第3章 关系数据库设计

本章要点

- 了解数据库的设计方法以及设计步骤。
- 掌握 ER 图的绘制方法。
- 掌握数据库概念结构的设计方法。
- 掌握数据库逻辑结构的设计方法。

数据库设计(Database Design),狭义上是指利用选定的数据库管理系统,针对某个具体的应用领域,构造合适的数据库模式,建立基于这个数据库的信息系统或应用系统,以便有效地存储和检索数据,满足各类用户需求。随着数据库技术本身的不断发展和数据库更广泛的应用,数据库设计的任务已远远超过上述内容,它不仅包括设计数据库结构,也包括设计应用程序等内容。

1. 数据库设计的特点

数据库设计需要结合多门学科的知识和技术,工作量大而且比较复杂。大型数据库的设计是一项庞大的工程,其开发周期长、耗资大、失败风险高。数据库设计的很多阶段都可以和软件工程的各阶段对应起来,软件工程的某些方法和工具同样也适合于数据库设计。为了降低风险,可以将软件工程的原理和方法应用到数据库设计中。因此,数据库设计的人员应该具备多方面的知识和技术。

2. 数据库设计的方法

两个主要的数据库设计方法分别是“自底向上”的方法和“自顶向下”的方法。使用“自底向上”的设计方法首先需要识别出系统潜在实体型或联系型的属性,分析这些属性之间的联系并据此将这些属性分组,通过分组属性就可以识别出实体型和联系型。规范化理论常用在“自底向上”的设计过程中,通过分析属性之间的依赖关系,可以将这些属性按照一定规范分组从而形成规范化的关系。“自底向上”的设计方法适用于设计包含较少属性的小型数据库系统。

如果数据库系统规模非常大,可能包含成百上千的属性,分析这些属性之间的联系将变得非常困难。对于复杂的数据库系统,使用“自顶向下”的设计方法更为合适。这种方法首先识别系统中主要的实体型和联系型,然后通过进一步分析找出低层次的实体型、联系型和属性。ER 模型通常用在“自顶向下”的设计方法中。使用 ER 模型首先找出用户关心的实体型以及实体型之间的联系,然后再找出实体型和联系型的属性。

3. 数据库设计的步骤

目前设计数据库系统主要采用的是以逻辑数据库设计和物理数据库设计为核心的规范

设计方法。按照规范设计的方法,考虑数据库及其应用系统开发的全过程,可以将数据库设计过程可分为以下 6 个主要阶段,分别是:需求分析阶段、概念结构设计阶段、逻辑结构设计阶段、数据库物理设计阶段、数据库实施阶段、数据库运行维护阶段。设计一个完善的数据库应用系统往往是上述 6 个阶段不断反复的过程。


视频讲解

3.1 实体联系模型

在数据库设计过程中,各类人员(如数据库设计人员、程序开发人员、最终用户)看待数据的角度是不同的。然而,数据库设计人员如果与其他人员在如何操作数据的问题上不能获得共识,那么数据库的设计很可能由于不能满足用户需求而失败。因此,需要一种非技术的工具对数据进行描述,以使参与数据库设计的各类人员能够通过这个工具进行沟通。实体联系模型(Entity Relationship model,ER model)用 ER 图来抽象表示现实世界中客观事物及其联系的数据特征,是一种语义表达能力强、易于理解的概念数据模型。

ER 模型支持“自顶向下”的数据库设计方法。通常,在数据库的需求收集和分析阶段完成以后就可以使用 ER 模型对数据库进行概念结构设计,并利用 ER 模型与用户进行沟通,使设计思想能够满足用户的需求。下面以 P. P. S. Chen 于 1976 年提出的 ER 模型方法为例,介绍 ER 图的绘制方法。

3.1.1 实体联系模型的要素

ER 模型中包括 3 个主要的要素,分别是实体(Entity)、联系(Relationship)和属性(Attribute),首先来看一下实体的概念和表示方法。

1. 实体

现实世界中客观存在并可以相互区别的事物称为实体。实体可以是物理存在的,如一本书或一名学生,也可以是概念性的,如一次销售行为或一次面试等。实体概念的关键之处在于一个实体能够与另一个实体相互区别。例如一个班里有 30 名学生,即使这些学生中有重名的情况,任何一名学生也都能与其他学生区别开来(例如,每个学生都拥有一个唯一学号)。

实体型表示具有相同属性的同一类实体。实体型可以刻画出全部同质实体的共同特征和性质。例如学生具有共同的属性(学号、姓名、入学日期等),则这些属性构成一个“学生”实体型。

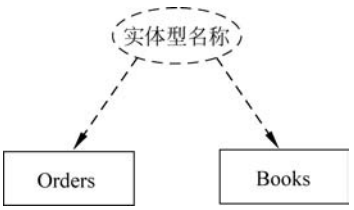

图 3-1 实体型的表示方法

同一类实体的集合称为实体集,例如某学校的全体学生就是一个“学生”实体集。实体型表示抽象的实体集,例如实体型“学生”表示全体学生的概念,并不具体指学生甲或学生乙等。在不引起混淆的情况下,可以将实体型简称为实体。

在 ER 图中,使用矩形框来表示实体型,框内标注实体型的名称。如图 3-1 所示,分别表示 Books(图书)实体型和 Orders(订单)实体型。

2. 联系

现实世界中，事物内部以及事物之间通常存在着一定的联系，这些联系在信息世界中反映为实体型内部以及不同实体型之间的联系。

实体型之间全部联系的抽象称为联系型。在 ER 图中，联系型用菱形框表示，框内标注联系型的名称，并用连线将菱形框分别与对应的实体型相连接。联系型的名称通常为动词。在不引起混淆的情况下，可以将联系型简称为联系。如图 3-2 所示，由于订单中包含图书，所以在“订单”实体型和“图书”实体型之间存在着“包含”的联系。

1）联系的元(Degree)

联系的元是指参与联系的实体型的个数。在图 3-2 中，有两个实体型参与 Include(包含)联系，所以称该联系为二元联系。有时，参与某个联系的实体型的个数可能更多。如图 3-3 所示，每名学生选修的每门课程都有一个授课教师，所以这个 Study(选修)联系是一个三元联系。

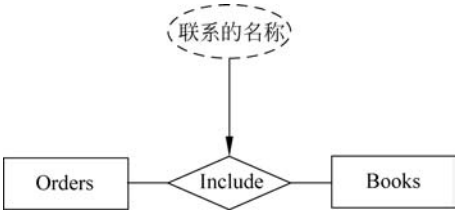

图 3-2　联系的表示方法

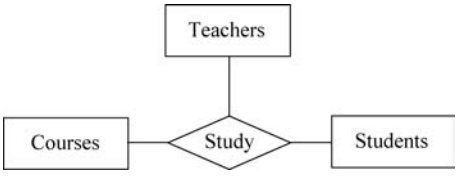

图 3-3　一个三元联系

2）递归联系

同一实体型中的实体以不同的角色参与到一个联系上，这个联系被称为递归联系。Teachers(教师)实体型中的实体可以按角色分为教研室主任和普通教师，教研室主任管理(Supervise)教师。如图 3-4 所示，“教师”实体型以不同的角色两次参与到“管理”联系上，这个联系被称为递归联系。

3）实体在联系上的参与度

如果实体型中的每个实体都参与到一个联系上，则使用双线将该实体型与联系连接起来，称为完全参与。如果实体型中的实体不是全部参与到联系上，则使用单线将该实体型与联系连接起来，称为部分参与。

在图 3-5 中，因为所有的“订单”中都包含“图书”，但是并不是所有的“图书”都包含在“订单”中，所以 Orders 完全参与联系 Include，而 Books 部分参与该联系。

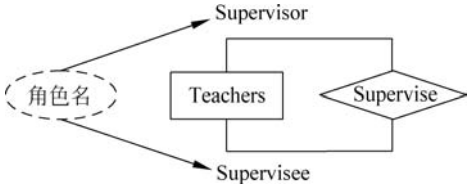

图 3-4　递归联系

图 3-5　完全参与和部分参与

4）基数映射

基数映射是建立在联系上的一种约束机制，表示某个实体型通过联系与另一个实体型中

的一个实体产生联系时,可能涉及该实体型中的实体个数。对于二元联系来说,两个实体型产生联系的类型可能是一对一联系(1∶1)、一对多联系(1∶n)或多对多联系(m∶n)三种情况。

(1) 一对一联系:设 A、B 为两个实体型,若 A 中的每个实体最多和 B 中的一个实体有联系,反之亦然,则称 A 与 B 之间是一对一联系,记作 1∶1。

图 3-6 是一个 1∶1 联系的例子。在 Locate(位于)联系中,一个 Bookstore(书店)只位于一个 City(城市),而且一个"城市"也只开办一家"书店"。

(2) 一对多联系:设 A、B 为两个实体型,若 A 中的每个实体可以和 B 中的多个实体有联系,而 B 中的每个实体最多和 A 中的一个实体有联系,则称 A 与 B 之间是一对多联系,记作 1∶n。

图 3-7 中的 Produce(生成)联系是一个 1∶n 联系,因为一位 Customer(顾客)可以生成多张"订单",但是一张"订单"只属于一位"顾客"。

图 3-6 1∶1 联系

图 3-7 1∶n 联系

在一对多联系中,"一"的一方被称为父实体型,"多"的一方被称为子实体型。在如图 3-7 所示的例子中,"顾客"称为父实体型,"订单"被称为子实体型。

然而,在前面所讲到的一对一联系中,如果一方是部分参与,另一方是完全参与,那么可以将部分参与的一方视为父实体型,完全参与的一方视为子实体型。在如图 3-6 所示的例子中,"城市"可以被视为父实体型,"书店"可以被视为子实体型。

(3) 多对多联系:设 A、B 为两个实体型,若 A 中的每个实体可以和 B 中的多个实体有联系,反之亦然,则称 A 与 B 之间是 m∶n 联系。

图 3-8 中的 Include 联系是一个 m∶n 联系,因为一张"订单"可以包含多本"图书",而且一本"图书"也可以包含在多张"订单"中。

5) 复杂联系的基数映射

二元联系基数映射的确定方法比较简单,如果参与一个联系的实体型的个数超过两个,那么确定其基数映射的过程稍微复杂一些。对于一个 n 元联系,可以首先确定 $n-1$ 个实体型中的一组具体实体,然后分析第 n 个实体型的参与情况。再按照这个方法依次确定每个实体型在联系上的参与情况。

如图 3-9 所示,Study(选修)是一个三元联系。通过分析发现每名学生选修的每门课程有且仅有一位教师教授;每位教师可以教授每名学生 0 门或多门课程;每名教师教授的每门课程都会面对多名学生。

图 3-8 m∶n 联系

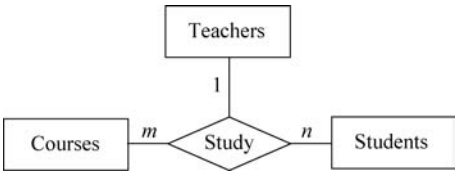

图 3-9 三元联系的基数映射

由于参与“选修”联系的实体型中存在多对多联系，可以认为这个三元联系的类型是多对多联系。

由于四元以及更复杂联系出现的可能性较小，而且其基数映射的分析方法与三元联系相同，所以这里不再赘述。

3. 属性

属性用来描述实体或联系的特性。例如，“图书”实体的属性包括“书号”“书名”“isbn”“单价”等。

在 ER 图中，用椭圆形表示属性，并用连线与实体型或联系型连接起来。如果属性较多，为使图形更加简明，有时也将属性另外单独用列表表示。图 3-10 是“图书”实体型的属性。

联系也可以有属性。例如图 3-11 所示的“选修”联系描述了学生选修教师的课程，在该联系上包含选修的学期、选修的成绩以及选修标志属性。如果将这些属性放在学生、课程或教师实体型上都无法正确地表达其含义。

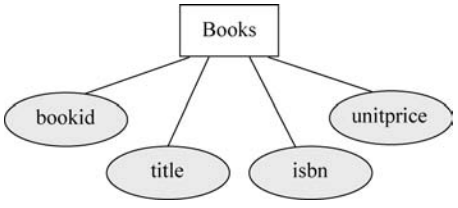

图 3-10　“图书”实体型的属性

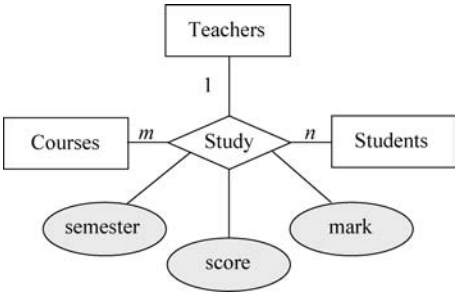

图 3-11　选修联系的属性

实体或联系的属性通常都有一个取值范围，这个取值范围称为该属性的域。例如学生的性别属性的域应该是一个只包含元素“男”或“女”的集合，选修的成绩属性的域应该是 0～100 的实数，成绩标志的域应该是包含元素“缺考”“缓考”“免考”“作弊”“通过”的集合等。

1）简单属性与复合属性

简单属性是原子的、不可再分的。复合属性可以细分为更小的部分。如图 3-12 所示，Customers（顾客）实体型的顾客编号、顾客姓名、邮编属性都是简单属性，而地址属性就是一个复合属性，因为地址还可以细分为城市、街道、门牌号等部分。

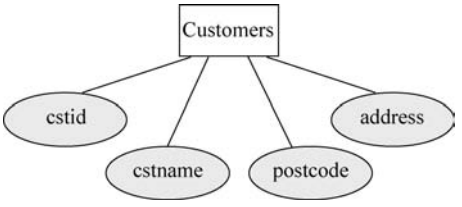

图 3-12　简单属性与复合属性

在设计 ER 图时，保持复合属性还是将复合属性分解为更小的属性取决于用户操作的需要。例如，在访问顾客的联系地址时，通常都使用完整的地址，这时就可以使用地址属性。如果还需要统计顾客所在城市的分布情况等信息，就可以选择将地址属性细分为城市、街道和门牌号三个属性。

2）单值属性和多值属性

如果某个属性对于实体型中的任意一个实体只有一个值，则该属性为单值属性；如果

某个属性对于实体型中的一个实体可能有多个值,则这个属性是多值属性。多值属性用双边椭圆形表示。

如图 3-13 所示,“顾客”实体型中的顾客编号、顾客姓名、邮编属性都是单值属性,因为每位顾客都只有一个顾客编号、一个姓名和一个邮编。而电话属性就是一个多值属性,因为一位顾客可以保留家庭电话、办公电话和移动电话等多个联系电话。

在进行数据库逻辑结构设计时,由于多值属性无法表达为基本的关系,所以需要进行特殊的处理,例如将多值属性转换为单值属性或将多值属性去除并通过单独的关系来表示。

3) 派生属性

如果某个属性的值可以由其他属性导出,则称该属性为派生属性。派生属性用虚边椭圆形表示。如图 3-14 所示,因为顾客的年龄可以由其出生日期导出,所以顾客的年龄属性就是一个派生属性。

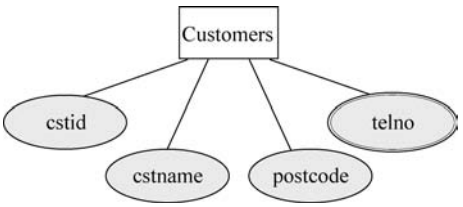

图 3-13 单值属性与多值属性

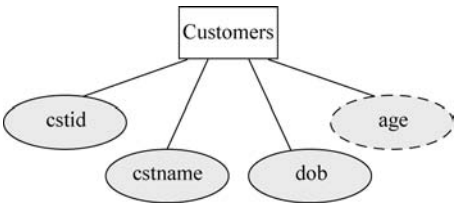

图 3-14 派生属性

某些情况下,某个实体型的派生属性的值可能需要通过其他实体的属性值计算得出。有些实体型的派生属性的值可能需要计算实体型本身或其他实体型中实体的个数来导出,例如如果“班级”实体型中有一个班级人数属性,这个属性的值可以通过计算“学生”实体型中学生实体的个数来得出,这个属性就是一个派生属性。

3.1.2 码

码(Key)是实体型中用来标识每一个具体实体的重要工具。下面将介绍候选码和主码的概念。

1. 候选码

候选码(Candidate Key,CK)是实体型中的属性或最小属性组,可以用来唯一标识实体型中的每个具体的实体。

例如在“图书”实体型中,因为每册图书都有一个唯一的书号,所以可以通过指定一个书号来确定唯一的一册图书,因此书号属性就是该实体型的候选码。这个例子说明候选码具有唯一标识性。

2. 主码

主码(Primary Key,PK)是从候选码中选出的,用来唯一标识实体型中的每一个实体的一个候选码。在 ER 图中,可以使用下画线来标识主码,如图 3-15 所示。

一个实体型中可能包含多个候选码,例如在“图书”实体型中,由于每本图书都有一个唯一的“书号”和唯一的 isbn 值,所以“书号”和 isbn 都是这个实体型的候选码。因为“书号”比 isbn 更简洁高效,所以可以选择“书号”作为“图书”实体型的主码。

3. 组合码

如果一个实体型中的两个或两个以上的属性共同组成实体型的候选码，则称该候选码为组合码。

假设有一个 Addresses(地址)实体型，该实体型中的每个实体表示一个联系地址，包括城市、街道和邮编 3 个属性。由于城市和街道两个属性组合的值能够唯一标识一个地址，所以城市和街道两个属性构成“地址”实体型的组合码，如图 3-16 所示。

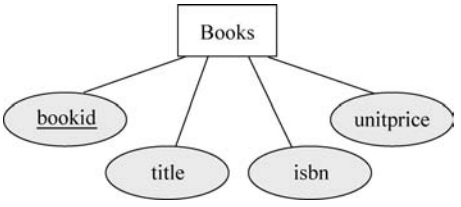

图 3-15　主码的表示方法

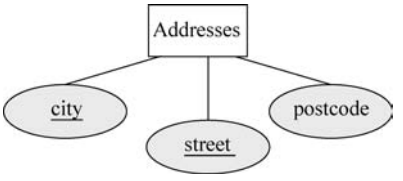

图 3-16　组合码

3.1.3　强实体型与弱实体型

1. 强实体型

如果一个实体型中实体的存在不依赖于其他实体型，则称该实体型为强实体型。强实体型的特性是它有候选码，实体型中的每个实体可以通过候选码被唯一标识。如图 3-17 所示，Courses(课程)实体型是一个强实体型，课程编号是其主码，每门课程都可以通过课程编号来唯一标识。

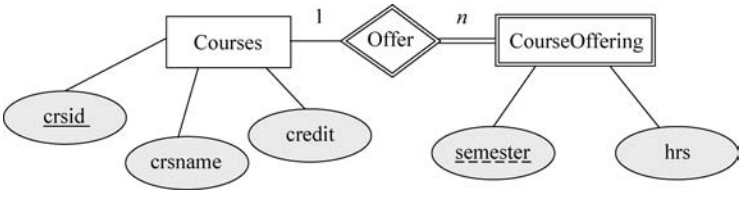

图 3-17　强实体与弱实体

2. 弱实体型

弱实体型是指其实体的存在必须依赖于其他实体型。弱实体型的特性是它没有候选码。例如有一门课程“数据库开发”，需要分别在第 3 学期完成 60 学时，在第 4 学期完成 80 学时，则可以创建一个“课程提供”实体型，该实体型包含学期和学时两个属性，如图 3-17 所示。CourseOffering(课程提供)实体型就是一个弱实体型，因为该实体型中的属性不能构成候选码，也就是说学期、学时或这两个属性的组合都不能作为实体型的候选码。在 ER 图中，使用双边矩形来描述弱实体型。

如果要想标识出“课程提供”实体型中的实体，则必须借助于“课程”强实体型，使用“课程”强实体型中的课程编号属性以及“课程提供”弱实体型中的学期属性可以共同标识“课程提供”弱实体型中的每个实体。虽然弱实体型没有候选码，但是某个或某些属性可以结合强实体型中的属性来共同标识弱实体型中的实体，这样的属性或属性组称为分辨符，在 ER 图中使用虚线下画线来描述，如图 3-17 所示，“学期”属性就是一个分辨符。并不是所有的弱

实体型都需要有分辨符,如果弱实体型中的每个实体都可以通过与其关联的强实体型来标识,则弱实体型不需要分辨符。

弱实体型所依赖的强实体型又称为标识实体型,每个弱实体型必须和标识实体型相关联,弱实体型与标识实体型之间的联系称为标识性联系。标识性联系是从弱实体型到标识实体型的多对一或一对一联系,并且弱实体型完全参与联系。在 ER 图中,使用双边菱形描述标识性联系。

弱实体型可以参与标识性联系以外的其他联系。弱实体型也可以作为标识实体型参与到与另一个弱实体型的标识性联系中。一个弱实体型也可能与不止一个标识实体型关联,这样一个特殊的弱实体型可以通过来自标识实体型的实体组合来标识。弱实体型的候选码可以由标识实体型的候选码的并集加上弱实体型的分辨符组成。

如果弱实体型只参与一个关联性联系,而且它的属性也不是很多,则建模时可以将其表述为强实体型的属性,如图 3-18 所示,将“课程提供”表述为“课程”实体型的一个多值复合属性。如果弱实体型参与到标识性联系以外的其他联系中,或其属性比较多时,则建模时将其表述为弱实体型更为恰当。

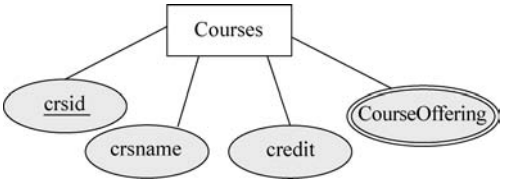

图 3-18 将弱实体型表述为多值复合属性


视频讲解

3.2 概念结构设计

设计数据库的概念结构,首先需要做的就是建立一个或多个概念数据模型。概念数据模型主要由实体型、联系型、属性和属性的域、主码和候选码以及完整性约束组成。概念数据模型开发的结果是一系列文档,文档中主要包括 ER 图以及支持概念数据模型的规格文档。

3.2.1 局部概念数据模型

在需求分析阶段,如果使用需求集中法,那么针对系统全部需求设计的概念数据模型是全局概念数据模型。如果使用视图集成法,那么将根据各个局部视图的需求设计局部概念数据模型,然后再将这些局部概念数据模型合并为全局概念数据模型。接下来以“图书销售”视图的需求为例介绍如何设计概念数据模型。

1. 实体型的识别

现实世界中一组具有某些共同特性和行为的对象就可以抽象为一个实体型。在用户需求文档中,实体型通常表现为名词的形式。实体型的组成成分可以抽象为实体型的属性。

实体型与属性是相对而言的,很难有截然划分的界限。同一事物,在一种应用环境中作为“属性”,在另一种应用环境中就可能会作为“实体”。一般说来,在给定的应用环境中,如

果一个事物满足以下两个条件之一的，一般可作为属性对待。

（1）属性不再具有需要描述的性质，属性在含义上必须是不可分的数据项。

（2）属性不能再与其他实体型具有联系。

例如，在“图书销售”数据库系统的需求分析中可以识别出以下实体型：图书（Books）、类别（Categories）、顾客（Customers）、订单（Orders）和评论（Comments）。

识别出实体型以后，需要为实体型确定一个名称，并将实体型的名称、描述等信息保存为实体型规格文档，如表 3-1 所示。

表 3-1　实体型规格文档

实体型名称	描　述	别　名	实　现
Books	表示系统中全部图书	图书	每册图书都有一个唯一的书号，而且都属于并且只属于一个类别
Customers	购买图书的顾客。顾客可以在购买图书之前在系统中进行注册	顾客	每位顾客都有唯一的顾客编号
……	……	……	……

2. 联系型的识别

找出实体型以后需要判断所有实体型之间是否存在联系，这种联系在用户的需求文档中通常表现为动词的形式，例如订单“包含”图书、顾客“生成”订单、顾客“发表”评论等，这些都是潜在的联系。

联系一般出现在两个实体型之间，但是也有特殊情况。有些联系可能出现在三个或更多实体型之间，这种联系型被称为多元联系。有些联系也许是某个实体型与自己的联系，这种联系被称为递归联系。确定联系以后需要进一步确定联系的基数映射。

例如，在“图书销售”数据库系统的需求分析中可以识别出以下联系型：订单包含（Contain）图书、顾客生成（Produce）订单、图书属于（Belongto）类别、顾客针对图书发表（State）评论。

识别出联系以后，需要为联系确定一个名称，并将联系的名称、描述以及同实体型之间的关系等信息保存为联系型的规格文档，如表 3-2 所示。

表 3-2　联系型规格文档

联系型名称	描　述	别　名	实　现	
			实体型	基数映射
Produce	顾客生成订单	生成	Customers	每张订单对应的顾客有且仅有一个（1…1）
			Orders	每名顾客可以没有订单，或有多张订单（0…n）
State	顾客针对图书发表评论	发表	Customers	每本图书的每条评论对应的顾客有且仅有 1 个（1…1）
			Books	每名顾客针对每本图书可以不发表评论，或发表多条评论（0…n）
			Comments	每名顾客发表的每条评论对应的图书有且仅有 1 本（1…1）
……	……	……	……	……

3. 属性的识别

属性在用户的需求文档中表现为名词的形式。属性是实体型或联系型的特征的描述。

另外,需要注意属性的冗余,例如派生属性。假如在“顾客”实体型中找到一个“出生日期”属性,那么“年龄”属性就是一个派生属性。如果派生属性所基于的对象不会发生变化,那么派生属性是多余的,可以去掉,但是如果派生属性所基于的对象有可能发生变化,例如消失,那么在实体型上保存派生属性是有必要的。

是否保留复合属性取决于用户的需求,如果用户不会访问复合属性中的子属性,那么就可以保留复合属性,否则应该将复合属性分解为若干子属性。

属性一般都是单值的,但是也可能出现多值的情况。例如起初设计“顾客”实体型时可能每名顾客只有一个联系电话,那么“联系电话”属性是一个单值属性。但是随着时间的变化,逐渐有顾客拥有多个联系电话,这时“联系电话”属性就成为了一个多值属性。通常,多值属性需要被识别为一个独立的实体,但是在 ER 模型中也可以保留多值属性,因为在数据库的逻辑结构设计过程中,也会将多值属性映射为一个独立的关系。

随着属性的确定,ER 模型中的实体型可能会发生变化。例如系统中的两个实体型的属性相同或相近,这时可能需要考虑将两个实体型合并成一个实体型。

通过检查实体型上的属性,也有可能找出实体型之间的新的联系。例如,若在创建 ER 模型时没有在“课程”实体型和“系部”实体型之间建立联系,但是“课程”实体型上有一个“系部”属性,表示课程属于哪个系部管理,此时应该去掉“课程”实体型上的“系部”属性,取而代之的是在“课程”和“系部”实体型之间建立一个“属于”联系。

联系上也可能存在属性,在处理联系的属性时需要注意属性的布局。通常一对一和一对多联系的属性都可以转变为实体型上的属性,而多对多联系型上的属性通常都不能被转变为实体型上的属性。

例如,在“图书销售”数据库系统的需求分析中,实体型和联系的属性如下所示。

```
类别(类别代号,类别名称)
图书(书号,书名,出版号,作者,单价)
顾客(顾客号,顾客姓名,联系电话,邮编,地址,电子邮箱,登录口令)
订单(订单号,订购日期,发货日期)
评论(评论号,等级,评论内容)
```

识别出属性以后,需要为属性确定名称,并将属性的名称、描述、数据类型、是否为空、类型以及对应的实体型或联系型的名称等信息保存为属性规格文档,如表 3-3 所示。

表 3-3 属性规格文档

实体型或联系型名称	属 性	描 述	数据类型及宽度	NULL	派生属性	多值属性
Categories	ctgcode	类别代号	不超过 20 个 Unicode 字符	×	×	×
	ctgname	类别名称	不超过 50 个 Unicode 字符	×	×	×
Contain	quantity	数量	整型数值	×	×	×
	price	售价	定点型数值,包含 2 位小数	√	√	×
……		……	……	……	……	……

4. 属性域的识别

属性的域就是属性的取值范围,例如“图书”实体型中的“单价”属性的取值是一个保留两位小数的实数。更完善的域的定义除了属性允许的一些值以外,还包括这些值的尺寸或

格式等信息。例如“顾客”实体型的“电子邮箱”属性值由 Unicode 字符组成，并且包含符号@，在该符号前是邮箱用户名，该符号后是邮件服务器的域名。

识别出属性的域以后，需要为属性的域建立规格文档，如表 3-4 所示。

表 3-4　属性域的规格文档

实体型或联系型	属　性	数据类型及宽度	NULL	取值范围或格式	举　例
Orders	orderid	整数	×	从 1 开始的整数	1、16、100 等
	orderdate	日期型	×	4 位年，2 位月，2 位日，中间用“-”分隔	2021-04-14
	shipdate	日期型	√	4 位年，2 位月，2 位日，中间用“-”分隔，并且不能早于同一张订单的订购日期	2021-04-17
……		……	……	……	

5. 码的识别

候选码是指实体型中的某个属性或最小属性组，其值可以唯一标识实体型中的每个实体。一个实体型可能包含一个以上的候选码，此时需要为该实体型确定一个主码。可以考虑使用以下原则来从若干候选码中选择一个做主码。

(1) 候选码中包含的属性数量越少越好；

(2) 候选码中的值从不或很少发生变化；

(3) 如果候选码的属性是字符型，那么属性值中包含的字符个数越少越好；

(4) 如果候选码的属性是数值型，那么属性值中的最大值越小越好；

(5) 选择最方便用户使用的候选码做主码。

例如在“类别”实体型中，属性“类别代号”和“类别名称”都是该实体型的候选码，但是“类别代号”属性值中包含的字符个数少，而且最方便用户使用，所以选择这个属性作为“类别”实体型的主码。

强实体型都可以找到候选码，而弱实体型中的属性不能构成其候选码。在数据库逻辑结构设计部分将介绍如何确定弱实体型的候选码。

识别出主码以后，需要为主码建立规格文档，这里不再赘述。

图 3-19 是“销售”视图下对应的 ER 图。为简便起见，图中只画出主码属性，省略了其他属性。

3.2.2　全局概念数据模型

到目前为止，数据库开发人员应该已经创建了各个用户视图的局部概念数据模型，包括 ER 图以及支持概念数据模型的一系列规格文档，利用它们比较各个数据模型的异同点，并为合并做好准备。图 3-19 是“销售”视图对应的 ER 图，图 3-20 是“管理”视图对应的 ER 图。

1. 消除冲突

如果系统中存在多个局部概念数据模型，则可以采用两两合并的方式最终合并为全局概念数据模型。在合并概念数据模型的过程中首先需要解决冲突问题。冲突可能发生在以下几种情况当中。

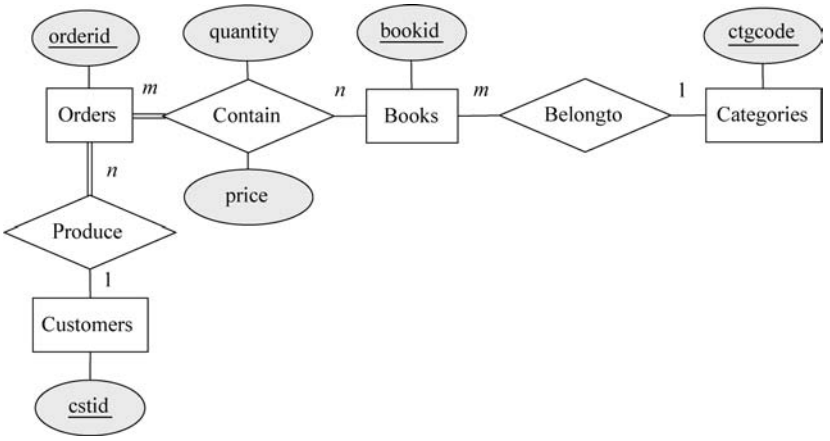

图 3-19 “销售”视图的 ER 模型

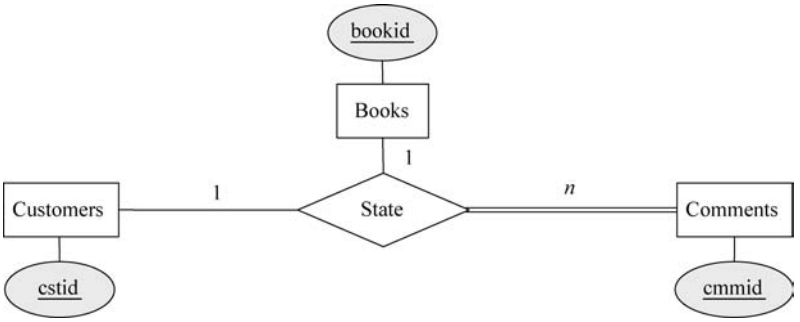

图 3-20 “管理”视图下的 ER 模型

(1) 属性冲突,包括属性值类型、取值范围或取值集合以及取值单位的冲突。这种冲突可以采用讨论协商等方式解决。

(2) 命名冲突,包括实体型名、联系名、属性名之间的同名异义和同义异名等命名冲突。这种冲突也可以通过讨论协商等方式解决。

(3) 结构冲突,主要包括三种情况。一是同一对象在不同的局部概念数据模型中具有的抽象不同,例如“评论”,在有的局部模型中被设计为实体,而在有的局部模型中被视为属性;二是同一实体在各局部概念数据模型中包含的属性不完全相同,例如“顾客”实体型的属性在“销售”视图和“管理”视图中的属性不同;三是实体型之间的联系在不同局部视图中呈现不同的基数映射。为了消除结构冲突,使各个局部概念数据模型相互匹配,如实反映应用需求,必须返回到需求分析阶段,做更加细致的调查研究,经过认真分析再做一致性调整。

2. 概念数据模型的合并

在解决了各个局部概念数据模型冲突的基础上,重新修正各个局部视图的 ER 图和规格文档,并为概念数据模型的合并做好准备。

(1) 对于各个局部概念数据模型中相同的实体型,直接合并为全局概念数据模型中的实体型。如果主码不同,需要重新选择实体型的主码。对于某个局部数据模型中单独存在的实体型,直接将这些实体型添加到全局概念数据模型中。

（2）将各个局部概念数据模型中相同的联系直接合并为全局概念数据模型中的联系。将某个局部概念数据模型中独立存在的联系添加到全局概念数据模型中。

（3）检查全局概念数据模型中是否丢失了实体型或联系型。在系统全局需求中存在的实体型或联系型不一定会出现在局部概念数据模型中。例如在一个局部概念数据模型中有一个实体型 A，在另一个局部概念数据模型中有一个实体型 B，在全局概念数据模型中 A 与 B 应该是有联系的，但是各个局部概念数据模型反映不出这种联系，通过简单的合并也不能得到这种联系，这时就需要认真检查全局概念数据模型，发现所有丢失的元素。

3. 建立全局概念数据模型

重新验证全局概念数据模型是否满足规范化设计需要、是否满足完整性设计需要，并进行适当的修改和调整，然后建立全局概念数据模型。建立全局概念数据模型的工作主要包括重新绘制反映全局应用需求的 ER 图并重新修订支持全局应用的规格文档。“图书销售”数据库系统的全局 ER 图如图 3-21 所示。

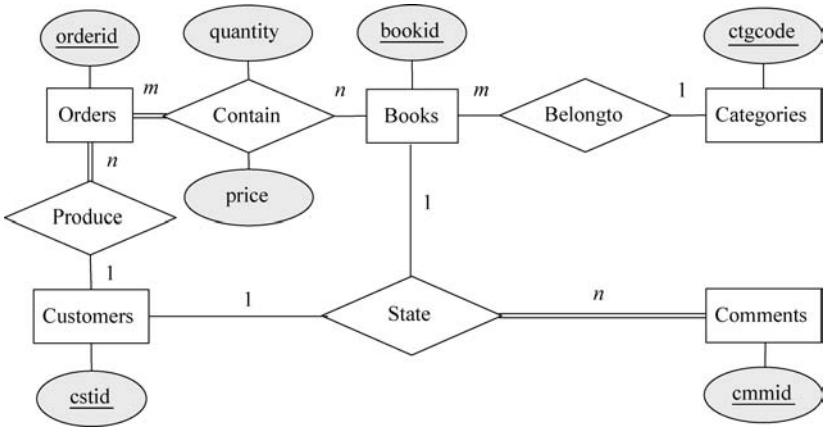

图 3-21　“图书销售”数据库系统全局 ER 图

最后需要用户确认全局概念数据模型能够反映企业的实际应用需要，结束数据库概念结构设计阶段的工作并开始下一个阶段的工作。

3.3　逻辑结构设计


视频讲解

概念结构设计是各种数据模型的共同基础，独立于任何一个 DBMS 系统。为了能够用某一数据库管理系统实现用户需求，还必须将概念结构进一步转化为相应的数据模型。数据库的逻辑结构设计就是在概念结构设计的基础上，将与数据库管理系统无关的概念数据模型转化成某个具体的 DBMS 所支持的逻辑数据模型，这些模型在功能性、完整性、一致性以及数据库的可扩充性等方面均应满足用户的各种要求。

数据库逻辑结构设计是在概念结构设计的基础上实施的，其中主要的一项工作是将概念数据模型中的 ER 图转换为关系模型中的关系模式，另外还包括数据模型的验证、修正以及支持文档的更新等任务。接下来将以图 3-21 所示的全局 ER 图为例介绍如何将 ER 图转换为关系模式。

提示：有些时候也可以将各个局部概念数据模型转换为局部逻辑数据模型，然后再进行合并，得到全局逻辑数据模型。

3.3.1 实体型和属性的转换

在 ER 图中，实体型分为强实体型和弱实体型。下面分别介绍这两种实体型转换成关系模型的方法。

1. 强实体型

将每个强实体型转换为一个关系模式，实体型的属性作为关系模式的属性，实体型的主码作为关系模式的主键。

例如，图 3-21 的 ER 图的强实体可以转换为如表 3-5 所示的关系模式。这些关系可能不是逻辑结构设计的最终产物，在接下来的步骤中，这些关系可能会有所变化。

表 3-5 “图书销售”系统的强实体关系

编 号	关 系 模 式	主 键
1	Categories (ctgcode, ctgname)	ctgcode
2	Books (bookid, title, isbn, author, unitprice)	bookid
3	Customers (cstid, cstname, telephone, postcode, address, emailaddress, password)	cstid
4	Orders (orderid, orderdate, shipdate)	orderid
5	Comments (cmmid, rating, comment)	cmmid

2. 弱实体型

弱实体型到标识实体型的联系类型是多对一或一对一联系，不同的联系类型将导致不同的转换结果。

(1) 如果弱实体型到标识实体型之间的联系是多对一联系，如图 3-22 所示。将弱实体型转换为一个关系，关系的属性包括弱实体型自身的属性、标识性联系的属性以及标识实体型的主码属性。弱实体型转换得到的关系的主键由标识实体型中的主码属性和弱实体型中的分辨符组成，并且该关系独立存在，不与其他关系合并。图 3-22 将转换为如下所示的关系。

```
Courses (crsid, crsname, credit)
CourseOffering (crsid[FK], semester, hrs)
```

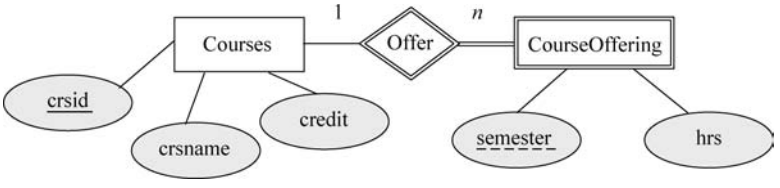

图 3-22 多对一标识联系

(2) 如果弱实体型到标识实体型之间的联系是一对一联系，且标识实体型部分参与联系，那么将弱实体型转换为一个关系，关系的属性包括弱实体型自身的属性、标识性联系的

属性以及标识实体型的主码属性。弱实体型转换得到的关系的主键由标识实体型中的主码属性组成，并且该关系独立存在，不与其他关系合并。如果将图 3-22 中联系类型改为一对一联系，则弱实体型 CourseOffering 中不再需要分辨符 semester，转换的关系如下所示。

```
Courses (crsid, crsname, credit)
CourseOffering (crsid[FK], semester, hrs)
```

提示：实际上，也可以将关系 Courses 和 CourseOffering 合并为一个关系 Courses (crsid, crsname, credit, semester, hrs)。但是由于不是所有的课程都有课程提供，所以将来在关系 Courses 的 semester 列和 hrs 列中会出现空状态。然而，将 Courses 和 CourseOffering 处理为两个独立的关系则不会出现空状态。

(3) 如果弱实体型到标识实体型之间的联系是一对一联系，且标识实体型完全参与联系，则将弱实体型的属性以及标识性联系的属性移动到标识实体型中。如果将图 3-22 中的标识性联系改为一对一联系，实体型 Courses 的参与约束改为完全参与，则转换得到的关系如下所示。

```
Courses (crsid, crsname, credit, semester, hrs)
```

3. 多值属性的转换

将多值属性转换为一个关系，另外将实体型中的主码属性复制到该关系作为外键。有时多值属性自己就可以在新关系中形成主键，但是有些时候需要将多值属性和实体型的主码属性组合起来形成新关系的主键。例如图 3-13 中“顾客”实体型中的联系电话属性是一个多值属性，所以将“顾客”实体型转换为两个关系，一个对应顾客，一个对应联系电话。考虑到有些顾客可能使用相同的联系电话的情形，那么 telno 属性就不能作为 Telephones 关系的主键，所以该关系的主键是 telno 和 cstid 的组合。

```
Customers (cstid, cstname, postcode)
Telephones (telno, cstid[FK])
```

3.3.2　联系的转换

由于实体型之间的联系种类很多，所以联系转换为关系也相对较复杂。尽管如此，联系转换为关系的方法还是有规律可循的。

(1) 将联系转换为关系。首先将联系转换为关系，关系的属性包括联系的属性以及参与该联系的实体型的主码属性。

(2) 确定主键。通常情况下，来自子实体型或多方实体型中的主码属性构成新关系的主键，但是在某些特殊场合，主键中还可能需要包含其他的属性。

(3) 合并关系。如果由联系转换得到的关系与由实体型转换得到的关系具有相同的主键，那么将具有相同主键的关系合并。

接下来分别介绍二元联系、多元联系和递归联系转换为关系的处理方法。

1. 多对多二元联系

在多对多二元联系中，需要特别注意的是“确定主键”。多对多二元联系转换得到的关

系中,来自两个实体型的主码属性的组合通常可以构成这个关系的主键,但是在某些情况下还需要包含其他的属性才可以构成主键。

如图 3-23 所示,Join(参与)联系是一个多对多联系,转换为一个关系。所以图 3-23 的 ER 图转换得到如下关系模式。

```
Projects (prjid, ...)
Teachers (tchid, ...)
Joining (prjid[FK], tchid[FK], order, task)
```

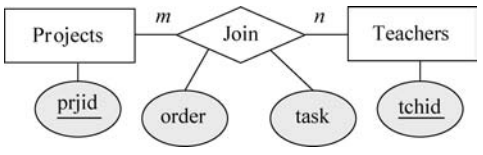

图 3-23　教师参与项目的多对多联系

但是在如图 3-24 所示的读者 Borrow(借阅)图书的多对多联系中,由于一个读者可以多次借阅同一本图书,所以转换的关系的主键除了包括属性 bookid(书号)和 rdrid(读者号)以外,还需要包括属性 borrowtime(借阅时间)。所以图 3-24 的 ER 图可以转换得到如下关系模式。

```
Readers (rdrid, ...)
Books (bookid, ...)
Borrowings (rdrid[FK], bookid[FK], borrowtime, returntime)
```

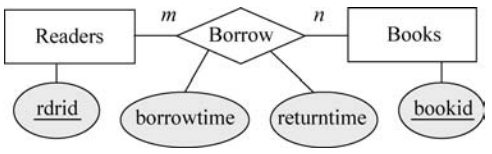

图 3-24　读者借阅图书的多对多

多对多二元联系转换的关系具有独立性,通常不与任何关系合并。

2. 一对多二元联系

例如,图 3-21 的 ER 图中有两个一对多二元联系,分别转换为如下所示的关系。

```
Producing (ordertid, cstid)
Belonging (bookid, ctgcode)
```

接下来将所有具有相同主键的关系合并。将以上两个关系与表 3-5 中的关系合并,得到如表 3-6 所示的关系。

表 3-6　合并后的关系

编　号	关 系 模 式	主　键
1	Categories (ctgcode, ctgname)	ctgcode
2	Books (bookid, title, isbn, author, unitprice, **ctgcode**)	bookid

续表

编　号	关 系 模 式	主　键
3	Customers (cstid, cstname, telephone, postcode, address, emailaddress, password)	cstid
4	Orders (orderid, orderdate, shipdate, **cstid**)	orderid
5	Comments (cmmid, rating, comment)	cmmid

提示：可以将一对多联系转换为关系的过程合并为一步完成，即将父实体型的主码属性复制到子实体型所在的关系中作为属性并成为该关系的外键。如果子实体型部分参与联系，那么这个外键中可能有空状态出现。如果一对多联系上存在属性，则应该将这些属性同时移动到子实体型对应的关系中。

通常情况下，一对多二元联系转换得到的关系都会与实体型转换得到的关系合并。

3. 一对一二元联系

在一对一二元联系中，实体型的参与约束将影响一对一联系转换为关系模型的结果。

(1) 在一对一联系中，如果一个实体是完全参与，另一个实体是部分参与，则部分参与的实体作为父实体，完全参与的实体作为子实体，选择子实体型中的主码属性作为联系对应的关系的主键。因此，最终这个关系将与子实体型中的关系合并。

在如图 3-25 所示的 Locate(书店位于城市)的联系中，首先转换为如下所示的关系。

```
Bookstores (bstid, bstname)
Cities (cityid, cityname)
Locating (bstid, cityid)
```

合并以后得到如下所示的两个关系。

```
Bookstores (bstid, bstname, cityid[FK])
Cities (cityid, cityname)
```

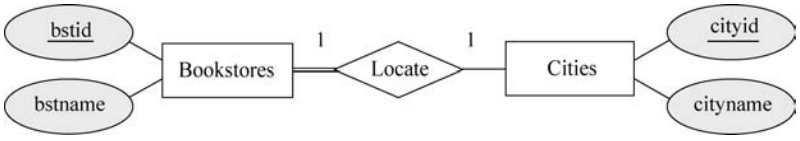

图 3-25　“书店位于城市”的一对一联系

提示：实际上，将关系 Bookstores 和 Cities 合并为一个关系 Cities(cityid, cityname, bstid, bstname)也是一种选择。但是由于不是所有城市都有书店，所以将来在关系 Cities 的 bstid 列和 bstname 列中会出现空状态。然而，将 Bookstores(书店)和 Cities(城市)处理为两个独立的关系则不会出现空状态。

(2) 在一对一联系中，如果参与联系的两个实体型都完全参与联系，那么它们的主码是等价的，最终可以将两个实体型以及它们的联系合并为一个关系，选择一个实体型的主码作为新关系的主键，另一个实体型的主码成为候选键。例如，若 Locate 的联系中，Bookstores 和 Cities 都完全参与联系，那么最终得到的关系如下所示。

```
Cities (cityid, cityname, bstid, bstname)
```

(3) 在一对一联系中,如果两个实体型都是部分参与,则根据实际情况选择一个实体型作为父实体型,处理方法与一个实体型为部分参与,另一个实体型为完全参与的一对一联系相同。

总的来说,一对一二元联系转换得到的关系都会与实体型转换得到的关系合并。

4. 多元联系

如果一个多元联系等效于一个多对多联系,这个多元联系会成为一个独立的关系。图 3-26 的"选修"联系等价于多对多联系,所以转换得到的关系不会被合并到其他关系中。该三元多对多联系转换为如下关系模式。

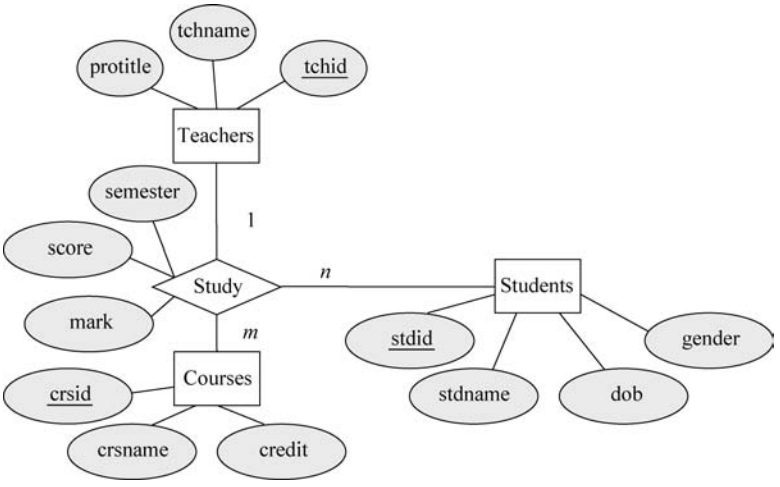

图 3-26 三元多对多联系

```
Teachers (tchid, tchname, protitle)
Students (stdid, stdname, dob, gender)
Courses (crsid, crsname, credit)
Studying (stdid[FK], crsid[FK], tchid[FK], semester, score, mark)
```

如果一个多元联系等效于一对多联系或一对一联系,由这个多元联系转换得到的关系通常会在后续的步骤中与其他关系合并。例如图 3-27 是顾客针对图书发表评论的联系。首先转换为如下所示的关系。

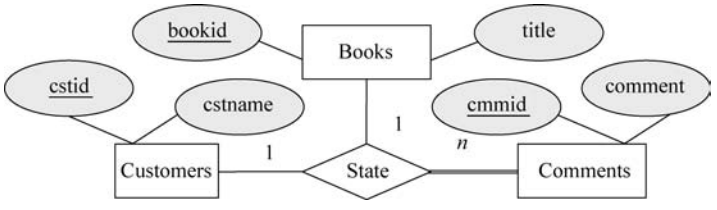

图 3-27 等效于一对多的三元联系

```
Books(bookid, title)
Customers(cstid, cstname)
Comments(cmmid, comment)
Stating(cmmid, cstid[FK], bookid[FK])
```

由于 Comments 和 Stating 具有相同的主键，将它们合并，最终转换为如下所示的关系。

```
Books(bookid, title)
Customers(cstid, cstname)
Comments(cmmid, comment, cstid[FK] , bookid[FK])
```

如图 3-21 所示的“图书销售”系统全局 ER 图最终转换为如表 3-7 所示的关系模式。

表 3-7　“图书销售”系统全局 ER 图转换为的关系模式

编　号	关 系 模 式	主　键	外　键
1	Categories (ctgcode, ctgname)	ctgcode	
2	Books (bookid, title, isbn, author, unitprice, ctgcode)	bookid	ctgcode REFERENCES Cagegories (ctgcode)
3	Customers (cstid, cstname, telephone, postcode, address, emailaddress, password)	cstid	
4	Orders (orderid, orderdate, shipdate, cstid)	orderid	cstid REFERENCES Customers (cstid)
5	Comments (cmmid, rating, comment, bookid, cstid)	cmmid	bookid REFERENCES Books (bookid) cstid REFERENCES Customers (cstid)

5. 递归联系

将一个递归联系转换为一个关系，关系中的属性包括联系本身的属性以及相连的实体型主码属性的两个副本，将这两个副本重命名以表示参与联系的两个角色。根据联系的类型确定关系的主键。最后考虑是否将这个关系与实体型转换得到的关系合并。

如图 3-28 所示，Teachers(教师)实体型自身存在一个递归联系 Supervise(管理)。由递归联系转换得到的关系的两个属性都是从 Teachers 实体型中复制的主码属性 tchid，并可以重命名，以达到“见名知意”的效果。

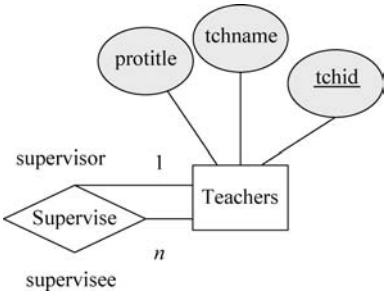

图 3-28　一对多的三元联系

关系 Teachers 表示全体教师，关系 Supervising 表示递归联系 Supervise。两次复制 Teachers 关系中的主键属性 tchid，第一次命名为 superviseeid，表示“被管理”的教师编号；第二次命名为 supervisorid，表示“作为管理者”的教师编号。确定 superviseeid 为主键是因为“被管理”的教师在一对多联系中是多的一方，即子实体角色的一方。

```
Teachers (tchid, tchname, protitle)
Supervising (superviseeid, supervisorid)
```

superviseeid 和 tchid 虽然名称不一样，但是在实际应用中表示的都是教师的编号，所

以认为以上两个关系具有相同的主键,合并为如下所示的一个关系。

```
Teachers (tchid, tchname, protitle, supervisorid[FK])
```

这个关系比较特殊的地方在于,外键 supervisorid 参照关系自己的主键 tchid。

3.4 实践练习

假设你计划为企业开发一个内部消息系统,该系统用于企业信息发布及企业员工之间的信息交流。系统中的全部数据都保存在数据库中。在系统中需要记录所有用户的信息,每位用户有一个唯一的用户编号,以及用户姓名和登录密码。每位用户都属于一个部门,每个部门拥有 1 或多名用户,每个部门都有一个唯一的部门编号,以及部门名称。系统还要保存全部消息,每条消息有一个唯一的消息编号,以及消息的标题、主体和发送时间。每条消息都属于一个类型,每个类型可以包含 0 或多条消息,每个类型有一个唯一的类型编号,以及类型名称。每位用户可以发送 0 或多条消息,也可以接收 0 或多条消息。每条消息有且仅有一位发送者,但同时可以有 1 或多位接收者,系统还需要记录每位接收者接收到消息后是否阅读了该消息(即消息的状态)。请绘制系统的 ER 图并进行必要的文字说明。