HZ BOOKS

Special Edition

Using Oracle 8/8i

特版精品系列

(美) William G. Page, Jr. 等著

王磊 蒋蕊 王焱 等译

附 CD-ROM 盘

Oracle 8/8i

开发使用手册

机械工业出版社
China Machine Press

que

特版精品系列

Oracle 8/8i 开发使用手册

（美）William G. Page, Jr. 等著

王磊 蒋蕊 王焱 等译

机械工业出版社
China Machine Press

本书着重于介绍Oracle 8/8i的新特性、性能调整、数据库管理、Oracle接口和应用工具、Oracle网络、Oracle应用服务器等主题。本书由十余位Oracle资深专家所著，他们在书中提供了大量实践经验，并配合示例进行讲解。

本书版权登记号：图字：01-1999-2603

图书在版编目(CIP)数据

Oracle 8/8i 开发使用手册 /（美）佩吉 (Page, W. G.) 等著；王磊等译. -北京：机械工业出版社，2000.3

(特版精品系列)

书名原文：Special Edition Using Oracle 8/8i

ISBN 7-111-07821-7

Ⅰ.O... Ⅱ.① 佩... ② 王... Ⅲ.关系数据库-数据库管理系统，Oracle 8/8i-手册 Ⅳ.TP311.132.3-62

中国版本图书馆CIP数据核字（2000）第10810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：陈剑颐 赵红燕

北京昌平第二印刷厂印刷 · 新华书店北京发行所发行

2000年3月第1版第1次印刷

787mm × 1092mm 1/16 · 47.25印张

印数：0 001-6 000册

定价：88.00元（附光盘）

译 者 序

众所周知，信息时代的一个重要技术支柱是数据库技术日新月异的发展，而Oracle数据库又是众多数据库中的佼佼者，在全球商务、科学和军事领域都有广泛的应用。我在Oracle数据库上已经有8年以上的工作经验，而且持续地跟踪着Oracle数据库的发展，深知对于一个Oracle数据库系统管理员、开发者或用户来说，什么是他们所最需要的。本书全面细致地介绍了Oracle 8i数据库的一切相关特性，本书的第一、二版已经成功地成为一些大学教授Oracle数据库知识的教科书。但我个人认为本书的作用远远不止于此。作为一名经验丰富的Oracle DBA，我在翻译本书的过程中，经常为书中的精彩之处叫好，一些困扰自己的难题也在译书过程中找到了答案。本书不仅给予你解决问题的答案，而且系统地给予你分析、解决问题的方法，它既有广度，也有相当的深度，你再也不需要在书堆中寻找答案了，可以说这是我所看到的Oracle书籍中最棒的。

在此我要感谢我们这个充满协作精神的工作小组。参加本书翻译工作的有王磊、蒋蕊、王焱、钟亦亮、邹复建、刘蕴秀、罗坤、曹鲁湘、徐兴杰、李大鹏、巩国、赵鹏、吴磊、蒋天仪、王毳、高志刚、黄蕾、赵鹏、周燕燕等等。

虽然我们尽了百分之百的努力，也难免会出现一些不尽人意的地方，我们诚恳地请读者朋友批评指正。如果对本书的翻译工作有哪些看法，不妨也通知我们，我们一定虚心接受。

1999年10月1日

前　言

适用本书的读者

在过去的几年中，Oracle数据库环境已成为世界上最流行的数据库平台之一，尽管Oracle数据库的主要目标依旧是服务于那些要求海量数据处理能力的大型公司或政府部门，但Oracle已经推出了许多将Oracle环境介绍给较小组织的新产品。在流行的微软Windows NT平台上已有了可用的功能强大的Oracle数据库，将Oracle集成到各种类型的企业和组织的计划正在执行，这意味着需要在Oracle技术方面要进行更多的专业训练——并不只是《财富》上列出的五百家企业需要Oracle数据库。

本书是为Oracle数据库专业人员而写的，目标是把中级技术人员培训为高级的Oracle数据库管理员（Database Administrator，DBA）；但对于其他的数据库专业人员如开发者、设计者、工程技术人员或体系结构设计者来说，这本书也是非常有用的。在本书中你将发现那些你所需要的让技术成为现实的信息，而不是关于它应该如何工作的条文。终端用户也将发现这本书十分有用，尤其对于那些只有基本的Oracle或关系数据库管理系统（RDBMS）经验的用户是非常有帮助的。想要在合理地建立、维护及调整的Oracle 数据库上贮存数据的Web管理员和开发者，也应该看一看这本书。如果UNIX和NT系统管理员（SA）必须管理运行Oracle产品的系统，那么他们很快会发现这本书非常有用。最后，对于那些必须管理数据库管理员和使用Oracle的与数据库相关的其他专业人员的技术经理来说，在做出有关Oracle 产品的关键性技术决定时，会发现在这本书简直是无价之宝。

本书的重点在于Oracle的实现和使用，以及数据库管理、开发、维护和调整，但是，书中也提供了关于关系数据库理论和设计的一个坚实基础，这是每个数据库专业人员都应该理解的。本书的作者都是经验丰富的Oracle专业人员，他们来自企业、政府部门、高等学府和Oracle公司。除了每章的内容外，本书还提供了非常有用的经过提炼的信息，以提示、注意和警告的形式给出。本书包含的信息是在Oracle现实环境中多年工作的经验总结。尽管此书是在广义上讲述Oracle，但把重点放在了Oracle最新的RDBMS产品系列的主线上，尤其对那些有兴趣快速学习Oracle 8和Oracle 8i的人来说，这本书是必不可少的。本书为读者提供了一个坚实的Oracle 7基础、Oracle 8特性的综合概括以及Oracle 8i特性的广泛概述。

用哪种版本：Oracle 7、Oracle 8还是Oracle 8i

对Oracle RDBMS的发展过程进行一次快速的讨论，将会帮助你更好地用描绘这个过程，可以更好掌握本书中的内容。这个概述是必须提供的，因为围绕Oracle 8i版本出现混淆并不鲜见。首先，当本书提到Oracle 7时，指的是Oracle 7.x，尤其是Oracle 7.3.x；当提到Oracle 8时，指的是Oracle 8.0.x；当提到Oracle 8i时，是指Oracle 8.1.x；当提到Oracle 8.x时，是指Oracle 8和Oracle 8i，或者换句话说，指的就是Oracle 8.0x和Oracle 8.1.x。

当Oracle 8第一次发行时，它提供了一些优于Oracle 7的重要性能和选项，但是，基本的

RDBMS Oracle 7 引擎并没有被完全重写。不过，从Oracle 7到Oracle 8，仍然是一个重要的升级。Oracle 8i现在正处在发行过程中，它的基础的RDBMS引擎与Oracle 8相比大部分仍未改变，甚至比从Oracle7到Oracle8的改变还要少。Oracle 8i可以被看作是Oracle 8功能的扩展集。

尽管这样，它们代表了两种不同的产品。Oracle 8和Oracle 8i有着并行的开发途径，所以它们的小版本几乎同时出现。例如：Oracle 8.0.5发行后，不久就看到Oracle 8.1.5的发行版了，下一个可望的版本显然是8.0.6和8.1.6。很明显，Oracle 8.0.x的技术形成Oracle 8.1.x的基础，甚至比Oracle 7.3.x给予Oracle 8.0.x的还要多。

想一想下面的问题：什么事情Oracle 8能做而Oracle 8i不能做？没有，因为Oracle 8是Oracle 8i功能的子集。那么什么事情Oracle 8i能做而Oracle 8不能做？除了其他的一些事情，Oracle 8i还提供了许多与因特网有关的能力，最重要的是它将Java集成为一种内部的数据库语言，这种语言可充当SQL的补充或替代品，而且它还集成了一个Web服务器和开发平台（WebDB）。根据Oracle公司的声明，对Oracle 8.0.x的开发（不是支持）将终止于8.0.6版，并且在那之后，Oracle 8.1.x将全面取代Oracle 8.0.x，换句话说，Oracle 8i将最终全面地取代Oracle 8。

随着Oracle 8i的出现，出现了Oracle应用服务器（Oracle Application Server，OAS）和其他产品，例如涉及基于Web开发的开发器。这些产品的目的是什么呢？概括来说，WebDB只是OAS功能的子集。另外，WebDB也不能做设计器（Designer）和开发器（Developer）所能做的所有事。尽管有一些产品的功能重叠，但没有重叠到足以使OAS和其他产品被舍弃的地步。相反，Oracle 8i可以看作是对中小型应用（实际上，是工作组或部门）的彻底解决方案，这种应用只要求简单的设计和一些Web开发。当必须合理地设计和建造大型应用时，要涉及复杂的多层软件体系机构，OAS、设计器和开发器等产品将被用来实现这种需求。因此，尽管出现了一些功能上的重叠，仍存在着保留所有这些产品的需要，它们被设计为满足不同的应用需要。

怎样使用此书

本书是一本实用参考手册，也是一个学习的工具。把本书放在伸手可及的地方，因为一线Oracle专业人员可能会面临许多典型问题，这些问题的答案都可以在本书中找到。

那些已经接触过关系数据库理论的有经验的Oracle 数据库系统管理员可能想跳过第一部分，直接进入本书的核心部分。对于需要了解Oracle 7体系结构和数据库管理的读者，复习一下第二部分。对于那些精通Oracle 7的读者，可直接进入第三部分，这部分全是有关Oracle 8的内容。对于需要了解在Oracle 7或Oracle 8中如何进行性能调整的读者，接着看第四部分。缺少经验或不熟悉关系数据库背后的理论与概念的数据库管理员不应跳过第一部分的学习。第一部分包含的概念对于一个数据库管理新手来说是一个良好开端。

剩余的部分可以作为Oracle 7和Oracle 8的参考。对一个数据库管理员来说，除了可以参考第二部分外，还可以参考第五、六部分。对于需要了解Oracle网络的读者，可以参考第七部分。对于需要了解Web服务器的读者，可以参考第八部分。对于需要了解并行与分布式数据库的读者，参考第九部分。关于特定平台的说明信息，可以参考附录。

不仅仅是数据库系统管理员，技术经理也会时常发现本书作为参考书是很有用的，这依赖于他们需要指导或参考的方面。就这点而论，本书对所有用户都是适用的。你可以按照你

所喜欢的任意方式使用本书：为了学习而读本书或者把本书当作参考书使用。

每章中都有足够的示例，帮助你学习、掌握你需要知道的东西。实际上，有了大量的例子和清晰的解释，本书的前一版已经被成功地用作教授大学一、二年级学生的Oracle数据库教科书。

原书出版社网址：www.quecorp.com
原书书号：ISBN 0-7897-1975-4

第一部分 数据库管理的原则

第1章 数据库、DBMS原理和关系型模型

本章要点：

了解数据库
了解DBMS
了解RDBMS

1.1 了解数据库

在过去的许多年里，有许多关于“数据库”这个名词的定义。数据库是一个服务于一个核心目标的数据的有组织的集合。数据库中的数据是有组织的，从某种意义上说，数据库中存储的数据采用一种不变的方式被存储、格式化、存取以及显示。因为数据库不含有无关的或冗余的数据，它可以适用于一个核心目标。一本电话簿就是一个很好的数据库例子，它包含有关的数据（名字），让人们能够查找电话号码；它不包含无关的数据，如某人的电话机的颜色；它只贮存那些与它的目标相关的信息。最常见的，一个数据库的目标是商务应用，但是也可能贮存科学、军事或其他数据，这些数据通常不能当作商务数据看待。因此，有商业数据库、科学数据库、军事数据库以及其他的数据库等等。另外，数据不仅能根据它的应用分类，还能根据它的格式分类，现代数据库包括多种类型的数据。例如，现在数据库贮存图像、图表、声音、视频或包括两种或多种类型的复合文档，已经是很普通的事了。

当讨论数据库特别是数据库设计时，通常称数据库所服务的核心目标为它的业务(business)，而不管它属于什么特殊的领域，如太空、生物医学或其他。此外，在实际生活中，你会发现数据库通常有它明确的业务应用。

在早些年，编写程序实现自动数据处理（Automatic Data Processing，ADP）要求的程序员发现，在每次运行中，他们需要频繁地贮存数据，这就是通常所说的对永久内存的需要，即从程序的一次运行到下一次运行时，需要将数据存留或贮存。这个基本的需要成为数据库发展的开端。其次的需要——简单的数据存储，也促进了数据库的产生。在线归档和历史数据就是一对特别的例子。尽管文件、目录和文件系统能满足多数普通的数据存储需要（包括索引变化），但数据库可以做文件系统能做的工作并且还可做更多的。

现代数据库通常为上级组织或企业的部门，或它们的小型组织单位实现存贮处理需求。因此，你用术语“企业范围（enterprisewide）”来指代整个组织的商务范围，用“部门范围(departmentwide)”来指代一个部门级的范围，用“工作组（workgroup）”指一个部门内的一些单位。通常，在部门范围和工作组级上查找数据库。

偶尔，你会查找服务于企业范围的数据库，例如工资单和人事数据库，但是它们在数量上远远少于那些较小的数据库。实际上，当几个部门的数据库放在一起或合并成为一个大数

据库时，这就是建立一个数据仓库（Data Warehouse、DW）的本质。那些充当大型数据库的数据源的小型数据库，称为可操作数据库。然而，这不是新东西，一个操作数据库就是产生数据的数据库，多年来一直被叫做成品数据库；只有在建立数据仓库的前提下，你才会发现成品数据库是指可操作数据库或有时是指可操作的数据存贮。随着因特网技术的出现，数据库和数据仓库现在常常作为前端Web浏览器的后端使用。

当工作组数据库合并起来以满足一个大型部门需要时，结果通常相当于一个数据中心（Data Mart、DW），一个DM就是一个部门规模的数据仓库。和术语“数据库”那样，术语“数据仓库”也会产生多种定义。然而，当你把几个小型数据库集成为一个大型数据库，为一个较广泛的组织服务时，如果该数据库存储历史数据、提供决策支持、提供数据汇总、提供只读数据，并且实质上充当所有向它提供数据的相关成品数据库的数据接收器，那么它通常被看作是一个数据仓库。

另外，如果一个数据库增大的原因是因为该数据库是一个长时期贮存数据的历史数据库（例如一个人口普查数据库），或因为它必须贮存数据的类型（例如一个图像数据库），或者因为它必须采用的贮存数据的频率（例如一个卫星遥测数据库），那么它通常被称为一个非常大的数据库（Very Large Database、VLDB）。

随着时间的发展，由于磁盘容量的增大和价钱的下降、均衡多处理技术机器的出现、冗余廉价磁盘阵列（Redundant Array of Inexpensive Disks，RAID）技术的发展、数据库软件的增多或规模化，判定一个数据库是否为VLDB的条件不断变化。当前，一个常用的准则是任何100GB或100GB以上的数据库就可以被看作是一个VLDB；而就在几年前，10GB就被看作是分隔点了。

1.2 理解DBMS

数据库管理系统（Database Management System，DBMS）就是管理一个数据库的软件，它充当所有数据的知识库，并对它的存储、安全、一致性、并发操作、恢复和访问负责。DBMS有一个数据词典（有时被称为系统目录），其中贮存着它拥有的每个事物的数据，例如名字、结构、位置和类型，这种关于数据的数据也被称为元数据（metadata）。在一条数据的生存周期里（从它的创建到删除），这条数据的逻辑和物理信息都被记录在数据词典中。数据库系统管理员（Database Administrator，DBA）应该熟悉DBMS的数据词典；在数据库的整个生命周期内，数据词典为他或她服务。

1.2.1 数据的安全

在成品数据库中，安全性一直是一个重点，在开发或测试数据库中也常如此。这通常不是有没有安全性的问题，而是有多大安全性的问题。除了操作系统和网络安全设施外，一个DBMS通常提供几层安全措施。最常见的情形是，DBMS要求用户登录用户帐户的口令，口令被验证为真，才能访问数据库。

DBMS也提供其他的机制，例如组、角色、权限以及简档，这些都提供更加精良的安全措施。提供这些安全级不仅是为了加强安全性，而且为了建立商业安全策略。例如，只有一个经过验证属于航空组的用户才能存取航空数据；又例如，只有经过验证拥有操作者角色的用户才能备份数据库。

1.2.2 维护和实施完整性

数据的完整性是指它的一致性和正确性。对于一致的数据，在它所有的出现的场合中，必须以相同的方式建模和实现，对于正确的数据，它必须是正确、精确和有意义的。

DBMS维护完整性的一个方法是在一条数据项发生改变的过程中进行锁定。数据库通常是在数据库页级或行级锁定数据。锁定偶尔也允许并发操作，在下面你将涉及到这种情形。

DBMS实施完整性的另一个方法是：如果一条数据贮存在多个地方，那么把变化复制到这条数据上。DBMS实现完整性的最后一个方法是通过监视输入的或改变的数据值，使它们全部符合要求的规格（例如：一个范围检查）。

如果接着的是合适的建模和实现过程（在后面的第2章“逻辑数据库的设计和标准化”中讨论），DBMS会自动地帮助实施这种完整性，例如，通过一个触发器或约束条件，这两个概念在第28章“PL/SQL基础”中被定义和说明。没有完整性，数据是没有价值的。有了完整性，数据就是信息。完整性不仅能增强数据，它还给予数据以价值。

当DBMS提供多用户存取时，它必须管理并发操作。那就是说，当几个人同时访问相同的数据库（特别是同一条数据）时，DBMS必须保证这种并发访问能够以某种方式实现。并发可以广义上定义为同时发生，即两个或多个用户在同一时期访问同一数据。

DBMS实现并发操作的方法倒不太复杂，但背后的实际程序却很复杂。本质上说，当两个或多个人只想简单地浏览一下同一个数据而不改变它时，一切还好。但当至少一个人想改变数据而其他人想浏览或也想改变数据时，DBMS必须贮存多个备份。当每个人都完成改动后，DBMS必须将所有改变的备份保存为一条正确的数据。

前面已提到并发管理的一种方式是加锁。通常来讲，锁越精密（越小），并发性就越好（那就是说，更多的用户无需等待即可同时访问）。行通常比最小的数据库页或块都小，因此，行级锁可较好地管理短小、杂乱数据的处理，块级锁可较好地管理长的、连续数据的处理。

这就是并发性和完整性是如何结合的。当一个人想查看或更改一条数据时，这个人就是执行数据库的一个事务。

1.2.3 理解事务

作为DBMS标准的一部分，DBMS有一个事务管理器（transaction manager），它的目的是管理并发操作和确保事务的完整性。事务管理器的工作是艰巨的，因为它必须允许许多人同时访问相同的数据，并且在访问之后要把数据放回到数据库中，就好象在某个时间上只有一个人存取数据，他完成工作后另一个人才工作，这样确保数据的正确性。DBMS解决数据的多个备份的基本方案就在这中间。如果（并且是只有）数据是串行的，那么在保持数据的准确性的同时也进行了事务处理。简单地说，DBMS必须重新整理所有改变，以使得它们的最终结果仿佛发生在一个文件中。

事务是并发或工作的单位。不能产生比一个事务更小或更少的东西，也就是说，没有人能够只完成数据改变工作的一部分。全部事务必须都是原子的，所以每个单独的事务要么完成，要么不完成。直到20世纪现代物理发展起来以前，原子一直被当作物质的最小单位。同样，事务也是并发的最小单位，它要么全有，要么全无。一个被完成的事务可以说是被提交了，没有完成的事务则是被回滚了。

DBMS用事务作为恢复的单位来控制恢复，正常完成、手工要求中止以及意料之外的退

出都要求DBMS重新访问数据的多个备份来提交或回滚数据。为了回滚或前滚，DBMS保持了一个事务日志。回滚是一个撤销操作。前滚是一个重做操作，例如，由于一个硬件或软件错误，一个已提交的事务无法将它所做的操作从内存中存储到磁盘就会发生前滚，DBMS只是简单地重做这个操作。因此，在DBMS中事务恢复的关键是一个事务必须是原子的，而且在必要时可以做、不做或重做。

1.2.4 与数据库通信

如果你不能和一个DBMS对话，那么这个DBMS就不是很好的。你可能会问怎样和DBMS对话。可以通过一种存取或查询语言访问数据库。结构化查询语言（Structured Query Language，SQL）是当今主要的查询语言，它主要用于管理主流类型的DBMS——关系型DBMS（RDBMS）。所有与数据库相关的通信往来都将通过DBMS完成，为了做这件事，你可以使用SQL或其他类似的东西。数据库系统管理员（DBA）使用查询语言来建立并维护数据库，用户使用查询语言来访问数据库并查看或更改数据。

1.3 理解RDBMS

1970年，E. F. Codd创立了关系模式的概念。在RDBMS（例如DB2）产生之前，层次（IMS）和网状（IDMS）模式是常见的。在这些模式之前，使用平面文件（操作系统文件不一定平面）来建立数据库，并且使用第三代语言（3GL）访问例程。实际上，一些专用系统仍然是按这种方式建立的，只是进行了修改或根本没有变化。在大型机和微机中依然存在着许多这样的遗留数据库。CODASYL（数据系统语言协会）是数据库任务组（Database Task Group，DBTG）创建的一种数据库标准，这是一种基于COBOL的网络数据库标准，并且IDMS是一个厂商的实现。但是，从70年代起，RDBMS已经逐渐地控制了市场，如Oracle、Sybase、Informix和Ingres。

最近，面向对象（Object-Oriented，OO）的DBMS已经成为最为突出的数据库管理系统，并找到了许多适当的应用环境，如在CAD/CAM、工程、多媒体等等。面向对象DBMS适于在这些领域中应用，因为在一个几乎作事务性的环境中，它们具有控制复型数据类型实力。由于竞争，RDBMS厂商为了提供包括文本、音频、图像和视频数据类型的面向对象/多媒体性能，已经制造了商业可用的通用服务器。Oracle的Universal Server就是一个例子。另外，用户定义的数据类型或可扩展类型，已经被扩大或增加到核心数据库服务器中，Oracle 8就提供了这样的性能。类似这样的RDBMS产品被认为是混合的，然而它们明显比以前的RDBMS更具有主流性。

此外，多维数据库（Multi-Dimensional Database，MDD）也分享了部分市场份额，这些数据库为带有许多必须被多维存取或列表的变量（例如行为科学数据）的应用提供了高度索引化的数据。在传统的RDBMS中，这几乎是不可能实现的，数据库只允许单独使用。再者，为和MDD竞争，RDBMS供应商提供了一些他们自己的层次产品，这些产品提供超级索引化的数据，并使用了特殊的技术，例如位映射索引。Oracle的Express就是一个多维数据库的例子。

1.3.1 关系模型

你已经了解了DBMS的主要任务，为了进一步了解一个RDBMS是由什么构成的，你必须

先了解关系模型。下列情况出现在一个关系模型中：

■ 数据的基础项是关系。

■ 在这些表上的操作只产生关系（关系型闭合）。

什么是关系？这是一个描述两个集合的元素如何相互联系或如何一一对应的数学概念。因此，关系模型是建立在数学基础上的。然而，对你来说，关系只是一个带有一些特殊属性的表，一个关系模型把数据组织到表中，而且仅在表中。客户、数据库设计者、数据库系统管理员和用户都以同样的方式——即从表中——查看数据。那么，表就是关系模型的近义词。

一个关系型表有一组命名的属性（attribute）或列，以及一组元组（tuple）或行。有时列被称为域，行被称为记录，列和行的交集通常被叫做单元。列标示位置，有作用域或数据类型，例如字符或整数。行自己就是数据。表1-1有三列四行。

表1-1 汽 车 表

制 造	模型品牌	价 格
Toyota	Camry	$25K
Honda	Accord	$23K
Ford	Taurus	$20K
Volkswagen	Passat	$20K

一个关系表必须符合某些特定条件，才能成为关系模型的一部分：

■ 贮存在单元中的数据必须是原子的。每个单元只能存贮一条数据，这也叫信息原则（Information Principle）。尽管在过去的数年中按某些违反这一条的方式已经建立了许多系统，但违反这一条将不能运用良好的设计原则。当一个单元包含多于一条的信息时，这叫做信息编码（information coding），一个很好的例子是一个车辆识别号码（Vehicle Identification Number、VIN）。如果它被存贮成一列，这将违犯信息原则，因为它包含了多条信息，例如产地、型号、出厂等等。在这样的情况下，是否采用违背理论的方案是一个设计的选择问题，尽管在多数情况下，结果证明这对数据的完整性是一不利的。

■ 贮存在列下的数据必须具有相同数据类型。

■ 每行是唯一的（没有完全相同的行）。

■ 列没有顺序。

■ 行没有顺序。

■ 列有一个唯一性的名称。

除了表和它们的属性，关系模型有它自己特殊的操作。不需要深入研究关系型数学，只需说明这些操作可能包括列的子集、行的子集、表的连接以及其他数学集合操作（如联合）等就足够了。真正要知道的事情是这些操作把表当作输入，而将产生的表作为输出。SQL是当前RDBMS的ANSI标准语言，它包含这些关系型操作。在SQL占主导地位之前，一种具有竞争性的语言是来自Ingres的QUEL或QUEry语言，另一种是UDL（统一数据语言、Unified Data Language）。ANSI（美国国家标准化组织）是一个具有广泛范围的标准实体，其中包括计算机软件语言（如SQL）的标准。

允许数据操作或数据处理的主要语句是SELECT、INSERT、UPDATE和DELETE。因此，这些数据处理操作中任何一个都是一个事务。

允许数据定义或结构化处理的基本语句是CREATE、ALTER和DROP。所有这些语句都可

以使用一组子句来替代，这些子句可以利用多种变化来定义和存取组成数据库关系表的结构和数据。因此，SQL既是一种数据定义语言（Data Definition Language，DDL），也是一种数据操作语言（Data Manipulation Language，DML）。统一的DDL和DML肯定比两种不同的语言和接口更有效、更有用。数据库系统管理员和用户可以通过完全相同的语言访问数据库。

关系模型要求的最后一件事是两个基础的完整性原则。它们是实体完整性原则（entity integrity rule）和引用完整性原则（referential integrity rule）。首先，让我们看看两个定义：

- 主键（primary key）是能唯一标识行的一列或一组列的集合。有时，多个列或多组列可以被当作主键。
- 由多个列构成的主键被称为连接键（concatenated key）、组合键（compound key）、或者更常称为复合键（composite key）。

数据库设计者决定哪些列的组合能够最准确和有效地反映业务情形，这并不意味着其他数据未被存贮，只是那一组列被选作主键而已。

剩余有可能被选为主键的列被叫做候选键（candidate key）或替代键（alternate key）。一个外键（foreign key）是一个表中的一列或一组列，它们在其他表中作为主键而存在。一个表中的外键被认为是对另外一个表中主键的引用。实体完整性原则简洁地表明主键不能全部或部分地空缺或为空，引用完整性原则简洁地表明一个外键必须为空或者与它所引用的主键当前存在的值相一致。

一个RDBMS就是一个建立在前面这些关系模型基础上的，一般能满足所提到的全部要求的DBMS。但是，在70年代末到80年代初，RDBMS开始销售的时候，SQL超越了本质为非关系型的系统，受到普遍欢迎，并被称作关系型。这引发了一些修正活动，即Codd十二条法则（1985）。

1.3.2 Codd十二条法则

DBMS应该遵循Codd提出的十二条法则，才能被分类到完全关系型：

1）信息法则。信息表现为贮存在单元中的数据，正如前面所讨论过的，将VIN作为一个单个的列使用，违反了这条规则。

2）授权存取法则。每一个数据项必须通过一个“表名+行主键+列名”的组合形式访问。例如，如果你能用数组或指针访问一个列，就违反这条规则。

3）必须以一致的方式使用空值。如果由于缺少数字值，空值（Null）被当作0来处理，或者由于缺少字符值而被当作一个空格处理，那么它就违反了这条规则。空值仅仅是指缺少数据而且没有任何数值。如果缺少的数据需要值，软件提供商通常提供使用缺省值的能力满足这一目的。

4）一个活跃的、在线数据字典应作为关系型表被存储，并且该字典应该可以通过常规的数据存取语言访问。如果数据字典的任何部分贮存在操作系统文件里，就违反了这条规则。

5）除了可能的低级存取例程外，数据存取语言必须提供所有的存取方式，并且是存取的仅有方式。如果你能通过一个实用程序而不是一个SQL接口来存取支持一个表的文件，就有可能违反了本规则。参见规则12。

6）所有能被更新的视图应当是可更新的。例如，如果你能将三个表连结起来，作为一个视图的基础，但却不能更新这个视图，则违反本规则。

7）必须有集合级的插入、更新和删除。目前，大多数RDBMS提供商都在某种程度上提供了这种能力。

8）物理数据的独立性。应用不能依赖于物理结构，如果一个支持某表的文件从一张盘移动到其他盘上或重新命名，不应该对应用产生影响。

9）逻辑数据的独立性。应用不应依赖于逻辑结构。如果一个表必须被分成两个部分，那么应该提供一个视图，以把两段连接在一起，以便不会对应用产生影响。

10）完整性的独立性。完整性规则应该贮存在数据字典中。主键约束、外键约束、检查约束、触发器等等都应该贮存在数据字典中。

11）分布独立性。一个数据库即使被分布，也应该能继续工作。这是规则8的一个扩展，一个数据库不仅能在一个系统（本地地）分布，也能在通过系统的网络（远程地）分布。

12）非破坏性法则。如果允许低级存取，一定不能绕过安全性或完整性规则，这些规则是常规的数据存取语言所遵守的，例如，一个备份或载入工具不能绕过验证、约束和锁来备份或载入数据。然而，软件供应商出于速度的原因，通常提供这些功能。那么，数据库系统管理员就有责任确保数据的安全性和完整性，如果瞬间出现问题，应该立即恢复。例如当载入VLDB时，可以临时禁止并重新打开约束检查。

如果一个DBMS能满足本章中讨论的所有基本原则（两个定义、六个属性、关系型操作以及两个完整性规则）和这十二条法则，那么它就可以被当作一个RDBMS。Codd用他的法则0总结了这一切："对于一个有资格成为RDBMS的系统来说，该系统必须排他地使用它的关系型工具来管理数据库。"

第2章 逻辑数据库的设计和标准化

本章要点：

实体-关系模型

将实体关系图映射为关系模型

理解标准化

2.1 实体-关系模型

对一个数据库管理员来说，所能为他的数据库做的最好的事情就是使之开始于一个合理的逻辑设计。不幸的很，数据库设计常常被匆匆地完成以致于做错，甚至在数据库建立后重新返工。一个见闻广博的和聪明的数据库管理员知道对数据库进行很好的设计，会大大提高数据库的性能，而不是减损数据库的性能，这种思想与流行的思想相反。事实上，直接投入物理设计或更深层的工作，只会带来麻烦，不仅在性能方面，而且在数据完整性方面同样如此。如果一个数据库运行得很快，但收藏的数据却是错误的，这又有什么好处呢？而且，在数据库系统的早期设计阶段，创建一个合理的逻辑设计，可以让它接受以后创建和维护阶段物理设计改变的考验。可是，如果你在逻辑设计阶段走捷径，你将不但可能需要重新设计逻辑模型，而且还可能需要重新构造下面的物理模型。间接的代价（职员的工作时间、停工期等等）可能会是令人吃惊的。在进行和建立数据库之前，需要了解逻辑数据库设计和标准化背后的基本原则。

在70年代中期，关系数据库模型逐渐超越其他的数据模型占据主导地位，关系模型技术的风靡使设计性能得到规范化。这其中最流行的是实体关系图（Entity-Relationship Diagram，ERD），它是P.P.Chen 在1976年提出来的。这就是语义数据模型，因为它试图捕获业务要素（业务本质）的语义或正确含义。因为关系模型本身几乎就是一个依据语法的模型，是一种主要处理结构的模型，实体关系图（ERD）通常用于补充它。实际上，ERD建模必然先于关系建模。当一个ERD结束时，它或多或少地被直接映射到关系模型上，而后关系模型再被映射到它的物理模型上。

一个实体是一个业务元素，比如一个雇员或一个项目。一个关系就是两个实体之间的联系，比如工作于不同项目的雇员。属性即组成实体的特征，比如一个雇员的工资或项目的预算。属性被认为是来自定义域中的取值或值的集合，它们所取的值是它们以后在关系模型中所用到的数据。它们是对一个事物全部抽取或部分抽取。ERD有许多画法，只要你选择一种并在整个使用过程中保持含义一致即可。

使用方框代表实体画高级图（那些不带属性的），将实体的名字列于方框的中心。低级图的实体名称列于方框中的上部，后面跟着属性名称。在方框之间画有箭头，代表关系类型。有三种基本类型的关系：一对一、一对多以及多对多。一对一的关系根据一对一关系的类型，在线条的一端或两端使用单箭头。一对多使用双箭头，多对多在两边使用双箭头。当一个实体的每一个值都和另一个实体的一个值并且只有一个值有关时，就存在着一个纯粹的一对一

关系，反之亦然。这种类型的关系是很少见。图2-1展示一种一对一关系，一个丈夫只能和一个妻子结婚，而且一个妻子也只能和一个丈夫结婚（没有考虑一夫多妻或一妻多夫的情况）。

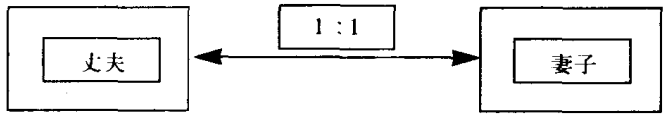

图2-1 一对一（1:1）关系

一种更为普遍的一对一关系是子类型关系，这是面向对象分析和设计的基础之一。在面向对象系统中，这被看作是类和子类（或者更简单地说，类的级别）。图2-2显示一个一对一子类型关系是如何被模拟的，该图显示了一个经典的例子：正方形是长方形的子类型。箭头的方向指明了继承的方向，继承是有关类级别的另一个面向对象概念。换句话说，在更为普遍的实体中的属性（长方形）上，将属性（如长和宽）送给更为特定的实体（正方形）。因此，继承的方向是从一般到特殊。

子类型关系比纯类型的一对一关系更为常见，但这两种都不常用。通常，当一个设计者偶然遇到一对一关系时，他必须问下列问题：

- 这两个实体能结合吗？
- 它们对于自己的目标是否是完全相同的？
- 它们是否由于某些业务原因必须保持独立和不同吗？

通常情况下，一对一实体是可以合并的。

在关系模型中，使用得最多的关系是一对多关系，图2-3显示了一种一对多关系。一个州有许多城市，但所有这些城市只属于一个州（这是正确的。然而，你会发现一个城市的名字被不同的州重复使用，这只意味着一件事：作为一名设计者，你选择的主键一定不能是一个城市的名字，例如：主键可以是州名+城市名。前面的章节包含了主键的定义）。

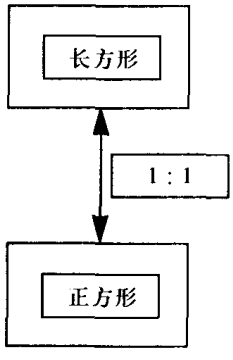

图2-2 一个一对一（1:1）的子类型关系

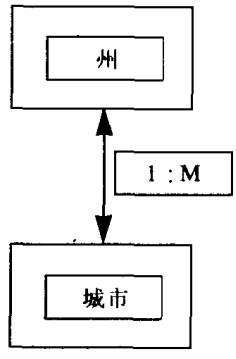

图2-3 一对多（1:M）关系

最后，图2-4表示许多雇员正在从事多个项目——这是一个多对多关系。注意带有下划线的属性是标识符属性，代表着你当前推测它们有可能最后成为关系模型中的主键。

在这种情况下，作为一名设计者，你所能为自己做的最好的事情就是把自己从所有的多对多关系中解脱出来；并不是真正地除去它们，但你可以在它们原先位置上使用两个或多个一对多关系来代替多对多关系。之所以要这样做，是因为关系模型并不能实际处理一个多对多关系的直接实现。仔细想想，如果有许多从事多个项目的雇员，你怎样来贮存外键？你不能在一列中贮存多个值，这样违反了数据必须是原子的关系型要求，这意味着没有一个单元

能够持有一条以上的信息。信息法则（在第1章已讨论过）的这一示例也说明：它是第一范式（First Normal Form）的一个特殊情况。稍后我们将讨论第一范式。因此，为确保数据的原子性，每条多对多关系都被两个或者多个一对多关系所取代。

因此，你所要作的工作就是分割多对多关系，这样多名正从事多个项目的雇员就变成了一个雇员拥有多项职位和一个项目分配给多个职位，而职位成为新的实体。图2-5表示了这种新的关系，注意，标识符属性已经被合并。

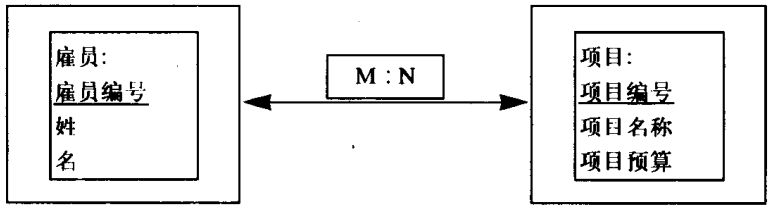

图2-4　多对多（M: N）关系

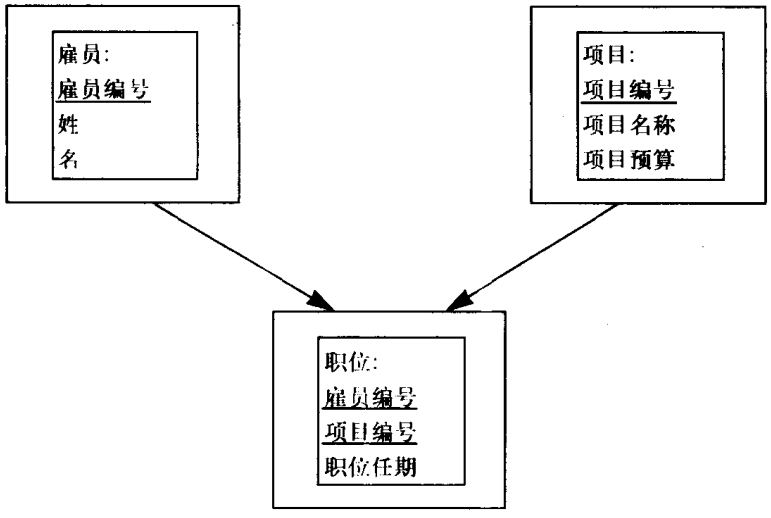

图2-5　使用两个一对多（1:M和1:N）关系修改多对多（M:N）关系

在关系模型中，被称为职位的新实体通常被叫做交叉表，因为它代表着与其相关的两个表中每一对实际值的交集，有时也称为叫纽带表（junction table）或连接表（join table），交叉表是这样的一个实体：它不一定总是一些业务元素的真实抽象，但是它是解决和实现关系模型中多对多关系的基本方法。

2.2　将实体关系图映射为关系模型

一个ERD可以准确地映射到关系模型上，因为它就是为这个目的而创建的。实质上，实体变成了表，属性变成了列，标识符属性变成了主键。如果不通过交叉表，关系并不能真正实现。外键是这样创建的：把一个单方表中的主键放入一个多方表中。例如：一个州对应许多城市的关系要求你把州的主键放到城市表中，在那里创建一个外键，这样在两者之间就建立了关系。

当前市场上有许多自动化计算机辅助软件工程（Computer Assisted Software Engineering，CASE）工具能够帮助你实现这个映射。这样的工具包括LogicWorks公司的ERwin和Oracle公