高等学校教材

面向对象数据库

李也白　范春晓　编著

高等教育出版社

高等学校教材

面向对象数据库

李也白 范春晓 编著

高等教育出版社

（京）112号

内 容 提 要

关系数据库系统已经成为数据处理应用的标准。然而，还存在一大类应用问题，对此类问题，关系数据库的数据建模能力非常有限。面向对象数据库系统就是为了满足这些复杂应用数据建模需求而发展起来的。这种数据库系统从关系模型中脱离出来而强调在数据库框架中发展类型、数据抽象、继承和永久性的概念。

本书的主要目的在于，向读者详细介绍面向对象编程语言研究和面向对象数据库研究两方面的进展情况，综合叙述语义数据模型技术、校正、可重用性、面向对象数据建模、类结构、继承、面向对象的查询处理、永久性、并发控制、恢复和分布、实现、簇集、版本控制、面向对象知识库技术等基本概念和问题。

本书内容安排循序渐进，语言叙述深入浅出，例子丰富，示图多样，可作为大学高年级学生和研究生的面向对象数据库教材，也可供想拓宽自己在面向对象编程语言和系统、数据库设计与实现方面知识的科研工作者和计算机爱好者学习和参考。

图书在版编目（CIP）数据

面向对象数据库/李也白，范春晓编著．—北京：高等教育出版社，1998
高等学校教材
ISBN 7-04-006419-7

Ⅰ．面… Ⅱ．①李… ②范… Ⅲ．面向对象语言-数据库系统-高等学校-教材 Ⅳ．TP311.13

中国版本图书馆 CIP 数据核字(98)第 03101 号

*

高等教育出版社出版
北京沙滩后街 55 号
邮政编码：100009 传真：64014048 电话：64054588
新华书店总店北京发行所发行
化学工业出版社印刷厂印装

*

开本 787×1092 1/16 印张 10.75 字数 260 000
1998 年 4 月第 1 版 1998 年 4 月第 1 次印刷
印数 0 001—3 192

定价 9.20 元

凡购买高等教育出版社的图书，如有缺页、倒页、脱页等质量问题者，请与当地图书销售部门联系调换

责任编辑 辛治运
封面设计 李卫清
责任绘图 黄建英
版式设计 辛治运
责任校对 翁咏梅
责任印制 宋克学

序

目前，在计算机应用领域中，面向对象数据库技术受到了广泛的关注。它把已经广泛应用的数据库技术和面向对象软件开发技术和方法相结合，有利于把常规的管理信息系统(MIS)等数据库应用系统推广到更为复杂的应用领域。常规的关系数据库适合管理大量日常数据表格(以多行多列构成的矩形表格为主)，然而当前有很多应用领域需要管理的信息更为复杂，这些应用领域的信息对象（例如大量的建筑工程图纸、大规模航空测量工程或卫星遥感设备所得到的图片、从信息网络下载的超媒体文本信息等）具有复杂的结构，而且很有应用价值，不仅需要很好地存储保存，而且需要开发新型的信息检索技术来实现对这些信息的快速查询。这就要求我们扩展常规数据库软件的功能，使之具有管理大量的具有复杂数据结构的信息的能力，面向对象数据库技术就是面向这一应用需要而发展起来的。近 2 年来，国际上已经推出了一些具有面向对象特征的数据库管理系统软件，这些软件大致上可以分为两大类，一类是“对象-关系数据库”系统，它通过对常规关系数据库管理系统软件功能的扩展，使系统具有存储管理复杂对象的能力。由于它们能够在传统的 MIS 系统基础上扩展，推广时遇到的阻力较小，目前已经开始占领市场。另一类则是完全新型的面向对象数据库管理系统，目前仍在发展中。凡是关心数据库发展和面向对象技术的计算机工作者，都很有必要了解以上这些方面的技术进展。

本书参考了国际上这方面的专著和资料，也融汇了作者长期以来的研究成果。对于关心数据库发展和面向对象技术的计算机工作者，从本书中可以得到很多启迪和对一些重要问题的解答。本书比较全面系统地介绍了面向对象数据库技术的各个方面的基本原理，如面向对象方法的主要特征；面向对象数据库和常规的关系数据库在数据模型上的主要差别；面向对象程序设计语言和面向对象数据库两者的紧密关系等。对于一些在设计和实现面向对象数据库系统时必须解决的一些技术性问题，如内存优化管理、查询优化、并发控制、数据恢复和数据分布等问题，本书也特别安排了专门章节对它们详细讨论。

目前，国内出版的计算机教科书对面向对象数据库已经做了不少介绍，但仍然很缺乏从原理上深入讨论面向对象数据库的专著。本书内容涵盖全面，结构选材合理，对填补这一空白做出了有益的贡献。对于准备在这一领域进行研究的研究生，以及在信息技术领域的科研工作者和大学高年级学生，它都是一本很有益的参考书。

许卓群

1998 年 1 月于北京大学

前　言

在商业领域里，关系数据库系统已经成为数据处理应用的标准，这在很大程度上归功于它们使用的方便性和灵活性，以及现有许多强大和有效的关系数据库管理系统可供商业界使用。在过去的几年中，这些系统已成功地提供了大规模事务处理应用所必须的执行环境。然而人们普遍认识到，在实际应用中还存在一大类问题，对这些问题，关系数据库的数据建模能力非常有限。这些应用问题的特征是高复杂性、大规模、数据密集，它们在计算机辅助设计(CAD)、计算机集成制造(CIM)和多媒体领域中常常出现。面向对象数据库系统就是为了满足这些复杂应用的数据建模需求而发展起来的。这种数据库系统从关系模型中脱离出来，强调在数据库框架中发展类型、数据抽象、继承和永久性（Persistent）的概念。然而，这种系统引起实现上的严重问题，特别是在永久性和并发控制方面，目前还在不断研究之中，受到了人们的高度重视。为了研究其中的某些问题，在过去的几年里，人们设计和实现了许多面向对象编程语言和系统。

面向对象编程语言已有许多年的历史。早在1966年，俄亥俄大学的O. Dahl和K. Nygaard就设计了被认为是第一种面向对象语言的Simula语言，从本质上讲它是Algol 60在面向对象方面的扩充。在20世纪70年代，出现了一些基于Pascal的强类型、模块化编程语言(如Pascal Plus、Concurrent Pascal和Ada等)，这些语言受Simula语言的影响很大，但却没能保持Simula语言的一些基本的面向对象的特征，特别是继承特征。Smalltalk语言也产生于20世纪70年代，由Xerox公司的R. Kay和A. Goldberg等人设计。尽管它的设计也受Simula语言的影响，但是它的基本原理与Simula语言却不同，它强调一种自由的无类型风格(Free Typeless Style)和动态联编(Dynamic Binding)。

在封装特性上，编程语言对象和数据库对象是相似的，但是它们之间有一些重要的区别。首先，数据库对象必须比创建它们的程序存在的时间长。其次，许多数据库应用要求有创建和访问某一对象多种版本(Version)的能力(这方面的例子可以从历史的数据库、用于软件管理的数据库和计算机辅助设计中找到)。第三，高度活跃的数据库，如可用在空中交通控制和电量分布管理中的数据库，对当条件满足后触发了动作的对象，要有把它与条件(Condition)和动作(Action)结合起来的能力。最后，数据库完整性控制(Integrity Control)要求有把对象和它的约束结合起来的能力。编程语言并不典型地提供永久对象(Persistent Object)和多对象版本(Multiple Object Version)。它们永远也不提供把对象和触发器、约束结合起来的能力。

目前在研究领域中，有一个很流行的趋势是倾向于把面向对象语言向数据库方向扩展，而同时扩展数据库系统使其具有面向对象的特征。一方面，面向对象数据库系统必须提供一种综合的语义数据建模概念集，以便对现实世界复杂应用的实体和联系建模。另一方面，应用程序必须能存取并处理对象，如同这些对象处于一个无限的虚拟内存中一样。

本书的主要目的是向读者详细介绍面向对象编程语言研究和面向对象数据库研究两方面的进展情况。

第一章综述了语义数据模型技术，以独立于模型的扩展实体联系方法开始，详细地描述了关系数据库模型、扩展关系模型、RM/T 和函数数据模型。

第二章以面向对象编程语言的基本原则为基础，讨论处理软件质量的重要主题、校正和可重用性。最后讨论在面向对象编程中为实现上述目的所使用的技术。

第三章介绍面向对象数据建模方面的课题，其中数据库采用复杂的内在联系和紧密的对象集合形式。这一章包括一些实例研究，阐明了传统关系数据库方法和面向对象数据库方法之间最重要的区别。

第四章深入讨论类结构和继承等面向对象方法的中心问题，并以一个广域的面向对象编程语言和系统为例列举了类描述和继承的例子。

第五章讨论面向对象的查询处理。这个课题与数据库系统相比多少有些不成熟，在过去的几年里引起过一些争论。在扩展 SQL 和函数数据模型基础上，我们将描述面向对象查询语言中较有影响和前途的一些方法。

第六章主要讨论数据库系统的一个基本特征——永久性。从带有对象关系的使用到将永久性作为所有数据的正交特性，讨论了各种永久性模型。同时深入地讨论了某些面向对象数据库系统的最合适模型的问题。与永久性有关的主要实现结果将在第八章介绍。

第七章讨论面向对象数据库中与并发控制、恢复和分布有关的问题。这一章首先详细描述传统数据库中的并发控制问题，同时强调了串行性理论和双阶段封锁。然后描述了用于编程语言和数据库领域的各种并发控制模型，讨论了它们与面向对象数据库之间的关系。由于总的来说传统数据库的并发控制机制对于面向对象环境太严格，所以我们研究了在面向对象数据库中与并发有关的一些特殊问题及解决这些问题的建议。面向对象范例对分布数据库的作用也是热门的研究课题，我们也适当讨论了在一个分布式方案中面向对象方法的优点。

第八章讨论实现问题。我们知道，有效的实现是面向对象数据库研究领域里最富有挑战性的工作，并且这种现象可能延续多年。在本章，我们先阐述了在面向对象系统中对象存储和内存管理技术，然后我们讨论与数据库有关的其他重要的实现问题，诸如簇集和版本控制等问题。

在第九章，我们集中讨论面向对象的知识库技术。在这一领域仍有很多待研究的课题，而且面向对象模型在该领域中的作用仍未被人们充分认识。我们将此章的重点集中在知识表达上，从这里我们能很好地认识到面向对象模型的优点。最后，我们指出目前的一些研究方向和正在解决的问题。

在本书编写过程中，全国高等学校计算机教学指导委员会副主任、北京大学计算机科学系博士生导师许卓群教授在百忙中审阅了全稿，为本书做序，并提出了许多宝贵意见，使本书得以改进和提高。另外，本书的出版还得到了冯晓君老师的大力支持和协助，她对这本书的内容编排、选材及撰写工作提出了许多中肯的建议，付出了艰辛的劳动。对以上诸位师长和学友，编者在此致以诚挚的谢意。

由于编者水平有限，书中难免存在缺点和错误，殷切希望广大读者批评指导。

编著者

1997 年 12 月

目　录

第一章　语义数据建模

1.1　简　　介

在过去的 10 年中，数据库技术一直在迅速发展，这不仅表现在数据库系统的数量上，而且也表现在数据库系统所存储的大量信息和它在多种应用上的发展上。这种日益增长的多样化、复杂化的数据库系统促使人们利用更高级的概念、工具和技术来设计和发展数据库。因此，数据库设计方法学近年来有了很快的发展，其目的是在对数据库进行详细的物理和逻辑设计之前，为设计者提供高级抽象方法来对一个计划建模。这些方法学对使用先进的编程语言设计复杂抽象结构有重大影响，例如数据类型的详细说明和独立实现的对象。

在本章中，我们将提出关于数据库应用的语义数据建模基本原则。数据模型是数据库技术的核心，它为概念化数据的深入应用提供了一个基础，并且为完成这些应用而发展的语言和系统提供了一个规范的基础。数据建模主要分为下面两个阶段。

1. 概念设计阶段，包括对现实世界某种状态的抽象和概念模式的设计。
2. 逻辑数据结构的设计，参照概念模式能映射成一个真实的实现。

第一个阶段是分析有关的应用信息需求、准备一个需求的说明并且从中构造一个高级数据模型。一个数据模型应该以参照逻辑组织的数据形式来定义，它由被命名的数据逻辑单元和由现实世界中某些说明决定的数据间的某些联系组成。数据模型必须能够保持基础应用的动态和静态两个方面的特性。系统的静态特性与其包含的对象和对象包含的特性（或属性）及对象的相互联系等概念的抽象有关。这些特性用某些数据描述语言在数据模式中定义。系统的动态特性与对对象的操作有关，而且与对特性的处理和查询等操作有关，也就是说，动态特性模型是对系统行为和操作数据库的说明。另外，数据模型必须能够处理系统的动态方面，也就是说，系统的发展变化应和随时间而改变的应用需求保持一致。

数据模型的另一个重要功能是必须依附于数据库对象和对对象操作的详细、完整的规则(对象的状态绝不能违背完整性规则)。完整的规则或完整性约束常常与静态及动态特性有关。语义完整性约束是高度依赖于数据模型的。传统数据模型(如层次、网络和关系模型)只体现很少的语义完整性并且约束必须由用户自身清楚地描述。在许多语义更为丰富的数据模型中，这种约束可能是模型自身固有的。在我们对数据模型的评价中，完整性是一个重要的评价指标，这一观点将贯穿本书。

本章我们讨论一种数据库设计方法学，重点放在概念数据建模和语义数据建模上。20 世纪 80 年代，人们已发表过很多文章，提出了许多语义数据模型，但是，这些文章在数据库研究团体以外很少引起注意，这可能是由于这些模型过于复杂并且难以实现。然而，语义数据建模中的大多数重要概念都能用扩展实体-联系模型(Extended Entity Relationship，简称 EER)来充分描述。此模型不同于其他模型之处是它在商业应用环境中引起了巨大的关注，并且此模型在计算机辅助软件工程(CASE)的许多工具中扮演了一个基础的角色。此模型最初于 1976 年提出，在当时大多数传统数据处理应用中被证明是充分可行的。然而，在过去的

10 年里，数据库设计者都已面临着应用复杂化的增长对附加语义建模概念需求的增加。这样，EER 方法就必须加以修改并且补充许多其他的方法来丰富其语义能力。对原始模型的重要补充包括子类和超类的概念，而且它们间的紧密联系机制被认为是特性的继承。

由于 EER 模型易于表达和使用，在过去的 10 年里，EER 模型在产业界和研究机构中引起了广泛的关注，并且在概念模式设计中，它仍然是首选的模型。EER 模型如此普及的主要原因之一是它提供了一种借助数据抽象概念的自顶向下的数据库设计方法，在这方面，EER 模型方法与现代软件工程准则是一致的。我们将在第一级设计策略中使用 EER 模型，并且研究数据的 EER 模型与现代数据库管理系统基本形式的可实现模型之间的转换。在下一章中我们将看到 EER 模型是面向对象数据库设计者的有力工具。

在本章中间部分，我们将介绍关系数据模型的基本原理。由于关系数据模型容易使用并能够高效实现，所以关系模型成为 20 世纪 80 年代商用数据库管理系统的事实上的标准，并且它为多数更先进的模型(如面向对象模型)提供了一个重要的检测尺度。回顾关系模型，我们主要关心的问题是：数据建模、数据操作和完整性。我们将把关系模型和面向对象模型的一些我们所关心的特性做一个比较。

在本章最后，我们要简单研究一下其他两个语义数据模型：一个是 RM/T 模型，这是一个获得了更多语义的扩展关系模型；另一个是函数数据模型。和函数程序设计一样，函数数据模型在关于获得语义且保护校正等方面比其他模型有许多固有的优点。

1.2　扩展实体-联系模型

扩展实体-联系模型(EER)的信息是借助于以下三个基本概念表达出来的。

1. 实体：表达被模型化的对象。
2. 属性：表达这些对象的特性。
3. 联系：表达各种实体间的联系。

在下一小节我们将分别详细描述这些基本概念。

1.2.1　实体和属性

在《简明牛津英语字典》中，实体一般是这样定义的：“实体”是相对于不存在的存在，存在由事物的性质或关系来区别。在数据库应用中，实体是指可以用存储的消息来描述的事物，它独立存在并且能唯一标识。实体可以是一栋房子、一个学生或一辆汽车等对象，或者是一场足球赛、一个假日或维修汽车等事件或活动。

一个实体由它的属性来描述其意义。例如，一栋房子可以由它的地址(ADDRESS)、风格和颜色等属性来描述；一辆汽车可以用车牌号(REGISTATION_NO)、式样、型号和注册年月日等属性来描述；一场足球比赛可以用主队、客队、日期、主队进球数和客队进球数来描述。如果一个属性本身就有描述信息，那么就应该将其划归成实体。例如，如果我们想记录一辆汽车型号的其他信息(除了型号名)，那么我们就要引进表示汽车型号的实体类型 MODEL，它与实体类型 CAR 存在着一定的联系。

我们可以用一个实体的名字与它的属性一起定义实体类型。实体类型可以有许多实例，实体类型与实例之间的区别恰好类似于编程语言中数据类型与它的值之间的区别。实体类型

的一个实例是这个类型的指定属性的实际值。例如，属性值(18 主街，独立式，蓝色)定义实体类型 HOUSE 的一个实例——一栋独立的房子、漆成蓝色并位于 18 主街。属性值(埃弗顿，利物浦，18/11/87，2，1)定义了实体类型 FOOTBALL_MATCH——埃弗顿队和利物浦队于 1987 年 11 月 18 日在埃弗顿主场踢的一场比赛，主队以 2:1 获胜的实例。

属性或属性集的值如果唯一地标识实体类型的一个实例，则称它为这个实体类型的候选关键字。例如，属性 ADDRESS 是实体类型 HOUSE 的一个关键字； REGISTATION_NO 是实体类型 CAR 的一个关键字。既然我们把一个实体定义为可以独立存在的事物，那么这个实体的关键字就必定永远存在。然而，一个实体类型可以有多于一个的候选关键字，例如，学生可以用一个唯一的标识符（学生证号）来识别，也可以用他的名字和地址的联合来识别。所以在众多的候选关键字中能够选出一个主关键字是非常有用的，主关键字在我们把实体-联系模型付诸实施时将扮演重要的角色。如有可能，我们将力图避免实体中的复合主关键字，即包括数个属性的关键字，如上面提到的学生名字和地址的联合。

1.2.2　联系

联系是两个或多个实体类型之间被命名的结合。例如， PLAYS_FOR 是实体类型 PLAYER 和 TEAM 之间的联系； CITIZEN_OF 是实体类型 PERSON 和 COUNTRY 之间的联系。

联系可以是 1 对 1(1:1)、 1 对多((1:N)和多对多(M:N)。例如，一个公司数据库可能会包括以下几种联系。

1. 1:1 的联系。 HEAD_OF 是实体类型 MANAGER 和 DEPARTMENT 之间的联系，它是 1:1 的。这意味着一个部门最多只能有一个负责人，而且一个经理领导一个部门。

2. 1:N 的联系。 SUPERVISES 是实体类型 MANAGER 和 EMPLOYEE 之间的联系，它是 1:N 的。也就是说，一个经理可以管理多个雇员，但一个雇员至多只受一个经理管理。

3. M:N 的联系。 ASSIGNED_TO 是实体类型 EMPLOYEE 和 PROJECT 之间的联系，它是 M:N 的。即一个雇员可以分派到不同的项目中，并且每一个项目可以有多个雇员参加。

这些联系可以有自己的属性，例如，实体类型 EMPLOYEE 和 PROJECT 之间联系 ASSIGNED_TO 可能有属性 DATE(分派日期)和 ROLE(参加项目雇员扮演的角色)。这些属性不仅仅只与实体类型 EMPLOYEE 相联系，也不仅仅只与实体类型 PROJECT 相联系，实际上，他们是通过联系 ASSIGNED_TO 把每个“雇员-项目”对的特性联系起来。因此，如果我们说雇员 John Smith 于 1989 年 10 月 15 日被委任为宣传局长，为了使这个声明有完整的意义，我们必须标识与此值有联系的“项目”（ PROJECT ），如 DATE （声明日期）、 DISCLAIMER （声明人）等。关于被分配的每一个“项目”，一个“雇员”通常对于这些属性的值会有一个不同的值集合。因此，这些特性最好被看作是联系的属性。

下面我们谈谈成员级别的问题。

如果一个联系的语义规定实体类型的每一个实例必须参与到这个联系中，那么实体类型的成员级别在那个联系中是强制的；否则，成员级别是可选的(非强制的)。

例如，假设在某公司的数据库中，实体类型 REPORT 与实体类型 PROJECT 之间存在着 1:N 的联系 PUBLISHES 。那么，每个项目可以发表许多报告，但是每个报告要准确地联系到一个指定的项目上。 REPORT 在联系 PUBLISHES 中的成员级别是强制的，因为每一个报告必须与一个项目相联。然而 PROJECT 的成员级别在 PUBLISHES 联系中是可选的，因为一个项目可以不

发表任何报告。

决定一个实体类型的成员级别在某些联系中是强制的还是可选的，有时可由数据模型设计者自己来处理。例如，我们再考察一下实体类型 MANAGER 和 EMPLOYEE 之间的 1:N 的联系 SUPERVISES 。如果我们希望执行“每一个雇员必须有一个经理”的规则，那么在联系 SUPERVISES 中 EMPLOYEE 的成员级别是强制的。但是，如果我们允许一个雇员可以不受雇于任何一个经理的情况存在，那么 EMPLOYEE 的成员级别在联系 SUPERVISES 中就是可选的。在本章的后面我们将看到，实体类型的成员级别在一个联系中可以影响实现联系的方法。如果一个实体类型在某些联系中是强制的成员，那么实现时将强调完整性约束，不可能有实体类型的实例不参加联系而存在于数据库中。

1.2.3　EER 模型的图解描述

图解实体-联系模型使用简图来描绘数据的自然结构。在简图中，用矩形来描述实体类型，用菱形来描述联系。联系由弧线联接到他们的构成要素（实体类型），联系的度(级)在弧线上表示。完整的 EER 模型还包括每一个实体类型和联系的属性表，它们有时也包括在简图中，但在本书中，为清楚起见，这些属性将被单独列出。

例如，某公司数据库的 EER 模型包括实体类型： DEPARTMENT 、 PROJECT 、 EMPLOYEE 和 REPORT 。这些实体之间的联系如图 1.1 所示。

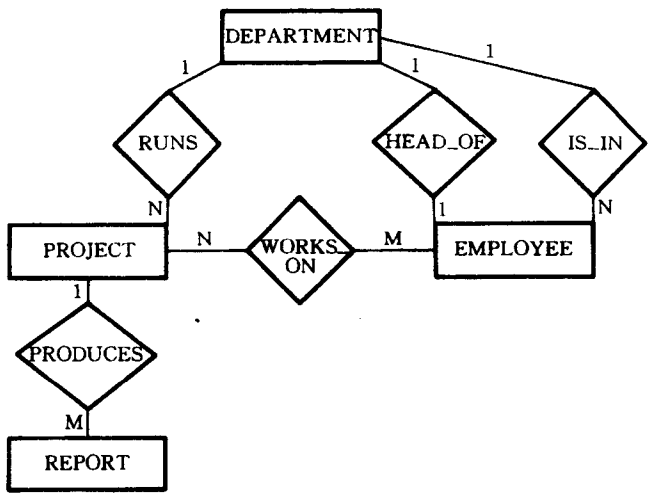

图 1.1　某公司数据库的实体-联系简图

从图 1.1 可以看出，每一个部门可以有多个雇员，但是只能有一个经理。一个雇员只属于一个部门。一个雇员可以工作在由不同部门管理的许多不同项目中。每一个报告与一个特定的项目相联系，并且一个项目可以生成多个报告。这种场合的实体类型(主关键字属性用下划线注示)如下：

1. 实体类型 DEPARTMENT 有属性 DNAME(唯一的部门名)、 LOCATION 等。

2. 实体类型 PROJECT 有属性 P#(该公司内部唯一的项目代码)、 TITLE 、 BUDGET 、 START_DATE 、 END_DATE 等。

3. 实体类型 EMPLOYEE 有属性 EMP#(唯一的雇员代码)、 ENAME 、 ADDRESS 、 SEX 等。

4. 实体类型 REPROT 有属性 R#(唯一的报告代码)、 TITLE 、 DATA 等。

实体类型之间的联系如下：

1. HEAD_OF 是实体类型 EMPLOYEE 和 DEPARTMENT 之间的 1 对 1 的联系。这个联系的成员级别对 EMPLOYEE 是任选的(比如一个雇员可以是一个部门的负责人，也可以不是)，但是它对 DEPARTMENT 是强制的(如果我们希望每个部门必须有一个负责人)。

2. IS_IN 是实体类型 EMPLOYEE 和 DEPARTMENT 之间的 N 对 1 的联系。这个联系的成员级别对 EMPLOYEE 是强制的，也就是说，一个雇员必须隶属于一个部门。

3. RUNS 是实体类型 DEPARTMENT 和 PROJECT 之间的 1 对 N 的联系。这个联系的成员级别对 PROJECT 是强制的，因为每一个项目须由一个部门来实施。

4. WORKS_ON 是实体类型 EMPLOYEE 和 PROJECT 之间的 M 对 N 的联系。正如本章前面所说，这个联系可以有 DATE 和 ROLE 等属性，它只能与雇员和项目对相联而不能与他们两者之一单独相联。

5. PRODUCES 是 PROJECT 和 REPORT 之间的 1 对 1 的联系。如果我们要求每个报告书必须有相关的项目，这个联系的成员级别对 REPORT 是强制的。

1.2.4 复杂的联系

现实世界中，实体类型之间的联系通常不是简单的 1:1 、 1:N 或 M:N 的，而是非常复杂的。例如，我们可能有同类型实体间的联系或包括两个以上实体类型间的联系。下面我们讨论一些更复杂的联系。

复杂联系是同一实体类型不同实例之间的联系。这些联系也可以是 1:1 、 1:N 或 M:N 的，下面用例子来说明。

例 1　一个 1:1 的复杂联系。

实体类型 PERSON 的一个实例可以通过联系 MARRY 与另一个成员(实例)相关联。如果我们假设一个人至多只能和另一个人结婚，且不计以往的婚姻史，那么这个联系由图 1.2 所示，它属于 1:1 的复杂联系。

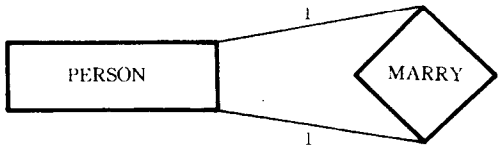

图 1.2　一个 1:1 的复杂联系

在这个联系中， PERSON 的成员级别显然是任选的，因为一个人可以结婚也可以不结婚。

例 2　一个 1:N 的复杂联系。

实体类型 EMPLOYEE 的一个实例可以管理其他实例。如果我们假设一个给定的雇员至多只有一个主管，那么我们就得到了一个 1:N 的复杂联系 SUPERVISES ，如图 1.3 所示。