为值得指出的是，我很有可能可以利用这里面的一些想法在没有数据库管理系统（DBMS）[1]支持的情况下来指导一个“自助式”的实施（针对视图更新）。但在这里其实我真心希望数据库管理系统工程师们能够看到这本书，并由此能在他们自己的产品中提供对视图更新的支持。注意：我同时也想提一下，我会举行一场基于书中内容的现场研讨会。想了解更多内容，请访问下面网址：www.justsql.co.uk/chris_date/chris_date.htm。

本书的结构

如之前所说，我假设你知道什么是关系、属性、数组。更进一步来讲，我假设你也至少大体知道什么是视图。视图最初被讨论（当时并不用“视图”这个名字）是在 Codd 的第一篇关于关系模型的论文中：

[1] 数据库管理系统（DBMS）= database management system。很显然，数据库管理系统和数据库是有区别的！然而不幸的是，业界通常都把一些商业产品，如甲骨文，或者安装在特定电脑上的此类产品的特定版本称为“数据库”。在本书中我不会跟随这个潮流。问题在于，如果你把数据库管理系统称作“数据库”的话，那你要怎么称呼真正的数据库呢？

E.F.Codd: "Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks," IBM 研究报告 RJ599（1969 年 8 月 19 日）

现在，正如 Codd 自己在这篇论文中所预见的，支持视图最主要的理由，是原则上可以通过它们来达成逻辑数据独立这个重要目标（逻辑数据独立指的是能够改变一个数据库的逻辑，而不需要相应改变用户的使用习惯的能力，由此可以保护针对既有用户培训、既有应用以及其他部分的投入。在第 3 章我们会有更深入的讨论）。换句话说，视图存在的主要理由，严格来讲就是为了实现逻辑数据独立这个目标。但如果我们想在实际中实现它，而不仅仅是停留在原则上，那么视图必须是可更新的。

所以针对视图的更新是一个很重要的问题。由此重要性导致的结果是，无论是对商业领域，还是研究领域而言，视图已经成为大家关注的焦点有一段时间了（至少有 35 年左右），而且一些不同的方案已经被提出，有些甚至被实施过。但可惜的是，所有这些方案都没能对视图更新给出一个令人满意的解决办法（这里我要强调，不仅仅是我，其他作者也这么认为）。我们举例来说，在当今主流的 SQL 产品中，视图更新机制具有以下两个典型的缺陷。

- 不完整，也就是说，它们没能使理论上应该能够更新的视图全都实现更新。
- 不正确，也就是说，即使对那些能够实现更新的视图，它们的实施方式也不对路，至少在某些情况下是这样的。

（再次说明，第 3 章中有关于这些问题的深度讨论。）至于那些研究文献，我觉得它们通常都会忽视一些重要因素，这些因素对于一个系统化的、综合性的以及正确的解决方案至关重要。与前面的例子不同，我相信在本书中将要详细讲解的，恰恰就是一个“系统化的、综合性的以及正确的” 解决方案。同时我也相信（在这里我必须说明，我本身并不是一名工程师），这个解决方案可以被整合进关系型数据库管理系统，并给系统架构带来适度的概念化扩展。

作者注：请注意我很谨慎地提到“关系型数据库管理系统”。在后面你将会看到，我提出的这个解决方案非常依赖对完整性约束的声明（当然，这需要数据库管理系统的强制执行能力来做保障）。从我的角度来说，我认为这些能力是构成一个真正的关系型系统的必要条件。不过相信大家都知道，当今绝大部分的 SQL 产品在这个环节都有缺陷。现在我正式介绍一下本书的大体结构。

- 第 1 章给出了一个抛砖引玉的例子，表明了在大家熟知的简单条件下（其实就是 SQL 的条件下）如何看待视图更新，在后面的章节中我们还会有更具体的

说明。具体来说，它证明了“更新就是更新”，无论被更新的是视图还是基础数据。这就是为什么我之前说这本书所涉及的问题应该被称为“更新理论”（一种既适用于视图，同时又适用于基础数据的理论）的原因。

- 接下来如前所述，第 2 章对关系理论的相关内容做了简要的回顾。其中特别强调了数据库的本质就是“一个真正的变量”，而它这个特质也恰恰让它成为对所有更新操作都适合的目标对象。

- 第 3 章对视图概念和相关内容做了更具体的描述。当然我前面说过我假定你了解视图的基本概念，但这一章包含了你可能不太熟悉的内容，而这些内容对于正确理解后面的内容至关重要。

- 第 4～13 章会针对各种各样你所熟悉（有的内容可能不那么熟悉）的关系操作逐个进行讨论——约束、投影、连接等。特别是在第 4 章，对于约束视图，我还介绍了很多额外的基础内容（其实，这几章也可以看作是第 3 章的一个继续）。第 4 章同时也给出了后面 9 章如何展开的计划。

- 第 14 章研究了合并操作（例如，更新一个由两个约束组成的连接，或者更新一个由两个连接组成的合并都需要什么）的问题，这个问题引发了一些相当有趣亦或出人意料的情况。

- 在整本书的讨论中，大家可能会对其中一些方案所提到的问题或内容不明确或产生混淆，最后在第 15 章我将为大家对这些不明确和混淆做出解释说明。

- 书后面还有两个附录。附录 A 的内容是关于所有重要的“关系赋值”操作的细节。附录 B 则收录了在本书主体中所使用或提到的大量关系操作的详细定义。

注意：目前给大家描述的大纲其实已经足以说明本书是按照循序渐进的方式来撰写的，我本人也强烈建议大家按照写作顺序来阅读本书。

技术性注释

还有几点我要在这里提前说明。首先请注意，在本书中我按照惯例使用一般术语，用小写字母表示的“更新”是代表包含“INSERT”“DELETE”以及“UPDATE”等所有操作语句的总称（正如我刚刚所提到的“所有重要的关系赋值操作”，详见第 2 章）。如果想表达“UPDATE”这个操作语句，我会全部使用大写字母表示，至于“INSERT”和“DELETE”操作就不会引起混淆。不过一味地使用大写会显得有些烦人，尤其是当它们作为修饰语使用的时候，例如“INSERT 规则”（“插入规则”？）。所以，我决定在本书中两种格式一起使用，我会根据前后语境自我发

挥（我也不会装作在这方面前后有多么一致）。

第二点，请注意我使用“SQL”来表示 SQL 语言的标准版，而不是什么专有的术语（另有明确说明的除外）。特别是按照标准读音“SQL”应该念为“ess cue ell”，而不是“sequel”（因为现在很多地方都按后者来发音），所以我对“一个 SQL 表”的书写会按照标准读音的写法。注意：SQL 标准版随着时间推移出现了数个版本，在本书写作的时候最新版本是 SQL：2011。以下是完整的版本信息。

国际标准化组织（ISO）：Database Language SQL, Document ISO/IEC 9075:2011 (2011)

第三点也是最后一点，我需要对我所谓的“用户”做一下说明，特别是我需要解释一下经常用到的一些短语，如“用户看到的”或者“用户对数据库的感觉”。总体来说，你可以把“用户”这个词理解为一个交互用户[1]、一个应用程序开发人员或者两者都是，请根据上下文理解。至于“用户看到的”和类似的短语中的用户，我指的是那些并未与整个数据库交互而更多的是与一些子集交互的用户，根据“子模式”定义。另外，根据视图的机制，子集可以并且经常参与到一些逻辑重建中。实际上，我们可以（至少我会这么做）简单而不失全面地假设，子集是由视图组成的，即使一些由基本数据转换而来的视图实际上跟导出它们的基本数据等价。当然，对于这个子集的用户来说，视图的集合就是数据库本身！换句话说，“数据库”在某种意义上来说是个相对项。所以，至少是为了本书使用，我们可以稍微有点随意但有效地把数据库定义为一个给定的数据，例如给定的基本数据的集合，或者是这个集合下的一些具体的，很可能是经过重构的子集。注意：当我在这里说到“随意”的时候，我考虑到的是这个数据库不仅仅只有数据，如相应的完整性约束也需要被考虑进去，我们在第 2 章和第 3 章将会见到。

[1] 这些用户依然是那些懂一点数据库问题的人，而不是那些纯的“最终用户”，他们可能对这些问题完全没有概念。

致谢

我想再次感谢我的夫人 Lindy 对我在撰写本书期间以及在撰写本系列前作时的大力支持。我还要感谢我的挚友和同事 Hugh Darwen、David Livingstone 以及 David McGoveran，他们对本书的早期草稿给予了大量细致的、综合的审阅。他们和他们所做的大量工作对本书都起到了重要作用，但我特别要感谢 David McGoveran。首先，感谢他最开始为本书的主题所提出的一些建议和想法，视图更新在本书中的描述大都基于此发展而来；其次，感谢他就这个主题在过去 20 年间跟我所做的沟通和协调；最后，感谢他在这个领域所进行的深远的理论工作。而且 David 所做的工作还大大超出了审阅本身，他不仅仅对词句给出建议，实际上他还针对本书主题的很多方面汇编并提供了许多短篇的论文或笔记。这些笔记在我重新修改文章的时候给了我非常大的帮助，我相信正是这些笔记让我的文章大有起色。当然，我并没有把他给的所有建议都采纳进来，我相信没有哪个作者完全按照审阅者的建议来行文！但是我已经尝试着尽可能采纳那些对我来说最重要和最真实的建议了。当然不用说，任何遗留的错误都是我自己的责任。

C.J.Date

于加利福尼亚，希尔兹堡

2013 年

序

在关系型数据库及其实际应用领域中，一直有两个特别棘手和富有争议的问题，而业界并没有得出一个大家都能满意的解决办法，这两个问题是：信息丢失问题和视图更新问题。对第 1 个问题，Chris Date 在过去 30 年左右的时间里已经撰写了大量著作，现在他开始着手应对第 2 个问题。

当然他在之前并非没有涉及视图更新这个问题。在他著名的著作，被作为教科书广泛使用的《An Introduction to Database Systems》中就有对该问题的讨论。在 1975 年第 1 版发行的时候，书中就有少量篇幅涉及这个问题；到第 8 版时（2004 年），书中已经有多达 16 页左右的篇幅来进行讨论。他第一次用整个章节讨论这个问题是在 1986 年开始撰写系列丛书《Relational Database Writings》的时候。到了 1995 年，在该系列第 4 本书中，他和 David McGoveran 带给了我们两个章节的内容来展示对这个问题思考的变迁，这些内容更多地源自 McGoveran 的研究。这个思考后来在《Databases, Types, and the Relational Model: The Third Manifesto》（2007）的附录中进化，在《Database Explorations》（2010）的章节中成长，最终到了今天的地位。

最初由 E. F. Codd 在 1969 年提出的基础理念，至今一直都没有变化。想象现在给定一个数据库，根据定义是由（a）可变关系或者“关系变量”[1]组成的集合，同时由（b）一套完整性约束管理着这些关系变量的可用值。这些给定的关系变量都是“基础”变量。总体来说，被选出的设计是在许多个设计中可以被选出来代表完全相同信息的那一个。从被选出的设计中我们可以根据基础关系变量的关系表达式，通过定义虚拟关系变量，从而得到一个备选方案。由于各种各样的原因，这样一个备选设计，实际上是一个备选的数据库视图，可能相对基础设计来说更适合某些特定用户。更重要的是，这个备选设计可能实际上并不包含底层的或者“真正的”数据库。有些用户对这些其实也不感兴趣，或者根本没有权限看到。除此之外，如果必须针对基础设计做出一些变化，代表原始设计的虚拟关系变量可以在新设计上被重新定义，这样一来已经存在的用户视图就无需改变，那些可能发生的令人讨厌的突变就可以避免。这就是著名的终极目标“逻辑数据独立”

[1] SQL 喜欢称这些关系变量为“表”。关于更多对于关系变量的名词解释和相关内容，请参看第 2 章。

背后的基础理念。

如果用户把虚拟变量视为他们的数据库，当实施与虚拟关系变量冲突的数据库更新时，棘手的问题就出现了。数据库管理系统到底是如何判定那些对真正的数据库实施真正的更新会对虚拟关系变量产生特定的改变的？如果这里有数种方式可以实现需要的效果，那么到底该选用哪种方式呢？举个简单的例子，假设一个用户使用常见的“供应商与零部件”数据库（第 1 章有详细介绍），他看到一个虚拟关系变量或者视图命名为“PS”，其中只显示位于巴黎的供应商。毋庸置疑，PS 的定义表达式是“S WHERE CITY = 'Paris'”。现在，我们假设这个用户让数据库管理系统从视图 PS 中删除数组 S2。那么数据库管理系统是否应当这样理解用户的意图：供应商 S2 不存在了，需要从基础关系变量 S 中删除相应的底层数组？还是说应当因为用户表述不清而拒绝客户的删除指令，因为将供应商 S2 的 CITY 值换成巴黎以外的地方也会达成相同的效果？或者在另一种情况下，假设用户知道供应商 S2 已经搬迁到伦敦，因此尝试在视图 PS 中对 S2 进行“更新数组”的操作来达到从视图 PS 中删除数组 S2 的效果，那么数据库管理系统是否应当接受这个更新操作呢？现在我们进一步假设视图 PS 没有 STATUS（状态）属性。那么数据库管理系统应当如何回应该用户对视图尝试插入数组的操作，而用户要求这些数组必须要删除 STATUS 值？

针对以上和更多的此类问题，Date 都试图在他详细的、全面的、细心的、有条理的分析中为我们做出解答。他计划在最开始的 3 个章节中对这些问题发起进攻。他清楚地定义了“2 个数据库被设计为在表达相同信息的时候是等价的”是什么意思，然后在接下来的 10 个章节，他将对他所使用的方法进行详细讲述。这个方法需要轮流检验每一个关系代数中的操作。例如，那个“只包含巴黎供应商”的视图 PS 被他称为约束视图，比如，定义一个虚拟关系变量只使用约束操作。同样地，现在定义一个不包含 PS 中 STATUS 属性的视图使用投影。既然这个视图是一个约束的投影，我们可以来推断它的更新效果。首先根据 Date 的更新规则通过投影来决定对底层约束的效果，然后根据更新规则通过约束来决定对底层基础关系变量 S 的效果。

对由一些关系操作定义的视图应用这些规则的时候会产生一个非常有趣，甚至可以说是很有争议性的问题，这个问题 Date 在第 14 章会做阐述，那就是如果两个表达式在语法上不同，但逻辑上等价。例如，数学表达式 $x(y+z)$ 和 $xy+xz$ 在语法上不同，但逻辑上是等价的。那么定义于它们之上的视图在执行更新操作的时候运行行为（如过程、动作或效果等）是否一定要完全相同？

现在，Date 的一些观点在我前面提到的 2007 年和 2010 年的出版物上出现的时候，

被证明是具有争议性的。例如，当我们向 R1 和 R2 的合并视图插入 1 个数组时，这个数组是否应该同时出现在 R1 和 R2 之内？如果我们从 R1 和 R2 的交集视图中删除一个数组时，是否会导致这个数组从 R1 和 R2 中同时消失呢？作为针对那些具体观点表达不同意见的一员，在这里我要强调，在我和 Date 长时间的合作过程中，我们之间唯一非常严重的技术分歧已经出现。这些争议点在本书中也有展现，并且 Date 已经针对这些问题加强了基本原理的解释，不过这可能仍然无法让所有人都信服他的观点。而我发现，在他的最后一章“歧义问题再回顾”中出现了希望之光，就好像是隧道尽头的那一盏明灯。在其中他描述了一个想法的大概轮廓，由 David McGoveran 提出的一种与以往我们更新关系型数据库所采用的方式完全不同的方式，它可以很有效地替代，或者至少是拓展，我们所熟知的“插入”“删除”和“更新”这些早在关系型时代之前就与我们紧密相连的操作。而这种不同寻常的方式所带来的好处是，我之前提到的会产生争议的问题根本不会出现了。

Date 告诉我，他并没有期待，甚至没有想过让本书成为他关于视图更新故事的结尾，但他希望本书能够给现存的争论提供一个坚实的理论基础，以使得整件事情可以向前推进。我想这正是他在本书中所提供的，我在这里也希望本书对你有相同的效果。

Hugh Darwen

于英格兰 Shrewley 区

2013 年

目录

第 1 章　抛砖引玉 …… 1
1.1　可交换性原则 …… 3
1.2　仅限基表：约束 …… 5
1.3　仅限基表：补偿性操作 …… 6
1.4　视图：约束和补偿性操作 …… 7
1.5　规则至上 …… 8
1.6　小结 …… 9
第 2 章　技术背景 …… 11
2.1　关系和关系变量 …… 12
2.2　关系赋值 …… 15
2.3　一致性约束 …… 19
2.4　关系变量谓词 …… 21
2.5　MATCHING，NOT MATCHING 以及 EXTEND …… 25
2.6　数据库与数据库变量 …… 27
第 3 章　视图概念：近距离观察 …… 31
3.1　视图是伪变量 …… 33
3.2　数据独立性 …… 34
3.3　如何预防 …… 37
3.4　约束和谓词 …… 40
3.5　信息等价 …… 44
3.6　小结 …… 47
第 4 章　限制视图 …… 53
4.1　“抛砖引玉”再回顾 …… 53
4.2　更多关于补偿性操作的内容 …… 57
4.3　关于触发器 …… 61
4.4　关于显式更新操作 …… 63
4.5　供应商与设备供应 …… 65
4.6　再谈抛砖引玉 …… 68

4.7 概括总结……70
4.8 最后一点……71
4.9 重叠限制……73
4.10 小结……75
第5章 投影视图……77
5.1 示例1：一个无损分解……77
5.2 示例1续：投影关系变量……83
5.3 示例1续：视图……85
5.4 示例2：另一个无损分解……85
5.4.1 投影关系变量……90
5.4.2 信息隐藏……91
5.4.3 视图……91
5.5 示例3：一个有损分解……92
5.5.1 投影关系变量……94
5.5.2 信息隐藏……95
5.5.3 视图……95
5.6 小结……96
第6章 连接视图I：一对一连接……99
6.1 示例1：信息等价……100
6.2 示例2：信息隐藏……102
6.2.1 编译指示……107
6.2.2 对称……108
6.3 小结……111
第7章 连接视图II：多对多连接……115
7.1 示例1：信息等价……115
7.1.1 补偿性操作……118
7.1.2 视图更新……121
7.2 再谈投影视图……123
7.3 示例2：信息隐藏……124
7.4 小结……126
第8章 连接视图III：一对多连接……127
8.1 示例1：信息等价……127
8.2 示例2：信息隐藏……131
8.3 小结……133