第3章 数据库静态结构设计和实现——数据库设计

第2章讲述了数据库设计应该遵循的规范，即范式理论及其在实践中的灵活运用，本章则要解决如何把用户的需求转换为合理并能充分反映用户需求的数据库设计。数据库的设计是整个数据库应用程序设计的基础，从第2章的讨论中可以看出，数据库的设计好坏直接关系到数据库应用程序整体的设计、开发和运行的效率。

把用户需求转换为一种合理的数据库设计的方法是把用户需求抽象为概念模型，然后由概念模型产生数据库的逻辑结构设计。

建立概念模型可以借助数据库模型设计的软件工具，使用工具的最大好处是设计者在进行上述的转换过程中，只需要把用户需求转换为概念模型，而概念模型转换为数据库设计完全可由软件自动完成，工具甚至能产生针对某个数据库管理系统的构建数据库的数据定义语句。所以，本章的重点是讨论概念模型以及如何把用户需求转换为一个好的概念模型的方法，所谓一个好的概念模型，是设计者很少或不必为满足某种需求而去修改由概念模型产生的数据库的逻辑结构。选用的工具为 PowerDesigner 9，版本并不重要，本书涉及的仅仅是概念模型的核心概念，这对所有版本都是一样的。

本章首先介绍概念模型的一般概念，然后通过实例讨论如何使用 PowerDesigner 根据一些典型需求建立概念模型，并比较同一需求建立的不同的概念模型的优劣，讨论并纠正一些建立概念模型过程中常见的错误。

3.1 概念模型的一般概念

概念模型用于信息世界的建模，是现实世界到机器世界的一个中间层次，是数据库设计的有力工具，也是数据库设计人员和用户之间进行交流的语言。基于上述目标，就要求概念模型具备较强的语义表达能力，能够方便、直接地表达用户需求，并且简单、清晰、易于用户理解，同时还要求其易于向数据模型(如关系模型)转换。

3.1.1 概念模型的两个要素：实体和关系

实体和关系是构成概念模型的两个要素，简单的构成充分体现了概念模型清晰和简单的原则。

1. 实体

(1) 实体(Entity)、实体型(Entity Type)和实体集(Entity Set)。

客观存在并可相互区别的事物称为实体。实体可以是具体的人、事、物或抽象的概念，如某名学生。

具有相同特性的实体称为实体型。如学生，这里的学生并非指具体某名学生，而是泛

指，是对具有学号、姓名、性别、年龄、在读学校、所在年级和班级等特性的所有学生的抽象。

实体和实体型的关系类似高级语言中数据类型和变量的关系，也类似于面向对象程序设计中类和对象的关系。实体型是构成概念模型的第一要素，在不至于引起混淆的情况下，通常把实体型简称为实体。

同属一个实体型的实体的集合称为实体集。如某班的所有学生构成一个实体集。

(2) 属性(Attribute)。

实体所具有的某一特性称为属性。一个实体可以由若干属性刻画。如学生实体型可以用如下形式表示：

学生(学号,姓名,性别,年龄,家庭电话,手机)

属性是实体的主要构成。

(3) 域(Domain)。

域是一组具有相同数据类型的值的集合。例如，整数、实数、介于某个取值范围的自然数(如年龄)、长度指定的字符串(如姓名)和介于某个取值范围的日期(如生日)等。

为某个属性指定一个域，也就限定了该属性的取值范围。不同的属性可对应同一个域，如性别，所有涉及人的实体型都可能有“性别”属性，不必为每个这样的属性指定可取的值，只需要定义一个称为“性别”的域，其取值为{“男”,“女”}，然后所有实体型中“性别”属性的数据类型指定为“性别”域，这样就可以最大限度地确保“性别”取值范围的一致性，也节省了设置取值范围的工作量，这是引入域概念的主要目的。

(4) 码(Key)。

唯一标识实体的属性集称为码。这里的码和二维表的候选码有完全相同的要求和特点，同样必须具备唯一性和最小性，关于候选码的定义见 1.1.1 节。

码是区别实体集中不同实体的关键属性，所以很多场合码也称为键。

2. 关系

概念模型中的关系(Relationship)是指两个或两个以上实体集中实体之间的对应关系。可以把这种对应关系分为一对一、一对多和多对多 3 种类型，分别记为 1∶1、1∶n 和 m∶n。

(1) 1∶1 关系。

如果一个实体集 A 中的实体，在另一个实体集 B 中至多有一个实体与之对应，反之，实体集 B 中的实体，在实体集 A 中也至多有一个实体与之对应，则称 A 和 B 具有 1∶1 的关系。

假设一名学生至多拥有一张有效的校园卡，则学生和校园卡之间就是 1∶1 关系。要注意的是，即使允许学生可以不办校园卡，即某些学生没有对应的校园卡，这种 1∶1 的关系依然成立。

(2) 1∶n 关系。

如果一个实体集 A 中至少有一个实体，在另一个实体集 B 中有多于一个实体与之对应，同时，实体集 B 中实体在实体集 A 中至多有一个实体与之对应，则称 A 和 B 具有 1∶n 的关系。

班级和学生就是 1∶n 的关系，即一个班级可对应多名学生，而一名学生只能对应一个班级。一个商场的供应商和商品的关系通常为 1∶n 关系，即一个供应商一般会同时向商场提供多种商品，而一个商场同一种商品在某个时期的供应商通常只有一个。学生和校园

卡一般为1∶1关系，但只要允许一名学生可申请一张以上的校园卡，则学生和校园卡关系就变成了1∶n。

(3) m∶n关系。

如果一个实体集A中至少有一个实体，在另一个实体集B中有多于一个实体与之对应，同时，实体集B中也至少有一个实体在A中有多于一个实体与之对应，则称A和B具有m∶n的关系。

学生和课程关系就是m∶n关系，一名学生在读期间必须要修多门课程，而学校开设的每门课程一般会有多名学生送修。一个商场的供应商和商品的关系通常为1∶n关系，但只要有一个商品对应一个以上供应商，则供应商和商品的关系就变成了m∶n关系。

实体之间的关系会随具体需求的变化而变化，如一个工厂的产品和零件的关系，可能为1∶1关系，也可能为1∶n关系，如果所有产品的零件各不相同，即同一种零件不可能使用在两个不同的产品上，则产品和零件就是1∶n关系，如有零件可同时使用在两个或两个以上的产品上，则产品和零件就是m∶n关系。

(4) 3种实体间关系的关系。

显然，给出3种实体间关系的定义，互不包含和交叉，两个实体之间的关系属于也只能属于其中的一种类型，这是和一些数据库理论书籍中的定义方式的不同之处。这种定义方式可以使开发者更好地明确两个实体间的关系。

m∶n关系在m=1时就退化为1∶n关系，1∶n关系在n=1时就退化为1∶1关系，所以在对3种关系的计算机处理中，1∶1关系可以看成1∶n关系的特例，1∶n又可以看成m∶n关系的特例。即对1∶n的关系处理适用1∶1关系，对m∶n关系的处理方式适用1∶n和1∶1关系，自然，对m∶n关系的处理方式最复杂。

3. 实体关系图

一般用实体关系(Entity-Relationship，E-R)图来表达和构建概念模型。E-R图表达了概念模型的核心概念实体型、属性和实体之间的关系。

在E-R图中，用矩形表示实体型，矩形框内写明实体名。用椭圆形表示实体的属性，并用无向边将其与相应的实体连接起来。用菱形表示实体之间的关系，菱形框内写明关系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上关系的类型1∶1(一对一)、1∶n(一对多)或m∶n(多对多)。关系本身也可以有属性。如果一个关系具有属性，则这些属性也要用无向边与该关系连接起来。

【例3-1】 图3-1是一个E-R图示例，一名学生可选多门课程，而一门课程会有多名学生选，所以学生所选课程的成绩既无法作为学生实体的属性，也无法作为课程实体的属性，它应该作为“选课”关系的属性。

图3-1所示的概念模型是依据需求而建立的，可以把根据需求建立概念模型的步骤归纳为：

(1) 根据需求确定概念模型由哪些实体构成。

确定每个实体的属性，但不要包含反映实体关系的属性，关于这点，3.1.2节有专门的讨论。

(2) 确定各实体的关系。

(3) 确定关系中所包含的属性(一般仅仅对m∶n关系有此需要)。

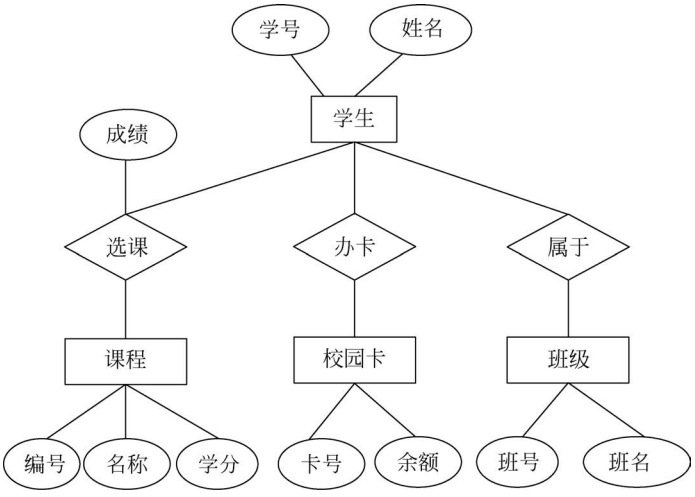

图 3-1　E-R 图示例

每个实体实际的属性可能远不止图 3-1 所示的内容，对一些包含几十甚至上百个属性的实体，用以上形式的 E-R 图是难以表达的，在实际应用中，图 3-1 形式的 E-R 图更多地用于表达系统包含哪些实体以及实体之间的关系。即使能把所有的属性都反映在 E-R 图上，由 E-R 图所包含的信息仍不足以产生关系数据模型，还必须补充属性的数据类型、取值范围等其他信息，这在后面介绍 PowerDesigner 的章节中会有详细说明。

建立了完整的概念模型后，可以很容易地把它转换为关系数据模型，并进而产生适合某个数据库管理系统的数据定义语句。

4. 概念模型向关系模型转换的方法和规则

概念模型转换为关系模型，就是要把概念模型中的两个要素“实体”和“关系”转换为关系模型中关系，即二维表。概念模型中的每个实体被转换为关系模型中的一个关系，实体的属性即为关系的属性，实体的码即为关系的码。

概念模型中实体之间的关系将分下列情况按一定的规则转换为关系模型中的关系或某个关系中的属性，其目标是使用关系模型中的关系（二维表）能表达概念模型中实体之间的关系。

（1）1∶1 关系的转换。

在任意一方实体对应的关系（二维表）中，加入另一方实体对应的关系（二维表）的码作为其属性（即外码）。

如例 3-1 中的学生和校园卡，转换后的关系可以是以下两种关系模型的一种：

```
学生(*学号,姓名,性别,…,校园卡号),校园卡(*校园卡号,办卡日期,余额,…)
学生(*学号,姓名,性别,…),校园卡(*校园卡号,办卡日期,余额,…,学号)
```

这样的转换实际上体现并非 1∶1 关系，而是 1∶n 关系，因为第一种设计可允许一张校园卡对应多名学生，而第二种设计可允许一名学生对应多张校园卡。

还有一种转换方式是独立地建立一个关系：学生_校园卡（*学号，*校园卡号），这种转换允许学生和校园卡存在 m∶n 的关系。

要真正保证关系模型只能接受 1∶1 的关系，必须引入属性的 Unique 特性，关于这点，可参见 3.3 节。

(2) 1∶n 关系的转换。

在 1∶n 的两个实体中的“多”的一方实体对应的关系(二维表)中，加入另一方实体对应的关系(二维表)的码作为其属性(即外码)。如学生和班级，转换后的关系为：

```
学生(*学号,姓名,性别,…,班号)
班级(*班号,班名,…)
```

这两个关系，通过学生中的“班号”，可以反映一名学生只能对应一个班级，而一个班级可对应多名学生的实体关系。同 1∶1 关系一样，也可以独立地建立一个关系：学生_班级(*学号，*班号)，而实际上这种转换允许学生和班级存在 m∶n 的关系，一般很少被采用。

(3) m∶n 关系的转换。

m∶n 的两个实体的关系，转换为关系模型后将通过一个独立的关系予以反映，该关系的属性由这两个实体的码组成，并把两个实体的码合并作为该关系的码。

“学生”和“课程”的关系是 m∶n 关系，转换为关系模型后对应关系为：

```
选课(*学号,*课程号)
```

其中，“学号”和“课程号”分别是“学生”和“课程”两个实体的码。

3.1.2 确定实体属性的重要规则

综上所述，概念模型中的实体关系在概念模型转换为关系模型时，将被转换为属性或独立的关系，所以在根据需求创建概念模型时，实体应该避免包含反映实体关系的属性，因为这些属性在由概念模型生成关系模型时将被自动生成。

例如，在概念模型中，不少设计人员会把“班长”作为“班级”实体的一个属性，而这实际上是一个错误的概念模型，“班长”事实上反映的是学生和班级的一种 1∶1 的关系，这种关系用 E-R 图表示如图 3-2 所示。

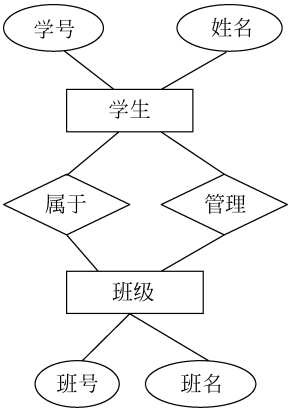

图 3-2 两个实体的双重关系

由此概念模型，按上述规则，可生成下列关系模型中的关系：

```
学生(学号,姓名,班号)
班级(班号,班名,学号)
```

其中，学生关系中的“班号”由班级和学生的 1∶n 关系产生，用于记录每名学生所在班级，而班级实体中的“学号”则是由学生和班级的 1∶1 关系产生，用于记录每个班班长的学号。如果班级实体本身错误地包含了“班长”属性，则生成的关系模型中，班级关系中出现“学号”和“班长”两个指向班长的属性。

同样，把辅导员作为班级的一个属性也是一个错误的概念模型，如果一个辅导员可带多个班级，则班级实体中的辅导员属性实际反映的是班级实体和辅导员实体之间的一对多的关系。

3.2 PowerDesigner 概述——概念数据模型

PowerDesigner 是 Sybase 公司的提供的一个功能强大的数据建模工具，使用它可以更方便、直观地分析和建立数据模型，它几乎包括了数据库模型设计的全过程。

对数据库的模型设计而言，PowerDesigner 可以根据需求建立概念数据模型，然后由概念数据模型生成关系数据模型，同时依据用户选择的具体数据库管理系统，产生数据定义语句(DDL)。所以，使用 PowerDesigner 为关系数据库建模的主要和关键的工作是建立正确、有效的概念数据模型，这也是本章下面重点介绍的内容。

3.2.1 概念数据模型概述

PowerDesigner 的概念数据模型(Conceptual Data Model，CDM)实际上是对 E-R 图表达形式的改进和表达内容的扩展，使之适应计算机的特点以及由 CDM 生成物理数据模型(Physical Data Model，PDM)的需要。CDM 的主要作用如下：

(1) 以图形方式表示数据的组织结构(E-R 图)。

(2) 检验数据设计的有效性。

(3) 产生 PDM，其中包含了数据库的物理实现。

(4) 可以产生一个使用 UML 标准表达的面向对象模型(Object-Oriented Model，OOM)。

建立 CDM 的依据是需求，它独立于任何软件或数据存储结构，不需要考虑实际物理实现的细节，明白这一点对设计正确、有效的概念模型很重要，一些软件设计人员常常会不知不觉地把设计思想放到 CDM 中，包括本书中的一些例子也常见这种错误。

在 PowerDesigner 中，CDM 是存放概念数据模型内容的文件扩展名，由概念数据模型生成的关系数据模型即数据表的逻辑结构被称为物理数据模型，PDM 是存放物理数据模型内容的文件扩展名。

用户也可以在 PowerDesigner 中直接建立物理数据模型或对生成的物理数据模型进行修改。下面将通过实例介绍 PowerDesigner 概念数据模型的基本概念，重点要关注的是概念数据模型所有对象特性(Properties)的不同设置的实际意义和对由此生成的物理数据模型将产生的影响。

3.2.2 CDM 分析设计的一般流程

本节用一个实例展示如何使用 PowerDesigner 根据需求建立 CDM，由 CDM 生成 PDM 以及 MySQL 的数据定义语句的整个过程。

【例 3-2】 仍以 3.1.1 节的实例为例，合并图 3-1 和图 3-2 中的 E-R 图得到如图 3-3 所示的 E-R 图。

(1) 使用 PowerDesigner 建立 E-R 图。

在使用 PowerDesigner 建立上面的概念数据模型时，仅使用图上的信息是不够的，必须对实体属性和实体关系的特性进行更详细的刻画，因为这是生成物理模型所必需的。图 3-4 是在上面手工画的 E-R 图的基础上，补充了实体的主码特性、属性的数据类型等诸多特性后，在 PowerDesigner 中建立的 CDM。

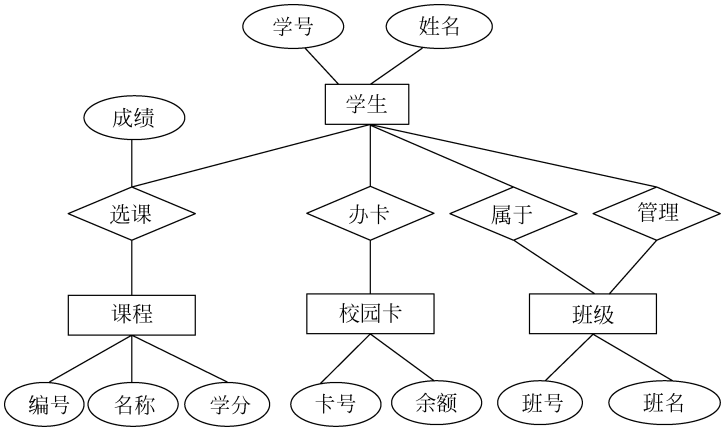

图 3-3　原始 E-R 图

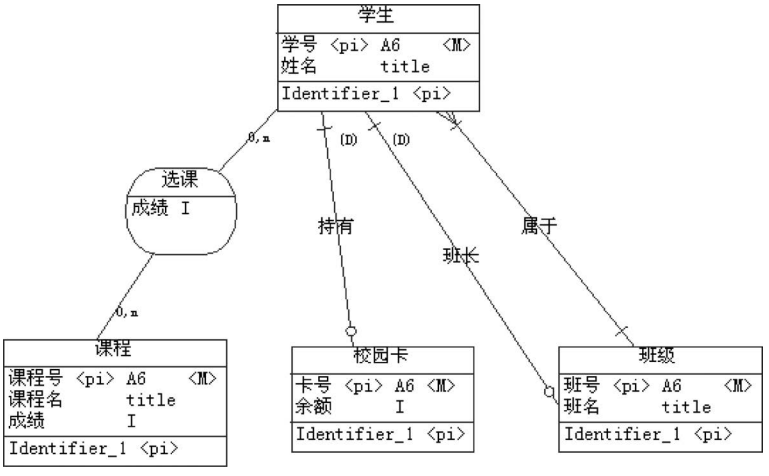

图 3-4　PowerDesigner 的 CDM

根据 CDM 由 PowerDesigner 可生成 PDM，对图 3-4 的 CDM 生成的 PDM 如图 3-5 所示。

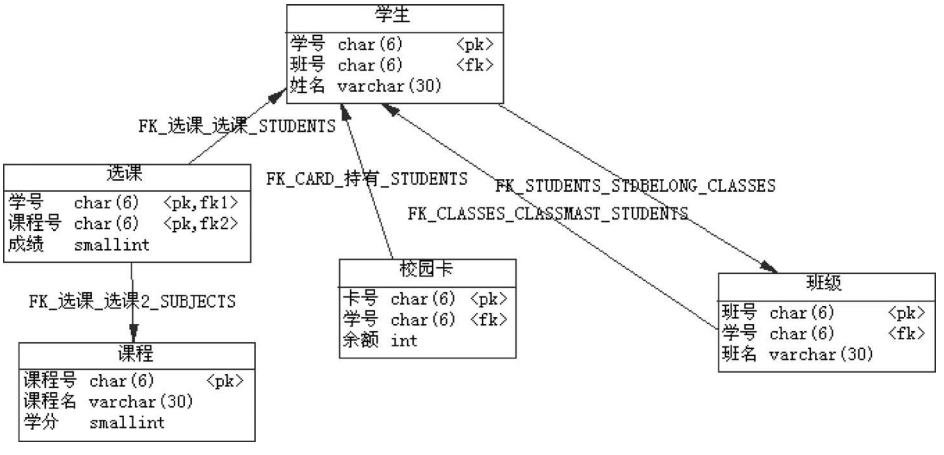

图 3-5　PowerDesigner 生成的 PDM

PDM 中每个框对应就是关系数据库中的一张表，每一个有向的线段表示两张表之间的引用关系。概念模型中的选课关系是 $m:n$ 关系，所以被转换为一张选课表以及选课表和学生表、选课表和课程表之间的引用关系，其中选课表以学生表和课程表的主码属性组合起来作为主码，并包含了选课关系中的属性“成绩”。概念模型中的其他关系，根据 3.1.2 节中的规则通过实体对应表中加入外码属性及引用关系来实现，如“学生”和“校园卡”之间的 1∶1 关系，通过在校园卡属性中加入“学号”并建立与“学生”的引用关系来实现；学生和班级的 1∶1 和 1∶n 的关系分别通过学生中加入“班号”（学生所在班）和班级中加入学号（班长）得以实现。

（2）生成 DDL。

在产生 DDL 的同时，PowerDesigner 同时生成了由用户选择的数据库管理系统的数据定义语句。如对学生表，其对应的 DDL 为：

```
create table Students (
StdId       char(6)     not null,
ClassId     char(6)     not null,
Name        title       null,
constraint PK_STUDENTS primary key (StdId)
)
alter table Students
add constraint FK_STUDENTS_STDBELONG_CLASSES foreign key (ClassId) references classes
(ClassId)
```

其中，create 语句用于在数据库中建立学生表（Students），title 为 PowerDesigner 中定义的一个域（Domain）；alter 语句在学生表（Students）中加入学生表和班级表（classes）通过外码“班号”（ClassId）而建立的引用关系。

PowerDesigner 中，CDM 中的实体和属性等都有一个 Code 特性供用户设置，该特性就是产生上述 DDL 语句中的表名和列名的依据。

3.2.3 建立 CDM 的一般操作

建立一个新的 CDM 的步骤如下。

（1）新建一个概念模型。

选择 New（新建）菜单，然后在弹出的对话框中选择 Conceptual Data Model（概念数据模型），确认后，在 Browse（浏览）窗口中显示的树节点的根节点 Workspace（工作空间）下将产生一个名为 ConceptualDataModel_1 的概念数据模型节点，在该节点下已自动加入一个 Diagram-1 节点（即图节点），Diagram-1 显示为一个空白的窗口，用户可以在该窗口中构建 E-R 图，并以图形化的方式显示概念数据模型，一个概念数据模型下至少有一个图，自动显示概念数据模型下的所有 Entity（实体）、Relation（关系）等对象。图如果被关闭，则可通过双击浏览窗口中的 Diagram（图）节点打开它。

右击 Browse（浏览）窗口中的 ConceptualDataModel_1，使用弹出的快捷菜单中的 Rename 菜单，把概念数据模型更名为 TeachingCDM。

（2）概念模型中对象的增加、删除和修改。

概念数据模型图中的元素可以取自称为 Palette（调色板）的窗口中的元素，在 Palette（调色板）窗口中单击需要的对象，然后单击 Diagram（图形）窗口的合适位置，该位置处就会

出现该对象，选择 Palette(调色板)窗口中的 Pointer(指针)，然后双击新加入的对象，就可以对该对象的所有 Properties(特性)进行设置。对实体，特性包括 General(一般特性)、Attributes(属性)和 Identifiers(标识)等。对于在图中增加的对象，在 Browse(浏览)窗口的树状节点中也可以看到。若 Palette(调色板)窗口不小心被关闭，则可以右击工具栏，在弹出的快捷菜单中选择 Palette(调色板)菜单打开该窗口。

在概念数据模型中增加或修改对象的另一个方法是右击 Browse(浏览)窗口中的概念数据模型，在弹出的快捷菜单中选择 New 菜单，然后选择要加入的对象，被加入的对象自动显示在图中。同样可以通过双击加入的对象，设置其所有特性。

所有对象的一般特性中均有 Name(名称)和 Code(代码)特性，名称特性用于 PowerDesigner 中显示，所以一般取中文，而代码特性用于生成物理模型的对象名(如表名、列名等)，一般用能表达该对象含义的英文字母组成。删除概念数据模型中对象的方法是选择要删除的对象，然后按 Delete 键。

概念数据模型对应文件的默认名为该概念数据模型名，也可以以其他名称命名，但不改变概念数据模型的名称。在获得概念数据模型中的所有实体的属性信息后，必须分析、整理出具有相同特性或必须保持一致特性的属性，把它们定义为域(Domain)或数据项(Data Item)，这对保证概念数据模型中数据定义的一致性，非常重要，尤其是在团队开发的情况下。

3.3 实体定义——域、属性和数据项

3.3.1 域定义

域由域名、数据类型、长度、默认值、最大值和最小值等特性组成，定义了一个域后，实体的属性的类型就可以用域表示，具有同一个域的属性就具有和该域一样的数据类型和数据约束。

使用域可以保证一组实体属性的数据类型和数据约束的一致性，一旦需要修改，不必再逐个对实体属性进行修改，而只要对定义的域做一次修改，减轻了设置的工作量。在 PowerDesigner 中，在一个概念数据模型下定义一个域的方法是右击 Browse(浏览)窗口中的概念数据模型，在弹出的快捷菜单中选择 New(新建)→Domain(域)菜单，然后在弹出的对话框中设置域的特性。

域的一般特性包括名称(Name)、代码(Code)、数据类型(Data Type)、数据长度(Length)和精度(Precision)。标准检查(Standard Checks)特性中包括最小值(Minimun)、最大值(Maximun)、默认值(Default)以及数据格式(Format)等。

【例 3-3】 定义一个名称域 title，其数据类型为 variable character，长度为 20，有了该域的定义后，学生和课程实体中的名称属性的域就可选取 title，而不必再定义这两个属性的数据类型。也可以为一个教学管理系统中所有实体的成绩属性定义一个域，其数据类型为 integer，最大值为 100，最小值为 0。

标准检查(Standard Checks)中设置的所有内容，将对选取该域的实体属性生成物理模型中的 SQL 语句产生影响，如定义年龄域，数据类型为 int，默认值为 18，最大值为 22，最小值为 16，并且在此区间内，只能取值 16，18，20，22，如果学生实体中年龄属性取该域，则生成

的 SQL Server 的数据定义语句中将包含：

```
…
age int null default 18
    constraint CKC_AGE_ENTITY_1 check(age is null or(age between 16 and 22 and age in(16,18,
20,22)))
```

3.3.2 数据项

在为实体定义属性后，PowerDesigner 会在 Browse(浏览)窗口的概念数据模型下建立一个数据项目录，并自动把所有实体的属性作为数据项罗列在该目录下进行统一管理，相同代码的属性被视为是同一个数据项，即系统会自动保证具有相同代码的属性(对应一个数据项)具有相同的数据定义。

(1) 增加新的数据项及实体属性对数据项定义的引用。

有两种途径增加新数据项：一种是在实体中设置的所有属性将被自动加入在数据项目录下作为数据项进行集中管理；另一种是直接在数据项目录下使用弹出式菜单增加数据项，这样定义的数据项不属于任一个实体，若某个实体属性使用该数据项定义，只要在该属性的代码特性中输入该数据项的代码即可。

(2) 修改数据项的定义。

具有相同代码的实体属性对应的是一个数据项，对这些实体属性中的任一个属性的修改，都可以改变该数据项的定义，也就改变了所有这些代码相同的属性的定义。也可以直接修改数据项的定义，结果是所有与该数据项具有相同代码的属性定义被同步改变。

(3) 数据项的删除。

删除某个实体属性，PowerDesigner 并不会自动删除对应的数据项，要删除某个数据项，必须在数据项目录下选择要删除的数据项，然后按 Delete 键予以删除。删除了数据项，也就删除了所有实体中与该数据项代码相同的所有属性，这点要特别注意。

(4) 数据项机制的意义、限制及与域的异同。

引入数据项机制的意义在于：

① 相同的数据项只需要定义一次，节省了设置的工作量。

② 最大限度保证了不同实体中相同性质的属性定义的一致性。

③ 保证了相同性质的列名的一致性。

其中，第①、②条和域的作用相似，第②、③条在团队开发情况下非常有用，第③条是和域作用的不同之处。

【例 3-4】 在销售管理系统中，订单、进货单、销售单中可能都会有“价格”属性，假如进货和销售由两个不同的人员进行分析设计，则很可能他们会为“价格”属性取不同的名称和代码，这会为以后的开发带来困难，编码人员在使用价格时，他不得不去分清并记住不同单据中“价格”列的名称。

相反的情况，由于相同代码的实体属性对应的是一个数据项，而代码在转换为物理模型时就是列名，因此会要求代码必须能表达属性的含义，引出的问题是相同含义的实体属性，如果具有不同的数据定义时就要避免取相同的代码。

【例 3-5】 在商场管理信息系统中，很多单据都有“数量”属性，若这些“数量”属性具有

相同的长度和精度要求，则其属性的代码可以相同。但实际情况是，一个零售企业的进货数量和销售量通常不在一个数量级，为满足同数量含义属性的不同的数据定义，必须为它们取不同的代码，例如进货的“数量”属性的代码取 OrderQty、销售单的“数量”属性取 SaleQty 等。

(5) 选作实体主码数据项的排他性。

在 PowerDesigner 的默认配置的情况下，若一个实体属性是一个实体的主码，那么它对应的数据项就不能再作为其他实体的属性，即下列在软件开发中常见的做法不被允许。

① 在销售管理系统中，所有单据的单据编号属性取相同的代码，如 sheetno，它们同时是这些单据实体的主码。

② 在教学管理中，教师和学生的“编号”属性取相同的代码，如 Id，它们同时是教师和学生实体的主码。

③ 把班长学号 Id 作为班级的属性，Id 为学生实体的主码。

这种限制的合理性在于 PowerDesigner 假设与某一实体主码同名(Code 相同)的属性必是外码，而外码是在生成物理模型时根据实体间的关系自动生成的，在概念模型中无须设置。

上面列举的例子中，①和②中单据号和编号都是对应实体的主码，所以属性的代码名不能相同，而③班级的班长不应该作为实体班长的属性，班级的班长应该通过学生和班级之间的 1∶1 关系来表达，这在 3.1.2 节中已做详细阐述。

(6) 突破限制的方法。

对上述数据项的诸多限制，认为是合理的并且是建立正确、有效的 CDM 必须遵守的规则，这对以后生成合理、稳定的物理模型具有重要的意义。

尽管如此，PowerDesigner 作为一个功能强大的软件，还是提供的解除这些限制的方法。

这些配置在选择 Tools(工具)→Model Options(模型配置)菜单弹出的对话框的 Model(模型)目录下，其选项包括 Unique Code(代码唯一)和 Allow Reuse(允许重用)，默认情况下，这两个配置均被打开，关闭 Unique Code(代码唯一)将允许两个数据项有相同的代码，关闭 Allow Reuse(允许重用)，将使一个数据项只能被一个实体的属性使用。

在生成的物理模型的对话框中，有选项：Convert Names into Codes(把名称转化为代码)，打开它 PowerDesigner 将用属性的名称取代代码作为表的列名。

在此，强烈地建议不要改变这些配置，在遇到问题时，应该尽可能地去改变 CDM，很有可能这些问题是因为建立了错误的 CDM 所造成的。

(7) 对数据项的小结。

综上所述，可以得到关于使用域和数据项代码设置的建议。

① 对存在不同含义但具有相同数据定义的属性，可以使用域。

② 对存在相同含义又具有相同数据定义的属性，使用相同的属性代码，即视为同一个数据项。

③ 对具有不同数据定义的属性，必须取不同的代码，尤其对那些含义相同但具有不同数据定义的属性，要避免属性代码相同。

④ 对主码属性，要确保其代码在整个 CDM 中的唯一性，可以用实体的代码名(或部分

缩写)作为其代码的前缀。

3.3.3 实体

在实体的特性中,主要介绍属性(Attribute)、规则(Rule)和标识(Identifier)。

1. 实体属性

在实体的特性窗口的属性框中设置实体的属性。选择某个实体属性,并双击该属性前的→符号,可设置该属性的各种特性,其内容与域的设置相同。

数据类型和域设置:只需要设置其中的一个,选择了数据类型后再选择域,域的定义将取代选择的数据类型,如果先选择域然后去设置数据类型,而两者不一致,系统会提示,可以选择用域的定义将取代选择的数据类型,也可以保留两者,但在生成物理模型时将出现错误。

属性的 M、P 和 D 含义分别如下。

M=Mandatory(强制):选中表示属性值必须非空。

P=Primary Identifier(主标识):选中则表示该属性为主码或是主码的组成部分。

D=Displayed(显示):选中则在图中显示该属性,否则不显示。

2. 规则

对单个实体属性的取值约束,可以通过属性的标准检查(Standard Check)特性设定其取值的约束条件(参见 3.3.1 节),有时一个实体属性之间的取值可能存在一定的约束关系,这时就必须使用规则(Rules)。

首先,在 Browse(浏览)窗口的概念数据模型下,新建一个业务逻辑(Business Role),在 Expression(表达式)下的 Server(服务)框中输入某个实体各属性要满足的表达式,在 General(一般)框中的 Type(类型)中选择 Constraint(约束)。

然后,在实体的特性设置窗口中,选择 Rules(规则)框,然后单击 Add Objects(增加对象)按钮,系统将列出所有可供选择的规则,选择要应用于本实体的规则,确定后就完成了实体规则的设置。实体约束类型的规则将在生成物理模型的数据定义语句中表现为表级约束。如成本价格和销售价格为商品实体的两个属性,其必须符合规则:成本价格≤销售价格。

假设成本价格和销售价格属性的代码分别为 Costprice 和 Saleprice,则规则的表达式应该设置为:Costprice <=Saleprice,生成物理模型的 SQL Server 的数据定义语句为:

```
create table goods(
BarCode       char(15)          not null,
GName         varchar(20)       null,
Gunit         varchar(10)       null,
CostPrice     decimal(10,2)     null,
SalePrice     numeric(10,2)     null,
constraint PK_GOODS primary key(BarCode),
constraint CKT_GOODS check(Costprice < = Saleprice)
)
```

3. 实体标识

所谓标识指一个实体的属性或属性组合,其值能唯一地标识一个实体。对那些属性或属性组的值必须唯一的属性,可以把它们定义成标识。定义方法为:在 Identifiers(标识)框

中增加一个标识，然后双击新增标识前的→符号，在弹出的对话框中选择 Attributes（属性）框，单击 Add Attributes（增加属性）按钮，在列出的可用属性前选择取值必须唯一的一个或一组属性。

一个实体可有多个标识，但只能指定一个为主标识（Primary Identifier），即主码，其他标识一定为非主标识。在生成物理模型时主标识对应属性即为主码，非主标识对应属性被定义为唯一性（Unique）约束。主标识和非主标识的共同点是对应的属性或属性组的值都必须唯一，区别在于两方面：一方面是主标识对应属性不能为空，但非主标识的属性或属性组的值在确保唯一的情况下可以为空，若把“姓名”属性定义为非主标识，则所有实体姓名不能相同，但允许存在一个空值；另一方面是实体的主标识只能有一个，而非主标识可以有多个。

在按照前面介绍的方法给一个实体属性或属性组定义主标识后，PowerDesigner 会自动创建一个 Identifier_1，其对应的属性就是构成主标识的属性。但去除已定义成主标识属性的 Primary Identifier 标识，并不会自动删除 Identifier_1，但对应属性被自动删除，Identifier_1 就变成一个不对应任何属性的空标识，这时生成物理模型时会出现错误，必须手工删除 Identifier_1。反之，删除 Identifier_1 或去除对应属性或去除其 Primary Identifier 标识，则实体中被标以主标识（P＝Primary Identifier）的属性将自动去除该标识。

如在一个销售系统中，商品实体的条形码可以作为主码，在一品一码的条件下，可设置商品名称属性为标识，但如果同一商品不同包装单位的条码不同而名称相同，就只能把商品名和包装单位两个属性构成属性组作为标识。生成的物理模型的 SQL 语句为：

```
create table goods (
barCode     char(15)          not null,
GName       varchar(20)       null,
Gunit       varchar(10)       null,
constraint PK_GOODS primary key(barCode),
constraint AK_IDENTIFIER_2_GOODS unique(GUnit, GName)
)
```

3.4 实体之间的关系

PowerDesigner 提供了两种方法建立实体之间关系：E-R 模型表示法和 Merise 表示法，Merise 为一种信息系统设计和开发的方法，类似 UML。

E-R 模型表示法通过使用 Palette（调色板）窗口中的工具 Relationship（关系）建立实体之间的各种关系，Merise 表示法则通过使用 Palette（调色板）窗口中的工具 Association（关联）和 Association Link（关联连接）建立实体之间的各种关系。在一个 CDM 中，实体之间的关系可以只使用关系或只使用关联来表示，也可以两者同时使用。

3.4.1 关系

选择 Palette（调色板）窗口中的关系，假如要建立实体 A 和实体 B 的关系，在概念数据模型的图中，把鼠标指针移动到实体 A 上，按住鼠标左键，然后把鼠标指针移动到实体 B 上放开鼠标左键，这样就建立了实体 A 和实体 B 的一对多的关系，在图形上显示为一根连接

实体 A 和实体 B 的两端带有特殊形状的线段。

然后根据需要对建立的两个实体的关系进行修改，即要修改关系的特性，双击图形上连接两个实体的线段，将弹出关系特性的设置窗口，其中最重要的是 Detail（细节）框中的内容，在这里可以对 1∶1、1∶n 和 n∶m 三类关系进行更细致的设置，在生成物理模型时，这些设置将对生成的物理模型产生影响。

在 Detail（细节）框中，关系的基本特性包括关系类型、1∶1 关系中的主导作用（Dominant）、依赖关系（Dependent）、强制关系（Mandatary）和基数（Cardinality）。

关系类型在 Detail 窗口表示为单选框，其选项包括“One-One(1-1)”“One-Many(1-多)”“Many-One(多-1)”和“Many-Many(多-多)”。不同的关系类型在图形上表示为连接两个实体的线段的两端所具有的不同的形状，如实体 A 和实体 B 为 One-Many 关系，则实体 A 与线段为一线（表示 1）连接，实体 B 与线段为发散的三线（表示 n）连接。

以下假设建立关系的两个实体的名称分别为 A 和 B，在 Detail（细节）框中分别有“A to B”和“B to A”两个面板，其中均包含了角色名、依赖与强制的选项，其中两个面板中的“角色名”分别用于描述“实体 A 和实体 B”和“实体 B 和实体 A”的关系，对生成物理模型不产生影响。

1. 主导作用

该选项只有在 One-One(1-1)情况下可选，其选项包括“None”“A to B”“B to A”。“学生”和“校园卡”之间的 1∶1 关系中，“学生”为起主导作用的实体，总是先有学生，然后才会有对应的校园卡，所以应该选“学生 to 校园卡”，生成的物理模型中将在校园卡对应的表中出现“学号”列作为学生表的外码。None 表示两个实体没有哪个实体起主导作用，产生的物理模型将在两个实体对应的表中分别产生外码。这种情况在实际应用中比较少见。

通常，可以把两个 1∶1 关系的实体合并成一个实体，但如果这两个实体分别处在两个相对独立的子系统中，自然合并就不合理，如校园卡系统可能就是一个相对独立的系统，教学管理系统中的学生实体的很多信息在校园卡系统中可能并不需要，合理的做法是把它作为一个单独的实体，具有自己特有的属性，然后在整个系统中，与教学管理系统中的学生实体建立 1∶1 的关系。

2. 依赖关系

“A to B”的依赖关系表示要确定实体 A 的一个实例，必须首先确定实体 B 的一个实例，或者说 B 的标识是构成 A 的标识的一个部分。

“A to B”的依赖关系反映了一个实体 A 对实体 B 的依附关系，其基本特征是实体 A 必须通过实体 B 来唯一标识

【例 3-6】 对一个学校，“学生”实体的学号仅仅为他所在班级中的序号，那么已知一名学生的学号，要确定这名学生，首先就要确定他所在的班级，所以“学生”实体和“班级”实体的关系为多对一的依赖关系，如图 3-6 所示。

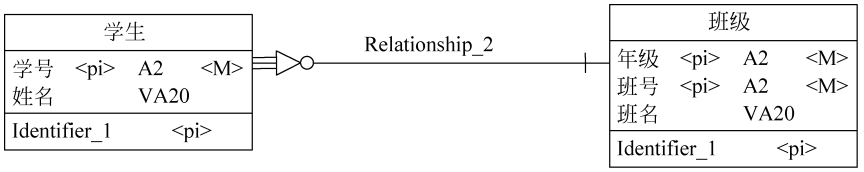

图 3-6 “学生”实体对“班级”实体的依赖关系

“A to B”的依赖关系对生成的物理模型的作用是：实体 B 的主码和实体 A 的主码合起来作为 A 表的主码，同时实体 B 的主码作为 A 表的外码。就例 3-6 而言，生成的学生表将以年级、班号和学号作为主码，其中学号为“学生”实体的主码，年级和班号为“班级”实体的主码，同时在学生表中年级和班号被设置成外码，反映了学生表与班级表的引用关系。

以上的概念数据模型中学生对班级的依赖关系，是以标识学生的学号仅仅为班级中的序号即学号本身不包含班级信息为前提的，如果学号本身包含了班号，则这种依赖关系自然就不再成立。如仍然用依赖关系建立学生和班级的关系，就会产生问题。如在实际的教学管理中，分配给学生的学号通常在学校范围内是唯一的，即包含了学生所在班级的信息，“学生”实体仅以“学号”单列就能构成主码，图 3-7 所示的学生表中由年级、班号和学号构成的主码就不符合码的最小性原则。

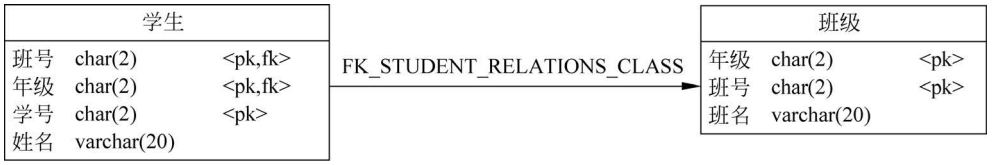

图 3-7　依赖关系生成的物理模型

学号中包含班级信息在 2.3.2 节中已说明这种设计不属于 1NF，这是造成以上概念数据模型中概念混乱的根本原因，这从一个侧面再次证明了 2.3.2 节中对 1NF 论述的有效性。所以建议不要采用惯性思维把教学管理中的组合了诸多信息的学号直接作为学生的属性，而应该用学生在班级中的序号作为学生的学号，只要确保学生的学号属性在他依赖的“班级”实体中是唯一的。在需要输出教学管理中的学号时，只需要把相关的属性组合起来。

这样就引出了一个问题，即在引入依赖关系后，对一个实体的主码属性的唯一性必须做一个修正：如果一个实体存在与另一个实体的依赖关系，则该实体的主码属性只需要在其依赖的实体范围内唯一。另外，从以上对依赖关系含义的分析上可以看出，依赖关系通常只能发生在多对一（如学生对班级）的关系中，因为从逻辑上前者通常从属于后者，后者包含了前者。实体的依赖关系具有传递性，如图 3-8 所示。

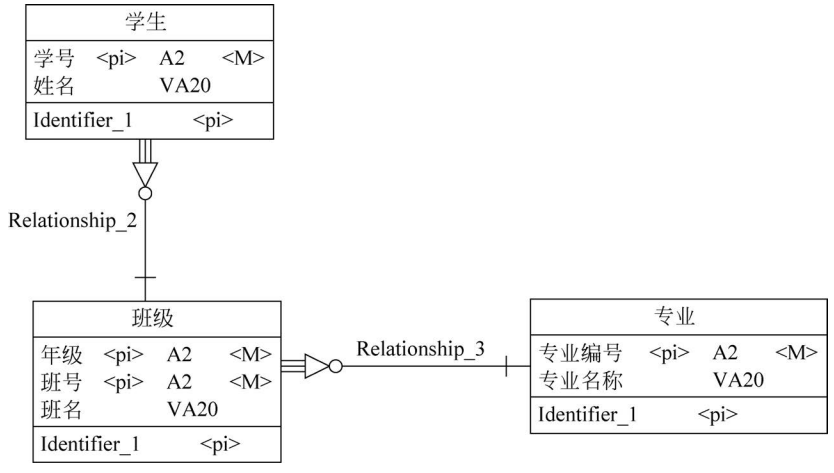

图 3-8　传递的依赖关系

【例 3-7】 例 3-6 中，“班级”实体又是多对一地依赖于“专业”实体的，所以真实的教学管理系统中完整的概念数据模型以及产生的物理模型如图 3-9 所示。

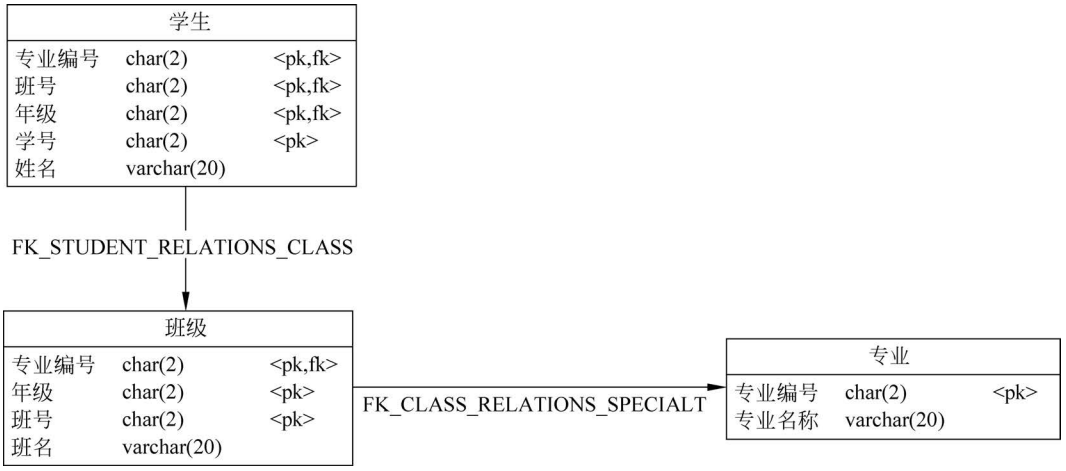

图 3-9 传递的依赖关系生成的物理模型

3. 强制关系

“A to B”的强制关系的含义是 A 的每个实例需要一个 B 的实例与之对应，在生成的物理模型中，A 实体对应表中对 B 实体对应表的外码属性被定义为非空。“学生”和“班级”的多对一关系通常情况下就是一个强制关系，因为每名学生必须对应一个班级，在生成的物理数据模型中，学生表中的班号被定义成非空，即每名学生必须有一个班号。

强制关系在图形上表示为在连接 A 和 B 实体线段的 B 端，出现一根与线段垂直的小线段，可以理解成“1”，表示 A 的每个实例必须至少有一个 B 的实例与之对应，参见图 3-8 中学生和班级关系中“班级”这一端。非强制关系则在同样位置上出现一个小圆圈，可以理解成“0”，表示 A 的每个实例可以没有 B 的实例与之对应，参见图 3-8 中学生和班级关系中“学生”这一端。

对 One-One(1-1)情况，如“学生 to 校园卡”，假设学生可不办校园卡，则“学生”和“校园卡”非强制关系，否则为强制关系。对 Many-One(多-1)情况，如学校为学生开设了若干兴趣班，每名学生最多可参加一个，则“学生 to 兴趣班”为多对一关系，假设一名学生可不参加任一个兴趣班，则学生和兴趣班的多对一的关系为非强制关系；假设一名学生必须参加一个兴趣班，则就为强制关系。

并非所有的强制关系都会对生成的物理数据模型产生影响，在学生和班级的多对一关系中，若设置班级对学生为强制关系，其含义是一个班级必须至少有一名学生，这样的设置将不对物理数据模型产生影响，事实上，物理数据模型无法做到这一点。从实际应用的角度讲，总是先设置班级再设置学生信息，即在设置班级信息时，学生信息可能还不存在，所以上述的班级对学生强制关系并不成立。

4. 基数以及和其他特性的关系

“A to B”的基数可选“0,1”“1,1”“0,n”“1,n”，其中，“m,n”表示一个实体 A 可对应 $m\sim n$ 个实体 B。

(1) 基数和关系类型的关系。

基数是对“1-1(One-One)”“1-多(One-Many)”“多-多(Many-Many)”关系类型的更细的划分。如果 A 和 B 为“1-多(One-Many)”关系,则:

① “A to B”可选的基数为“0,n”和“1,n”,前者表示对 A 中任一个实体,B 可以没有实体和它对应,后者则至少要有一个实体和它对应。

② “B to A”可选的基数为“0,1”和“1,1”,前者表示对 B 中任一个实体,A 可以没有实体和它对应,后者则一定有一个实体和它对应。

可以通过改变“A to B”和“B to A”的基数来改变 A 和 B 的“1-1(One-One)”“1-多(One-Many)”“多-多(Many-Many)”关系类型,系统将自动保证基数和关系类型的一致性。

(2) 基数和依赖、强制之间的关系。

基数和依赖、强制之间存在下列约束关系。

① 0,1:不强制、不依赖。选择“0,1”,系统自动选择不强制和不依赖。

② 0,n:不强制、无依赖。选择“0,n”,系统自动选择不强制,依赖选项不可选(灰色)。

③ 1,1:强制、依赖可选。选择“1,1”,系统自动选择强制,依赖可选。

④ 1,n:强制、无依赖。选择“1,n”,系统自动选择强制,依赖选项不可选(灰色)。

“A to B”的基数为“1,1”或“1,n”,表示 A 中任一实体一定有 B 的至少一个实体与之对应,这就是强制关系的含义。

若“A to B”为依赖关系,根据分析,表明 A 和 B 为多对一的关系,所以只有“A to B”的基数为“0,1”和“1,1”才可能存在依赖关系。对“0,1”,表示 A 中实体可以在 B 中没有对应实体,就图 3-6 的实例来讲,如果学生对班级的基数为“0,1”,表示学生可以没有班级,那么学生就不能依赖于班级,所以基数为“0,1”则一定不能是依赖关系,最后结论是只有基数为“1,1”的情况才可能存在依赖关系。

5. 多对多关系

实体之间的“多-多(Many-Many)”关系在生成物理模型时将生成一个独立的表,该表包含了两个实体的主码属性,并以此为生成表的主码和外码。

【例 3-8】 假如“学生”和“课程”的关系为多对多关系,如图 3-10 所示,图 3-11 是它们的概念模型以及由此生成的物理数据模型。

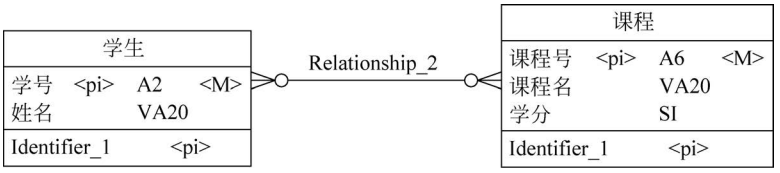

图 3-10 多对多关系

上面这种做法仅仅表达了两个实体的关系,若关系本身包含了属性,如“学生选课”的“成绩”,就必须把关系转换为实体,具体做法是在概念数据模型的图中右击表示多对多关系的线段,在弹出的快捷菜单中选择 Change to Entity(转换为实体)→Standard(标准),多对多的连线将被分解,并出现一个实体,修改实体的名称为“学生选课”,然后在实体中加入“成绩”属性,得到如图 3-12 所示的概念数据模型。

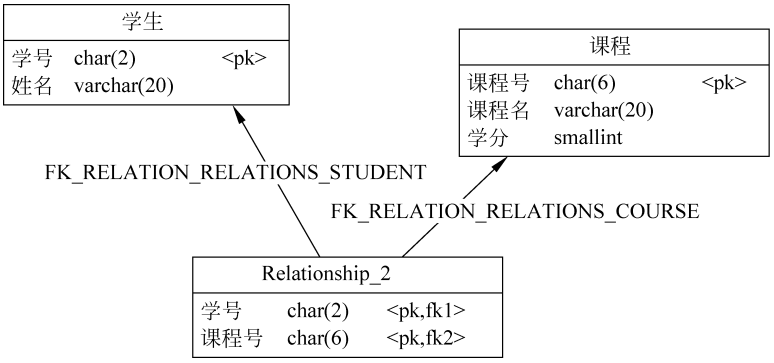

图 3-11　多对多关系生成的物理数据模型

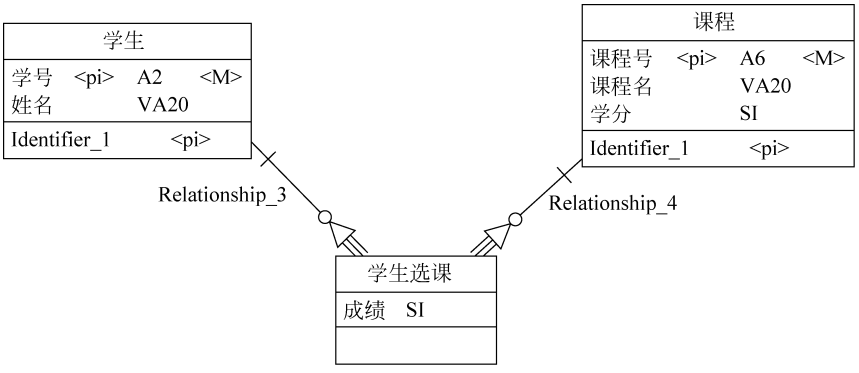

图 3-12　多对多关系转换为实体

从图 3-12 中可以看到，多对多关系被分解成两个多对一的强制的依赖关系，依赖关系表明最后生成的“学生选课”表将以“学生”和“课程”表的主码属性为主码，强制关系表明“学生选课”表中的“学号”和“课程号”必须非空，但由于“学号”和“课程号”为“学生选课”表的主码，因此非空要求必然会被满足，图 3-13 是生成的物理数据模型。

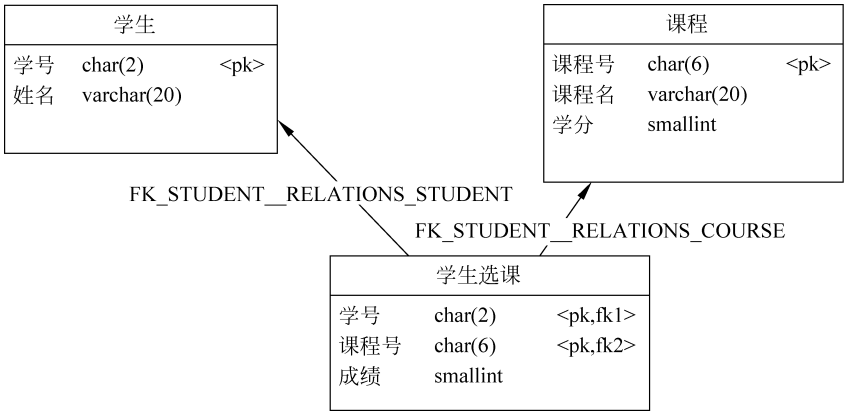

图 3-13　包含属性的多对多关系生成的物理数据模型

3.4.2　关联和关联连接

关联和关系相同，也是用于建立实体之间的关系，主要区别是一个关联可以通过关联连

接与多个实体连接，并可包括自己的属性。

关联连接的使用方法和关系基本相同，即可以直接用关联连接连接两个实体，结果将产生一个关联和两个连接关联与实体的关联连接。也可以先加入一个关联，然后用关联连接分别把关联和两个实体连接起来。

图 3-14 中间为关联，连接学生和班级的分别为两个关联连接，双击关联连接可设置其特性，主要内容为“基数”和“是否为标识”。

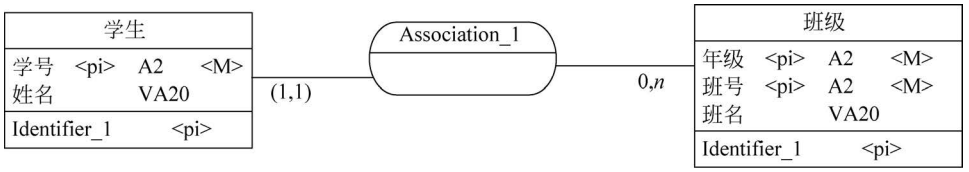

图 3-14　使用关联建立的学生和班级关系

基数的含义和关系一致，但是是相对于关联而言的，“1,1”表示一名学生在关联中出现并只出现一次，而“0,n”表示一个班级在关联可出现 0～n 次，其实质还是表示学生和班级的多对一关系。注意，不要像关系一样把“1,1”和“0,n”直接理解为学生和班级关系的基数，那样会得出相反的结论。

在关联中没有属性的前提下，生成的物理数据模型和关系建立的学生和班级关系一样，“1,1”外加的括号表示该关联连接中选中了标识选项，其作用等同于在关系中设置了学生对班级的多对一关系为依赖关系，图 3-14 生成的物理数据模型与图 3-6 所示的用关系建立概念数据模型生成的物理数据模型相同，见图 3-7。

可以在关联中设置属性，设置的方法和在实体中设置属性的方法一样。

【例 3-9】 图 3-15 是使用关联建立的“学生”和“课程”的多对多关系，其表达的含义等效于图 3-12，生成的物理数据模型与图 3-13 相同。

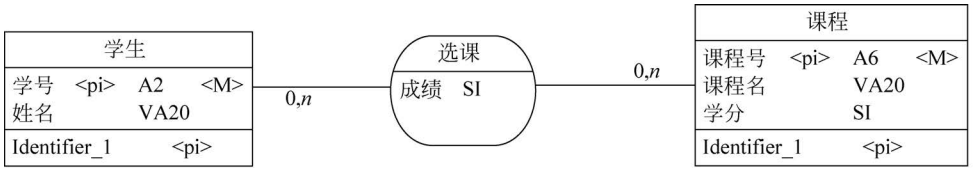

图 3-15　用关联建立的多对多关系

3.4.3　关系和关联的使用特点

针对关系和关联的使用特点，建议在一个概念数据模型下，根据不同的情况使用不同的方法。

- 对 1∶1、1∶n 和不包含属性的 m∶n 关系，使用关系。
- 实体之间的 m∶n 关系且包含属性的使用关联比较简单。
- 处理两个以上实体的 m∶n 关系，使用关联。
- 在处理比较复杂的问题时，如要建立关系之间的关系或关系与实体之间的关系，只能使用关系，因为关系可以把关系转换为实体，该实体又能和其他实体建立关系，这点关联无法做到。

在 3.6 节中，可以看到关于概念数据模型设计的更多实例。

3.5 继承关系

引入继承是借鉴了面向对象的程序设计思想，是对概念数据模型的又一个扩展。继承关系的一端连接具有普遍性的实体，称为父实体(Parent Entity)，继承关系的另一端连接具有特殊性的一个或多个实体称为子实体(Child Entity)。

一般可以把具有诸多相同属性的实体中的公共属性抽象出来作为父实体，其他实体从父实体继承这些属性，这样做可以确保公共属性的一致性，也节省了重复设置这些属性的工作量，在子实体之间不存在相互引用的情况下，也可以避免主码属性名的排他性。

【例 3-10】 一个业务系统中可能会处理大量的各种类型的单据，这些单据可能都具有单据号、日期等公共属性，可以把这些公共属性组成一个抽象的“单据”实体作为父实体，具体的单据实体继承这个抽象的“单据”实体，这时所有单据的单据号(主码)都从父实体继承过来，所以可以同名。

【例 3-11】 “学生”可以作为“本科生”与“研究生”的父实体，后者为前者的子实体，可以建立如图 3-16 所示的概念数据模型，由于“学生”实体为一个抽象的实体，实际并不存在，因此在生成物理数据模型时并不需要生成对应的表，可以在该实体的特性窗口的 General(一般)特性框中关闭 Generate(生成)选项，生成的物理数据模型如图 3-17 所示。

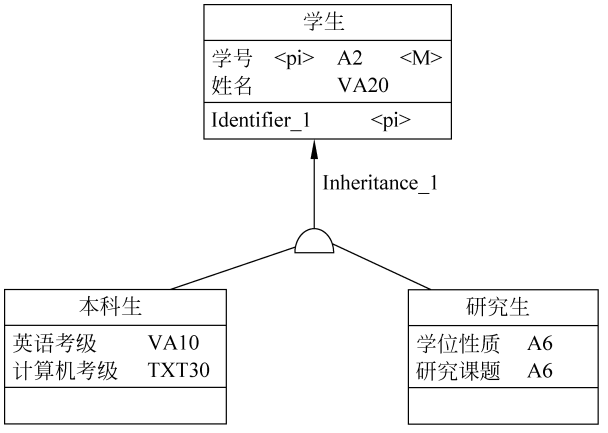

图 3-16 包含继承关系的概念数据模型

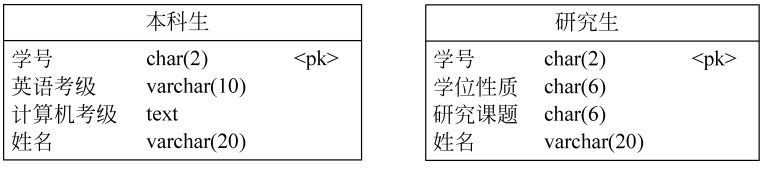

图 3-17 仅生成子表的继承关系生成的物理数据模型

通过对继承关系的特性设置可以控制由此生成的物理数据模型。

在特性窗口的 Generation(生成)框中，Generate Parent(生成父表)的设置和父实体的特性窗口中的 Generate(生成)的设置作用是一样的，关闭这个选项，父实体对应的表将不在物理数据模型中生成。打开这个选项，则父实体对应的表将在物理数据模型中生成，并且

子实体对应的表将以父实体的主码属性为主码同时为外码。

在打开 Generate Children(生成子表)选项后,可选 Inherit All Attributes(继承所有属性)或 Inherit only Primary Attributes(仅继承主属性),这两个选项确定在物理数据模型中对应的表是包含父实体所有属性还是仅包含父实体的主属性。

如关闭 Generate Children(生成子表)选项,则仅生成父实体对应的表,该表将包含子实体所有属性并可在特性窗口的下方设置 Specifying Attribute(指定属性),这些属性将出现在父实体对应的表中,通常用作区分每一行对应的是哪个子实体。

【例 3-12】 本科生实体和研究生实体共同属性很多,而在不同属性很少的情况下,可仅生成父表,即用一个表来存放本科生和研究生实体信息,在图 3-16 所示的概念数据模型中,在继承关系的特性窗口的 Specifying Attribute(指定属性)中设置一个“学生类型”属性,该属性将出现在生成的父实体对应表中,以区分每行是本科生还是研究生,最后生成的物理数据模型为如图 3-18 所示的单个表。

学生		
学号	char(2)	<pk>
姓名	varchar(20)	
学生类型	bit	
学位性质	bit	
研究课题	char(6)	
英语考级	varchar(10)	
计算机考级	text	

图 3-18 仅生成父表的继承关系生成的物理数据模型

另外,在继承的特性设置中,有一个称为 Mutually Exclusive Children(互斥性继承)的选项,其含义是同一事件不能出现在同一父实体的两个子实体中,这种继承称为互斥性继承,反之,则称为非互斥性继承。如父实体中一名学生只能是本科生或研究生,两者取其一,则“本科生”和“研究生”为“学生”的互斥性继承,否则,如一名学生可同时为本科生和研究生,则为非互斥性继承。此设置只影响文档而不影响生成的物理数据模型。

综上所述,在使用继承时,应注意以下事项。

- 使用继承的前提条件是子实体具有较多的共同属性。使用继承可避免共同属性重复维护并保持其一致性。
- 在子实体之间共同属性相对少而差异大的情况下,可选择仅生成子实体对应表。在子实体之间非互斥情况下,会有冗余。如上例中,同一名学生可能既是本科生又是研究生,其共同的信息如性别、出生日期必须同时存放在两张表中。
- 在子实体之间共同属性多而差异小的情况下,可选择仅生成父实体对应表。如子实体之间互斥,其中差异部分的属性值会有空值,会占用一定的空间。
- 子实体之间共性属性多而差异也大的情况下,可同时生成父实体对应表和子实体对应表,此情形子实体仅需要继承父实体的主属性。生成的物理数据模型中的子实体对应表会通过外码关联到父实体对应的表。互斥和非互斥情况均适用,互斥情况下父表的一行对应某张子表的一行,非互斥情况下父表的一行可能对应多张子表中的一行。

3.6 概念数据模型实例分析

本节通过分析几个典型的并在建立概念模型时比较容易出现错误的实例,进一步说明正确建立概念数据模型的方法和重要性。

3.6.1 单据的概念数据模型

在各种类型的业务系统中，需要处理大量的各种类型的单据，单据是记录业务流程数据的最常见的形式。如一个商场有订货单、进货单、退货单和销售单等单据；一个制造企业有采购单、领料单、出厂单等单据。由于单据通常都具有相似的数据形式，如何为单据建立正确的概念数据模型就显得非常重要，下面就以一个商场进货单为例进行说明。

【例 3-13】 商场一张进货单可能包含多个商品，其格式如图 3-19 所示。

进　货　单

进货单号：　　供应商名：　　进货日期：

货号	货名	单位	单价	数量	金额
合计					

图 3-19　商品进货单

同所有类型的单据一样，可以把进货单中的数据项分成两部分：一部分是和单据一对一关系的属性，如进货单号、进货日期，这些数据项组成单据摘要；另一部分是和单据一对多关系的属性，如货号、货名、单位等，这些数据项组成单据明细。

进货单是一个实体，进货单号和进货日期为它的属性，供应商名由于反映的是进货单和供应商实体的关系，因此不应作为进货单的属性，而应该通过进货单和供应商的多对一关系来反映，关于这一点由于和本例的主题无关，因此不予考虑。剩下的是货号、货名、单位等是否是进货单的属性，显然不是，因为这些数据项对进货单而言不是原子项。

所以就会看到一种常见的如图 3-20 所示的错误，如前所述，把单据分成单据摘要和单据明细两个实体，事实上这个概念模型是由物理模型反推出来的，其错误表现为把本为一个实体的单据分成两个实体（其实已经包含了设计思想），其中，进货单明细对商品实体和对进货单摘要的一对多关系被设置成依赖关系是为了使（单据号，商品条形码）成为单据明细的主码。

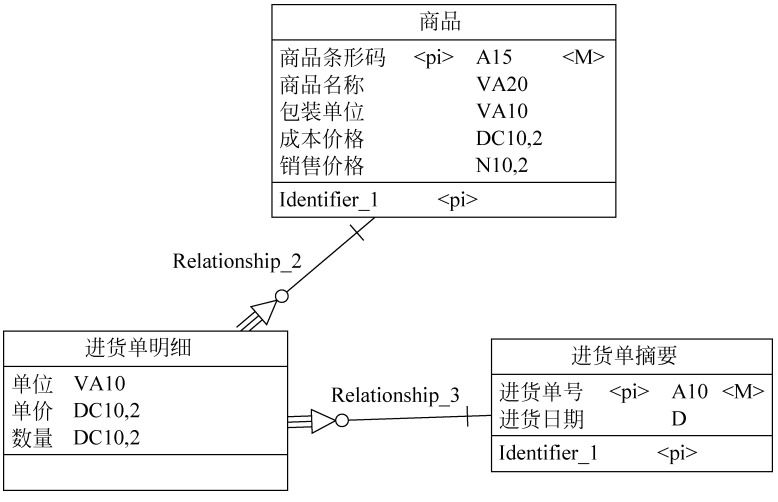

图 3-20　错误的单据概念数据模型

犯上述错误的原因是没能看出单据明细本质上是反映了单据实体和商品实体的多对多关系，所以更自然的符合实体概念的做法是如图 3-21 所示使用关联建立商品实体和单据实体的多对多关系，在多对多关联中增加单位、单价和数量属性，其生成的物理数据模型与图 3-22 所示的相同，只是进货单明细是由商品和单据之间的多对多的关联生成。

图 3-21　使用关联表达进货单明细

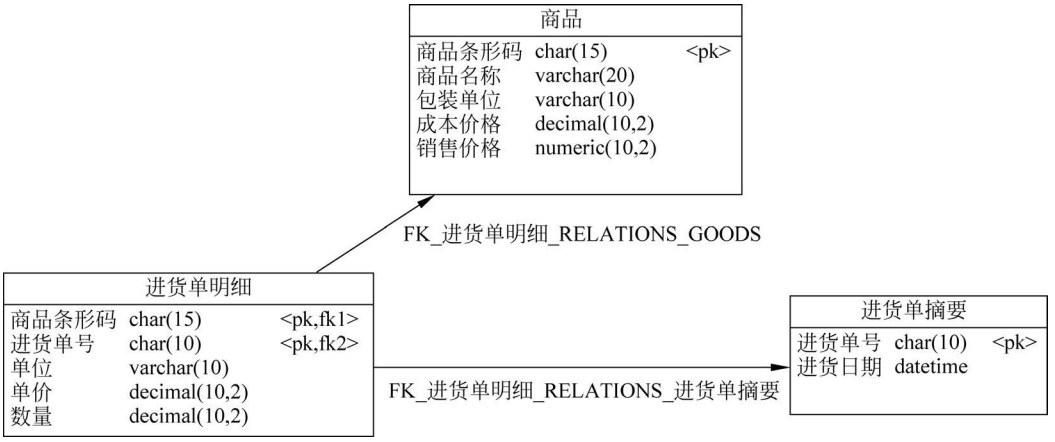

图 3-22　进货单物理数据模型

3.6.2　考勤系统的概念数据模型

【例 3-14】 某企业要求对员工的每日上下班、迟到、早退、出差、病事假进行记录统计，员工每日首次到公司必须打卡记录到公司时间，最后一次离开公司则必须打卡记录离开公司时间，其整天不到公司或工作日中间外出，则必须填写请假单。请假单分病假、事假和公假三种。设计概念数据模型如图 3-23 所示。

员工和考勤卡为 1∶1 关系，且员工对考勤卡为主导作用，考勤卡依赖于员工，请假单与员工为 1∶n 关系，且请假单对员工为强制关系。生成的物理数据模型如图 3-24 所示。

3.6.3　商品多供应商问题的概念模型

【例 3-15】 一个超市中某种商品的供应商一般一个时期固定为一个，所以商品和供应商的关系为多对一，但可能会有个别商品同时有多个供应商，即对个别商品，商品和供应商的关系为多对多。

通常的解决方案是把商品和供应商关系处理成多对多的关系，这也符合实体多对多关系的定义，生成的物理数据模型将包括商品表、供应商表和商品与供应商的对应表，因为对于极个别的商品，查询任何和商品及供应商相关的信息都必须连接三张表，查询效率较低。

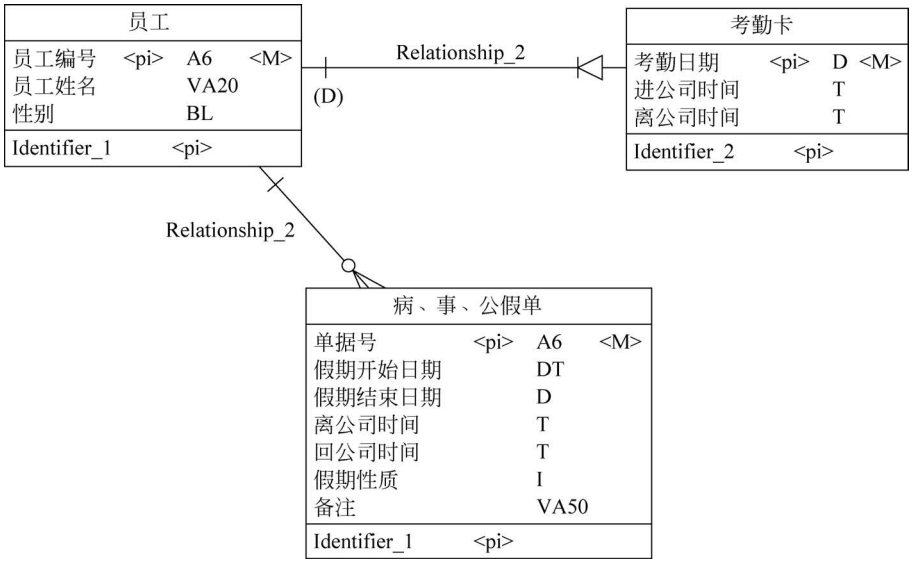

图 3-23　考勤系统概念数据模型

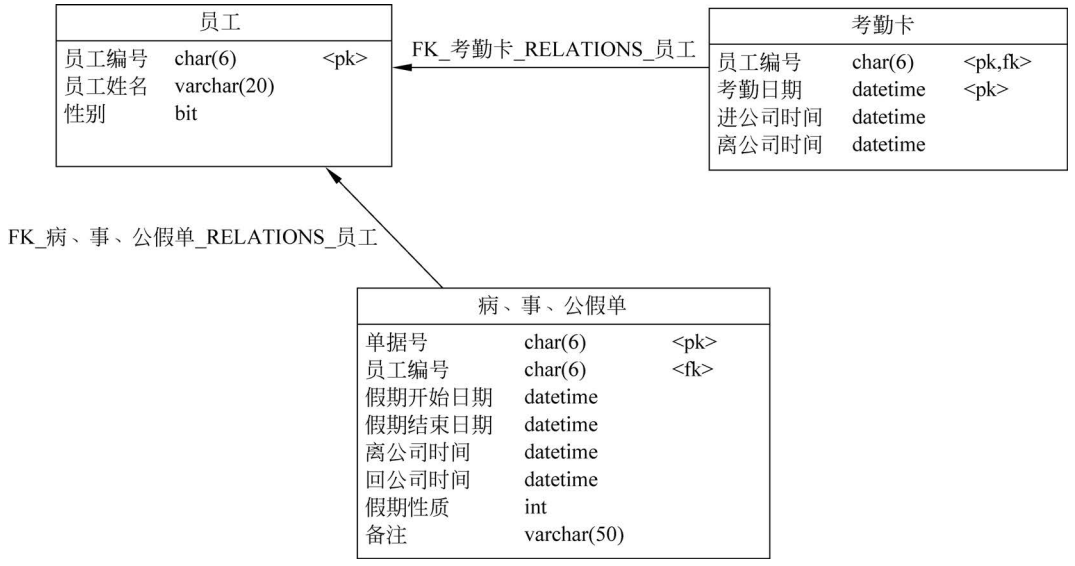

图 3-24　考勤系统物理数据模型

为了优化查询效率，在只有个别商品存在多供应商的情况下，设计人员在概念数据模型中把商品和供应商关系仍处理成多对一，即对于大多数商品，查询商品和供应商相关信息时，只与两张表相关，然后在生成的物理数据模型中增加一张表，存放少量的商品和供应商的多对多关系，只有在查询结果中包含这些个别商品时，才需要连接三张表。

如前所述，建立概念数据模型的一个原则是尽可能不要修改由概念数据模型生成的物理数据模型，问题是能否建立一个概念数据模型直接产生如上要求的物理数据模型。

答案是肯定的。

可以在商品和供应商上之间建立两个关系，分别为多对一和多对多的关系，然后在“商品”属性中增加一个“多供应商”标志属性，或指定一个特殊区段的商品号为多供应商的商品

编号。概念数据模型和生成的物理数据模型如图 3-25 和图 3-26 所示。

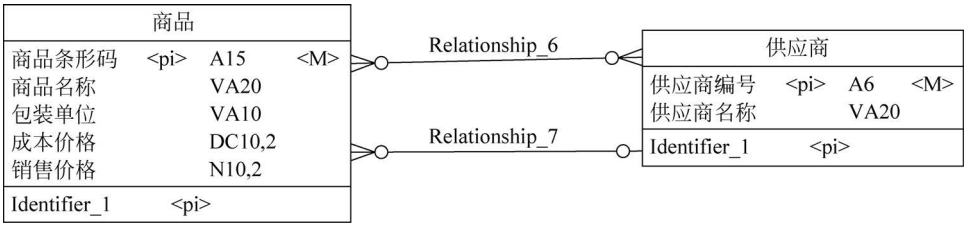

图 3-25　多供应商概念数据模型

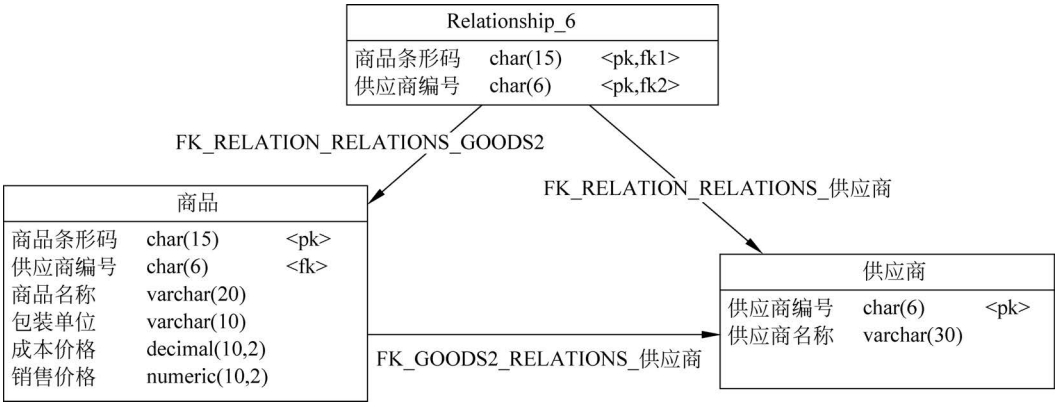

图 3-26　多供应商物理数据模型

对供应商唯一的商品，使用商品表的供应商编号外码指定该商品的供应商，对供应商不唯一的商品，使用供应商和商品对应表实现两者的多对多关系，若一个商场只有个别商品存在多供应商的情况，供应商和商品对应表可能只存在少量的数据。

此例表明，建立概念数据模型后，如果生成的物理数据模型需要优化，不一定要修改物理数据模型，而可以通过改变概念数据模型达到对生成物理数据模型优化的目的，前提是概念数据模型仍能如实地反映客观需求。如上例供应商和商品的两个关系分别表示对多数商品，商品实体和供应商实体为多对一关系；对个别商品，商品实体和供应商实体为多对多关系。

3.6.4　单据相关人员的处理

【例 3-16】　一个单据通常会包含多个相关人员，如输入员、审核员、记账员等，根据 3.1.2 节中的规则，这些人员不应该作为单据的属性，而应该通过单据实体和人员实体的关系来反映。

单据与相关人员的概念数据模型如图 3-27 所示，生成的物理数据模型如图 3-28 所示。这样的物理数据模型存在以下两个问题。

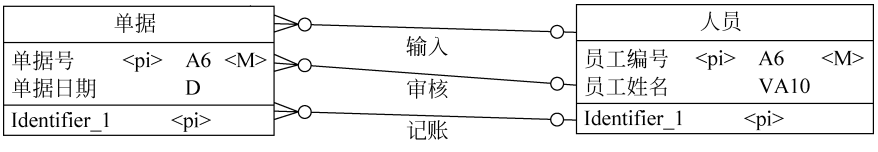

图 3-27　单据与相关人员的概念数据模型

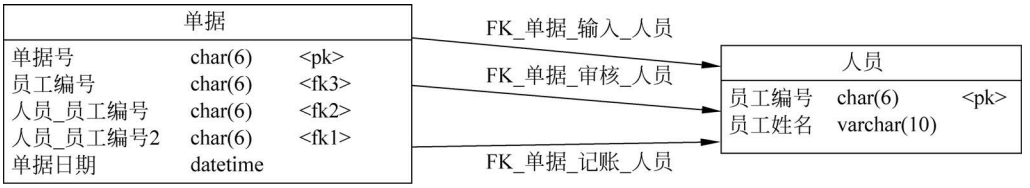

图 3-28　单据中员工的物理数据模型

第一个问题是由于在概念数据模型中无法控制生成对应表的外码名称，只能修改生成的物理数据模型，把这三个外码名称分别改为“输入员”“审核员”“记账员”，在每次修改概念数据模型、重新生成物理数据模型后，可能需要做重新修改物理数据模型。

第二个问题是从范式角度看，“输入员”“审核员”“记账员”为重复项，在单据的相关人员有变化预期的情况下，不符合 1NF。

是否有更好的方法建立概念数据模型，使得对生成的物理数据模型不必做修改而生成的物理数据模型符合所有范式？

由于“输入”“审核”“记账”分别表示员工对一张单据的责任，为此可以增加一个单据责任实体，包含单据的所有责任类型。

这里分析一下单据实体、人员实体和单据责任实体三者的关系。

- 单据和人员的关系为多对多关系，即一张单据对应多个责任人，一个人员可对应多张单据。
- 单据和单据责任也为多对多关系，即一张单据可有多个责任，一个责任对应多张单据。
- 人员和单据责任则要分两种情况：若一张单据中不同责任人可为同一个人，则人员和单据责任为多对多，即一个员工可以对应多个责任，而一个责任可对应多个员工；若一张单据中的责任人不能相同，则人员和单据责任为多对一，即一个人员只能对应一个责任，而一个责任仍能对应多个员工。

对上述第一种情况，可建立如图 3-29 所示的概念数据模型，生成的物理数据模型如图 3-30 所示。

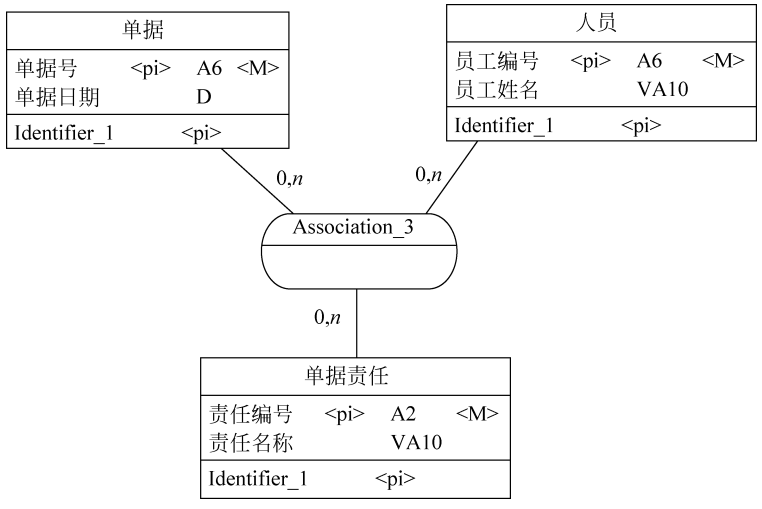

图 3-29　单据责任人概念数据模型一

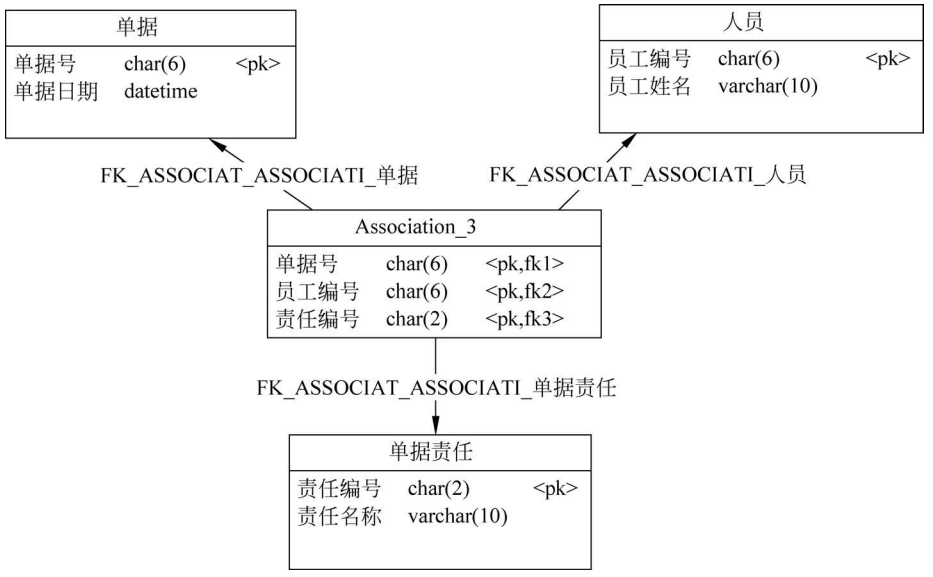

图 3-30　单据责任人物理数据模型一

其中，通过关联建立了三个实体相互之间的多对多关系，由于一个员工可以对应多个责任，因此对于一张单据的不同责任的责任人可为同一个人，如制单和审核可以为同一个人，这通常与企业的管理制度不一致。

所以建立人员和责任的多对一关系可能更符合实际的管理要求，而用一个关联建立的三个实体之间的关系，无法通过修改三个关联连接的基数同时满足它们两两之间的关系类型（原因读者自行思考）。

于是只能通过建立三个实体两两之间的关系来表达这个需求，概念数据模型如图 3-31 所示，生成的物理数据模型如图 3-32 所示。

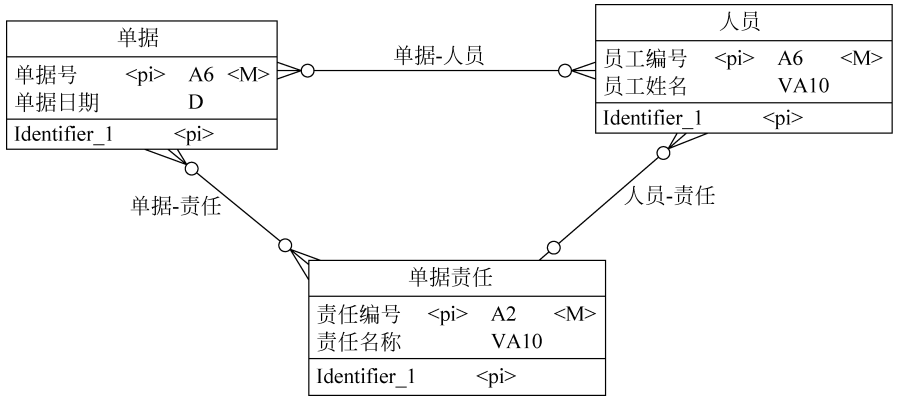

图 3-31　单据责任人概念数据模型二

通过“单据-责任”得到一张单据有哪些责任，通过“单据-人员”可以得到一张单据有哪些相关人员，至于这些相关人员对应哪些责任，由“人员”表中的每个人员唯一对应的责任编号来确定。

如此的概念数据模型仅适用于一种类型的单据，同时对一种类型的单据，单据和单据责

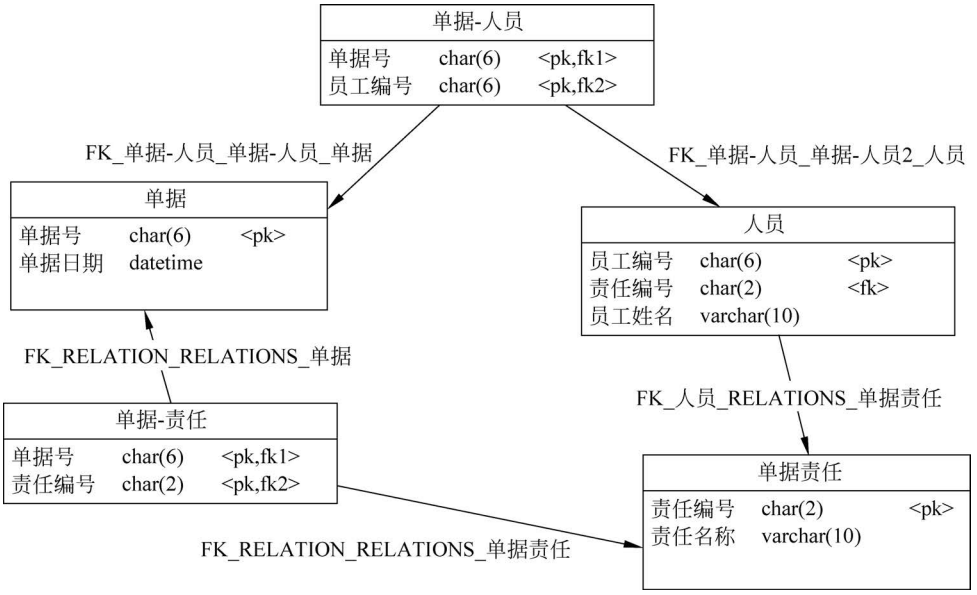

图 3-32　单据责任人物理数据模型二

任的关系是确定不变的。也就是说,“单据-责任”表中只需要存放一张单据和单据责任的关系就足够了,其他单据和单据责任的关系都是一样的,即该表存放了大量的冗余信息。由此就自然想到需要一个单据类型的实体,通过该实体和单据责任建立联系,就不会有冗余(一种类型的单据只对应一个单据类型的一个实体),同时也可以使概念数据模型适用于多种类型的单据。

单据和人员以及人员与单据责任的关系不变,原单据和单据责任的关系通过单据类型作为桥梁来建立:单据类型和单据责任的关系为多对多,即一个单据类型可以对应多个单据责任,而一个单据责任也可对应多个单据类型,如进货单要包括输入、审核、记账等多个单据责任,而输入的单据责任要出现在所有的单据类型中。单据和单据类型的关系很显然地为多对一关系,并且可以把它设置为依赖关系。概念数据模型和生成的物理数据模型分别如图 3-33 和图 3-34 所示。

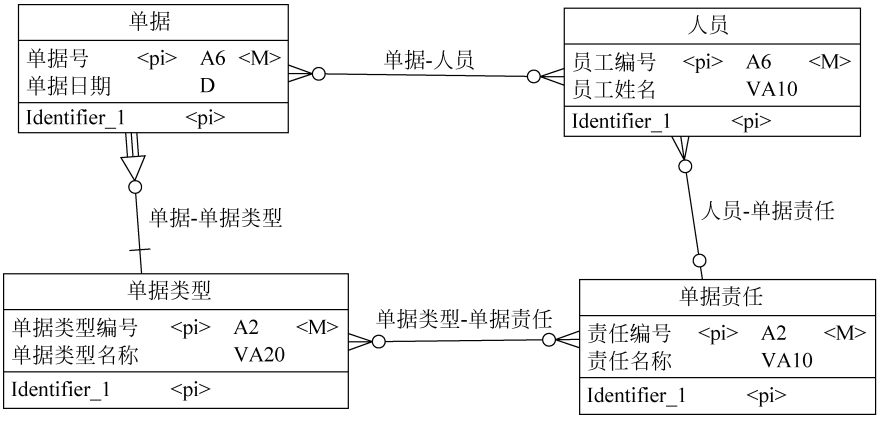

图 3-33　单据责任人概念数据模型三

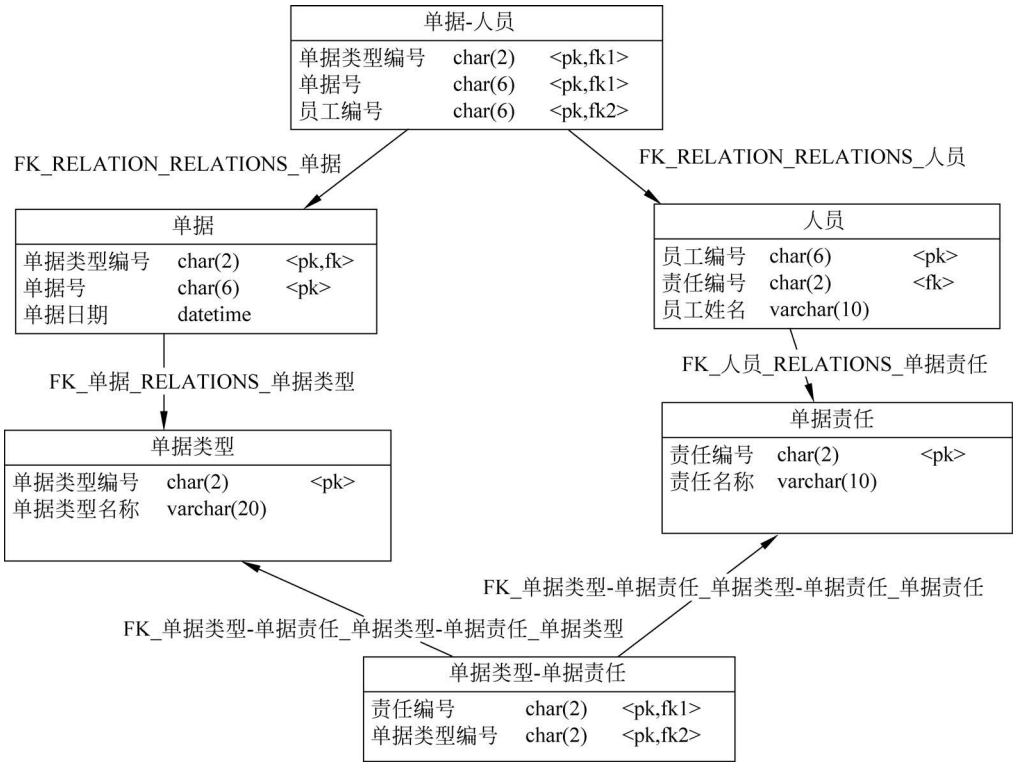

图 3-34　单据责任人物理数据模型三

上述的概念数据模型把所有单据作为一个实体处理，在实际应用中不同类型的单据数据项可能存在较大的差异，不同单据之间也可能存在关系，所以把所有单据作为一个实体不能满足这些需求，需要考虑把不同类型的单据作为不同的实体时，如何建立对应的概念数据模型。

可以在图 3-33 所示的概念数据模型三基础上进行扩展，把单据实体理解为由所有单据实体的公共属性构成的父实体，其他单据如订单、进货单等作为该实体的子实体，并且仅从父实体中继承主属性，并设置各自特有的属性，在 Inheritance（继承）的特性设置中同时选择 Generate Parent（生成父实体）和 Generate Children（生成子实体）对应的表，概念数据模型和生成的物理数据模型分别如图 3-35 和图 3-36 所示。

生成的物理数据模型中，所有单据表中都有单据类型编号，该编号对一种类型的单据来说值不变，是一种冗余的数据，但该冗余的数据是为了和各单据公共数据构成的单据建立引用关系，所以是必须和必要的冗余。

另一种方法是在概念数据模型中去除单据实体，让订单或进货单等单据实体直接和人员实体建立多对多联系，在生成的物理数据模型中，n 种单据将产生 n 个表达单据实体和人员实体多对多关系的表。读者可自行按此方法建立概念数据模型。

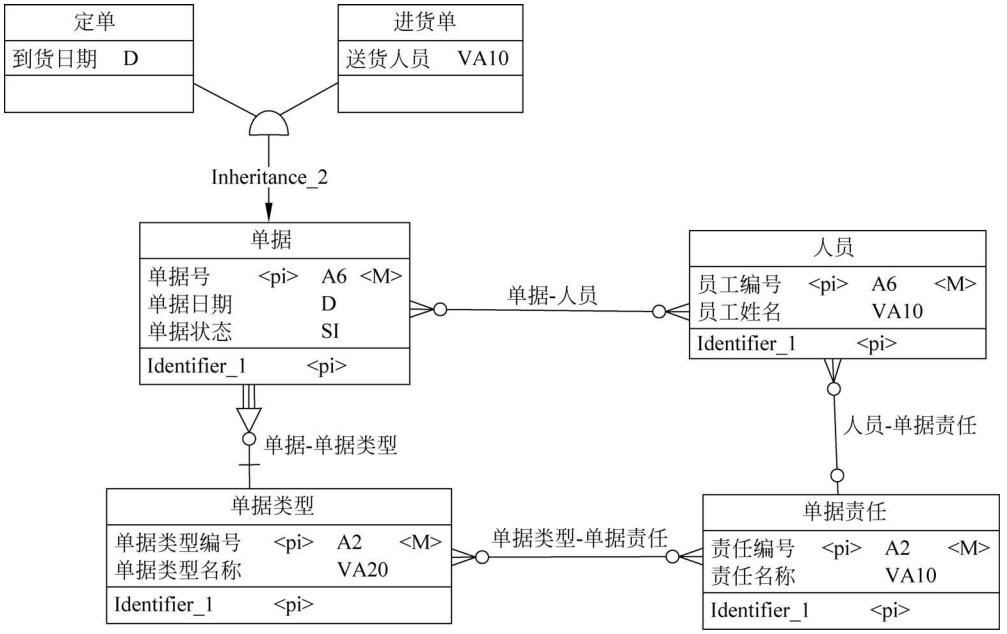

图 3-35　单据责任人概念数据模型四

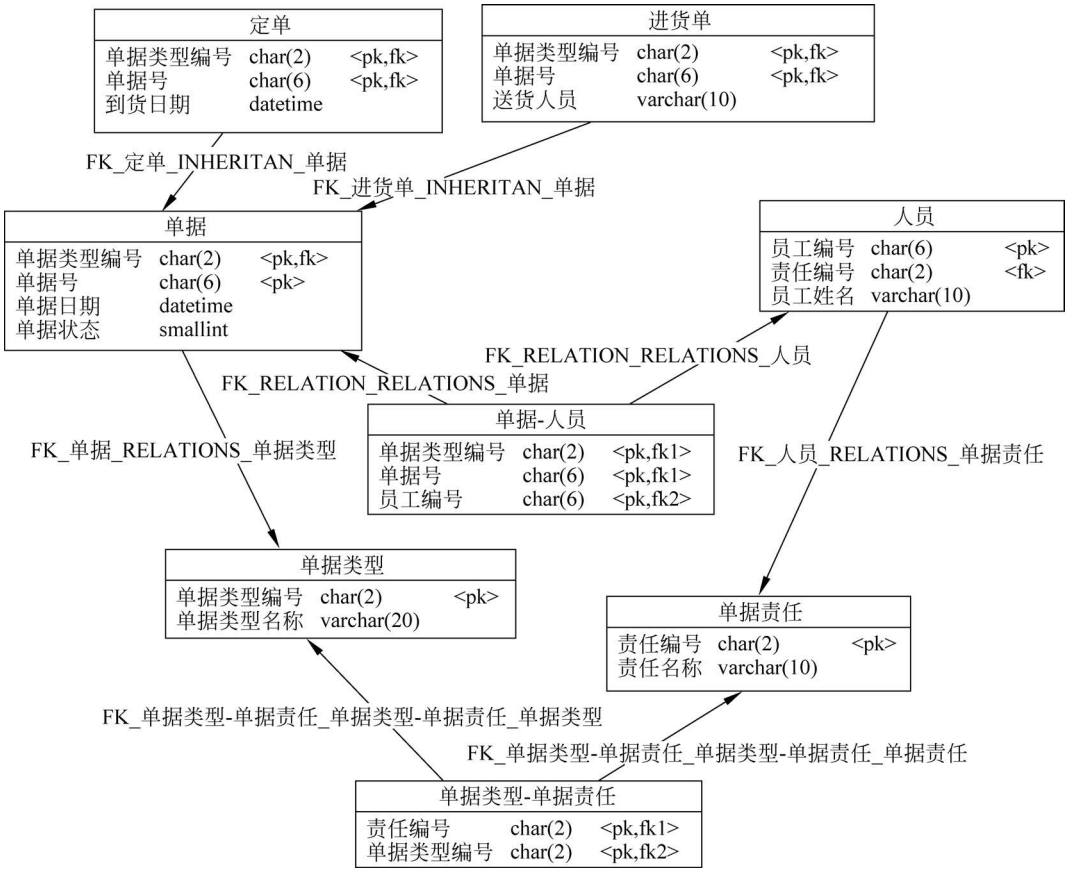

图 3-36　单据责任人物理数据模型四

3.7 PowerDesigner 的物理数据模型

建立了概念数据模型后，可以生成物理数据模型，即关系模型，正如 3.6 节看到的，物理数据模型同样以图形方式表达关系模型中各二维表的结构及其相互之间的引用关系。

物理数据模型并非一定要通过概念数据模型转换而来，也可以直接建立一个物理数据模型，或对生成的物理数据模型进行修改，建立和修改方法与概念数据模型类似。本节重点描述由概念数据模型如何生成物理数据模型以及两者之间的关系维护的基本思想和方法。

1. 生成物理数据模型

在完成概念数据模型的设计后，使用 Tools(工具)→Generate Physical Data Model(生成物理数据模型)菜单，弹出生成对话框生成物理数据模型。通过设置对话框中的 PDM Generation Options(PDM 生成参数配置)对生成进行控制。

若是第一次生成，则使用默认的选项 Generate New Physical Data Model(生成新的物理数据模型)，但必须选择数据库管理系统类型，这是由于不同的数据库管理系统要生成的数据定义语句可能略有不同，可以选择 Microsoft SQL Server 2000，单击“确定”按钮则生成物理数据模型。

生成过程首先是一个对概念数据模型的检查过程，错误分成两类：一类是警告；另一类是错误。警告不影响物理数据模型的生成，而一旦发现错误，则必须修改概念数据模型以排除这个错误，否则不能生成物理数据模型。

修改了概念数据模型后再次生成物理数据模型，仍可选择 Generate New Physical Data Model(生成新的物理数据模型)，此选择表示将生成一个新的物理数据模型，原来已生成的物理数据模型被保留，但默认的选择是 Update Existing Physical Data Model(更新存在的物理数据模型)，如果没有对生成的物理数据模型进行修改或不需要保留这些修改，可关闭 Preserve Modifications(保留修改)选项，单击“确定”按钮则重新生成 PDM 并覆盖原 PDM。

2. 修改物理数据模型

有时可能对生成的物理数据模型进行修改，此后又修改了概念数据模型，概念数据模型重新生成物理数据模型时，希望能保留原物理数据模型的修改。

解决方法是生成物理数据模型时，打开 Update Existing Physical Data Model(更新存在的物理数据模型)中的 Preserve Modifications(保留修改)选项，PowerDesigner 将根据 CDM 生成 PDM，并与原来生成后可能修改过的 PDM 比较，差异部分由用户选择哪些内容要根据新的 PDM 进行更新、删除或增加。

【例 3-17】 CDM 中有一名学生实体，包括“学号”“姓名”“性别”属性，“性别”类型为布尔型。

- 生成 PDM。
- 在 PDM 的学生表中增加“出生日期”。
- 将 CDM 中学生性别(Sex)类型由 Boolean 改成 Char(1)。
- 重新生成 PDM，选择 Update Existing Physical Data Model(更新存在的 PDM)中的 Preserve Modifications(保留修改)选项，单击“确定”按钮后弹出如图 3-37 所示的界面，左边是新生成的 PDM，右边是原生成并已增加“出生日期”的 PDM，其中，对于

所有比较后的不一致项用户可打钩表示用新 PDM 更新。更新方式是：不一致项上的“－”表示删除，“＋”表示增加，“＝”表示更新。

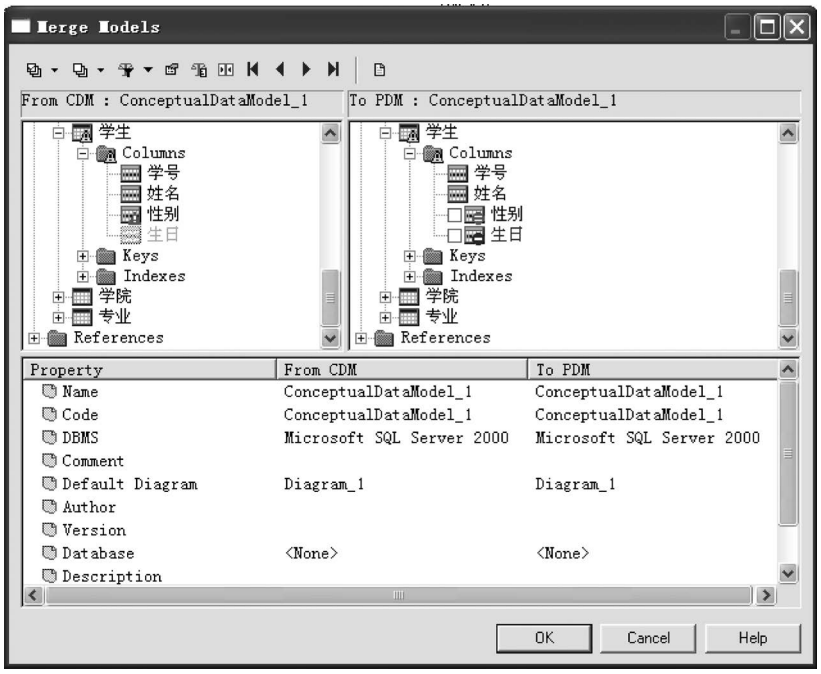

图 3-37　更新时原 PDM 和新生成 PDM 比较控制界面

本例中，希望保留 PDM 的“出生日期”，而更新 Sex 的定义，所以仅在右边“性别”前打钩。单击 OK 按钮，产生的新 PDM 将保留“出生日期”，性别类型改为 Char(1)。

显然，如果对物理数据模型有较多的修改，在概念数据模型生成物理数据模型时，对物理数据模型的一些修改可能要保留，一些修改要根据新概念数据模型产生，所以修改生成的物理模型的弊端是每次修改概念数据模型后，需要对生成的物理模型重复以上修改，尽管 PowerDesigner 提供的对生成的物理数据模型进行更新的可选机制，在更新的情况下可保留原来对物理数据模型的修改，但要记住哪些变更项要更新、哪些变更项要保留本身是一件容易出现差错的事情，所以应该尽可能地建立完善、合理的概念数据模型，不要或尽可能少地修改生成的物理模型。

3. 生成物理数据模型的其他控制

可以对概念数据模型生成物理数据模型进行其他的参数配置，其主要内容在生成物理数据模型对话框的 Detail(细节)选项卡中，如图 3-38 所示。

(1) Check mode(检查模式)：打开该开关，再生成 PDM 前检查模型是否有错，有错则停止生成。

(2) Save generation dependencies(保存生成的依据)：打开该开关，PowerDesigner 将跟踪每一个生成对象的标识，在以更新方式生成 PDM 时，即使同时修改了对象 Name 和 Code，在新老 PDM 的比较时，系统仍能识别是同一对象。

(3) Convert names into codes(把名称转换为代码)：在生成时把对象 Name 作为 Code，即以 Name 作为 PDM 的对象名(如表名和列名)。

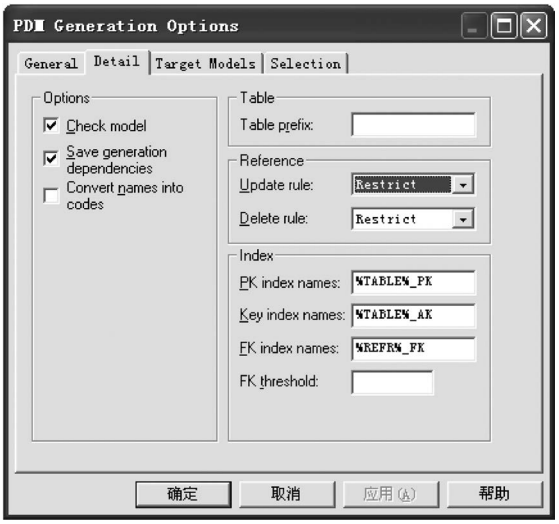

图 3-38　生成物理数据模型的参数配置界面

(4) Table prefix(表前缀)：输入的字符串将作为生成所有表表名的前缀，为了能区分一个系统中各子系统中数据表，可以为从属于某一个子系统的所有数据表的表名加一个前缀。

(5) Reference(引用)选项区域：其中，Update rule(更新规则)和 Delete rule(删除规则)分别可选 None(无)、Restrict(限制)、Cascade(级联)、Set Null(置空)和 Set Default(置默认值)，这些设置将决定对建立了引用(子表)和被引用(父表)关系的两个表，当被引用表的行中对应引用表外码的属性发生变化(被删除或修改)时，引用表应该如何做相应的变化，如学生表通过班号引用班级信息，当班级信息中发生了与班号相关的变化(被删除或修改)时，学生表应该如何变化。

① None(无)：被引用表修改，引用不变，即不进行任何控制。如班号被修改或删除，但学生表中的班号仍不变，这通常不符合实际需求，因为所有引用该班号的学生，班级信息丢失。

② Restrict(限制)：对被引用表中已经被引用的数据，不允许修改或删除。假如已经有学生的班号取值为“19001”，并且已经设置更新规则为限制，则班级表中班号为“19001”的行，班号不能被修改；如果已经设置删除规则为限制，则班级表中班号为“19001”的行不能被删除。

③ Cascade(级联)：被引用表修改了已经被引用的数据，则引用表将被同步修改。假如已经有学生的班号为“19001”，如果已经设置更新规则为级联，班级表中修改班号“19001”为“19002”，则所有学生表中班号为“19001”的行的班号被同步改成“19002”；如果已经设置删除规则为级联，删除班级表中班号为“19001”的行，则所有班号为“19001”的学生将被同步删除。

④ Set Null(置空)：被引用表修改或删除了被引用的数据，引用表中引用的数据被设置成空值。如商品通过分类代码对商品分类引用，在商品分类需要调整时，可能需要删除某些分类而保留该分类的商品，所以在删除该商品分类时，把该类商品的分类代码设置为空，表示这些商品的分类需要被重新定义。

⑤ Set Default(置默认值)：被引用表修改或删除了被引用的数据，引用表中引用的数据被设置成默认值，该默认值必须在被引用用表中存在。仍以商品和商品分类为例，在商品分类需要调整时，可能需要删除某些分类，而原属于这些分类的商品，统统把它们归类到一个特殊的分类中，此时，就可以选用本引用规则。建立该引用之前，在商品分类中设置一个特殊的分类，然后设置商品的分类代码的默认值为该特殊的分类代码。

在实际应用中，限制和级联是常用的选择，数据库管理系统提供的数据定义语句也并不支持所有的这些选择，对 SQL Server 而言，其数据定义语句也仅支持限制和级联两种规则的引用，如果上述选择为 Restrict(限制)，则不对生成的物理数据模型中的数据定义语句产生影响，因为 SQL Server 定义一个引用的默认方式就是限制方式，如果上述选择为 Cascade(级联)，则生成的数据定义语句中会出现 on delete cascade 或 on update cascade，分别表示级联删除和级联修改。对另外两种引用规则，SQL Server 可以使用触发器来实现，具体方法可见 7.4.1 节中的实例。

如果选择了数据库管理系统不支持的引用规则，在生成的物理数据模型中，双击图中生成的引用，在其特性窗口的 Preview 框中看不到生成的 SQL 语句，而是提示：

```
-- The preview is empty because of the setting.
-- Check the generation option.
```

正是由于这种局限，一些系统的分析设计人员放弃使用外码，做出这个决策应该是慎重的并且必须经过全面的分析和比较，因为放弃使用外码，也就放弃了数据库管理系统提供的确保引用数据一致性的强大功能，不得不自己写程序，例如编写触发器来确保这一点。

最后要指出的是，PowerDesigner 没有在概念数据模型中为要生成的每个关系提供引用规则的选择，而是在生成物理数据模型时提供选择，其后果是一个概念数据模型生成的所有关系都必须具有同样的引用规则，这在适用性上存在缺陷。另外，使用 PowerDesigner 9 在数据库管理系统选择为 SQL Server 的情况下，选择引用规则为级联，在生成的数据定义语句中没有包含与选择的引用规则所对应的语句，PowerDesigner 的后继版本如 12.5 修正了这个缺陷。

3.8 数据库的建立

有了物理数据模型以及数据定义语句，接下来就可以创建数据库了。创建数据库分两个步骤：第一步是使用 create database 语句建立一个空的数据库；第二步是在该数据库中建立数据表、约束、引用关系以及存储过程和触发器等内容。

1. 在 PowerDesigner 中建立数据库

物理数据模型中包含的 SQL 语句并不包括创建数据库的语句，所以可以使用 SQL Server 的企业管理器或查询分析器创建数据库，也可以在 PowerDesigner 中的物理数据模型的状态下，使用 Database(数据库)→Execute SQL(执行 SQL)菜单，在弹出的对话框中单击 Add(新增)按钮，新增一个 ODBC 数据源，该数据源可以对应任一 SQL Server 中已经存在的数据库，然后在 Machine data source(机器数据源)中选择该数据源，单击 Connect(连接)按钮后，将出现类似 SQL Server 查询分析器的界面，输入并执行建立数据库的命令。

由于原来建立的数据源并不对应新建的数据库，因此要重新建立一个数据源，使该数据

源对应新建的数据库，然后选择这个数据源，重新连接，连接成功后，在出现的类似查询分析器的界面中输入并执行创建数据库中对象的命令，这些对象将被创建在新建的数据库中。

2. 在数据库中建立数据库对象

可以在数据库中逐个建立物理数据模型中的对象，也可以一次全部建立。

(1) 逐个建立数据库对象。

在物理数据模型各个对象的特性窗口中，都有一个 Preview(预览)框，其中包含了在数据库中创建这个对象的数据定义语句，并且也包含了对需要创建的对象是否已经存在的判断以及存在情况下进行删除的操作，这样就使创建工作可重复进行。

【例 3-18】 下面是一个系统生成的建立数据表的数据定义语句：

```
if exists(SELECT 1
            FROM  sysindexes
            WHERE  Id     = object_Id('goods')
            and   Name   = 'goods_PK'
            and   indId > 0
            and   indId < 255)
   drop index goods.goods_PK
go
if exists(SELECT 1
            FROM  sysobjects
            WHERE  Id = object_Id('goods')
            and  type = 'U')
   drop table goods
go
/* ============================================================== */
/* Table: goods */
/* ============================================================== */
create table goods (
Id             char(12)            not null,
Name           varchar(20)         null,
Unit           varchar(6)          null,
costprice      price               null,
saleprice      price               null
)
go


/* ============================================================== */
/* Index: goods_PK */
/* ============================================================== */
create unique index goods_PK on goods(
Id
)
go
```

其中，if 语句用来判断索引和表是否已经存在，若存在则删除它。

建表语句中 price 为一个已定义的一个域，在概念数据模型中定义了该域，域的名称(Name)为“价格”，代码(Code)为“price”，所以在执行该语句前，先要建立 price 域，即先要执行下列语句，这些语句可以在生成的物理数据模型下的“价格”域的特性窗口中的 Preview 框中找到。

```
if exists(SELECT 1 FROM systypes WHERE Name = 'price')
      execute sp_unbindrule price, R_price
go
if exists(SELECT 1 FROM systypes WHERE Name = 'price')
begin
  execute sp_droptype price
   drop default D_price
end
go
if exists(SELECT 1 FROM sysobjects WHERE Id = object_Id('R_price') and type = 'R')
      drop rule R_price
go
create rule R_price as
       @column is null or(@column between '100' and '1000' )
go
/* ============================================================== */
/* Domain: price                                                  */
/* ============================================================== */
execute sp_addtype price, 'char(10)'
go
create default D_price
     as '500'
go
sp_bindefault D_price, price
go
execute sp_bindrule R_price, price
go
```

该域定义了最大值 1000、最小值 100 及默认值 500，SQL Server 是通过定义规则 R_price、默认对象 D_price 和使用过程 sp_addtype 定义域 price，然后把 R_price 和 D_price 绑定到 price。

(2) 一次建立所有数据库对象。

选择 Database(数据库)→Generate Database…(生成数据库…)菜单，在弹出的对话框中选择目录和文件名，所有建表的 SQL 语句将放入该文件中，并选中 ODBC Generation(ODBC 生成)选项，以建立数据表。

确认后将弹出 Connect to an ODBC Data Source(连接到一个 ODBC 数据源)的对话框，选择数据源，确认后将生成要执行的数据定义语句，单击 Execute 按钮执行这些数据定义语句。

也可以关闭 ODBC Generation(ODBC 生成)选项，在单击“确定”按钮后，系统将把所有数据定义语句放入上面指定的文件中，把这些语句粘贴到查询分析器中执行，可以达到相同的在数据库中建立所有数据对象的目的。