第3章 chapter 3

数据模型

本章学习目标

本章重点介绍数据模型的基本概念，从实体及实体间的联系、实体的属性出发，讨论了实体联系模型的意义与绘制方法，讲述了关系数据模型的概念与设计方法，介绍了从实体联系模型得到关系数据模型的方法与步骤。通过本章学习，读者应该掌握以下内容：

- 实体与实体之间的联系方式；
- 三种数据模型及其比较；
- 关系数据模型与模式；
- 从 E-R 模型到关系数据模型的转换方法。

数据库技术要求尽可能高效地获取和处理数据、更广泛地共享数据、更方便与可靠地处理数据，这些都要求设计最能满足上述要求的数据模式。设计数据模式一般采用的是先建立数据模型，再求取数据模式的方法。

3.1 数据模型概述

数据库是某个企业、组织或部门所涉及的数据的存储库，它存放所有的数据并且反映了数据彼此之间的联系。设计数据库系统时，先用图或表的形式抽象地反映数据彼此之间的关系，称为建立数据模型。常用的数据模型一般可分为两类，一是语义数据模型，如实体_联系模型(E-R 模型)、面向对象模型等；二是经典数据模型，如层次模型、网状模型、关系模型等。第一类模型强调语义表达能力，建模容易方便，概念简单、清晰，易于用户理解，是现实世界到信息世界的第一层抽象，是用户和数据库设计人员之间进行交流的语言。第二类模型用于机器世界，一般和实际数据库对应，例如层次模型、网状模型、关系模型分别和层次数据库、网状数据库和关系数据库对应，可在机器上实观。这类模型有更严格的形式化定义，常需加上一些限制或规定。设计数据库系统时通常利用第一类模型作初步设计，之后按一定方法转换为第二类模型，再进一步设计全系统的数据库结构。全系统的数据库结构通常包括数据结构、数据操作和完整性约束三部分内容，数据模型也应体现这三方面的内容，才能顺利地完成数据模式的设计。

1. 数据模型应表现数据结构

数据结构描述的是数据库数据的组成、特性及其相互间联系。在数据库系统中通常按数据结构的类型来命名数据模型，例如层次结构、网状结构和关系结构的模型分别命名为层次模型、网状模型、关系模型。这些数据模型都要能反映数据的情况及数据之间联系的情况。

2. 数据模型要定义数据操作的内容

数据操作指对数据库中各种对象的实例允许执行的操作的集合，包括操作及有关的操作规则。数据库的操作主要有检索和维护（包括录入、删除、修改）等两大类操作。数据模型要定义这些操作的确切含义、操作符号、操作规则及实现操作的语言。数据结构是对系统静态特性的描述。数据操作是对系统动态特性的描述。

3. 数据模型应描述数据的约束条件

数据的约束条件指数据完整性规则的集合，它是给定数据模型中数据及其联系所具有的制约和依存规则，用于限定符合数据模型的数据库状态及其变化，以保证数据的完整性。

数据模型关于上述三方面的内容，完整地描述了一个应用系统中数据与数据之间的联系，而其中数据结构是首要内容，下面介绍各种数据模型数据结构的表示方法及各种模型设计方法和它们之间的关系。

3.2 E-R 数据模型

3.2.1 数据之间的联系

现实世界的事物之间彼此是有联系的，代表实体的数据之间也存在联系，对于不同实体集之间的实体与实体的联系按关联的数量可分为三类。

1. 一对一联系（1∶1）

若对于实体集 A 中每一个实体，实体集 B 中至多只有一个实体与之联系，反之对于实体集 B 中每一个实体，实体集 A 中也至多只有一个实体与之联系。则称实体集 A 与实体集 B 之间具有一对一联系，记为 1∶1。

例如，企业和法人代表的联系，假如每个企业有一个法人代表，而一个法人代表只在一个企业任职，则它们是一对一的联系。一般来讲，如果 B 是 A 的代表，或者是 A 中独具特色的内容，则 A 和 B 的联系是 1 对 1 的。

2. 一对多联系（1∶N）

若对于实体集 A 中的每一个实体，实体集 B 中可有 N 个实体（$N \geqslant 0$）与之联系。而

对于实体集 B 中的每一个实体，实体集 A 中至多只有一个实体与之联系，则称实体集 A 与实体集 B 有一对多的联系，记为 $1:N$。

例如，一个系可有多个教研室，而一个教研室只属于一个系，则系与教研室是一对多的联系。一个教研室内有许多老师，而一个老师只属于一个教研室，教研室和老师之间是一对多的联系。一般来讲，如 A 是 B 的领导实体或 B 与 A 是所属关系或 A 和 B 存在层次关系，A 和 B 之间就是一对多的联系，如 B 是组成 A 的部分，而 B 中一个实体只组成到 A 中某一个实体之中，则 A 与 B 也是一对多的联系。

3. 多对多联系（$M:N$）

若对于实体集 A 中的每一个实体，实体集 B 中可有 N 个实体（$N\geq 0$）与之联系，反过来对于实体集 B 中的每一个实体，实体集 A 中可有 M 个实体（$M\geq 0$）与之联系，则称实体集 A 与实体集 B 之间有多对多联系，记为 $M:N$。

例如，前面所举学生与课程之间的联系是多对多的联系。如果一个产品由多个零件组成，而一个零件可组成到多个产品之中，则产品和零件是多对多的联系。一般说，如果 A 和 B 存在工作、学习、业务、组成等方面的关系，A 与 B 是多对多的联系。例如，A 中一个实体由 B 中多个实体组成，而且 B 中一个实体可能被组成到 A 的不同实体中，则 A 与 B 是多对多的联系。

实体集与实体集间联系的性质与系统环境有关，与研究的问题有关，某些问题与前提条件有关。例如，A 与 B 为组成关系，则它们的联系有可能为一对多的，也可能为多对多的，例如产品与部件之间、产品与零件之间的关系可能为一对多的，也可能为多对多的。在一个学校中，每个老师最多担任一门课主讲，但有些课程有多位主讲老师，则课程和老师间是一对多联系。在一个学校中，如果有的老师要担任多门课主讲，且有一些课程有多位主讲，则课程和老师之间是多对多联系。在一个学校中如果有的教师担任多门课主讲，而所有课程均只有一位主讲，则课程和老师间是多对一的联系。

如果在一个系统中，实体之间联系都是一对一或一对多的，可采用层次模型来描述，使用层次数据库效率高。如学校与系、系与教研室、教研室与老师、教研室与学生等之间联系都是一对多的，那么由学校、系、教研室、老师、学生构成的系统可采用层次模型来描述。

凡采用层次模型可以描述的问题也可以使用关系数据模型。设计时，模型 A 和 B 如果是一对一的联系，常可将 A 和 B 的数据合并存放在一个表之中。A 和 B 如是一对多的联系，则需将 A 和 B 的数据分别存放在两个表之中，而且要在 B 中增加一个字段：A 中的关键字。以便回答既涉及 A 又涉及 B 的检索和统计的应用问题。使用关系数据库实现容易，但效率可能较低。

如果在一个系统中，实体和实体间存在多对多的联系，则常采用网状数据模型来描述、使用网状数据库存储数据，其查询速度较快。

也可用关系数据模型来描述，A 和 B 的数据分别存放在两个表之中，同时另外建立联系关系：分别由 A 和 B 的关键字数据项构成联系关系表（如 1.2 节所举的关于学生和课程的例子）。

在一些问题中，可能涉及三个实体集之间的联系，例如在老师、学生、课程关系中，如老师和学生、学生和课程、老师和课程间均是多对多关系。而且要讨论某学生学习某课程是哪个老师教这一类问题，这时，仅建立老师和学生间联系及学生和课程之间联系关系还不能回答此问题。实际上它们三者是 $M:N:P$ 的联系。在关系模型中也较容易处理，方法是三者分别建立一个表，另外建立联系关系表，由三者的关键字作为数据项。老师、学生、课程相互关系问题可用“教学(学号、职工号、课程号)”来表现。

3.2.2 实体联系模型

实体-联系模型(Entity-Relationship Model)是 P. PS. Chen 于 1976 年提出的一种概念模型，用 E-R 图来描述一个系统中的数据及其之间关系。在 E-R 图中，用矩形表示实体集，在矩形框内写上实体名。用菱形表示实体间联系，菱形框内写上联系名。用无向边把菱形和有关实体相连接，在无向边旁标上联系的类型，如 1 或 M 或 N。用椭圆形或表格表示实体或联系的属性。如用椭圆形，将它与一个相应实体间以无向边相连。如以表格形式，表示方法为：实体名(属性 1，属性 2，…)。如果在例 1.2 中，添加描述如下：各教研室负责开出各专业课程，一个学生要学习许多课程，一门课程常有许多学生学。一个老师要带多门课，有一些课程有多位主讲教师。可画出其 E-R 模型如图 3.1 所示。

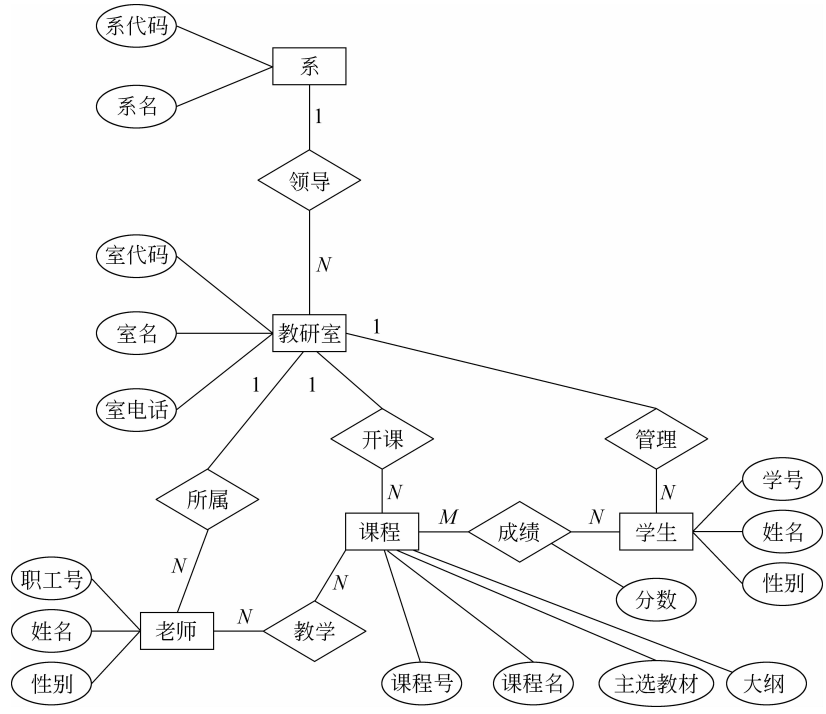

图 3.1 一个学校的 E-R 图

图 3.2 为另一种画法，将属性用“表”的形式表示，更突出实体及其之间的联系。

要注意以下几个问题：

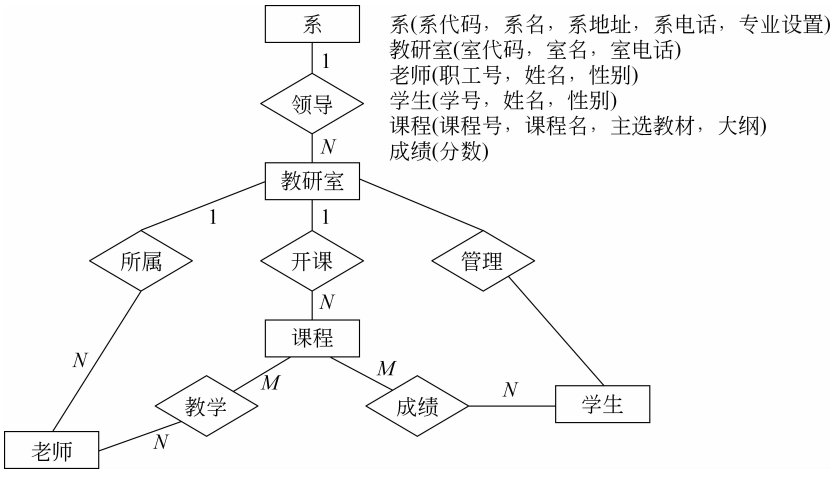

图 3.2　一个学校系统的 E-R 图的另一种画法，属性用“表”表示

(1) 某些联系，也具有属性，例如“成绩”有属性“分数”，它既非为学生独有，也非为“课程”独有，是涉及学生学习某门课程关系时产生的特性，因而是联系的属性。

(2) 对于三个实体 $M:N:P$ 的联系，如老师、学生、课程间联系可如图 3.3 所示描述。

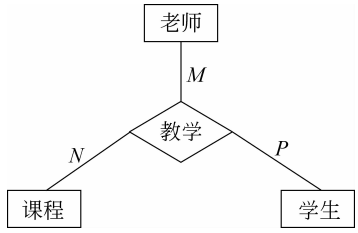

图 3.3　涉及三个实体的 E-R 图

(3) E-R 图可以表现一个实体内部一部分成员和另一部分成员间的联系。在一个学生班中，班干部和一般学生都是学生，但班干部和一般学生间存在一对多联系，可表示为图 3.4，这类联系称为自回路。

(4) E-R 图可以表现两个实体集间的多类联系。例如一个单位中职工和工作的关系，一个职工可承担多项工作，一个工作一般有多人承担，这种工作关系是多对多的关系。另外，有一些职工对一些工作是主要责任人，一个职工可对多项工作负责，但一项工作只有一个责任人，它们之间这种负责关系为一对多联系，可用图 3.5 描述。

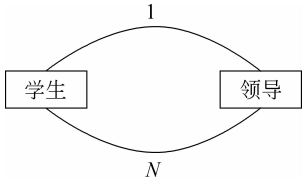

图 3.4　自回路的 E-R 图

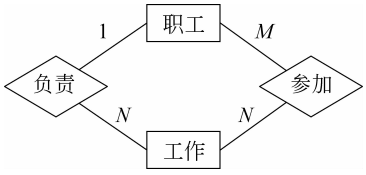

图 3.5　两个实体间有多个联系的 E-R 图

从以上 E-R 模型可见 E-R 图表述简单，与现实世界较接近。其画法：首先分析一个系统中有哪些实体集，分别用矩形框画出；再分析它们每两者之间是否存在联系，是什么类型的联系；如有联系，用菱型框表示并与相关实体框用线相连，标上联系的类型；最后分析各个实体集及联系的属性，用“表”表示就可得到 E-R 图。

这中间比较困难的是，有些时候实体与属性不易区分，对于一些具体问题，如对实际

数据构成了解不透，对环境和应用需求调查不深入，不容易弄清联系关系及联系的类型，容易漏或错。这些内容将在后面进一步讨论。

此外，根据这种模型无法直接建立机器上的存储结构，因此还要进一步转化出某一种经典数据模型。

3.3 关系数据模型

3.3.1 关系数据模型基本概念

用二维表格数据（即集合论中的关系）来表示实体和实体间联系的模型叫关系数据模型。它是经典数据模型中建模能力最强的一种，对于各种数据联系类型都可描述。它以关系理论为坚实的基础，因此成为当今实用系统的主流。目前流行的数据库管理系统，如 Oracle、Informix、Sybase、SQL Server（以上是网络系统上常使用的数据库）和 Access、Visual FoxPro（这两个是客户机上常使用的数据库）全都是关系数据库管理系统。

关系数据模型用二维表表示实体集。二维表由多列和多行组成，每列描述实体的一个属性，每列的标识称为属性名，在关系数据库中称为数据项或字段。表中每一行称为一个元组，描述一个具体实体，在关系数据库中称为记录，元组的集合构成表，称为关系，描述一个实体集中各类数据的集合，在关系数据库中也称为表。关系数据模型由多个关系表构成，每个表表示为：关系名（属性 1，属性 2，…，属性 n），例如，学生（学号，姓名，性别，出生年月，专业，班级，政治面貌，家庭住址，履历）。

在一个关系的属性中有的属性或属性组能唯一标识一个元组，称为主码，或称为关键字。

有些属性取值有一定范围，属性的取值范围称为域。一个域对应关系数据库中的表中的一个数据项的值的集合。多个数据项可对应同一个域。例如中国人“家庭住址”值的构成是汉字与数字的集合，汉字与数字的集合可以称为汉字数字域，即“家庭住址”对应汉字数字域；显然，“履历”也对应汉字数字域。

元组中一个属性值称为分量，对应关系数据库中一条具体记录的一个数据项的具体值。

在关系模型中对于联系有不同表示方法，例如对于一对多联系，可在“多”方实体集的表中加进“一”方实体集的主码。对于多对多联系则可以建立一个新表，由“联系”的两个实体的主码及“联系”自身的属性作为其属性。

在关系数据库中用户的检索操作实际是从原来的表根据一定的条件求得一个新表。

以上所述，关系模型概念单一，无论是实体还是联系，无论是查询检索源还是检索结果集都用二维表表示，结构清晰，用户易懂易用，容易维护，适应性强，容易扩充，其坚实的理论基础使之严密细致，这些都使它长期成为实用数据库系统的主流。

对于关系模型还要指出几点：

（1）关系是元组的集合，元组在关系中的顺序不影响关系。

(2) 同一关系任意元组不允许全同。对于每一表,要选定或设计主码,用于区分不同元组。

(3) 关系的每一属性都是不可再分的基本数据类型,这种特性称为原子性。如表 3.1 所示的表结构是不允许的。其中成绩可再分为单科成绩和总分两个数据项,且单科成绩还可再分为 C、OS、DB 三个数据项。成绩称为组项,组成它的两个数据项是不同级别的,单科成绩称向量,构成单科成绩的三个数据项是同级的。向量是组项的一种,是特殊的组项。

表 3.1 组项一例

姓名	成绩			
	单科成绩			总分
	C	DB	OS	
张三	90	85	65	240

在关系模型中不允许存在组项和向量。另外如表 3.2 所示的结构也不允许,这种结构称为重复组,其中有的行又分成多行,在关系模型中不允许存在重复组。

表 3.2 重复组一例

姓名	课名	分数
A1	C	80
	DB	85
	OS	92
A2	C	90
	DB	80

(4) 在一个表中属性排列顺序可以交换,不影响关系。

(5) 允许属性值为空值(null value),表示该属性值未知,空值不同于 0,也不同于空格。它使关系数据库支持对不完全数据的处理。在表中,不允许主码全部或部分为空值,否则它就无法唯一标识一个元组。

3.3.2 从 E-R 数据模型到关系数据模型

在数据库系统设计时,首先画出 E-R 模型后可继而转化出关系模型,画法为:

(1) 将每一个实体型(矩形)用一个关系表示,实体的属性就是关系的属性,实体的码就是关系的主码。例如图 3.2 中关于“教研室”要建立一个关系:教研室(室代码、室名、室电话),其中“室代码”是主码。对于一对一的联系可将原两实体合并为一个关系表示,关系属性由两个实体属性集合而成,如有的属性名相同但意义不同,则应加以区分。

(2) 对于一对多的联系,在原多方实体对应的关系中,添加一方实体的主码,多方实体主码是多方对应关系的主码。例如图 3.2 中关于“教研室”关系中需要增加数据项:

“系代码”，是系关系中主码，但不是“教研室”关系的主码；同样的道理，在老师和学生关系中要增加数据项：“室代码”，是对应的教研室关系的主码。

(3) 将多对多的联系转换为新关系，联系名为关系名，联系的属性加上相关两实体主码构成关系的属性集，相关两实体主码的集合是联系关系的主码。例如将联系集“成绩”转换为关系“成绩(学号、课程号、分数)”，其属性包括“成绩”的属性“分数”及相联系的两个实体集：“学生”与“课程”的主码：“学号”与“课程号”，它将是联系“学生”和“课程”两个关系的关系，(学号、课程号)构成“成绩”关系的主码。

综合上述，基于图 3.2 描述的 E-R 图转化为关系数据模型的步骤是：

(1) 分析图中实体集的情况。图中有系、教研室、老师、学生、课程等实体集，分别用系、教研室、老师、学生、课程等关系表示。

(2) 分析图中所有一对多联系的情况。系与教研室、老师、学生为一对多联系，因此在图 3.2 所示的 E-R 图的教研室、老师、学生等属性表中增加系的主码：系代码。教研室和课程之间也是一对多联系，在课程属性表中增加教研室的主码：室代码。

(3) 分析图中所有多对多的联系。老师与学生、学生与课程间是多对多联系，因此要建立两个联系关系：教学与成绩，其中教学的属性集为老师实体与课程实体的主码：职工号、课程号，成绩的属性集为学生实体与课程实体的主码：学号、课程号，另外分数是联系关系“成绩”原来自有的属性，保留在成绩关系中。

最终可以得到系统的关系数据模型如下。

系(系代码,系名,系地址,系电话,专业设置)
教研室(室代码,室名,室电话,系代码)
老师(职工号,姓名,性别,系代码)
学生(学号,姓名,性别,系代码)
课程(课程号,课程名,主选教材,大纲,室代码)
成绩(学号、课程号、分数)
教学(职工号、课程号)

特别注意，关系模型与 E-R 图中的属性表的表示相似，但增加了许多实现关联的属性，而且内容与意义是不相同的。

对于 $M:N:P$ 的联系，仿照多对多联系处理，联系转化为关系，原三个相关实体的主码及联系自身的属性构成联系关系的属性。

对于自回路，区分一对多和多对多。对于多对多情况，先复制原实体中主码及涉及的主要属性，改名后存另一个表，再仿照一对多联系和多对多联系处理，联系转化为关系，原实体中主码加上更名后原实体中主码作为联系的属性。例如学生和学生之间合作联系可用合作(a.学号,b.学号)表示，其中 a、b 为“学生”表的两个别名。

对于两个实体间的多种联系，对每一个联系区别一对多或多对多，依前面处理一对多联系及处理多对多联系的方法处理。

得到关系模型后将其中数据结构、数据约束等内容用规范化格式抽象出来，进一步设计各个属性的性质(是否主码、存储数据类型、宽度等)及具体的表示完整性约束条件，就可得到关系数据库模式，完成数据库系统数据逻辑结构的设计后上机实现。

3.4 其他数据模型

3.4.1 网状数据模型

广义的网状模型十分简单，它以矩形代表实体集，实体间用箭头线表示联系，箭头线为两头带箭头的连线，箭头分单箭头与双箭头，单箭头代表一，双箭头代表多。

从 E-R 模型转换为这种模型，只需将所有菱形及相关无向边改为箭头线，一对多联系改画为一边单箭头，一边双箭头的箭头线，多对多联系改画为两边都是双箭头的箭头线。原有矩形不变。

根据图 3.1 变成网状模型后如图 3.6 所示。这样一种模型仍然无法在机器上实现。

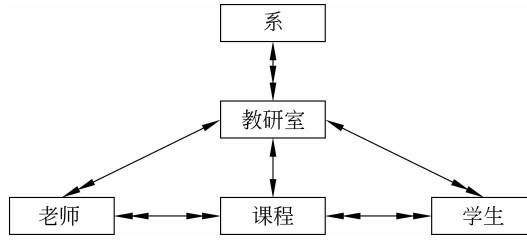

图 3.6 广义的网状数据模型

1971 年美国数据系统语言会议(Conference of Data System Language)组织的下属机构数据库任务组织(DBTG)提出了 DBTG 网状数据模型，它包括两种基本构件，记录类型和系类型，前者描述实体，后者描述实体间联系。记录类型是具有相同结构的一组记录的框架，相当于一个二维表的表头结构，它允许组项和向量。在依之而设计的 DBTG 网状数据库中每一记录对应一个实体，实体按实体集分区域存放。

根据应用需要建立“系”，系是由实体集为节点构成的二级树，树根实体集称为系主，叶节点实体集称为成员，这个树结构称为系型。系主由多条记录组成，每条记录和子节点中许多成员记录各构成一棵树，称为一个系值。这些系在实际存储时采用多种形式的链接实现数据之间的关联。DBTG 网状数据模型构成规则可大体归纳为：

(1) 一个记录类型可以参与多个系的组成，可以是多个系的系主记录型，也可为多个系的成员记录型。

(2) 任意两个记录类型之间可以定义多个系类型，这就使 DBTG 模型能表达两个实体集之间可能存在的多种联系关系的能力。

(3) 系主记录型与成员记录型之间只能是一对多联系。

(4) 在任何系中，一个成员记录值最多只能对应于一个系主记录值，即它不能属于同一系类型的不同系值。在实际问题中如出现一个成员记录值对应多个系主记录值的情况，则必须通过某个数据项标识以便区分。

(5) 允许一个系只有成员记录型而无系主记录型，这样的系称为奇异系，视“系统”为其系主，这样的系只有一个系值。

(6) 不允许一个记录型既是系主记录型又是成员记录型,这样的结构称为自回路。

从 E-R 图转换为 DBTG 数据模型时,可分为两步,第一步求取广义的网状模型,如图 3.6 所示。第二步根据应用需要设计“系”结构,如图 3.7 所示共设计了 5 个系结构,在实际应用中可能需要的“系”。多于 5 个也可能少于 5 个。

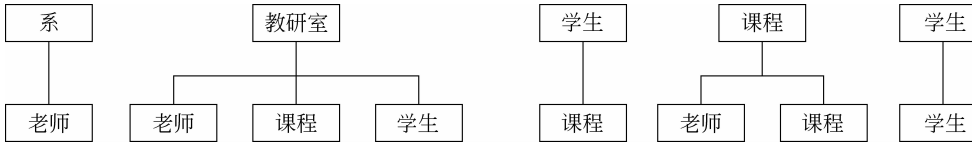

图 3.7 根据应用需要构成 DBTG 模型

一些特殊问题的转换方法:

(1) 自回路,如一对多联系,则将联系转为实体,作为成员记录型。例如图 3.4 自回路的 E-R 图可转换为图 3.8 所示的系型。

图 3.8 一对多自回路转换为 DBTG 模型

(2) 如系是多对多联系,利用在两个实体间可建立多个系的特性,在实体与联系间建立两个或多个系。例如学生中可开展多项活动,有些学生是活动的组织者,有些是参与者,一个学生可参与多位学生组织的活动,每个学生组织的活动常有多位同学参加,其 E-R 模型如图 3.9 所示。

(3) 在两个实体集之间存在多项联系时,针对每一联系可分别建立不同的系,可区分一对多或多对多再实现转换。学生与活动间可建两个系如图 3.10 所示。

图 3.9 “学生”与“活动”E-R 图

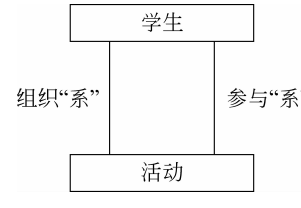

图 3.10 “学生”与“活动”转换为两个系

3.4.2 层次数据模型

层次数据模型用“树”结构表示实体集之间的关系。

它以实体集(用矩形框表示)为节点,父节点与子节点间数据联系均为一对多联系。有且仅有一个节点无父节点称为根节点,如图 3.11 中的“系”。其他节点有且仅有一个父节点构成树的支和叶节点。没有子节点的节点称为叶节点。

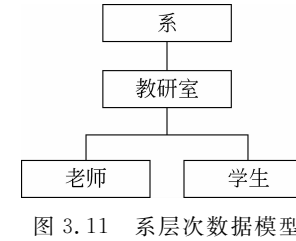

图 3.11 系层次数据模型

由于每个节点代表实体集,代表多个实体数据,每个实体数据都对应一个实体,这个实体又可能对应其子孙节点中一个或多个实体。因此按层次模型设计的数据库数据结

构模型由多棵树构成。层次数据模型根节点中每一个实体及其所有子孙节点中相关实体都构成一棵树，其数据集合称为记录，是层次数据库中一个存储单位。图 2.4 描述了数据关联的一种形式。

从 E-R 图到层次数据模型转换时可分为三步，第一步将所有一对多联系菱形及连线如前转换为箭头线；再将所有多对多联系菱形转换为矩形，其内部要写上联系的名字。

观察表 2.4 中学生表和表 2.6 成绩表记录之间的联系，学生表中一条记录对应成绩表中多条记录，成绩表中一条记录仅对应学生表中一条记录，显而易见，学生与成绩是一对多联系。同样从表 2.5 和表 2.6 联系可见，课程和成绩是一对多联系。因此可用箭头线将学生与成绩相连，学生方为单箭头，成绩方为双箭头。用箭头线将课程与成绩相连，课程方为单箭头，成绩方为双箭头。在 E-R 图中将多对多联系视为实体集后，不难分析，原联系相关实体与它的联系皆为一对多。因而可用箭头线将原相关实体矩形与联系矩形连接，原实体方为单箭头，联系矩形方为双箭头。照此将老师与教学、课程与教学分别用箭头线相连，如图 3.11 所示，便得到只含有一对多联系的网状数据结构。

第二步确定根节点。参见图 3.12，层次数据模型的父节点与子节点间总是一对多的联系，如果实体集 A 和实体集 B 间是一对多的联系，则 A 是父节点，B 是子节点，“根”无父节点。在图中寻找指向它的箭头全是单箭头的节点，将它视为根节点。如有多个节点满足这些条件，则有多个根节点，该系统将由多个记录型构成。

第三步，从根起，求每个节点的所有子节点。按从单箭头到多箭头方向遍历全系统各节点，如果到一个节点处它还与另一个实体间有一对多关系，则它还有下层子节点，对所经节点按层次在图中画出，到达某一个节点前一个节点为其父节点，由此即可得层次模型。从根起所经节点数即该节点所在层次数。

例如，图 3.12 可转换为层次模型如图 3.13 所示。在图中存在两个教学和两个成绩节点，称为冗余节点。这种冗余是必要的。如果要查询某个老师所教课程情况，需从根查起直到找到该老师数据，再在其下教学节点中查到课程代码，然后从根查起找到有关课程数据。反过来，若要查询某门课程由哪些老师教，则要先从系查到该课程数据，然后在课程的教学节点中找到有关老师代码，然后才可查询老师有关数据。在实际数据库中可有两种处理方法，一种如图冗余节点的处理方法，另一种是取两个节点中之一为虚节

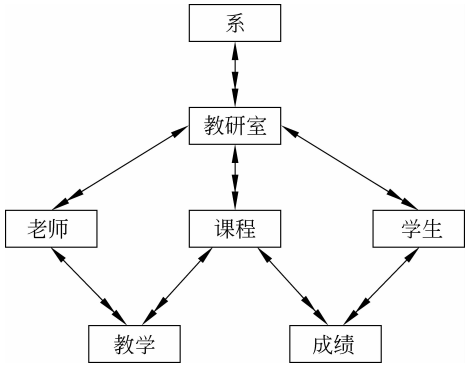

图 3.12　只含有一对多联系的网状数据结构

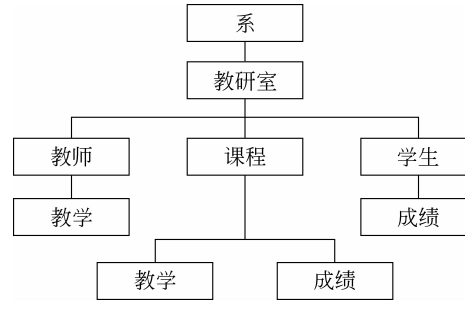

图 3.13　从图 3.12 转化为层次数据模型

点。例如课程下教学、成绩均作为虚节点，在其中只存放实节点的指针，而实际联系数据在实节点中存放。前者查询效率高，后者维护较方便。

3.5 小 结

本章通过讨论数据之间的联系重点介绍了数据模型的概念。包括用于描述实体及其属性和联系的 E-R 图和计算机上可以实现的关系数据模型、层次数据模型、网状数据模型。说明了根据 E-R 图求关系数据模型、层次数据模型、网状数据模型的方法与步骤。

用图或表的形式抽象地反映数据彼此之间的关系，称为数据模型。数据模型包括数据结构、数据操作、数据的约束条件等三方面的内容。数据结构表明数据库数据的组成、特性及其相互间联系；数据操作指对数据库中各种对象的实例允许执行的操作及有关的操作规则；数据的约束条件指数据完整性规则的集合。

习 题 三

1. 解释下列术语：

实体模型、数据模型、E-R 模型、关系数据模型、关系模式、实体。

2. 在关系数据模型中，什么是关系的原子性？若关系不具有原子性，应该如何处理？

3. 举出一对一、一对多和多对多的实例并用 E-R 图表示。

4. 为某个出版单位设计一个 E-R 图。假设在一个出版社要出版很多图书，每本图书只能由一个出版社出版。每本图书可以有多名作者，每个作者也可能参与多本书的编写工作。根据你所了解的出版社工作情况为每个实体设计属性并分析实体间的联系。

5. 在第 4 题的基础上建立对应的关系数据模型，并分析在转换过程中要注意哪些问题。

第4章 chapter 4

关系数据库基本概念

本章学习目标

本章深入讨论了关系数据库系统的基本概念、函数的依赖关系，并在此基础上介绍了关系规范化理论，以及关系数据库的基本元素，如实体、关系、表、关键字、索引等。通过本章学习，读者应掌握以下内容：

- 掌握函数的依赖关系（完全函数依赖、部分函数依赖和传递函数依赖）；
- 候选关键字、关键字和主属性的基本定义；
- 关系规范化的理论，掌握范式的基本概念和分解方法。

4.1 基本概念

按关系数据模型组织的数据库是关系数据库。其理论基础是集合代数。按集合代数理论，关系名及其属性序列称为关系模式或关系的型。一个元组为其所属关系模式的一个值，对应一个实体或一组联系。元组中每一个分量对应该实体或联系的一个属性值。

例如，一个关系名为 RELATION，其属性有 attrl、attr2……attr$N$，则关系模式简单写成 RELATION(attr1，attr2，…，attr$N$)，其一个属性或若干属性取值的范围称为域，同一域中数据是同质的，例如性别域{男，女}，由“男”、“女”两个值组成；姓名域{张明，王新，林军，…，彭烁映}，对于汉族人来说，一般由一个中文姓氏（一到两个汉字）加中文名字（一到两个汉字）组成等。各域中各取一值的完全组合称为这些域的笛卡儿积。性别域和姓名域的笛卡儿积为“性别×姓名”，如图 4.1 所示。

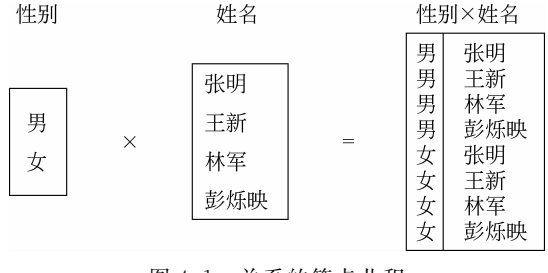

图 4.1 关系的笛卡儿积

一般说来，域 D1 和域 D2 的笛卡儿积是一个表，其属性为原 D1 域和 D2 域所有属性的集合，其行数为 D1 域值的个数和 D2 域值个数的乘积，每一行由 D1 和 D2 各取一值组成，所有各行均不重复。如果给定一组域 D1、D2……Dn，这些域中允许有相同的。则 D1×D2×…×Dn＝{(d1，d2，…，dn) | di∈Di，i＝1，2，…，n}，其中每一个元素(d1，d2，…，dn)叫做一个 N 元元组，或简称为元组。元素中的每一个值叫作元组的一个分量，也是它所对应的那个属性的一个值。多个属性构成的关系是这些属性所属域的笛卡儿积的子集，一般说来只有其真子集才有意义。图 4.1 的关系中同一位老师的性别不可能既为男又为女，因而“性别×姓名”中只有一半元组是有意义的。

按数据库理论，所有关系模式的集合(包括关系名、属性名、关键字、完整性约束和安全性要求)称为关系数据库模式，它表示一个关系数据库的逻辑结构。关系数据库模式中所有关系模式的具体关系的集合称关系数据库。关系数据库模式是数据的型的表示，而关系数据库则是数据的值的表示。元组中属性的个数称为度或目，对于一个 N 目的关系，又可称为 N 元关系。

数据库中的关系应具备如下性质：

(1) 每一列中的分量来自于同一个域，是同一类型的数据。

(2) 不同的列可来自于同一个域，每一列称为属性，要给予不同的属性名。

(3) 列的顺序的改变不改变关系。

(4) 在一个关系中任意两个元组不能全同。

(5) 元组次序可以任意交换而不改变关系。

(6) 每一分量必须是不可再分的数据项，即具有原子性，而表 3.1 所示的组项、向量和表 3.2 所示的重复组结构都是关系性质所不允许的结构。

4.2 函数依赖

4.2.1 函数依赖概念

关系理论中函数依赖是指关系中属性间的对应关系。如关系中对于属性(组)X 的每一个值，属性(组)Y 只有唯一的值与之对应，则称 Y 函数依赖于 X，或称 X 函数决定 Y。记作 X→Y。其中，X 称为决定因素。例如表 2.1 所示“系” 关系中：

系代码→系名，系代码→系地址，系代码→系电话，系代码→系专业设置

如果系名值是唯一的，即各系名均不相同，那么该关系还存在函数依赖：

系名→系代码，系名→系地址，系名→系电话，系名→系专业设置

显然系地址对任何其他属性皆不是决定因素，因为系地址为“机二楼”时对应任何属性都有两个不同值。

决定因素可能为两个以上属性构成的属性组。例如在表 2.6 的“成绩”关系中，课程号不是决定性因素，每门课都有许多学生学，同一个课程号有多个学号、多个分数与之对应。学号也不是决定性因素，同一个学生要学习多门课程，因此一个学号有多个课程号，

有多个分数值与之对应。只要每个学生每门课只有一个成绩，那么学号和课程号的值的集合在这个表中就是唯一的，任何两个元组中学号与课程号的值都不会相同，只要学号和课程号都确定，与之对应的分数值也唯一确定。因此，(学号，课程号)→分数。

在表 2.5“课程”关系中只有两行的课程名是相同的，但也因此存在这样的情况，当课程名为 Java 语言时，课程号有两个值与之对应，因而课程名不能唯一确定课程号。在分析函数依赖时一定要全面分析了解在实际应用中属性和属性组全部取值可能，只要存在一个元组的某个属性值不能唯一决定另一个属性的值，另一个属性对这个属性的函数依赖关系就不成立。

在一个关系中，如果一个属性(组)值不唯一，则这个属性(组)与任何属性(组)的函数依赖关系中，它都不是决定因素。

函数依赖是语义范畴的概念，对该关系模式下的任何关系都有效。

4.2.2 部分函数依赖

在一个关系中，可分析出许多依赖关系，但存在依赖程度的不同。

例如表 2.5 中，显然有课程号→课程名，课程号→开课单位。从另一角度看，只要课程号一定，同时课程名确定，开课单位也就唯一确定。因此{课程号，课程名}→开课单位。但它与前述课程号→开课单位是不同的，因为{课程号，课程名}存在真子集：“课程号”，课程号→开课单位。“开课单位”部分函数依赖于{课程号，课程名}。

定义部分函数依赖为：若 X、Y 为关系 R 中的属性(组)，如 $X \rightarrow Y$ 且 X 中存在真子集 $X'(X' \neq X \wedge X' \in X)$，满足 $X' \rightarrow Y$，则称 Y 部分函数依赖于 X，记作 $X \xrightarrow{P} Y$。因而表 2.5 中{课程号，课程名}$\xrightarrow{P}$开课单位。

部分函数依赖可以用图 4.2 示意：属性集 A 由属性 k_1、k_2、k_3 构成(要求至少两个属性)，表示为 $A=\{k_1, k_2, k_3\}$；如果 $A \rightarrow k_4$，且在 A 中的一个属性 $k_2 \rightarrow k_4$，那么，$A \xrightarrow{P} k_4$。

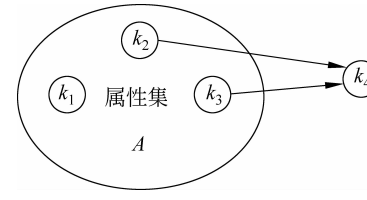

图 4.2 部分函数依赖示意

4.2.3 完全函数依赖

如 X、Y 是关系 R 中属性(组)，$X \rightarrow Y$ 且对于 X 的任何真子集 $X'(X' \neq X \wedge X' \in X)$，都有 $X' \not\rightarrow Y$，则称 Y 完全函数依赖于 X，记作 $X \xrightarrow{F} Y$。

前面所举的函数依赖例子中，除了{课程号，课程名}与其他属性之间的函数依赖之外的函数依赖皆为完全函数依赖。

4.2.4 传递函数依赖

在一个学校中，每门课均是某一位老师教，但有些老师可教多门课，则有关系“教学”如表 4.1 所示。

表 4.1 教学表

课程名	职工号	老师名	性别	出生日期	职称
英语	T1	张平	男	55.6.3	教授
数学	T2	王文	女	62.10.5	副教授
Java 语言	T3	李迎	女	62.10.5	副教授
数据库	T2	王文	女	62.10.5	副教授

由以上关系不难分析，课程名→职工号、职工号↛→课程名，但职工号和其他属性的函数依赖关系中都是决定因素，即职工号→老师名、职工号→性别，在这种情况下，职称传递函数依赖于课程名。同理，老师名也传递函数依赖于课程名。

一般说来，如 X、Y 为关系 R 中属性(组)，有 $X \to Y$，$Y \not\to X$ 但 $Y \to Z$，则称 Z 传递函数依赖于 X，记作 $X \xrightarrow{T} Z$。

上例中有：课程名$\xrightarrow{T}$职称；课程名$\xrightarrow{T}$老师名。

4.3 候选关键字与主属性

4.3.1 候选关键字

前面曾给关键字定义：在关系 R 中，如属性(组)X 唯一标识一条记录，则 X 称为关系 R 的关键字，显然这个定义是不严密的。从前面例中可以看到{课程号，课程名}能唯一标识一条记录，因为课程号就能唯一标识一条记录，课程号是关键字，课程号与课程名的集合不是关键字。

关键字的更严密定义是：在关系 R 中如记录完全函数依赖于属性(组)X，则称 X 为关系 R 中的一个候选关键字。

在表 2.1 中，“系代码”是关系“系”的候选关键字，表 2.3 中“职工号”是关系”教师”的候选关键字。在表 2.6“成绩”关系中，(学号，课程号)是候选关键字。

候选关键字有如下性质：

(1) 在一个关系中，候选关键字可以有多个。例如表 2.1 所示的系关系中，系代码、系名都是候选关键字。

(2) 候选关键字值是唯一的，因为若有两条记录候选关键字的值相同，它和记录的关系就不是决定因素。由于同名同姓的人很多，在人事管理中，姓名一般不是候选关键字，需要设计标识代码，例如“职工号”作为人事关系的关键字。

(3) 关键字可能由一个属性构成，也可能由多个属性构成。关键字不可能再与其他的属性构成新的候选关键字。分析一个关系中有哪些候选关键字时，一般首先一个个属性逐一分析判断，再两两判断，三三判断……

(4) 在任何关系中至少有一个关键字。因为根据关系的基本要求，在一个关系中任何两个元组不全同。因而在一个 N 元关系当中如单个属性都不是关键字，任何两个属性

的属性组也不是关键字，任何 K（$K<N$）个属性的属性组都不是关键字，则该关系全部属性构成的属性组是其关键字。

(5) 在实际应用中，只有在任何情况下值皆不重复的属性(组)才有可能是候选关键字。候选关键字只与关系的语义相关，关系的值无论怎么变，记录都是完全函数依赖于候选关键字的。

4.3.2 主属性

在一个关系中，如果一个属性是构成某一个候选关键字的属性集中的一个属性，则称它为主属性。如一个属性不是构成该关系任何一个候选关键字的属性集的成员，就称它为非主属性。例表 2.6 中 ，(学号，课程号)是关键字，那么“学号”是主属性，“课程号”是主属性，分数是非主属性。

4.4 关系规范化

4.4.1 问题的提出

设计关系数据库时，一种方法是分析并找出 E-R 模型再转换为关系数据模型，最后求取关系模式。称为自上而下设计方法。另外也有采用自下而上设计方法的。其方法是：

首先收集应用对象全部表格、凭证等各类数据，对它们的所有栏目归纳分类。

例如，人事部门的数据表有：

(1) 人事卡片，栏目有职工号、姓名、性别、出生日期、职务、职称、基本工资、政治面貌、所在部门、入校时间，关于家属的有关情况有爱人姓名、单位地址、性别、出生日期……还有社会关系情况包括姓名、与本人关系、出生日期、地址等。

(2) 人员报表，栏目有姓名、性别、年龄、职务、职称、部门、政治面貌。

有关职称职务工资计算办法，例如，处长 800 元，副处长 740 元，科长 700 元……，教授 1000 元，副教授 900 元等，同一职工如有职务又有职称，则职务工资取两个标准较高者。

财务部门有工资单，栏目有部门名、姓名、基本工资、职务工资、考勤补扣、行政费补扣、公积金等。

还有其他一些凭证、收据、发票、报表文件等。

其次，消除命名冲突和冗余属性，例如，对所有表的所有栏目汇总并经过分析可知所在部门、部门和部门名是同一概念，定义完全相同，可统一称为部门名，年龄可由系统当前日期及出生日期计算得到，年龄依不同年份而不同，但一个人出生日期是只有一个值且不会改变。职务工资可从职务、职称及有关职务工资计算公式求得。由此可知，人事报表数据源来自人事卡片，工资单有部分数据源来自人事卡片，另有一些数据与生产部门、行政部门相关，且有各自计算方法。最终可设想系统由三个关系构成：

人事卡片(职工号,姓名,性别,出生日期,职务,职称,基本工资,政治面貌,部门名,参加工作时间,爱人姓名,爱人性别,爱人出生日期,爱人职务,爱人职称,爱人单位,爱人地址,关系人姓名,关系人年龄,关系人性别,与本人关系,关系人单位,关系人地址)

考勤表(职工号,加班天数,早班数,中班数,晚班数,病假天数,旷工天数)

行政收费表(职工号,房租费,水电费,电话费,行政扣除,行政奖励)

但在按此模式建立数据库后在录入数据时发现有几个问题:

① 在使用中,对爱人情况和社会关系有关数据查找取用次数极少。

② 不少人尚未结婚,但表中仍需留下大量空位,使数据库文件规模扩大,影响效率。

③ 不少人社会关系人很多,如要对关系人利用数据库系统进行查找,必须对不同关系人数据分段存放,如表 4.2 所示。

表 4.2 人事卡片表

职工号	姓名	性别	爱人姓名	爱人年龄	关系人姓名	与本人关系	关系人职务	关系人职称
201	张平	男	李莉	30	张明	大哥	科长	高级工程师
201	张平	男	李莉	30	张武	二哥		工程师
201	张平	男	李莉	30	张文	妹妹	处长	工程师
202	王勇	男	吴方	43	王新平	父亲		高级工程师
202	王勇	男	吴方	43	张明	表兄	科长	高级工程师

显然一个职工数据多行重复存放,出现了严重冗余。这种冗余使表格文件规模增加了数倍,使检索速度降低,在数据录入和修改时需同时修改多处相关数据,工作量大且易出错。在实际运行中还发现,实际数据库如要求定义库结构时必须说明关键字,关键字数据不输入或不完整输入,数据都不能录入计算机。一个职工可有多个关系人,一个关系人可能和多个职工有关系,在这个结构中不难分析,(职工号,关系人姓名)构成候选关键字,一个职工如果一个关系人姓名也不填,他自己的记录也无法录入计算机。另外还不难分析,如某一个职工要删除自己全部社会关系数据,则自己的数据也无法留存。这些都与实际系统的要求相违背,实际上使这样的结构无法使用,必须修订。这种现象称为操作异常。

操作异常包括插入操作异常和删除操作异常两类。插入操作异常指要录入的数据因缺少关键字或关键字数据不完整而不能被录入的现象。删除操作异常指不应当被删除的数据因部分主属性删除而被删除的现象。操作异常与冗余一般是相伴而生的,因此常常通过检查冗余来发现是否可能存在操作异常。一旦有可能出现操作异常,设计就必须修改。例如上述例子中,首先针对冗余最严重的、在实际应用中出现最多的问题——职工基本数据冗余问题,将人事卡片分解为两个或三个表,第一个表包括职工本人基本数据,第二个表包括职工代码及爱人情况全部数据(上述两表可合为一个表),第三个表包括职工代码及社会关系全部数据。如表 4.3 所示,通过职工号的关联作用将可实现相关检索。

表 4.3 将人事卡片关系分解为两个关系

职工卡片

职工号	姓名	性别	爱人姓名	爱人年龄
201	张平	男	李莉	30
202	王勇	男	吴方	43

社会关系

职工号	关系人姓名	与本人关系	关系人职务	关系人职称
201	张明	大哥	科长	高级工程师
201	张文	妹妹	处长	工程师
201	张武	三哥		工程师
202	王新平	父亲		高级工程师
202	张明	表兄	科长	高级工程师

显然分解为两个表后，冗余减少，操作异常问题得到解决。如果爱人情况数据量大，且在实际工作中涉及爱人情况的查询极少，也可以将职工卡片与爱人情况分成两个表，通过职工号联系。从人们设计数据库的实践，根据不同设计出现冗余和操作异常的程度，分成若干标准，称为范式，范式级别越高一般说来冗余越小，发生操作异常的可能越小，但在读取数据时花费在关联上的时间越多，查询效率越低。

关系规范化的过程是以达到高级别范式的关系去取代原有关系的过程，随着规范程度的提高，冗余与操作异常可能性减少。

4.4.2 范式

1. 第一范式(1NF)

任给关系 R，如果 R 中每个列与行的相交处对应单元格的数据都是不可再分的基本元素，则 R 达到第一范式，简称 1NF。根据关系的基本性质可见，符合关系基本性质的关系均达到第一范式。表 4.3 就已达到第一范式。表 3.1 和表 3.2 所示表存在重复组、组项和向量，因而均未达到第一范式。可通过去掉上层的属性，并更改最下层的属性的名称使它达到第一范式。例如把表 3.1 结构改为成绩单(姓名，C 分数，DB 分数，OS 分数，总分)，则每个行与列相交处的单元格取值均不可再分。又例如设计人事卡片结构为：人事卡片(职工号，姓名，…，社会关系)，将一个人的所有社会关系列在一个行列交叉点上。如不准备对它进行分类检查，则它达到第一范式；否则就要按照表 4.2 或设计成表 4.3 那样以达到第一范式，但达到第一范式仍可能有冗余或操作异常的问题。

从表 4.3 来分析，此表中数据实际可看成是来自两个实体集：职工与社会关系。如

果一个关系人只对应一个职工，从函数依赖关系来看，候选关键字是(职工号，关系人姓名)，所有属性对它们都是函数依赖的，但是其中职工实体的数据对它们是部分函数依赖，因为它们完全函数依赖的是职工号。把职工实体自己的数据从原表中分离出来就可解决严重冗余的问题。由此得到对达到第一范式，但存在较严重冗余的一些结构关系优化的办法：如果这些结构中存在非主属性对关键字的部分函数依赖时，将之分解为两个关系，将对候选关键字存在部分函数依赖的属性分离出来建立新关系，注意加进它们完全函数依赖的主属性，剩余属性构成另一个关系。它们将达到第二范式。

2. 第二范式(2NF)

如果一个关系达到第一范式，且不存在任何非主属性对候选关键字的部分函数依赖，则称此关系达到第二范式，简称 2NF。经过上述分解后的“职工卡片”关系和“爱人情况”关系都满足第二范式的基本要求。第二范式还可用另一种形式表述，即如果一个关系达到第一范式，且不存在非主属性对构成候选关键字的部分主属性的完全函数依赖，则该关系达到第二范式。

在如图 4.3 所示的图中，k_1、k_2、k_3、k_4 是主属性，其他 p_j 是非主属性。如果 k_3 不是关键字，但出现 $k_3 \rightarrow p_3$、$k_3 \rightarrow p_4$、$k_3 \rightarrow p_6$ 的情况(只要出现了其中之一)，那么该关系即使达到第一范式，也未达到第二范式。要达到第二范式需作分解。方法是：将 p_3、p_4、p_6 等函数依赖于 k_3 的非主属性抽出来，加上 k_3 组合成新的关系，k_3 是其关键字；剩余非主属性、主属性包括 k_3 维持原有各关系不变。

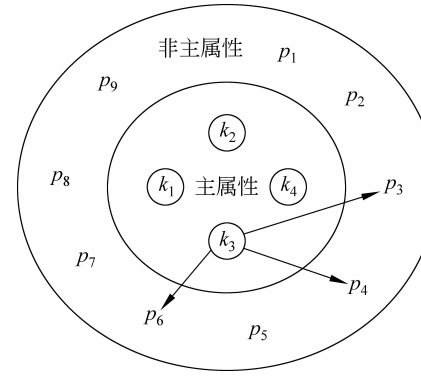

图 4.3 不到第二范式的关系示意

在关系规范化过程中，并不是规范化程度越高越好，要根据实际问题的需要综合考虑。例如在表 4.3 中的“社会关系”关系中，如一个关系人存在与多位职工的联系，则候选关键字是(职工号，关系人姓名)，职称、职务等其他数据对它是部分函数依赖，因而未达到 2NF。但一般应用问题，主要关心的是职工数据，是弄清某一个职工有哪些社会关系及其情况，在数据录入时一定是先为职工编号并录入职工数据，再录入社会关系数据，如果再分解，数据表过多，不便于管理，为求系统简单而不再分解，这样的设计是允许的。

但对于类似的另一些问题，例如学生与课程关系，经常要分析课程情况，例如大纲、教材等，而且从管理上也要求课程数据的录入与学生数据无关。仅仅把学生关系分解出来是不够的，应分解为学生、课程、成绩三个关系。

表 4.3 所示的关系中，如果经常需要对关系人情况进行分析与统计，社会关系表也需要再分解，可按表 4.4 所示进行分解，关系分解后，在社会关系表中已不存在冗余和操作异常。