第 3 章 数据库设计

学习完了第 1 章和第 2 章，读者已经基本理解了数据库中数据的组织方式——使用关系模型组织数据，也清楚了如何对关系进行查询操作。本章需要考虑另外一个问题：针对一个企业，应该如何将它的数据组织成关系模型呢？这个工作便是数据库设计。数据库设计通常经过需求分析、概念设计以及逻辑设计等步骤。概念设计是其中至关重要的一步。进行概念设计需要使用通用的概念模型。本章将对数据库设计的相关内容进行介绍，重点关注实体-联系模型，介绍从应用中鉴别实体和联系并进行正确表达的方法。同时，本章还关注如何将实体-联系模型向关系模型转换，从而能够得到一个基本的关系数据库模式。

3.1 数据库设计过程

数据库并不是天然就存在的，它的创建是一个复杂的任务。数据库设计是软件设计的重要组成部分，包含数据库模式的设计、索引的设计、安全机制的设计等。本节主要介绍数据库模式的设计。

3.1.1 设计目标

本节所说的数据库设计主要指数据库模式设计。一般来说，数据库是各种软件系统的支撑，例如一个企业的办公自动化系统需要有企业的数据库提供数据支撑，一个电商平台（如淘宝、京东等）需要巨大的商品及其购买情况的数据库做支撑。因此，在为一个企业进行数据库模式设计时，不仅要把企业的数据组织成关系模式，还要考虑这个关系模式实例化后的数据是否能有效支持企业的业务逻辑，是否利于提高应用的效率，是否会由于产生数据存储的冗余而增加企业的成本。这一系列问题都是衡量数据库设计的关键因素。一个合理的数据库应该在满足业务逻辑的基础上，以较小的冗余提供较高的应用效率，这也是数据库模式设计的基本目标。

3.1.2 设计步骤

为了设计好一个数据库，数据库的设计者需要充分理解应用系统的需求。但现实情况是应用系统往往非常复杂，只凭设计者自己是无法理解透彻的，因此设计者往往要与用户进行充分的沟通，以理解应用系统的需求，并将这种理解用高层次的模型进行表达以使用户也能理解，双方利用该高层次的模型进行充分的沟通，从而全面理解应用系统的数据

需求。然后，设计者再将高层次的模型转换为低层次的逻辑模型以进行数据库的实施。这个高层次模型也称概念模型，它为数据库设计者提供了一个概念框架，以系统的方式定义了用户的数据需求。基于这一过程，数据库的设计可分为以下几个阶段：

（1）需求分析阶段。该阶段也是软件需求分析阶段的重要组成部分。其主要任务是数据库设计者同领域专家和用户深入沟通系统的数据需求，并以文字的形式进行描述，可以形成需求规格说明书。

（2）概念设计阶段。基于需求分析阶段的成果，产生企业的数据需求，包括需要什么样的数据，数据间都有什么样的联系，都遵循什么样的约束，等等。用合理的方式进行表达，建立企业数据的概念模型，通过与用户的反复沟通，使概念模型表达的数据及联系能够全面符合企业的业务逻辑。该阶段一般使用实体-联系模型，用实体和联系的方式描述现实中的数据，有利于从宏观描述数据及其结构，使设计者能够从全局掌控数据，并能够反复检查，不断完善。

（3）从系统功能的角度出发，完善概念模型的设计。了解系统都需要为用户提供哪些功能、哪些操作，是插入、更新、删除还是查询，都涉及哪种类型的查询，是否涉及数据间的特别约束等，并分析概念模型表达的数据是否满足业务应用的需求，对其进行修改和完善。

（4）逻辑设计阶段。该阶段将高层概念模型映射到具体使用的数据库管理系统运行的数据模型，也是就逻辑数据模型。对于关系数据库而言此步骤要求将概念模型映射到关系模型，然后根据应用要求对关系模型进行相关的优化。

（5）物理设计阶段。该阶段根据应用需求情况，为数据库设计合理的空间、存储路径、索引、约束等。对深层次的应用还要设计数据的存储方式等。本书不对此进行介绍，有兴趣的读者可以参考相关的数据库书籍。

数据库的设计非常重要，它既决定系统的可用性和可靠性，也决定用户对系统的体验感，因此在应用程序开发之前一定要慎重设计系统所需要的数据库。

3.1.3　设计的平衡

在进行数据库设计的时候，一个基本问题是应该如何表达各种类型的事物？例如人、汽车、货物，它们之间有没有共性？有没有关联？有什么样的关联？我们希望利用事物的共性获得简洁有效且易于理解的设计，但也需要一定的灵活性，以保持它们各自的不同。因此，数据库设计需要设计者不断在简洁与全面、冗余与效率间保持平衡。例如在学生管理系统的数据库中，对于学生的信息除了有学号、姓名、性别等外，还有住址信息，而住址信息比较长，包含省市区县街道等一系列信息，多个同学甚至居住在同一个小区、同一个单元，他们的地址信息基本是相同的。如果把地址作为学生的属性，则会有信息的冗余，但这样的好处是，如果查询某个学生的地址，其查询效率会很高。如果想把地址中相同的部分只记录一次，则需要额外的关系以表达地址与学生的关系，这样降低了冗余度，减少了存储空间，但这样做的缺点是要查询学生的地址信息时就变得复杂，效率较低。

在进行数据库设计时还需要注意的另外一个问题是完整性，即，数据是否覆盖了系统的所有业务，以及数据间的联系与实际的业务处理是否相一致。例如，把学生的学号、姓

名和性别等信息与选修的课程组成一个关系存储是否可行？虽然看上去在查询时可直接获得学生姓名与所选课程的成绩，但这样做是否会对业务处理造成其他的问题？由此可见，数据库设计是一个综合考虑多种因素且优中选优的过程。

本章后面将对数据库设计的概念模型设计阶段和逻辑模型设计阶段进行详细讨论。

3.2 概念模型

3.2.1 什么是概念模型

在数据库项目开发过程中，在进行了需求分析之后，就可以着手进行概念结构的设计了。概念结构是用来与用户进行交流的，与数据库管理系统无关。在给出系统的逻辑设计之前，设计概念结构，听取用户意见，及时进行修正，以便正确地表达用户的需求，避免将隐患带入后面的逻辑结构设计阶段和应用系统的程序设计阶段。

从用户的需求中提取需要表达和存储的各个事物以及这些事物之间的关联，以用户的视角和观点进行抽象得到的模型就是概念模型，也称为信息模型。这是数据库设计过程中从现实世界到信息世界的抽象，是从用户角度进行的抽象。因此，概念模型的表达方式应该直观、简洁、用户易于理解。

3.2.2 常用的概念模型

常见的概念模型有 E-R 模型（Entity-Relationship model，实体-联系模型）、UML（Unified Modeling Language，统一建模语言）、ODL（Object Definition Language，对象定义语言）等。

下面通过一个例子说明这 3 种常用的概念模型是如何表达数据的概念结构的。

例 3.1 考虑某高校学籍管理系统中学生选课的问题。该系统的具体描述是：每一位学生属于一个班级，每一个班级有多位学生。学生可以选修多门课程，每一门课程有多位学生选修，学生选修课程会获得相应的成绩。学生的相关信息有学号、姓名、性别、生日，其中学号唯一标识学生。班级的相关信息有班号、类别（取值为实验班、普通班等）、专业，其中班号唯一标识班级。课程的相关信息有课号、名称、学分，其中课号唯一标识课程。

下面分别用 3 种概念模型表达此应用问题的概念结构设计。

（1）用 E-R 模型给出的设计如图 3.1 所示。该 E-R 图由不同类型的节点和边连接而成。

（2）用 UML 模型给出的设计如图 3.2 所示。这个概念模型中包含 3 个类：学生类、课程类和班级类。每个类分为 3 部分：名称、属性、方法。PK 表示主码。图 3.2 中的选课情况为关联类，用属性“成绩”表示其状态或结果。“0..1”和“0..*”表达联系的类型。

（3）使用 ODL 设计的概念模型如下：

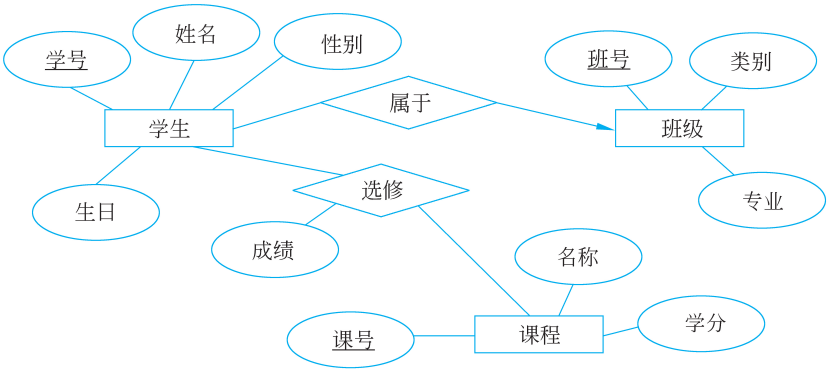

图 3.1　学生选课系统的 E-R 图

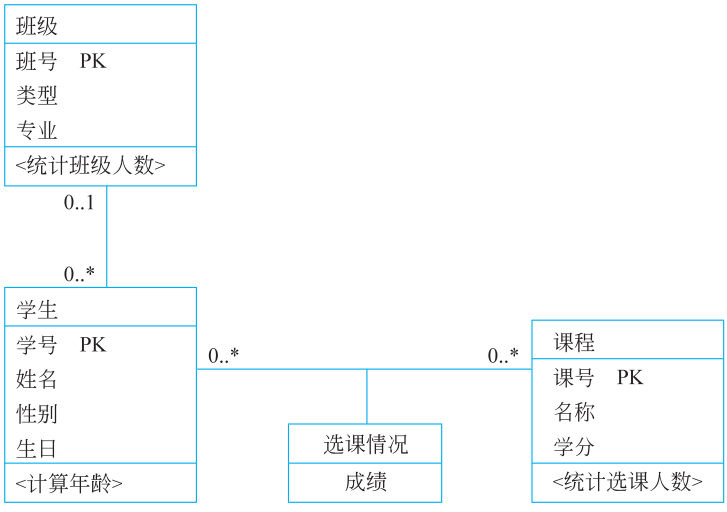

图 3.2　学生选课 UML 设计

```
class 班级
{  attribute string 班号;
   attribute string 类别;
   attribute string 专业;
    relationship include set<学生>;
    inverse 学生:: belongto;
    int countnum();                    //统计班级人数
}
class 学生
{  attribute string 学号;
   attribute struct name{string 姓,string 名} 姓名;
   attribute string 性别;
   attribute date 生日;
   relationship belongto <班级>;
   inverse 班级:: include;
   relationship take set<课程>;
   inverse 课程:: took;
   int age();                          //计算年龄
```

```
}
class 课程
{  attribute string 课号;
   attribute string 名称;
   attribute int 学分;
   relationship took set<学生>;
   inverse 学生:: take;
   int numberofstudents();            //统计选课人数
}
```

从这个例子可以看出，概念模型的表达简洁明了。大多数概念模型使用图形符号，用户易于理解，特别适用于数据库设计者与用户共同讨论交流。ODL 虽然不是图形方式，但近乎自然语言，并且易于转换为利用面向对象语言编写的程序，用来进行概念模型的设计也是十分方便的。

以上是比较简单的一个应用例子。事实上，每一种概念模型都有一套完整的表达规范，能够表达现实世界中的各种事物及其关联。本书重点介绍数据库设计中使用最为广泛的 E-R 模型。有关 UML 以及 ODL 更为详尽的内容，可以参看《数据库系统基础教程》[1]以及其他相关书籍，这里不再赘述。

3.3 E-R 模 型

E-R 模型是实体-联系模型的简称。该模型由 P.P.S.Chen 在 1976 年提出，它使用一套十分简洁而规范的图形符号表达数据的概念结构。由于 E-R 模型最终以图形的方式呈现，所以也称之为 E-R 图。由于 E-R 模型直观、易懂、规范且表达能力强，因此在数据库系统的概念设计阶段使用最为广泛。随着计算机技术的广泛应用，人们面对的应用问题越来越复杂，众多数据库技术专家也不断对 E-R 模型进行扩展，增加新的表达元素，以便适应新的应用问题。

E-R 模型的基本元素以及扩展部分的表达形式不是唯一的。本书选用斯坦福大学数据库原理教材——《数据库系统基础教程》中的表达方式。E-R 模型的扩展部分包括子类和弱实体的表达。

3.3.1 基本概念

1. 实体

所谓实体(entity)是现实世界中可区分的具体或抽象的事物。在需求规格说明书中常常是用名词来表达的。

所谓可区分，这里解释一下。例如，学生可以用学号加以区分。在学籍管理系统的数据库中，使用学号作为学生实体的标识，不同的学生有不同的学号。也就是说，每一个具体的实体在系统中可以通过唯一的标识查找、确认。再例如，教师实体使用职工号作为标识，说到某一职工号，则唯一代表一位教师。类似的例子还有课程、图书、合同、支票等。

同类型实体的集合。严格来讲称为实体集。例如，所有学生形成学生实体集，该集合中每个实体都是学生。但是，有时人们将实体集简称为实体，因此读者需要根据上下文来

理解。

2. 属性

同一类型的实体会有一些共同的特征。例如，学生有学号、姓名和性别这样一些共同的特征，需要存储于数据库中，供应用程序使用。这些共同的特性称为实体集的属性(attribute)，实际上实体都是由属性描述的。

属性的取值范围称为域(domain)。例如，姓名为15字节的字符串，年龄为14～24岁，等等，都是对属性取值范围的限定。

当描述实体的属性中有一个或一些属性可以让实体具有可区分性，或者说其能够唯一地标识具体的实体，使得该实体与其他实体不同，例如学生的学号可以标识具体的学生，不会存在多位学生同一学号的情况，此时称学号为学生实体集的码。

实体集的码可以由多个属性构成。例如，电影有片名、年份、片长、类别等信息。假定同一年没有同名的电影，那么(片名，年份)的组合就可以唯一确定一个电影，并且两个属性缺少任何一个就不能唯一标识一部电影，那么这个属性组是电影实体集的码。片名是码中的一个属性，称为码属性；同样，年份是另一个码属性。

从概念上讲，实体集的码有可能不是唯一的。例如，职工信息包含职工号、姓名、性别、身份证号；那么在公司中职工号唯一标识职工，可以作为码；同样，身份证号也唯一标识一个职工，是另一个码。当一个实体集有多个码时，这些码也被称为候选码，这是因为在概念模型设计阶段，绘制E-R图的时候仅仅标出其中一个码(即主码)，其他的码可以用文字或别的方式说明。

3. 联系

所谓联系(relationship)指的是实体集之间的关联，可以是一个实体集内部实体之间的关联，也可以是不同实体集的实体之间的关联。实体集之间的联系通常是由业务活动体现的，所以通常由动词描述。例如，学生“选课”这样一个活动就产生了学生实体集与课程实体集之间的联系。两个实体集之间的联系称为**二元联系**，例如“选课”是一个二元联系；三个或者三个以上实体集之间的联系称为**多元联系**，例如对于学生实体集、运动会实体集和运动项目实体集，学生参加运动会中的项目这一活动就体现了学生、运动会和项目3个实体集之间的联系，可称为多元联系。一个实体集内部实体之间的联系称为**一元递归联系**。有关多元联系和一元递归联系，将在3.3.2节介绍，这里先介绍最为常见的二元联系。

根据联系所涉及的实体的个数，二元联系可分为以下3种类型：**一对一联系**、**多对一联系**和**多对多联系**。

1) 一对一联系

一个实体集中的任意一个实体与另一个实体集中的至多一个实体有联系，反之亦然，则两个实体集之间的联系为一对一联系。如图3.3所示，班级实体集与班主任实体集之间为一对一联系，因为一个班级至多有一个班主任，而一个班主任至多管理一个班级。

读者或许注意到130731班没有对应的班主任，这种情况是允许出现的，即实体集中某些实体未参与联系，例如该班尚未指定班主任。但是对于那些参与联系的实体来说，两个实体集中的实体之间存在的关联关系只能是一对一的。

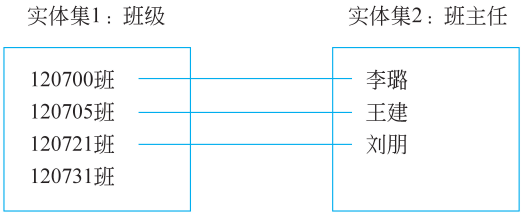

图 3.3　一对一联系

2）多对一联系

多对一联系指实体集 A 中的任意一个实体与实体集 B 中的任意数目的实体有联系，而实体集 B 中任意一个实体只能与实体集 A 中至多一个实体有联系。如图 3.4 所示，学生实体集和班级实体集之间就是多对一联系，因为班级实体集中的任意一个班级可以与学生实体集中的 0 个、1 个或多个学生有联系，而学生实体集中的任意一个学生只属于班级实体集中至多一个班级。

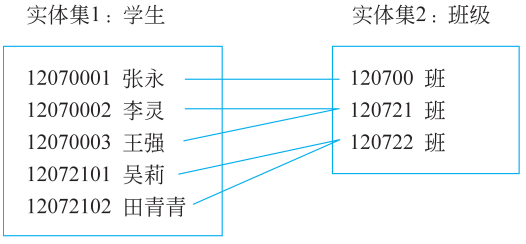

图 3.4　多对一联系

存在多对一联系的两个实体集中也可能会出现实体不参与联系的情况，例如，个别班级可能暂时还没有学生，或者个别学生暂时还没有安排到具体的班级，这些都不影响多对一联系的定义。

3）多对多联系

一个实体集中的任意一个实体与另一个实体集中的 0 个、1 个或多个实体有联系，反之亦然，则称两个实体集之间的联系为多对多联系。

图 3.5 给出了学生实体集与课程实体集之间多对多联系的例子。

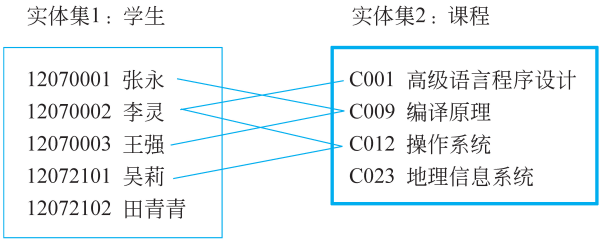

图 3.5　多对多联系

选修课程的一名学生可以选任意门课程，一门课程可以有任意数量的学生选修。当然，这两个实体集中也允许有实体不参与联系，例如田青青尚未选课，地理信息系统暂时

无人选修，这些都不影响多对多联系的定义。

二元联系是概念设计中最为常见的，也是初学者比较容易理解的实体集之间的联系。除了二元联系，实体集之间还会有多元联系、一元递归联系等较为复杂的联系，为了避免概念的混淆，多元联系和一元递归联系在 3.3.2 节介绍。

3.3.2　E-R 模型的表示方法

1. 实体集与属性

E-R 模型是一种图形化的模型，每种元素都有规范的表示方法。具体来讲，使用矩形表示实体集，使用椭圆表示属性，表示属性的椭圆与表示实体集的矩形之间用直线相连。实体集和属性的名称通常为名词。例如，学生实体集包含学号、姓名、性别、生日这 4 个属性，用 E-R 图表达如图 3.6 所示。

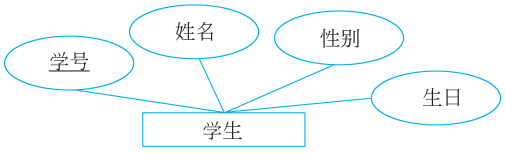

图 3.6　学生实体集与相应属性的 E-R 图

3.3.1 节在介绍实体概念的时候提到学号是学生实体集的码，在 E-R 图中用下画线标出作为码的属性，例如图 3.6 中的属性学号用下画线标出，表示它就是学生实体集的码。

如果一个实体集的码包含多个属性，例如（片名，年份），则相应的属性名都要用下画线标出。

如果一个实体集有多个码，例如职工有职工号、身份证号两个码，选一个系统常用的作为主码，在 E-R 图中用下画线标出；其他码不标出，可以用文字或其他方式说明。这样做是为了避免组合码与多个码在标识上混淆。

2. 联系及其属性

在 E-R 模型中，用菱形表示实体集之间的联系，并将菱形与相应的实体集间用直线连接。由于联系通常为某一活动的表达，因此联系的名称通常为动词。有关联系的类型，即“多”与“一”的表示有多种形式。本书使用的表示方法为：菱形到实体集的直线有箭头表示“一”，菱形到实体集的直线没有箭头表示“多”。如图 3.7 表达了班级实体集与班主任实体集间的一对一联系，名字为“对应”。注意，箭头一定要指向实体集，从而让用户清楚哪个实体集中的实体至多参与一个联系。为清晰起见，图 3.7 中将实体集的属性略去了。

图 3.8 为多对一联系的 E-R 图示例，表达了学生与班级的多对一联系“属于”，实体集的属性也被略去了。

图 3.7　班级与班主任间的一对一联系

图 3.8　学生与班级间的多对一联系

图 3.9 为多对多联系的 E-R 图示例，表达了学生与课程间的多对多联系“选修”，实体

集属性也被略去了。该联系有一个属性是“成绩”，表示学生选修课程的结果。

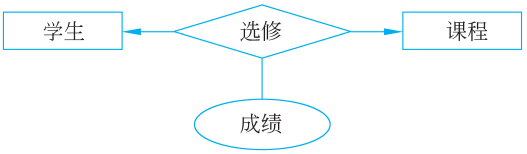

图 3.9　学生与课程间的多对多联系

读者或许注意到了，图 3.9 中“成绩”这一属性与“选修”这一菱形相连。这属性叫作联系的属性，即一个学生选修某一门课程这个活动发生时才会有的属性，它既不是学生的属性，也不是课程的属性。联系可以有属性，也可以没有属性，还可以有多个属性，要根据具体情况设定。

3. 多元联系

数据库应用中二元联系最为常见，但在一些特定的场合，也会出现 3 个或者 3 个以上实体集参与的联系，统称为多元联系。为了避免初学者概念上的混淆，3.3.1 节没有介绍多元联系，下面对其加以介绍。

先看三元联系，假定 3 个实体集为 E_1、E_2、E_3，它们共同参与一个活动，从而形成一个 3 个实体集之间的联系，这个联系就是一个三元联系。三元联系在 E-R 图中的表达依然使用菱形，该菱形与 3 个实体集 E_1、E_2、E_3 之间分别用线直线连接。但是对于 3.3.1 节介绍的二元联系中多与一的表达，在此怎么处理呢？原则是这样的：E_1 是否为“一”的一方，取决于 E_2、E_3 两个实体集。从 E_2、E_3 这两个实体集中分别任取一个实体，若能够确定 E_1 中唯一的实体，则 E_1 为“一”的一方，E-R 图中指向 E_1 的直线加上表示“一”的箭头；从 E_2、E_3 这两个实体集中分别任取一个实体，若不能确定 E_1 中唯一的实体，则 E_1 为“多”的一方，E-R 图中指向 E_1 的直线不加箭头。E_2、E_3 是否为“一”的一方，也按类似的方法处理。

推广一下，对于 n 个实体集 $E_i(i=1,2,\cdots,n)$之间的 n 元联系，E_i 是否为“一”的一方，取决于其他 $n-1$ 个实体集。即，从其他 $n-1$ 个实体集中分别任取一个实体，若能够确定 E_i 中唯一的实体，则 E_i 为“一”的一方，E-R 图中指向 E_i 的直线加上表示“一”的箭头；否则，E_i 为“多”的一方，直线不加箭头。

下面看一个具体的例子。

例 3.2　教师、班级与课程之间的授课联系。一位教师可以针对多个班级讲授一门课程，一个班级的一门课程仅由一位教师授课，一个班级可以有多门课程由一位教师讲授。与授课这个活动有关的还有 3 个属性：学期、教室与时段。

在这个授课联系中，一个班级、一门课程可确定唯一的教师，所以教师为“一”的一方；一个班级、一位教师，可能对应多个课程，所以，课程为“多”的一方；与课程类似，班级为“多”的一方。

图 3.10 表达了这个三元联系，简单起见，其中略去了 3 个实体集的相关属性，保留了 3 个联系的属性：学期、教室和时段。

4. 一元递归联系与角色

联系大多发生在不同实体集中的实体之间。但有一种特殊的联系称为一元递归联

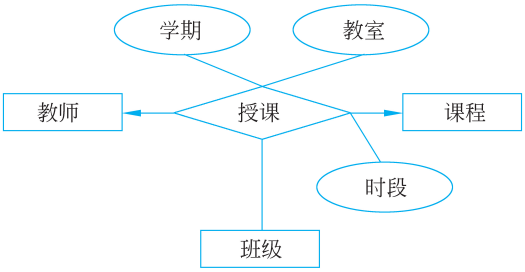

图 3.10　教师、课程、班级间的三元联系

系，也称环形联系，是在一个实体集内部的不同实体之间发生的。下面用两个例子加以说明。

例 3.3　一个公司的职工实体集中，职工与职工之间有直接上下级这样的联系，这种联系是职工实体集内部实体之间的联系，下级与上级之间的联系是多对一的，除了总经理之外，每一位职工仅由一位直接上级领导，每一位上级直接领导多位下级职工。图 3.11 给出了一个具体的上下级关系的例子。

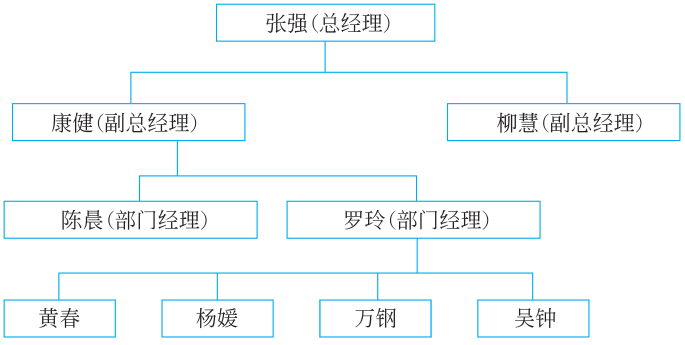

图 3.11　职工实体集内部上下级关系

这种一个实体集内部实体之间的联系被称为一元递归联系，其 E-R 图如图 3.12 所示。

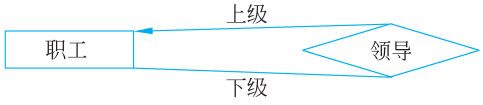

图 3.12　职工实体集的一元递归联系 E-R 图

在图 3.12 所示的一元递归联系中，连接实体集的两条线上均有文字标注："上级"和"下级"，这两个标注是说明实体集参与此联系时的角色。例如图 3.12 中"一"方实体表明其角色是上级，而"多"方实体表明其角色是下级，整个图表明多位下级由一位上级领导。

注意：当一个表达联系的菱形与一个表达实体集的矩形有多条连线的时候，一定要用角色进行标注，否则联系的表达含混不清。一元递归联系需要有角色的标注。角色也会出现在二元或多元的联系之中。

例 3.4　汽车厂生产的汽车由多个部件组成。例如，车门是汽车的部件，而车门又由

许多其他部件组成；螺丝钉是一个部件，汽车的很多部件都会用到螺丝钉，是螺丝钉的上级部件。因此汽车厂的部件之间是有“组成”这种联系的，而这种联系也是一元递归联系，并且联系的类型是多对多的，其 E-R 图如图 3.13 所示。

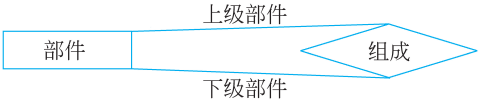

图 3.13　部件实体集的一元递归联系 E-R 图

这个例子说明，一元递归联系同样有一对一、多对一以及多对多等多种情形。

至此，本书介绍的 E-R 模型的相关知识已经可以表达大多数数据库应用问题的概念模型设计。

3.3.3　E-R 模型设计实例

下面给出一个较为完整的 E-R 图设计的例子。其中包含了最为常见的二元多对一以及多对多联系。

例 3.5　图书馆信息管理系统保存有关图书馆、图书、出版社、作者的相关数据。图书馆需要保存图书馆编号、名称、地址、电话，图书馆编号唯一标识图书馆。图书需要保存 ISBN、书名、类型、语言、定价、开本、千字数、页数、印数、出版日期、印刷日期。出版社需要保存出版社编号、名称、国家、城市、地址、邮编、网址，出版社编号唯一标识出版社。作者需要保存作者编号、姓名、性别、出生日期、国籍，作者编号唯一标识作者。

每一种图书收藏于不同图书馆，每个图书馆收藏若干图书，数据库保存收藏日期。每一种图书由一个出版社出版。每一种图书有若干作者，每一位作者编著/翻译若干图书。作者的类别为主编、编著者、译者。作者有相应的署名排位。

为了突出实体集之间的关联，本例将 E-R 图分为几部分。图 3.14 表达了实体集及其之间的联系，实体集属性仅标出码属性，实体集的其他属性没有给出。图 3.15～图 3.18 给出了相应实体集的完整属性。

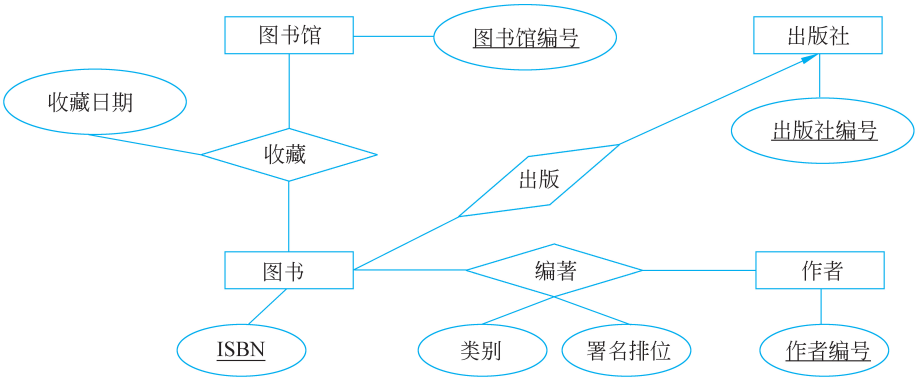

图 3.14　图书馆信息管理系统简略 E-R 图（实体集及其联系）

接下来，再看几个 E-R 图的例子。

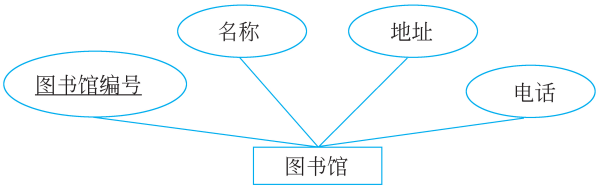

图 3.15　图书馆实体集与相关属性 E-R 图

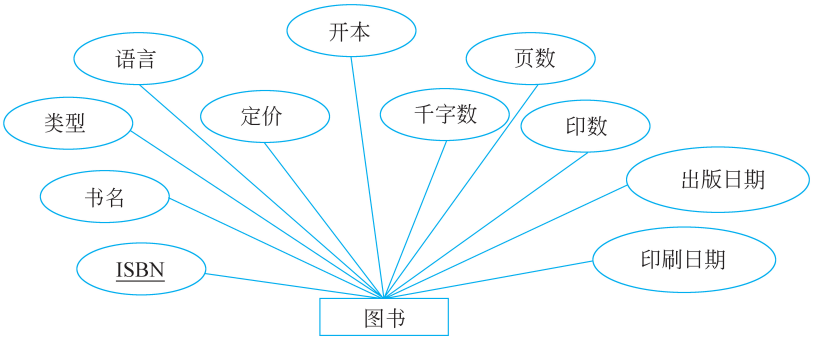

图 3.16　图书实体集与相关属性 E-R 图

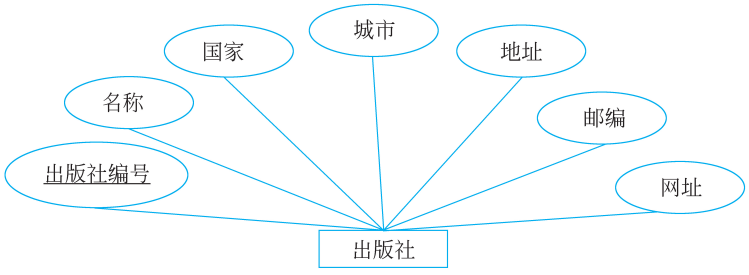

图 3.17　出版社实体集与相关属性 E-R 图

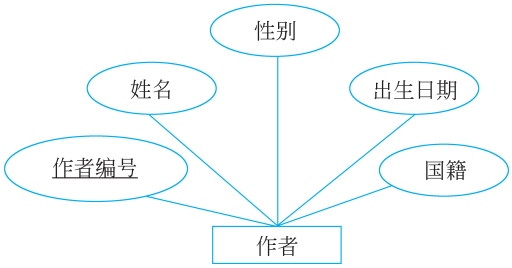

图 3.18　作者实体集与相关属性 E-R 图

例 3.6　教学信息管理数据库需要存储如下信息。学生信息保存学号、姓名、性别、年龄、手机号、Email，学号唯一标识学生。班级信息保存班号、专业、班主任号，班号唯一标识班级，班主任号为班主任职工号。课程信息保存课号、课名、学分、类别，课号唯一标识课程。教师信息保存职工号、姓名、性别、出生日期、职称、专业方向。

每一位学生属于一个班级，每一个班级有若干学生。每一个班级仅有一位班主任，每

一位教师可做多个班级的班主任。每一位学生可以选修多门课程，每一个课程可以有多位学生选修，系统记录选课成绩。一个班级的一门课程仅由一位教师授课；一位教师可以为一个班级讲授不同课程；一位教师可以为不同的班级讲授同一门课程；系统记录授课的学期、教室和时段。

图 3.19 给出了相应的 E-R 图。有关实体集与属性的完整表达方式参看例 3.5，这里不再赘述。

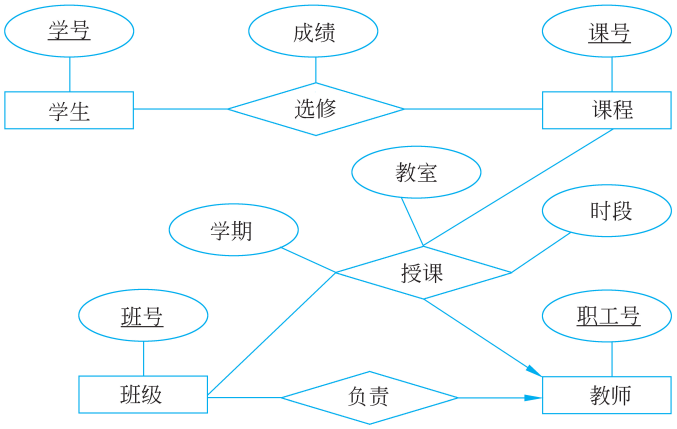

图 3.19 教学信息管理数据库 E-R 图（实体集及其联系）

例 3.7 为航空公司建立航班信息管理数据库需要存储如下信息。关于航线保存航线号、出发地、到达地、飞行距离，航线号唯一标识航线。关于航班保存航班号、日期、起飞时间、到达时间，航班号唯一标识航班。关于乘客保存乘客编号、姓名、性别、出生日期，乘客编号唯一标识乘客。关于空勤人员（后面称职工）保存职工号、姓名、性别、年龄、工龄、职务（机长，驾驶员，乘务长，乘务员等），职工号唯一标识职工。关于飞机类型保存每一类型飞机的相关属性，包括机型、名称、通道数、载客人数、制造商，机型唯一标识飞机类型。

每一航线有不同航班，每一航班飞唯一航线。每一航班对应唯一飞机类型，每一飞机类型对应不同航班。每一航班有唯一的机长，机长是航班的机组职工之一，同一人可以是不同航班的机长。每一职工可以工作于不同的航班，每一航班的机组有若干职工。乘客可以乘坐不同的航班，每一航班可以有多位乘客乘坐，系统记录乘客的座位号。

图 3.20 为对应的 E-R 图。有关实体集属性的完整表达形式参看例 3.5，这里不再赘述。

例 3.8 设计某高校实验室管理系统。每一个科研小组有唯一的实验室，该实验室仅供该小组使用。一个科研小组有若干教师，教师属于唯一的科研小组。一位教师有若干研究生，每一研究生由唯一的教师作其导师。实验室拥有若干计算机。每位教师及其研究生可使用多台计算机，每一计算机归一位教师管理。每一位研究生在校期间使用唯一的计算机，在不同的时期，计算机可以由不同的研究生使用。科研小组保存组号、名称、人数、研究方向，组号唯一标识科研小组。教师保存职工号、姓名、职称，职工号唯一标识教师。研究生保存学号、姓名、生日，学号唯一标识研究生。计算机保存计算机编号、厂家、

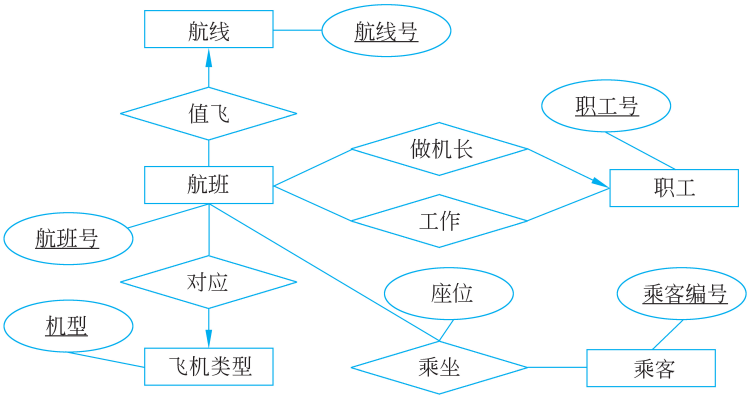

图 3.20　航班管理 E-R 图（实体集及其联系）

类型，计算机编号唯一标识计算机。实验室保存实验室编号、名称、面积，实验室编号唯一标识实验室。

图 3.21 给出了对应的 E-R 图。有关实体集属性的完整表达形式参看例 3.5，这里不再赘述。

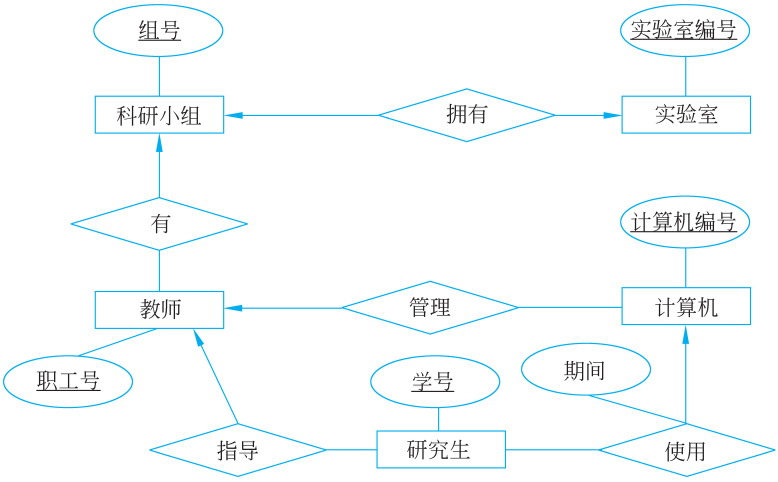

图 3.21　某高校实验室管理系统 E-R 图（实体集及其联系）

3.3.4　设计中的常见问题

至此，已经介绍了 E-R 图的基本表示方法。使用这些方法足以解决大多数数据库应用的模型设计问题。但是，对于初学者来说，在模型设计中仍然会有许多困惑。本节讲解模型设计中的常见的问题。

1. 客户需求

E-R 图是概念模型设计的工具，设计概念模型的最终目的在于正确表达用户的需求。因此，设计 E-R 图的时候，一定要与用户沟通，根据需求进行设计。

首先，需要注意系统边界问题。现实世界的信息有很多，但任何一个数据库系统都有

其具体的设计目标，数据库仅仅保存满足系统业务需要所必须存储的信息即可。例如，就公民信息而言，人口普查系统不必关注公民的银行账号和健康情况，这就需要设计者在设计数据库时要全面理解系统的功能需求和业务流程，从而能恰当地确定各实体集的属性。

其次，需要注意联系的类型问题。初学者看到书上的例子中说学生与班级为多对一联系，常常误以为这两者之间的联系类型就是多对一无疑，但实际情况并不总是如此。在高校中学生与所在班级的确如此，并且上大学期间，大多数情况下学号是不变的，班号也是不变的。那么在中小学呢？一些小学的班号随着每一年升到高年级而改变。一些中学不断按照考试的排名重新分班，学生会不断地变换班级。因此，在不同的情境下，学生与班级的联系的类型是不同的。在进行设计的时候，一定要了解用户的应用背景、具体的需求，根据需求确定合适的联系及其类型，从而保证概念模型（E-R 图）对现实世界的真实性。

2. 冗余问题

初学者在设计 E-R 图的时候经常出现冗余的信息。例如，图 3.22 中的虚线椭圆表达的属性就是冗余的属性。因为选课的菱形本身已经表达了学生与课程的联系，如果再用学号描述联系显然是多余的；同理，课号也是多余的属性。另外，从实体描述的角度看，学号也只能是学生的属性，不应该画在菱形上；课号也是如此。

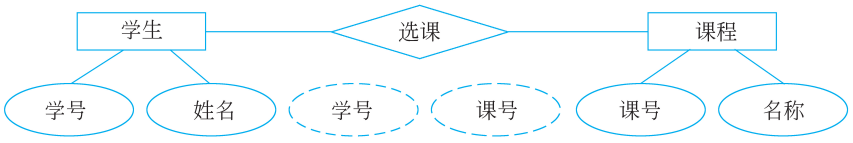

图 3.22　有冗余属性的错误设计

又如，图 3.23 中的设计出现了冗余的联系，即用虚线菱形表达的“属于”，这是将例 3.8 中有关研究生的信息略去后重画的 E-R 图。其中增加了计算机与实验室之间的“属于”联系，这样看起来很合理，计算机的确是属于实验室的。但是，因为教师属于唯一的科研小组，对应唯一的实验室，计算机由唯一的教师管理。通过这些联系，已经表达了计算机属于实验室这样一个事实。此时，再加上虚线部分，就出现了冗余的联系。冗余的联系在后期进行数据库逻辑设计的时候会产生有冗余的关系模式和基本表设计，这些冗余会给应用带来诸多后患。

3. 属性与实体集

在设计数据库系统的时候，会出现关于属性与实体集的困惑。例如，班号是作为学生的一个属性还是作为一个实体集？

这个问题取决于用户的需求。如果数据库系统不需要班级的各类信息，仅仅关注学生的班号，那么，就不必设计班级实体集，而仅仅需要在学生实体集上添加班号这一属性。

通常，一个实体集除了包含唯一的标识（码）所需要的属性之外，还有一些其他的属性。例如，班级除班号之外还有类型（如实验班）、专业方向等，这时班级就应该设计为实体集。

4. 联系与实体集

通常，联系对应于活动，例如选课、购买等，在需求规格说明书中通常用动词表达。而

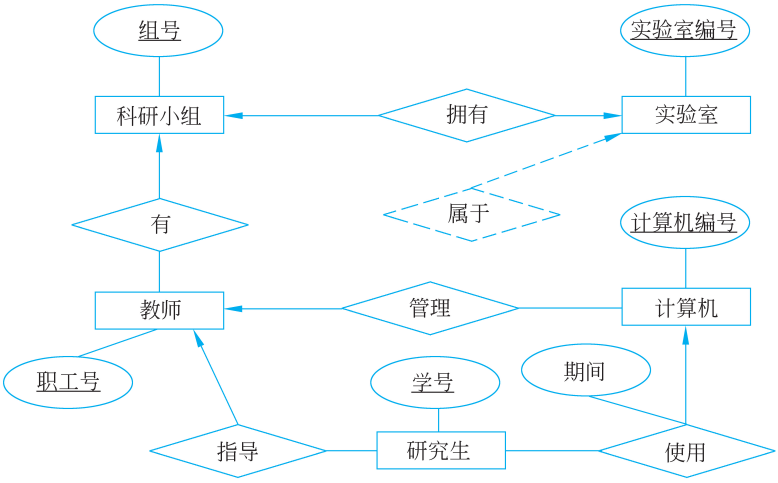

图 3.23　有冗余联系的错误设计

实体集对应于事物，在需求规格说明书中通常用名词表达。但在一些特殊的情况下，“选课”这样一个活动更适合用“选课记录”表达，这属于 3.3.5 节要介绍的弱实体集的情况。

3.3.5　子类实体集与弱实体集

现实世界千变万化，随着计算机技术的发展，应用问题的需求也越来越丰富，数据库技术的发展必然要适应这一变化。传统 E-R 模型也有了很多扩展，本节介绍这些扩展中的子类实体集与弱实体集。

1. 子类实体集

从一个实际的问题入手看看什么是子类实体集以及为什么要引入子类实体集的概念。

例 3.9　高校里的学生实际上分为本科生、研究生。本科生包含留学生，研究生分为硕士生与博士生，全日制研究生与在职研究生等。简单起见，在此仅仅考虑本科生、研究生两类。所有的学生都有学号、姓名、性别、生日等属性。本科生除了学生的公共属性之外，班级信息也十分重要，许多教学活动都按照班级来安排。研究生除了有作为学生的公共属性之外还有导师、研究方向等特殊属性。

作为学生，无论是本科生还是研究生，都有一些共有的活动，例如选课、参加社团、参加比赛等，这些活动对应着与其他实体集的联系。对研究生还要求参加学术活动，数据库中需要进行记载；本科生虽然也会参加学术活动，但数据库中无须记载，这里忽略它。也就是说，设计中仅仅考虑研究生有特殊的联系——参加学术活动。

于是，学生实体集有共同的属性和联系。而作为特殊的学生，研究生有特定的属性和联系，本科生则有特定的属性，用 E-R 模型表示如图 3.24 所示。

一般来讲，当若干实体集有公共属性和联系的时候，应该设计拥有这些公共属性和联系的超类实体集，称为父类实体集，而将父类实体集细分之后具有特殊联系和属性的实体集定义为子类实体集，子类实体集继承其父类实体集的所有属性和联系。

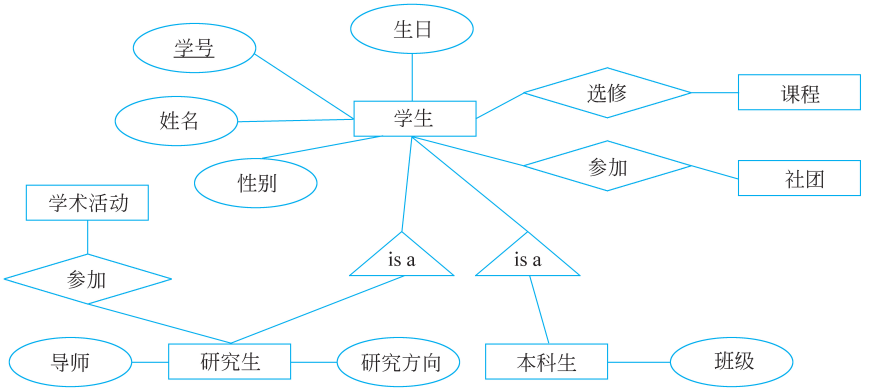

图 3.24　含本科生和研究生子类实体集的 E-R 图

在 E-R 模型中，子类实体集仅仅画出其所拥有的特殊的属性和联系。子类实体集与其父类实体集之间使用含有文字“is a”的等边三角形进行连接，以表达继承的关系，等边三角形的顶点与父类实体集相连，底边与子类实体集相连。

不是所有的子类实体集都需要在 E-R 图中表示，如例 3.10 所示。

例 3.10　将例 3.9 的需求修改一下，只考虑研究生的导师和研究方向两个特殊属性以及参加学术活动的特定联系，不考虑本科生的班级属性，则 E-R 图如图 3.25 所示。

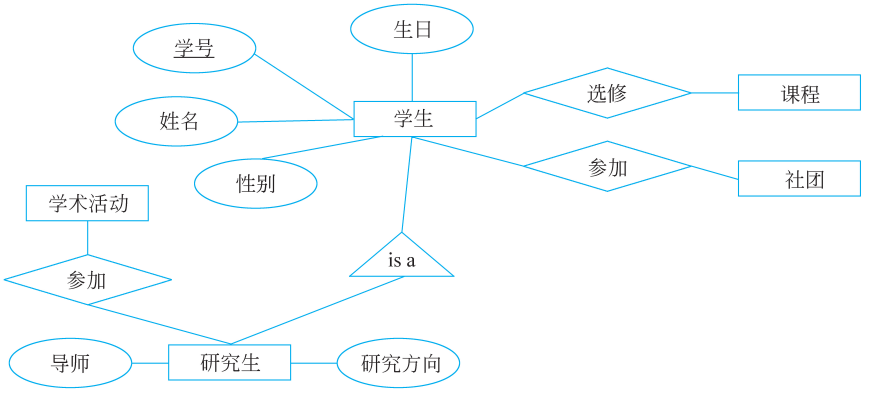

图 3.25　无本科生子类实体集的 E-R 图

为什么要引入子类实体集的概念呢？除了现实世界中存在固有的超类和子类关系这样一个原因之外，还有下面的原因。针对例 3.9，假定某一时刻有本科生和研究生的数据，如表 3.1 和表 3.2 所示。

表 3.1　本科生信息

学　号	类 别	姓 名	性别	生　日	本科班级
10070101	本科	张易	男	19921012	100701 班
10070002	本科	王东	男	19920123	100700 班
10070001	本科	吕平	女	19920908	100700 班
10070201	本科	陈明	男	19930715	100702 班

表 3.2　研究生信息

学　号	类别	姓名	性别	生　日	导师	研究方向
S19950002	研究生	肖剑	男	19901112	黄格	人工智能
S19950112	研究生	刘莹	女	19900318	董士林	模式识别

如果把本科生与研究生分别作为两个实体集考虑，那么作为参加社团的联系就要分别从两个实体集引出，但是本质上是学生到社团的一个联系，而且作为学生的公共联系，要从本科生和研究生两个实体集中去找，很不方便。

再考虑采用下面的方案，把本科生和研究生的所有信息都放在一起，那么这个合并后的学生表如 3.3 所示。

表 3.3　本科生与研究生信息

学号	类别	姓名	性别	生日	本科班级	导　师	研究方向
10070101	本科	张易	男	19921012	100701 班		
10070002	本科	王东	男	19920123	100700 班		
10070001	本科	吕平	女	19920908	100700 班		
10070201	本科	陈明	男	19930715	100702 班		
S19950002	研究生	肖剑	男	19901112		黄格	人工智能
S19950112	研究生	刘莹	女	19900318		董士林	模式识别

这个设计有什么问题呢？本科生没有导师和研究方向，这两列为空；研究生没有对应的班级信息，相应列为空。在使用定长字符串类型的时候，这将会浪费大量空间。当数据量巨大的时候，因为空值带来更多空间的占用，会增加磁盘输入输出的负担，也会降低查询的命中率，从而影响查询效果。真实的学生数据库包含大量的数据，这种空间浪费的问题是十分严重的。

因此，数据库研究者对 E-R 模型进行了扩展，增加了子类实体集的概念。子类实体集的提出，本质上是为了解决超类以及各个子类信息的空间浪费问题。

在设计中，一些实体集具有共同的属性或联系，应该将这些共同点抽取出来，设计一个超类实体集。超类实体集拥有这些共同的属性和联系。而子类实体集仅仅需要在 E-R 图中画出它们独有的属性和联系。子类实体集与超类实体集之间使用含“is a”的等边三角形连接。各个子类实体集的并集不一定等同于超类实体集。子类实体集还可以有下级的子类实体集。

子类实体集与下面介绍的弱实体集不同。子类实体集是具有特定属性和联系的实体集。子类实体集的成员对应着超类实体集中的唯一成员，故而，在后面的逻辑设计中，子类实体集与超类实体集具有相同的码。

2. 弱实体集

为了便于理解，先通过例子介绍弱实体集，然后再对其进行严格定义。

例 3.11 图书馆的图书是同一个书号确定的同一种图书，通常都有许多副本，读者借书时也仅仅借走这种图书的一个副本。这样就带来以下问题，数据库中每一本图书的标识是什么？即标识一本图书的关键字是什么？使用习惯的 ISBN 吗？那么一个 ISBN 对应一种图书的多个副本，应该怎么处理呢？

如果没有引入弱实体集的概念，用前面介绍的 E-R 图表示法，图书的标识应该是 ISBN 加上副本号。例如，《高等数学》的 ISBN 为 9787040396621，则 9787040396621-1、9787040396621-2 分别对应图书馆中该书的副本 1 和副本 2。

但是，在数据库中，ISBN 为 9787040396621 的这种图书的基本信息以及它与出版社、作者等的关联需要独立地表达，需要以 ISBN 为码。这将带来设计上的困惑。如果不使用弱实体集概念，将图书与图书副本设计为两个独立的实体集，图书副本的码应设计为(ISBN，副本序号)这样一个组合码，而这样的设计使得 E-R 模型的表达违反一事一次的原则，即一个 ISBN 在 E-R 图中多次出现。

下面引入弱实体集的概念。

弱实体集是这样的实体集：它依附于其他实体集，它的码的一部分取自其所依附的实体集。一个弱实体集可以依附若干实体集，该弱实体集的码中的一部分分别取自其依附的不同实体集。

图书副本是一个弱实体集，依附于图书，因此图书副本的码的一部分来自其依附的图书的码，即 ISBN。本例中图书副本的码是(ISBN，副本序号)。

E-R 图中使用双线矩形表达弱实体集，通过双线菱形指向其依附的实体集。图 3.26 是用弱实体集表达的图书副本的例子。图书仅仅给出 ISBN 和书名两个属性，其他属性没有画出；图书副本给出副本序号和购买日期两个属性，本例中只有副本序号参与码的构成，购买日期则作为普通属性。

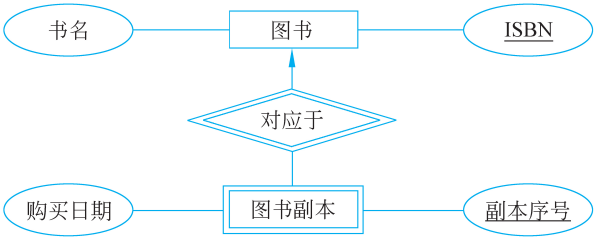

图 3.26 含图书副本弱实体集的 E-R 图

E-R 图中弱实体集用双线矩形标识，弱实体集与其依附的实体集之间的联系用双线菱形标识。使用双线矩形和双线菱形是为了给设计者提供明显的提示，将来做数据库逻辑设计的时候，弱实体集(如图书副本)的码需要顺着双线菱形的指向添加其依附的实体集的码。

这样的设计便于表达图书与出版社、作者等的关联，图书副本与读者、书库等的关联，同时，也明确给出了图书副本与图书之间的关联。

再看一些弱实体集的例子。

例 3.12 在大学里，学生属于唯一的班级，一个班级有若干学生；班级属于唯一的系，

一个系有若干班级。考虑以下两种情况。

第一种情况，学号唯一标识学生，如学号为“12070115”指的就是“王利”这个学生；班号唯一标识班级，如班号为“120701”指的就是“计算机实验 1 班”。那么，这里不存在弱实体集的情况，没有哪一个实体集的码的一部分取自其他实体集。这种情况下的 E-R 图如图 3.27 所示，其中的实体集仅仅标出码属性。

第二种情况，学号为班级的内部编号，也就是说，每一个班级的学号都是 1～n，不同班级中的学生会存在学号相同的情况，例如每个班都有 1 号学生；班号为系的内部编号，例如 1201 班，表示 12 级 1 班，不同的系都会出现 1201 班。这种情况下，学号本身不能唯一标识学生，需要依附于班级帮助其进行标识，因此学生是弱实体集；班号本身不能唯一标识班级，需要依附于系帮助其进行标识，因此班级也是弱实体集。此时的 E-R 图如图 3.28 所示，其中的实体集仅仅标出码属性。在这个例子中，班级的码为（系名，班号），学生的码为（系名，班号，学号），这样一来使用时有诸多不便，但是如果用户的需求是这样的，也只能将相应的实体集设计为弱实体集。

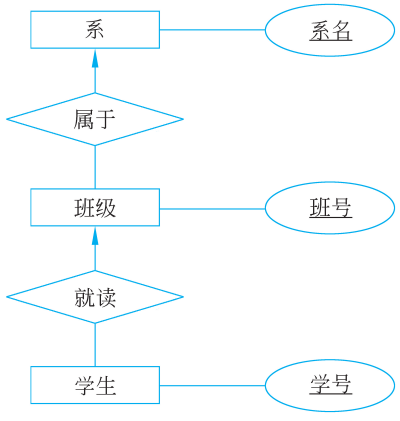

图 3.27　学生和班级为非弱实体集的 E-R 图

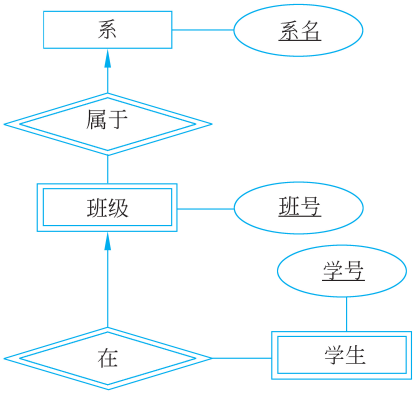

图 3.28　学生和班级为弱实体集的 E-R 图

弱实体集的本质是自身的码有一部分取自其他实体集，班级是否为弱实体集取决于数据库内班号是否唯一标识班级，这与用户提出的需求有关，E-R 图作为概念模型的设计手段是根据用户需求进行设计的。学生是否为弱实体集，道理也是一样的。

再看一个例子。

例 3.13　在学生选课系统中，每一位学生选修一门课程会有一个最终成绩。此外，每一位学生的每一门课有许多次单元测验，还要记录单元测验的序号以及测验成绩。如果按照常规将学生与课程设计为多对多联系，课程的总成绩是联系的属性，这些信息用 E-R 图表达没有什么问题。但是单元测验怎么处理呢？每一位学生与一门课程的选课活动对应多次单元测验，选课活动与单元测验是一对多联系，怎么表达呢？

采用前面介绍过的 E-R 图表示法，菱形都出现在矩形之间，也就是说联系都出现在实体集之间。现在的问题是选课活动这种联系与单元测验实体集之间发生了联系。

弱实体集恰恰可以表达这样的特殊情况。将选课这个活动不按照通常的方式用菱形表达为联系，而把它设计为选课记录实体集，而这个实体集的码取自学生实体集和课程实