北京希望电脑公司

面向对象数据库原理与应用

李东升　曾祥衡
刘映杰　马新枫　编写
费　丽　审校

面向对象数据库原理与应用

李东升　曾祥衡
刘映杰　马新枫　编写

费　丽　审校

北京希望电脑公司

内 容 摘 要

本书详细介绍了面向对象数据库原理,应用实例以及程序设计语言. 全书共九章,依次介绍了语义数据建模技术;面向对象程序设计基本原理;面向对象数据库建模技术;面向对象方法核心的类结构和继承性问题;面向对象的查询处理功能;数据库系统的持久性问题;面向对象数据库的并发, 恢复和分布相关技术;面向对象数据库技术的应用;面向对象的知识库.

本书适合于高年级的大学生、研究生、研究人员、计算机专业人员及所有有志于拓宽面向对象程序设计语言的程序设计及关系数据库设计和实现知识的人员使用.

需要本书的用户可直接与北京 8721 信箱联系

邮码 100080, 电话 2562329

(京)新登字 087 号

责任编辑: 刘莉蕾

* * * * *

面向对象数据库原理与应用

李东升 曾祥衡
刘映杰 马新枫 编写

费 丽 审校

* * * * *

京准印字: 3586—91586

内部成本价: 10.00 元

前　言

在商业界，关系数据库系统已成为数据处理应用的实际标准．其成功之处在于使用灵活、方便以及目前存在一些适用商用的关系数据库管理系统，其功能齐全，效率极高．在过去的几年中，这些系统成功地提供了大规模事务处理环境所需要的性能标准．然而，人们也广泛认识到，还存在大量应用，对这些应用而言，关系系统的数据模型功能非常有限．这些应用可以称为复杂的、大规模的、数据密集的程序，正如在计算机机辅助设计和计算机集成制造领域中发现的一样．目前在研究面向对象的数据系统以满足这些应用的复杂数据建模的需要．这些数据库系统不再建造关系模型，而把精力转向开发数据库框架中的一些模仿，如类型、数据抽取、继承性及一致性等．但是，这些数据库系统引起了一些严重的实现难题，尤其是关于一致性和并发控制方法．目前已引起学术界的高度重视．为了调查研究其中一些问题，在过去的几年中设计和实现了许多面向对象的数据库程序设计语言和系统．

当然，面向对象的程序设计语言已问世多年了．在 1966 年首次出现了由 Dslo 大学的 Dahl 和 Nygaard 设计的 Simula 语言．通常认为 Simula 语言是第一种面向对象的程序设计语言．其实 Simula 语言只是对 Algol 60 语言做了面向对象的扩展．在七十年代出现了几种语言．(如，Pascal+，并行 Pascal，Ada 等)．这些语言深受 Simula 的影响，但却没有保持 Simuula 的一些基本面向对象特性，尤其是继承性．同样于七十年代在 Xevo 的 Kay，Goldberg 和 Ingalls 也研制出 Smalltalk 语言．虽然 Smalltalk 也受 Simula 的影响，但其基本原理却完全不同，Smalltake 主要强调自由的无类型风格和动态联编．

程序设计语言对象和数据库对象比较相似．两者都只有封装特性，但它们之间也存在着几个主要区别．首先，当创建数据库对象的程序结束之后，该对象还须保持．其次，许多数据库应用要求能够创建和访问一个对象的多种版本．(该情况出现在历史数据库，软件管理数据库以及计算机辅助设计中)．再次，频繁使用的数据库，象用于航空交通控制和电力分布管理的数据库，要求能够把对象与条件、动作联系起来，当条件满足要求时就产生动作．最后，数据库完整性控制要求能够把对象和限制联系起来．一般来说，程序设计不支持一致性对象和多种对象版本，也不提供工具支持把对象限制和触发器关联起来．

然而，学术界存在一种很强的趋势，倾向于在数据库方面扩充面向对象的语言及用面向对象特性扩充数据库系统．一方面，面向对象的数据库系统必须提供一套齐全的语义数据模型模仿，用于为复杂的现实应用建立实体和关系模型．另一方面，应用程序设计者即使在不确定的虚拟存储中也必须能够访问和操作对象．

本书主要目的在于研究面向对象框架中数据库和程序设计语言，及这两方面取得的进展．第一章综合说明了语义数据建模技术，起始说明独立于模型的扩展实体关系方法，然后，详细描述了关系数据模型和扩展关系模型以及 RM／T 和函数数据模型．第二章专门论述了面向对象程序设计基础的基本原理，并处理了几个重要的问题，如软件质量，正确

性和复用性以及用于面向对象程序设计中支持这些目标的方法．第三章把面向对象的数据建模介绍给读者，在该模型中数据库采取了一批复杂的、高度耦合的对象形式．本章引入几个实例分析，用于说明传统关系方法和面向对象方法之间的区别．第四章更深层次地讨论了作为面向对象方法核心的类结构和继承性问题，从广泛的面向对象程序设计语言和系统中抽取一些类描述和继承性的例子．

第五章论述了面向对象的查询处理功能．对于数据库系统来说，这方面还不太成熟，在过去的几个中引起了一些争论，本章只是基于对 SQL 和功能数据模型的扩展来描述面向对象查询语言的一些有意义的方法．第六章致力于讨论持久性，持久性是数据库系统的一个基本属性，并说明了各种不同的持久性模型，范围的对象关系以及把持久性作为所有数据的正交特性的数据库程序设计语言并深入讨论了何种模型最适合于面向对象的数据库系统。

第七章描述了与面向对象数据库的并发，恢复和分布相关的问题．这一章详细描述了传统数据库的并发问题，重点在于串行性，理论及两相同步．接下来讨论了并发的各种模型（用于程序设计语言和数据库）以及面向对象数据库的相关性．一般来说，传统数据库并发机制过分限制了面向对象的环境．随后，研究一些与面向对象数据库的并发性有关的特例，并提出解决方法．深入研究的主题仍是分布式数据库中面向对象范例的作用．另外，还将讨论分布式方案中面向对象方法的一些优缺点．

第八章处理实现问题．当然，在学界面向对象数据库所面临的最富有挑战性的问题之一是如何高效实现面向对象数据库，这一难题很可能要持续很多年．本章先描述一些方法，在面向对象系统中用这些方法进行对象存储和存储管理，然后讨论其他与数据库有关的主要实现问题，如分类及版本控制等．

第九章转向面向对象的知识库．这一领域仍是很多研究项目的主题，而面向对象模型的作用还不太明确．本章主要考虑知识表示，这较好体现出面向对象模型的优势．另外还向读者介绍当前研究的一些方面及仍要解决的一些问题．

本书适合于高年级的大学生、研究生、研究人员、计算机专业人员及所有有志于拓宽面向对象程序设计语言的程序设计及关系数据库设计和实现知识的人员使用．由于编者水平有限，书中不免有不当之处，望读者批评指正．

编者

1992 年于北京

目　录

第一章　语义数据建模

1.1　引言

在过去的十年里，数据库系统，无论从数量，还是从存储信息容量及开发应用项目的复杂程度来看，都取得了飞速发展。这就要求系统的复杂度和精度不断增加，刺激了需求，需要更高层次的概念、工具和技巧。以便于数据库的设计和开发。于是，在近几年研究出的数据库的设计方法理论，在设计人员开始进行详细设计逻辑数据库的实际数据库之前，提供给设计人员一些方法，用于在高层次抽象中构造模型。这些方法学深受程序设计语言进展的影响。这些进展已开发出选项的抽象机制，允许独立于实现之外详细说明数据类型和对象。

本章为数据库应用提供了语义数据建模的基本原理。数据模型是数据库技术的关键。数据模型提供一个基本原则，以使数据密集型应用理论化。模型还提供一个规范标准，以便开发语言和系统中实现这些应用。实质上数据建模可分两个阶段，包括:

1. 初步设计阶段，包括设计一个初级模式，该模式是从现实状态中抽象而来的。
2. 设计一个逻辑数据结构，表示该模式。该数据结构也许会映射到实际的实现方案中。

第一步是分析应用的信息需要，准备需求说明书，并由此构造一个高层次的数据模型。根据逻辑上组织数据这一原则，数据模型可以定义为模板。由命名的数据逻辑单位组成，说明数据之间的关系。这些关系由一些对观察的解释而决定。数据模型必须能获取基础应用的静态特性和动态特性。系统的静态特性是关于抽取有关对象、概念、属性及其相互关系。在模式中使用数据描述语言的一些形式来定义这些特性。系统的动态特性是关于处理和查询所需对象和特性的操作。也就是说，动态特性构造系统的行为，并采取说明书形式以操作数据库。另外，数据模型必须能处理系统的动态特性。一旦应用需要随时改变时，即对系统修改。

数据模型的另一个重要功能是指定完整性规则，数据库对象（即:对象状态不能违反完整性规则）必须遵守，对象的操作也须遵守。完整性规则或限制通常与静态和动态特性相关（即:防止一次修改而产生对象的无效状态）。语义完整性限制高度依赖于数据模型。系统的数据模型（即分层模型、网络模型及关系模型）合并了较小的语义完整性，限制条件必须由用户暗示。在语义更丰富的数据模型中，限制条件对模型本身而言是内在的。完整性是评价数据模型的重要因素。

本章讨论了数据库设计的方法学，侧重于初步语义数据建模。文献中提出了许多语义数据模型（参看 Hull 和 King, 1987; Tsichritzis 和 Zochorsky, 1982; Bradie etal. 1984）。但是很少在数据库研究界之外引起人们的兴趣。这也许是由于许多

提出的模型过于复杂，难以在实际中实现。然而，扩展实体关系模型（EER）却足以表示语义数据建模中的大多数重要概念。与其他模型不同，EER 模型在商业应用环境中引起极大兴趣，对用于计算机辅助软件工程（CASE）的许多工具起着重要作用。这一模型起初由 Chen[Chen，1976]提出的，这一模型是供大多数传统的数据处理应用者使用的。然而，在近十年内，数据库设计面临着应用项目的复杂度不断增长，而这将需要增加语义建模概念。这样，Chen 的建议得到修改并在语义上由其他人加以丰富，主要添加于原始模型的附件包括子类（subchass）和超类（superclass）概念，以及密切相关的机制，称之为属性继承。

由于易于使用和表达，EER 模型在近十年来引起工业界和学术界极大的观注，并仍然作为初步模式设计的主要模型。受欢迎的主要原因是 EER 模型使用了数据抽象的概念，为数据库设计提供了自上而下的方法。从这些方面来看，EER 模型与软件工程的现代理论相一致。在初步设计时使用 EER 模型，并研究 EER 模型可实现模型之间数据转换。这就形成了现代数据库管理系统的理论基础。在后一章可看出 EER 模型为面向对象的数据库设计提供有效的第一步。

在本章第二部分提出了关系数据模型的基本原理。由于其易于使用，便于高效实现，八十年代关系模型成为商业数据库管理系统的实际标准，并由此提供一个重要的标准，用来检测更高级的模型（如面向对象的模型）。在考察关系模型时，主要关心象数据建模，数据操作和完整性这样的问题。一旦把关系模型和面向对象模型做一个比较，就会发现这些特性是至关重要的问题。

在本章末尾简要介绍两个其他语义数据模型:用于扩展关系模型以取得更多语义的 RM／T 模型[Codd，1979]，和函数数据模型。与函数性程序设计一样，在获取语义及确信正确方面，函数数据模型比其他模型具有更多的内在优点，但是，对是否能有效处理大规模应用数据库还有待于证实。

1.2 扩充实体关系模型

在扩充实体关系模型（EER）中，信息由三种原语概念来表示:

1. 实体　表示正建模的对象。
2. 属性　表示这些对象的特性。
3. 关系　表示实体间的关联。

下面几节中要详细描述各个概念。

1.2.1 实体和属性

在字典中实体的定义如下"即存在，作为不存在的反义词，与事物的质量和关系截然不同"。而对数据库应用来说，实体即要存储的描述性信息，它能独立存在和唯一标识。实体可以是一个对象，如一幢房屋，一个学生或一辆汽车等；实体也可以是一个事件或一项活动，如一场足球赛，一个假期或一辆汽车服务等。

从意义上来看，实体可用其属性来说明。比方说，一幢房屋可以用属性地址、式样和

颜色来描述，一辆汽车可以用属性车牌号，型号，式样和注册年份等来描述；一场比赛可由属性主队、客队、比赛日期、主队得分、客队得分等来描述。如果属性本身具有描述性信息，那么应把其归入实体类。例如:如果希望在汽车的式样上记录附加信息．那么就应当引入实体类型"Model"。"Model"与实体类型"Car"相关。

实体名称和属性共同定义了许多实例的实体类型。实体类型与实例之间的区别类似于程序设计语言中的数据类型和应用实例之间的区别。实体类型的一个实例是产生该实体类型．其属性的实际值已被指定．例如，属性值（18″大街，单独的房子蓝色）定义了实体类型 House 的一个实例．，即一幢单独房屋，刷成蓝色，在 18 号大街．属性值（Everton，Liverpool，18／11／87，2，1）定义了实体类型 FootBall__Match 的一个实例，即 1987 年 11 月 18 日进行一场足球赛，Everton 队对 liverpool 队，Everton 队为主队，并以 2:1 赢得这场比赛。

当一个或一组属性的值唯一确定一种实体类型的各个实例时，就称属性为该实体类型的"候选关键字"举例说明，属性"address"是实体类型"House"的关键字；Regfstvation "号"是实体类型"Car"的一个关键字．既然把实体定义为单独存在，则关键字必须经常存在。但是，一个实体类型可以只有一个以上候选关键字。例如:可通过唯一识别号（学生号码）来确认一个学生。也可通过其姓名和地址的组合值来识别。从候选要素中挑选一个主要的关键字，这非常有益。一旦开始把实体——关系模型映射到实现过程中，主关键字将起着重要作用。可能的话，总是努力避免为实体挑选复杂的主关键字，即一个关键字包含几种属性。

1.2.2 关系

关系是两个以上实体类型之间的有名称的关联。比方说:实体类型"Player"和"Team"之间的关系"Plays__For"，实体类型"Person"和"Country"之间的关系"Citizen__of"。

关系的功能度（functionality）可以是一对一（1:1），一对多（1:N），多对多(N:N)。例如，考虑一个可能存在的公司数据库，包含下列关系:

1:1　实体类型"Manager"和"Department"之间关系"Head__of"是1:1，这意味着一个部门最多有一个领导，而一个管理者至多领导一个部门。

1:N　实体类型"Manger"和"Employee"之间关系"Supervises"是1:N．这表示一个管理者可以管理任何数目不同的雇员，而一个给出的雇员至多由一个管理者来管理。

N:M　实体类型"Enployee"和"Project"之间的关系"Assigned__To"。这样可以让一个雇员从事许多不同的工作，而每项工作可以由许多雇员来完成。

一些关系具有一些特性，能够用关系的属性来表示。例如，实体类型"Employee"和"Project"之间的关系"Assigned__to"具有如"Date"（分配日期）和"Role"（雇员在项目中所起的作用）这样的属性。这些属性不可能只与实体类型"Employee"或"PRoject"相关联。它们是每一个雇员——项目对的特性。这些雇员——项目对是通过关系"Assigned__to"相关联的。这样的话，如果要说明在 1989 年 10 月 15 日委派雇员 Jone Smith，职位是"广告经理"，这样一件事，还须指定哪个项目，以便使说明只有完

整含义．一般来说，对于雇员的委派的各个项目的属性，他（她）具有一套不同的值．这样，最好认为这些特性为关系属性．

成员类

如果一个关系表示一个实体类型的各种实例必须具有这种关系，那么就说该实体类型的成员类在这种关系下是强制性的，否则，该成员类则是非强制性的．

例如，假定在某公司数据库中具有实体"Report"，而"Report"与实体"PRoject"存在N:1 的关系．"PuBlishes"也就是说，每个项目可以公布许多报告，而能给出的一个报告只与一个项目有关．因为每个报告必须与一个项目相关，可以在关系"PuBlishes"中，"Report"的成员类是强制性的，但是，在该关系中的"PRoject"的成员类是强制性的，但是，在该关系中的"PRoject"的成员类是非强制性的，因为一个项目可能不公布任何报告。

关于一个实体的成员类在某一关系中是强制性的，还是非强制性的，这需要数据模型设计者来决定．比方说，再一次考虑实体"Manager"和"Employee"之间的 1:N 的关系"Supervises"．如果希望增加一条规则，即每个雇员必须有一个管理者；那么"Employee"的成员类在关系"Supervises"中可以说是强制性的。然而，如果希望允许存在这样的情形，即可以存在不用管理者管理的雇员，那么就说"Employee"的成员类在关系"Supervises"中是非强制性的．

正如在本章后面所叙述的，在某关系中实体的成员类会影响实现关系的方式．如果一个实体是某关系的强制性成员，那么实现方案中应当增加一条完整性限制，即不具备这种关系的实体类型的实例在数据库中是不存在的。

1.2.3 ERR 模型的图解表示

图解式实体关系模型利用图表来描述数据的自然结构，在这些图表中，矩形代表实体类型，菱型代表关系，利用弧把关系连接到实体类型上，在弧线上标注关系的度数。

完整的 ERR 模型还包括对各实体类型和关系的属性的列表。这些有时绘在图表中，但是在本书中，为清楚起见，单独列出属性。

例如:用作小公司的数据库的 ERR 模型包括实体类型"Department"、"PRoject"、"Employee"和"Report"，可以如图 1.1 所示。每个部门拥有许多雇员，但只有一个管理者。一个雇员只属于一个部门。雇员可以从事许多不同项目，这些项目由不同的部门来管理。每个报告与一指定的项目关联，而一个项目可以产生许多报告。

这种情形的实体类型如下:（主要要素的属性之下划线）.

1. 实体类型DepartMent，带有属性DNAME（唯一的部门名），LoCation等等。
2. 实体类型PROJECT，带有属性P#（公司中唯一的项目号），TLTLE、BUDGET、START__DATE、END__DATE 等等。
3. 实体类型Employee，带有属性EMP#（唯一的雇员号）ENAME、ADDRESS、SEX 等等。
4. 实体类型REPORT，带有属性R#，（唯一的报告号），TITLE、DATE等等。

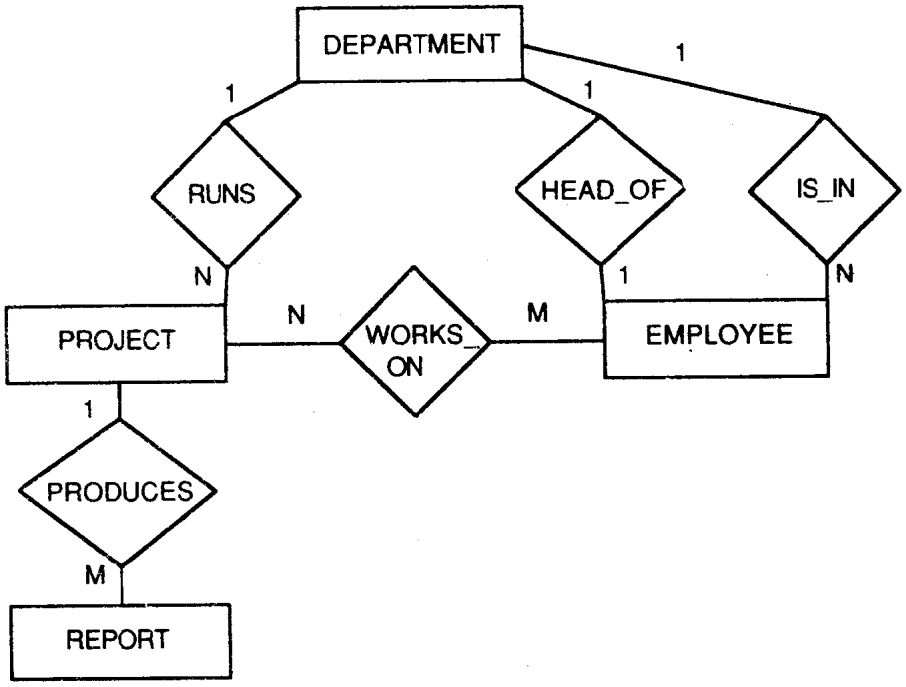

图 1.1　数据库的实体——关系图

实体类型之间关系如下:

1. 在"EMPLOYEE"和"DEPARTMENT"之间的关系Head__of是1:1的，这个关系的成员类对"EMPLOYEE"来说是非强制性的（即一个雇员也可以是部门的领导，也可以不是部门的领导)，但对"DEPARTMENT"来说却是强制性的（如果希望每个部门必须有一个领导的话)。
2. 在"EMPLOYEE"和"DEPARTMENT"之间的关系"IS__IN是N:1的，该关系的成员类对于"EMPLOYEE"来说是强制性的，也就是说，一个雇员必须属于一个部门。
3. 在实体类型"EMPLOYEE"和"PROJECT"之间的关系"RUNS"是1:N的，该关系的成员类对于"PROJECT"而言是强制性的，因为每个项目由一个部门来进行。
4. 在实体类型"EMPLOYEE"和"PROJZCT"之间的关系"Works__on"是N:M的，正如本章前面所描述的一样，该关系具有象 DATE（分配日期）和 ROLE 这样的关系。这些属性与雇员和项目相对关联，而不可以单独与雇员或者与项目相关联。
5. 在实体类型"PROJECT"和"REPORT"之间的关系"PRODUCES"是1:N的，该关系的成员类对于"REPORT"来说是强制性的，假如要求每个报告必须具有一个关联项目的话。

1.2.4　更多的复杂关系

现在只考虑简单的二元关系 1:1、1:N 和 N:M，但是现实情形常常包含实体之间更加

复杂的关系．例如，可以具有同一类实体间的关系，也可具有两者以上实体类型之间的关系．下一节将描述其中最主要的关系．

回旋关系（involuted relationship）

回旋关系是指在同一实体类型的不同实例之间的关系．这种关系如下述例子阐述的那样，可以是 1:1、!:N 或 N:M 的．

例 1:1:1 回旋关系

实体类型"PERSON"的一个实例通过关系 MARRY 可以与另一实例相关．如果做出简化性假设，假定一个人至多只可以与另一个人结婚，也就是忽略再婚，禁止一妻多夫制或一夫多妻制，则图 1.2 显示的关系为 1:1 的回旋关系．

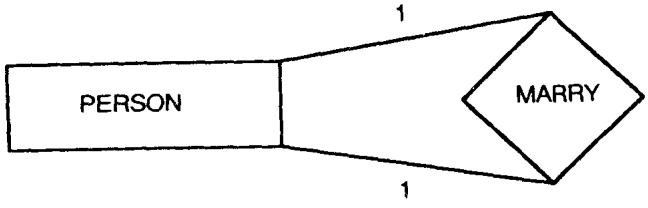

图 1.2　1:1 的回旋关系

在这种关系中"person"的成员类显然是非强制性的．即一个人可以结婚，也可以独身．

例 2: 1:N 的回旋关系

实体类型 EMPLOYEE 的一个实例可以管理其它实例．如果假定所给出的雇员至多可以有一个管理人员的话，那么就拥有了 1:N 的回旋关系"SUPERVISES，如图 1.3 所示．

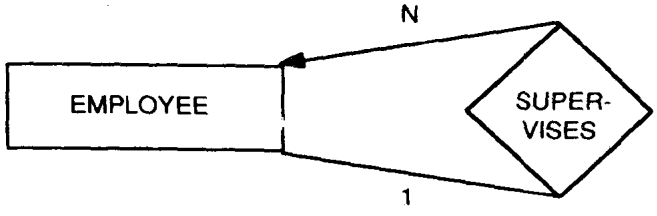

图 1.3　1:N 的回旋关系

(图中箭头必须指示关系的方向，即一个雇员管理许多其他的雇员而不是许多雇员来管理一个)，由于并非所有雇员都有一个管理者，故实体类型"EMPLOYEE"的成员类在该关系中是非强制性的．然而，如果多数雇员都拥有一个管理者的话，那么成员差不多是强制性的．在本章后面将重新讨论一问题．

例 3: N:M 的回旋关系

实体类型 PART 的一个实例可以由其他部分组成，而一个给出的部分也可以是许多部分的成员之一．图 1.4 所示的 N:M 回旋关系描述了这种情况．

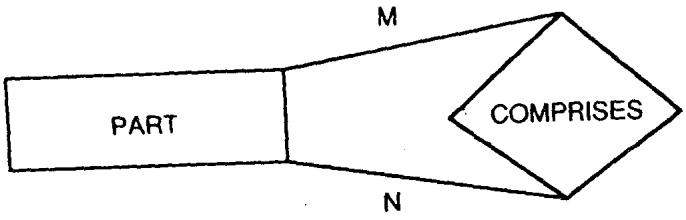

图 1.4　N:M 的回旋关系

三元关系

关系可以包括两个以上的实体类型．这种包含三个实体的关系并非不常见．例如，考虑图 1.5 所示的数据库，它要保持公司的信息，生产的产品信息以及出口产品所往的国家信息．

产品出口所往的国家集合随着产品的不同而不同，也随着公司的不同而不同．关系"SELLS"是三元关系，即它包含三元实体类型．图 1.5 所示的三元关系"SELLS"，其功能度为（N:M:P），这反映了下面关于关系的实事．

对于一个给出的对（公司、产品），一般来说存在许多产品销售图．对于一个给出的对（国家、产品）也可以存在几家公司，向该国家出口该产品．而对于一个给出的对（公司、国家），也存在许多由该公司向该国出口的产品．

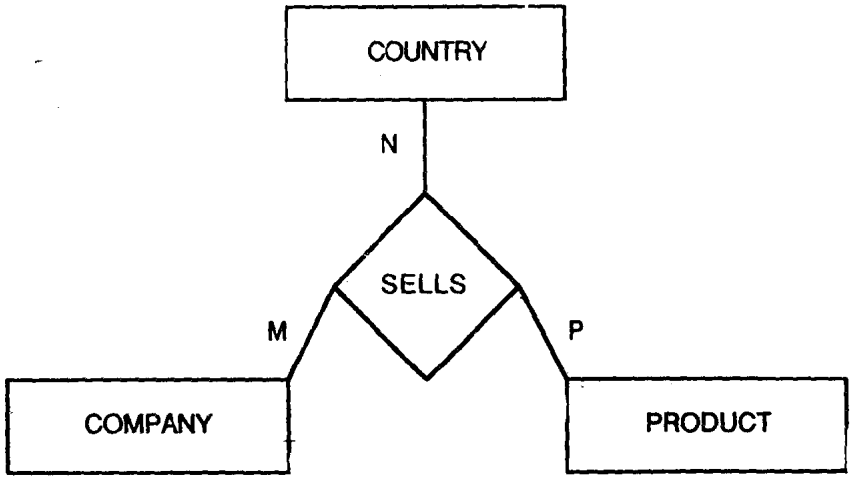

图 1.5　三元关系

三元关系的功能度可能是"一对多再对多"，"一对一再对多"或甚至是"一对一再对一"，必须仔细考察关系的语义以决定其功能度．在本章后面考虑关系模型的三元关系的表示时会重新研究这一问题．

只有当关系不能由有关的实体类型之间的几个二元关系确切表达出来时，才必须仔细定义三重关系并应加以介绍．否则的话，会如后面所述，在最终的数据库中可能出现不必要的冗余．例如:如果一个公司制造了许多产品，并把这些产品出口到几个不同国家，

那么如图 1.6 所示，应定义两个独立的二元关系 Exports 和 MATES 以取代三元关系。在这种情况下，必须注意到公司出口的国家集合独立于产品，并由数据模型反映出这一事实。在本章后面讨论概念"多值依赖"时还会重新加以讨论。

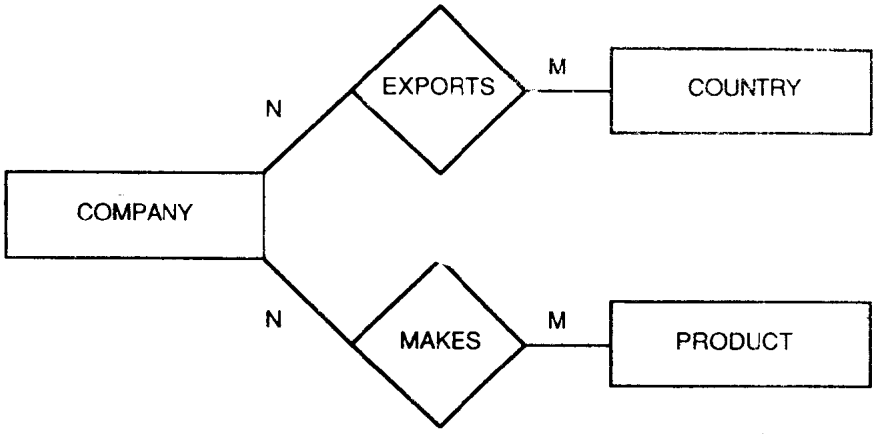

图 1.6　二元 N:M 的关系

子类型

由 Chen 提出的原始实体-关系模型存在一个弱点，该模型缺乏子类型概念以及对这种子类型的归纳概念。如果实体类型 E1 的每个实例同样也是实体类型 E2 的实例，那么 E1 即为 E2 的子类型。如果实体类型的每次出现同样也是实体类型 E1，E2，...，En 之一出现，那么 E 即为 E1，E2，...En 的概括。Smith[1977]和 Smith 在他们的语义模型中引入术语"归纳"，这为抽象数据模型中的子类型提供了有效基础。尤其在实体-关系模型[Seheuerman 等，1980；Navathe 和 Cheng，1983；SaKai 1983]方面，这些想法由其他人加以扩充。

作为子类型的一个例子，考虑上面描述的小公司数据库，把部门领导表示成管理人员，即雇员的特殊类更为恰当。同样，存在实体类 EMPLOYEE 的分类，象 SECRETARY、TECHICIAN 和 ENGINEER 等，希望能加以区分。由于这些实体集都可以被认为是实体类型 EMPLOYEE 的不同种类，因而各实体集都分享一些特性。其实，这些实体类型是 EMPLOYEE 的子类型，而 EMPLOYEE 可以说是这些实体类型的超类型。注意，如果一个子类型不是一个超类型的成员，那么该子类型的实例在数据库中不可能存在。也就是说，该子类型的一个实例所表示的真实实体与超类型的某一实例所表示的一样。实现阶段的任务之一就是保证这条语义规则被执行。但是，要注意到，没有必要使超类型的每个成员都成为其子类型之一的成员。同样，子类型不必相互排斥。

子类型自身可以有子类型，这导致了类型分层结构，例如，在一个航空工业公司中，可以把子类型 ENGINEER 分成三种不同的实体类型：Auto__ ENGINEER、AERO__ENGINEER、ELECTRONIC__ENGINEER。这将导致图 1.7 所示的类型系的产生。子类型和父类型之间的关系由一种被称为 IS__A 关系的特殊 1:1 关系来表示。

子类型共享其超类型的所有属性以及超类型的部分关系，而没有必要共享全部关系。作为子类型成员的实体可以认为是继承了其超类型的属性，一个子类型也可以拥有附加指定的属性和关系。比方说，在公司数据库中希望实施一条规则，即只有实体类型 MANAGER（EMPLOYEE 的子类型）的实例方可分享关系 HEAD__OF，在这种情况下，可如图 1.8 所示在实体类型 MANAGER 和 DEPARTMENJ 之间定义这种关系。管理人员享受一个雇员的所有属性，但可以拥有附加属性，只与管理人员有关。

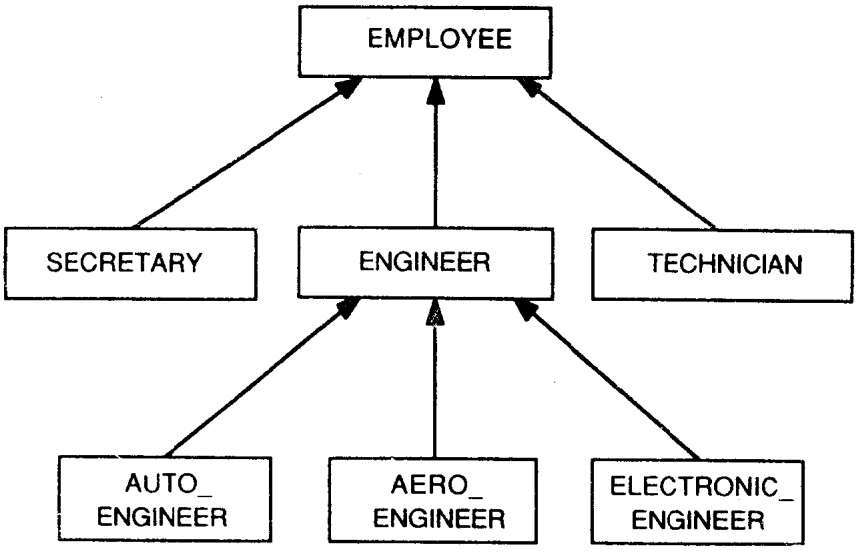

图 1.7　类型分层结构

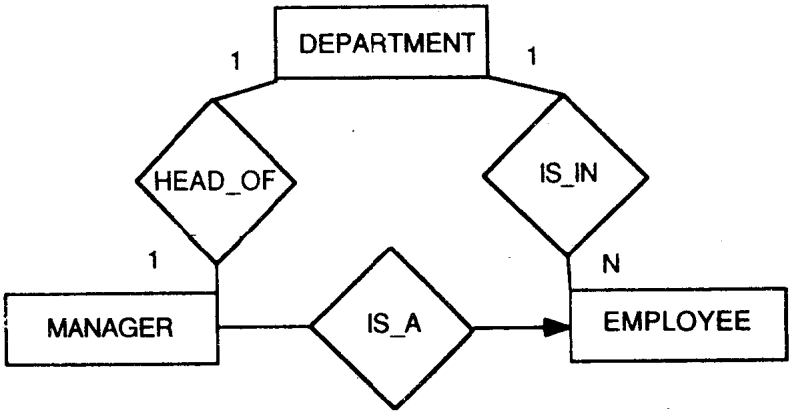

图 1.8　作为 EMPLOYEE 子类型的 MANAGER

由于子类型的内容使图解增加清晰度和准确度，并能提高实现的效率，因而一旦子类型产生，就认为是比较有益的。一旦对正在建模的工程存在许多不同的用户观点时，子类

型常常会出现，数据建模人员在构造全局数据模型时识别归纳和类型分层结构也是很重要的。

归纳和限定 (generalyrefin and specifigation)

从另一个观点来看，如果数据库中 EMPLOYEE 的每个实例也是这些子类型其中之一的实例，那么可以认为实体类型 EMPLOYEE 归纳了实体类型 SECRETARY、ENGINEER 和 TECHNICIAN。在这种情况下，实体类型 SECRETARY、ENGINEER 和 TECHNICIAN 形成了对实体类型 EMPLOYEE 的限定，而每种限定都由属性值加以区分。这样区分属性是 JOB__TITLE。在同一实体集中可定义各种限定，每种限定由不同的特性加以区分。例如，实体集 EMPLOYEE 的另一种限定可以是子类型 FIXED__TERM 和 LONG__TERM，其中雇员由属性 TYPZ__OF__CONTRACT 加以区分。然而另一种限定也可以是子类型 WEEKLY__END 和 HOURLY__PAID，这些子类型根据属性 MODE__OF__PAYMENT 值在雇员中间区分。

1.2.5 映射到实现中

ERR 模型的一个极大优点在于其可被非专业化人员理解。数据库中实体数目与数据项总数相比，一般要小得多。这样通过使用实体概念抽象出那些与希望收集信息有关的现实对象，从而大大简化了需求分析阶段和概念设计阶段。用户可以简化和扩充 EER 语义模型，直到模型能确切表示正在建模的数据结构。从这方面考虑，EER 模型独立于任何特殊数据库管理系统。这些系统一般把限制强加给表达数据的方式。而作为更高级的 EER 模型，能更加确切地反映出数据的自然结构。与许多实现过程不一样，EER 模型具有灵活性和可扩充性，可适应于新开发。

有理由认为把 EER 模型转化成实现过程是很简单的。为了表达关系之间的实体和联接，各数据库管理系统即提供了一些机制，但是自然要做出一些折衷方案以便有效地预先设置。比方，网络数据管理系统常常把限制强加到能表达出的关系类型上。对于这样的系统，有必要把 N:M 关系和内旋关系分解为更加原始的结构，以有利于实现。

从另一方面看，尽管关系模型相对简单，但还能凭借两维表来获取许多 EER 模型的信息并表达出这些信息。这些表格或关系可以认为是对文件的数据处理概念的限制方式做一次抽象。

1.3 关系数据模型

关系数据模型是由 Codd 在七十年代发表的一系列论文 (CODD，1970，1972a，1972b，1974) 创建而成的。该模型基于关系和一阶逻辑的数学理论，具有完善的理论基础，但却用二维表形式把数据的简单视图提供给用户。该模型具有较强的理论基础，允许应用基于高级抽象的系统方法，来设计和管理关系数据库。这大大减少了数据建模人员和终端用户所面临的复杂性。

在本节开始要考察关系的标准定义，接着要研究应用于关系的属性和要素的概念。然后提出一个方法论，以便把 EER 模型转化为关系形式。